ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 35 Выпуск 4

ISSN Online 2220-6426
Volume 35 Issue 4

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2023

ИСП РАН

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

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

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

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

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

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

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

Труды ИСП РАН реферируются и/или индексируются в:

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

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

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

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

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

УДК004.45

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

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

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2023-35(4)-1

Обзор методов динамического анализа программного обеспечения

[1,2] *В.В. Кулямин*, ORCID: 0000-0003-3439-9534 <*kuliamin@ispras.ru*>

[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

[2] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1.

Аннотация. Данная статья представляет собой обзор методов динамического анализа программного обеспечения (ПО), в котором основное внимание уделено методам, имеющим инструментальную поддержку, нацеленным на проверку безопасности и защищенности и применимым к системному ПО. Подробно рассмотрены техники фаззинга, верификационного мониторинга и динамической символьной интерпретации. Методы и средства динамического анализа помеченных данных исключены из обзора из-за трудностей сбора технической информации о них. При рассмотрении фаззинга и динамической символьной интерпретации больше внимания уделено не отдельным инструментам, которых известно уже более 100, а техникам решения различных задач, возникающих при их работе. Также рассмотрены техники снижения эффективности фаззинга.

Ключевые слова: динамический анализ программного обеспечения; верификация; фаззинг; динамическая символьная интерпретация; верификационный мониторинг; противодействие фаззингу.

Для цитирования: Кулямин В.В. Обзор методов динамического анализа программного обеспечения. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 7–44. DOI: 10.15514/ISPRAS–2023–35(4)–1.

Survey of Software Dynamic Analysis Methods

[1,2] *V.V. Kuliamin* ORCID: 0000-0003-3439-9534 <*kuliamin@ispras.ru*>

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

[2] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia.

Abstract. The article presents a survey of software dynamic analysis methods. The main focus of the survey is on methods supported by tools, targeted on software security verification and applicable to system software. The survey examines in detail fuzzing and dynamic symbolic execution techniques. Dynamic taint data analysis is excluded due to difficulty of gathering technical details of its implementation. Review of fuzzing and dynamic symbolic execution is focused mostly on the techniques used in supporting tools, not on tools themselves, because their number exceeds 100 already. Also, the techniques of fuzzing counteraction are surveyed.

Keywords: software dynamic analysis; verification; fuzzing; dynamic symbolic execution; runtime verification; fuzzing counteraction.

For citation: Kuliamin V.V. Survey of Software Dynamic Analysis Methods. *Trudy ISP RAN/Proc. ISP RAS,* vol. 35, issue 4, 2023. pp. 7-44 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-1.

1. Введение

Данный обзор посвящен методам и инструментам динамического анализа программного обеспечения (ПО), нацеленным, чаще всего, хотя и не исключительно, на обеспечение его защищенности и безопасности. Известно очень много различных видов и техник динамического анализа, а основным фокусом этого обзора являются методы, обладающие следующими характеристиками.

- Имеющие существенную инструментальную поддержку, позволяющую значительно снизить трудоемкость применения метода по сравнению с его выполнением вручную.
- Позволяющие оценивать защищенность и безопасность ПО. Техники анализа, не оказывающие существенного влияние на защищенность ПО и не способствующие систематическому выявлению ошибок и уязвимостей, не рассматриваются.
- Пригодные для оценки ПО как изолированного продукта. Например, методы обнаружения и анализа вторжений или аномального поведения [1,2], широко используемые для оценки защищенности крупномасштабных систем, в которых совместно работает множество различного программного обеспечения, тоже далее не рассматриваются.
- Достаточно широко применимые (или имеющие значимый потенциал применения) к системному ПО: к библиотекам, драйверам и ядрам операционных систем (ОС), к реализациям телекоммуникационных протоколов, к системам управления базами данных и ПО промежуточного уровня, к инструментам разработки и исполнения ПО (компиляторам, интерпретаторам и пр.). Специализированным методам анализа прикладного ПО уделяется меньше внимания.

Кроме того, методы динамического анализа помеченных данных исключены из обзора, помимо упоминания тех случаев, когда они используются в рамках инструментов, фокусирующихся на реализации других методов. Это сделано в связи с большим разнообразием таких методов и поддерживающих их инструментов, для которого, однако, в доступных источниках обычно имеются лишь частичные и не обладающие нужной систематичностью описания, опускающие множество важных технических деталей.

Далее изложение организовано следующим образом. В разделе «Основные понятия» определены основные рассматриваемые виды динамической верификации и динамического анализа. В следующих разделах последовательно рассматриваются техники и инструменты фаззинга, включая также техники противодействия ему, верификационного мониторинга и динамической символьной интерпретации. Заключение завершает статью.

2. Основные понятия

Динамический анализ (dynamic analysis, runtime analysis) объединяет все методы верификации и анализа программного обеспечения, использующие результаты его выполнения, включая исполнение отдельных модулей или групп модулей в специально созданных окружениях. При этом методы верификации, в качестве результата выдающие оценку качества ПО или, несколько уже, оценку его соответствия требованиям, делятся на две большие группы: тестирование и верификационный мониторинг. Остальные методы динамического анализа, которые достаточно разнообразны, часто служат вспомогательными техниками для методов этих двух типов.

У большинства рассмотренных далее видов динамического анализа есть соответствующие разновидности статического анализа, которые иногда также задействуются в качестве вспомогательных техник в инструментах, основной решаемой задачей которых является динамический анализ и/или верификация. С точки зрения эффективности поиска ошибок и контроля качества динамический анализ обладает недостатком, связанным с тем, что он

выполняется лишь для тех путей исполнения, которые задействуются в реальном выполнении ПО (или его компонентов). В то же время, достоинством динамических техник анализа является очень малое количество ложных срабатываний, при которых инструмент создает сообщение об возможных ошибках, которые в реальной работе никогда не происходят. Чаще всего, все ошибки, о которых сообщают инструменты динамического анализа, достижимы на каком-либо реальном сценарии работы проверяемого ПО.

Тестирование (testing) проверяет соответствие работы ПО требованиям на наборе заранее выбранных ситуаций.

Одной из основных задач при подготовке тестирования является выбор достаточно представительного набора ситуаций, чтобы по результатам работы в них можно было судить о соответствии ПО требованиям в целом, и построение на их основе ***тестов*** (tests). Каждый тест включает в себя описание используемой ситуации в виде некоторого сценария работы тестируемой системы (последовательности обращений к ее операциям/функциям, выполнения команд, использования элементов интерфейса, отсылок сообщений и т.д., с передачей сопутствующих данных) и набор проверок, выполняемых, чтобы убедиться, что система ведет себя корректно, в соответствии с требованиями. Ситуации, создаваемые в ходе выполнения тестов, называются ***тестовыми ситуациями***.

Для того чтобы оценивать представительность набора тестовых ситуаций, т.е. значимость набора ситуаций, в которые система попадает при выполнении тестов по сравнению со всеми возможными при ее работе ситуациями, и возможность судить на основе поведения системы в тестах о ее поведении вообще, вводятся ***критерии полноты тестирования*** (test adequacy criteria, test completeness criteria) [3] или ***критерии покрытия*** (test coverage criteria). Они обычно основаны на разбиении всех ситуаций, возможных при работе данного ПО, на некоторый набор классов или типов (не обязательно непересекающихся) и определении доли задействованных при выполнении тестов из этих классов ситуаций.

Типовые критерии покрытия основываются на задействованных при работе в некоторой ситуации элементах самого ПО, элементах требований к нему или гипотезах о возможных значимых ошибках, которые в данной ситуации могут произойти. Например, ***покрытие инструкций кода*** (code statement coverage) означает долю/процент инструкций, которые были выполнены в ходе работы тестов. ***Покрытие ветвлений кода*** (code branch coverage) означает процент веток в коде (возможных выполнений then и else в условных операторах if или различных случаев case, выполненных в операторах выбора switch, выполнений, проходящих внутрь цикла while и обходящих его, и пр.), которые выполнялись в ходе работы тестов. ***Покрытие требований*** (requirements coverage) означает процент требований (от всех сформулированных), которые должны были выполняться в затронутых тестами ситуациях. ***Покрытие состояний*** (state coverage) или ***покрытие переходов*** (transition coverage) в некоторой автоматной модели, описывающей поведение тестируемого ПО, означают, соответственно, процент состояний, в которых система побывала во время выполнения тестов, от всех в принципе достижимых состояний, и процент выполненных во время тестирования переходов из всех возможных.

При автоматизации тестирования стараются отслеживать достигнутое на каждом исполнении тестов покрытие, и определять повышение или снижение покрытия при очередном выполнении тестов по сравнению с предшествующими, чтобы иметь представление о возможном повышении или снижении значимости оценки качества ПО в целом по результатам тестирования. По этой причине важно, чтобы измерение покрытия по выбранному критерию легко автоматизировалось. Покрытия, основанные на простых элементах кода (покрытие инструкций или ветвлений), легче всего поддаются измерению за счет достаточно простой инструментации исходного или исполнимого кода и потому чаще всего используется на практике.

Для тестирования, нацеленного на оценку защищенности проверяемой системы, критически важным становится достижение как можно более высокого покрытия, чтобы значительно

снизить риски необнаружения возможных уязвимостей. При этом существенно возрастают усилия, необходимые для подготовки тестов, а также возрастает значимость автоматизации тестирования, означающей возможность автоматического исполнения тестов и как можно более детерминированного воспроизведения их результатов. В силу этих факторов в качестве одного из основных методов тестирования защищенности широкое распространение получил фаззинг.

Фаззинг (fuzzing) [4] или ***фаззинг-тестирование*** является частным случаем тестирования. В его рамках тестовые ситуации массово генерируются псевдослучайным и/или нацеленным образом, возможно, путем внесения псевдослучайных или нацеленных модификаций (мутаций) в небольшой заранее подготовленный набор исходных ситуаций (входных данных или сценариев). Проверяться в таких тестах может просто работоспособность ПО в построенных ситуациях (то, что оно не падает, не создает ошибок в системе или исключений) или более сложные свойства (корректность управления памятью, отсутствие обращений за границами буферов и пр.), обычно фиксируемые с помощью инструментации исходного или бинарного кода некоторыми проверками в тех местах, где такие свойства могут быть нарушены (см. далее о мониторинге).

Обычно, более эффективным вариантом фаззинга является ***фаззинг с обратной связью по покрытию*** (coverage feedback fuzzing), при котором сгенерированные тестовые данные прогоняются через инструмент, измеряющий покрытие, и из них отбираются лишь те, которые добавляют новые, не покрытые имеющимися тестами ситуации. Отобранные данные часто сохраняются в корпусе входных данных, чтобы использовать их мутации для дальнейшей генерации тестовых данных.

Другим продвинутым вариантом является ***фаззинг с использованием динамической символьной интерпретации***, в рамках которого тестовые данные генерируются не только случайным образом, а иногда вычисляются как решения систем символьных ограничений, получаемых как условия попадания в ту или иную ситуацию (выполнения заданной ветви или нарушения заданного ограничения в коде).

Использование обратной связи по покрытию, некоторых эвристик внесения мутаций и динамической символьной интерпретации иногда позволяет достаточно эффективно (с точки зрения быстрого получения высоких уровней покрытия) генерировать тесты для достаточно объемного кода, вскрывающие редко встречаемые ошибки. Бурное развитие техник и инструментов фаззинга за последние 15 лет и достигнутый в нем высокий уровень автоматизации сделали фаззинг неотъемлемым инструментом обеспечения защищенности и безопасности ПО.

Верификационный мониторинг или ***мониторинг утверждений/свойств*** (runtime verification) — это метод верификации ПО, при котором исходный или бинарный код проверяемого ПО подвергается инструментации, вставляющей в некоторые места код, проверяющий заданные свойства (в виде проверяемых утверждений, assertions) и либо заносящий записи о найденных нарушениях в некоторый журнал, либо просто прерывающий работу ПО при нарушения утверждения, после чего инструментированный код исполняется, чаще всего в обычном эксплуатационном режиме. Иногда такой инструментированный код исполняется в рамках тестов, соответственно, в этих тестах можно не проверять отдельно требования, связанные с зашитыми в инструментированный код утверждениями.

Динамическая символьная интерпретация или ***динамическое символическое выполнение*** (dynamic symbolic execution, DSE) является методом динамического анализа ПО, обычно используемым как составляющая часть других методов. *Символическая интерпретация* или *символическое выполнение* (symbolic execution) состоит в том, что код (исходный или бинарный) подвергается анализу, при котором входные, выходные и внутренние данные (переменные, объекты, регистры процессора или участки памяти) рассматриваются как символьные переменные, между которыми устанавливаются связи в виде символьных выражений (выражающих одни данные через другие) в ходе интерпретации кода как средства

построения и трансформации этих символьных выражений. *Динамическая символьная интерпретация* является символической интерпретацией, выполняемой в динамике, при исполнении анализируемого ПО, и поэтому, в качестве исходной точки обычно имеет некоторое конкретное выполнение ПО. При этом часть данных представляется одновременно конкретными значениями и символическими выражениями, что позволяет упрощать ряд получаемых ограничений и проводить анализ более эффективно. Динамическая символьная интерпретация используется как при фаззинге или обычной генерации тестов для извлечения символьных ограничений, решая которые можно получить тестовые данные, обеспечивающие попадание в нужную ситуацию или проявление некоторой ошибки, так при других видах анализа: выявлении дубликатов кода, выявлении обхода механизмов защиты и др.

Обычный ***анализ помеченных данных*** (taint data analysis) является методом анализа, нацеленным на выявление информационных потоков, порождаемых некоторой выделенной частью входных или выходных данных при работе ПО. Иногда такой анализ используется для выявления распространения влияния некоторой части входных данных (например, возможно, содержащих злонамеренно искаженные данные или эксплойт с целью запуска постороннего кода), иногда — для выявления влияния различных данных на выводимые результаты (например, возможность проявления в выходных результатах каких-либо непредназначенных для посторонних лиц данных: ключей, паролей, секретных данных и пр.). ***Динамический анализ помеченных данных*** (dynamic taint data analysis, DTA) [5,6] отличается от статического использованием результатов работы анализируемого ПО на некоторых наборах входных данных.

Помимо инструментов фаззинга, такой анализ часто используется для выявления подверженных атакам извне компонентов и интерфейсов ПО, возможности проникновения в систему специально испорченных данных и их влияния на сохраняемую информацию, информационных потоков, образуемых от конфиденциальной информации, возможностей компрометации или подмены паролей и ключей, для восстановления используемых алгоритмов, выявления компонентов, которые могут быть подвергнуты атаке извне, и пр.

В рамках самого анализа определяются помеченные элементы данных (чаще всего «пометка» означает просто принадлежность к множеству выделенных данных, т.е. может быть выражена одним битом, но иногда используются техники анализа, в которых метки могут быть сложными, как, например, в фаззере TaintScope [7]) и правила распространения пометок при выполнении различных инструкций в коде (какие операции могут создавать помеченные данные, какие могут снимать пометки, как пометки переносятся инструкциями, как сложные метки трансформируются инструкциями и т.д.). Анализ может быть построен на распространение меток вперед, в соответствии с последовательностью выполнения инструкций в коде, или может быть обратным, выполняемым в противоположном направлении.

Инструменты DTA достаточно разнообразны по конечному назначению: от общих фреймворков, поддерживающих проведение широкого множества видов анализа, до специальных инструментов, нацеленных на решение конкретных задач, например, проверки наличия конкретных видов уязвимостей в ПО, или использующих конкретный вид DTA в качестве вспомогательной задачи, например, для определения элементов входных данных, влияющих на проявление специфической ошибки.

Далее методы и инструменты динамического анализа помеченных данных отдельно не рассматриваются. Несмотря на их значительное разнообразие и долгую историю развития (достаточно зрелые для промышленного использования инструменты подобного типа появились еще в 2005 г.), для большинства из них довольно тяжело найти в открытых публикациях значимую информацию об используемых в рамках проводимого анализа технических решениях.

3. Фаззинг

Фаззинг [4] является разновидностью тестирования, в рамках которой тестовые ситуации массово генерируются псевдослучайным или нацеленным образом и тут же используются для выполнения тестов. Генерируемые тестовые ситуации при этом могут не являться ожидаемыми или валидными с точки зрения документации на тестируемое ПО и соображений «здравого смысла» по поводу его использования. Целью такого массового тестирования является выявление ошибок, в частном случае, уязвимостей, с помощью которых можно переключить исполнение на сторонний код. При невозможности выявить ошибки мерой успешности фаззинга служит высокий уровень покрытия проверяемого кода. Индикатором ошибки служит падение ПО или, при использовании дополнительных средств мониторинга, попадание в ситуацию, где мониторинг сообщает о некорректном поведении.

Тестовые ситуации могут генерироваться в общем случае в виде сценариев выполнения проверяемого ПО, но достаточно часто на практике используются только входные данные для одной или нескольких вызываемых функций или операций, иногда в виде файла или байтового массива (полноценные сценарии чаще возникают при тестировании протоколов или библиотек компонентов со сложным внутренним состоянием, существенно влияющим на выполняемые функции). Генерация может осуществляться псевдослучайным образом, может быть нацеленной с помощью заданных форматов или грамматик входных данных, может выполняться с помощью внесения случайных или нацеленных модификаций (мутаций) в ранее набранном корпусе исходных ситуаций (входных данных) или сценариев.

Поскольку критически важной для успешности фаззинга является массовость генерируемых тестов, он всегда выполняется с помощью инструментов, фаззеров.

Сам термин «фаззинг» (fuzz) был введен на семинаре Б. Миллера в университете Висконсина в 1988 г. Первый инструмент фаззинга [8] был создан на основе результатов работы этого семинара в 1990 г. Этот инструмент случайным образом генерировал строковые входные данные, что позволило обнаружить достаточно много ошибок в стандартных утилитах Unix. С тех пор фаззеры значительно эволюционировали, превратившись в одну из необходимых составляющих процесса разработки безопасного ПО. Агентство DARPA активно содействовало их развитию [9-11], организовывая соревнования между ними по эффективности выявления ошибок и уязвимостей. Крупные компании-разработчики ПО, такие как Cisco [12], Google [13-15], Microsoft [16,17], создали собственные линейки фаззеров, внедрили их в свои процессы разработки и тратят значительные усилия на их развитие и поддержание на современном уровне.

За прошедшие 30 с лишним лет было создано более сотни инструментов фаззинга. Оценить их количество и граф генеалогических связей между ними можно на сайте [18], где для большинства представленных инструментов есть ссылки на описывающие их статьи и Web-странички самих фаззеров. Опубликовано, как минимум, 3 книги [4,19,20], рассказывающие как о техниках фаззинга, так и о доступных инструментах. Однако, про некоторые инструменты крайне тяжело найти хоть какую-то значимую информацию. Ряд инструментов имеют только репозиторий с кодом и обрывочные инструкции по их использованию. Еще часть инструментов известны лишь по небольшим, в пределах 10 страниц, статьям, в которых авторам удается изложить только базовую идею инструмента (часто, совсем не оригинальную) и, иногда, какие-то данные о результатах его применения. Понимание специфики различных методов фаззинга затрудняют также использование разнородной терминологии в разных статьях (некоторые техники называются по-разному разработчиками различных инструментов, иногда, наоборот, один термин используется для достаточно сильно отличающихся техник или явлений) и ошибки в обзорах [21], иногда вводящие читателя в заблуждение о свойствах рассматриваемых инструментов.

В данном обзоре использован материал нескольких достаточно недавних обзоров инструментов фаззинга [22-24], обзор из работы [25] и дополнительная информация из описаний самих инструментов. Целью являлось рассмотрение основных техник реализации

компонентов инструментов фаззинга, обзор наиболее характерных примеров таких инструментов, обладающих значимой спецификой по сравнению с другими.

3.1 Структура обобщенного фаззера

Для более ясного понимания проблем создания эффективного фаззера необходимо иметь представление о решаемых им частных задачах и общих подходах к проектированию подобного инструмента. Обобщенный фаззер может быть представлен состоящим из компонентов, представленных на Рис. 1. В каждом конкретном инструменте некоторые из указанных компонентов могут отсутствовать, некоторые могут быть объединены или, наоборот, разбиты на более мелкие модули.

- *Препроцессор* получает на вход *тестируемое ПО* (system under test, SUT) в виде исходного кода или в виде исполнимого файла. Он может выполнять статический анализ кода SUT (полученная информация сохраняется в конфигурации, обычно в виде данных о возможных элементах покрытия, о выполняемых SUT проверках входных данных и пр.), его инструментацию (вставляя в код SUT конструкции, выполняющие функции монитора и тестового оракула, иногда также устраняя из кода несущественные для тестирования основной его функциональности элементы), и, если нужно, сборку, готовя ее к выполнению. В тех случаях, когда он предварительно инструментирует код, результатом его работы является *инструментированная SUT*. Иногда препроцессор предоставляет некоторую часть среды выполнения, осуществляющую инструментацию SUT на лету.
- *Обработчик конфигураций* получает на вход *конфигурации* инструмента, которые представляют собой корпус ранее отобранных или полученных наборов тестовых данных или сценариев, размеченный дополнительной информацией, используемой для их приоритезации и выбора алгоритма обработки, информацию о возможных и достигнутых элементах покрытия, а также, возможно, включают историю использованных ранее параметров работы инструмента. В его задачи входит первичная обработка входящих в конфигурацию исходных данных, чтобы выделить информацию, используемую далее для нацеленной генерации подходящих входных данных. Он также выполняет выбор конфигурации, а также набора значений параметров, управляющих работой инструмента, для текущего прохода фаззинга. Он может также модифицировать корпус данных, переупорядочивая их и меняя разметку, для повышения эффективности следующих проходов фаззинга.
- *Генератор данных* создает на основе обработанной конфигурации новые тестовые данные для очередного запуска SUT (отметим, что здесь мы генерацией называем любой способ построения данных, часто в работах по фаззингу термин «генерация» применяется только к методам, не использующим другие, ранее подготовленные входные данные). Генератор данных может использовать техники псевдослучайной генерации данных, небольшие изменения (мутации) в известных данных, эвристики поиска наиболее удачных таких мутаций (с точки зрения повышения вероятности выявления ошибки или покрытия ранее непокрытых участков кода), нацеленные техники генерации данных в соответствии с заданными или выявленными форматами входных данных SUT или грамматиками, получение данных с помощью разрешения ограничений, выявленных статическим анализом или динамической символьной интерпретацией, и т.д.
- *Монитор* отслеживает информацию об очередном исполнении тестируемого ПО, и передает ее для обработки. Монитор работает в рамках инструментированной SUT или может быть частью среды выполнения SUT.
- *Обработчик прохода* обрабатывает информацию, полученную монитором, и сохраняет часть результатов в конфигурации для более эффективного нацеливания дальнейшей

генерации. В его рамках может выполняться сбор данных о достигнутом покрытии, динамическое символьное выполнение, анализ помеченных данных и пр.

- *Тестовый оракул* выявляет ошибки, ситуации некорректного поведения тестируемого ПО. Тестовый оракул может быть обработчиком аварийного завершения процесса, в котором работает SUT, или он может быть частью инструментированной SUT, представляющей собой встроенный код обработки ошибок.
  Достаточно часто монитор и тестовый оракул представляют собой с трудом разделимые части инструментации, но для удобства понимания общих задач фаззинга мы рассматриваем их отдельно.
- *Обработчик ошибок* получает информацию об ошибке и, возможно, преобразует ее для дальнейшего использования, в частности, для использования в виде уязвимости (для этого может использоваться информация от обработчика прохода). Часть информации об ошибках может вноситься в конфигурацию.

Стоит отметить, что представленное здесь разбиение фаззера на компоненты предназначено для анализа различных используемых техник, для практической реализации инструментов оно может оказаться неудобным. Разбиение фаззера на компоненты с точки зрения эффективной и конфигурируемой реализации таких инструментов рассмотрено в работе [26].

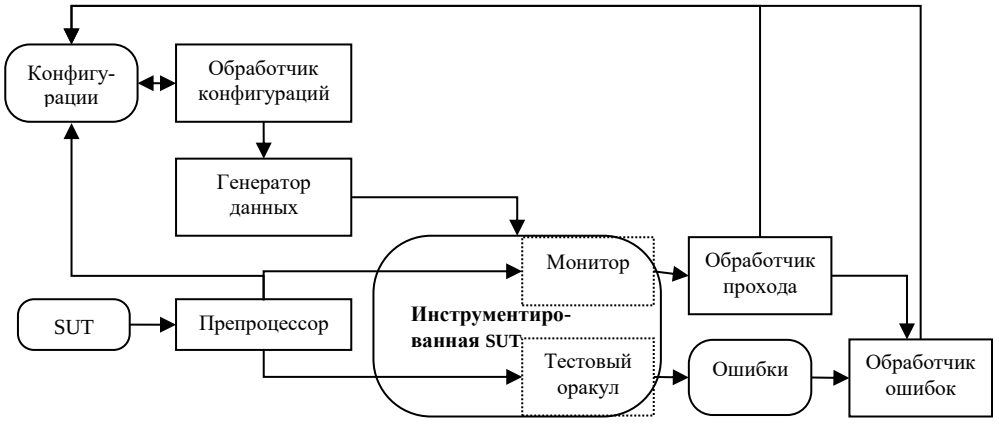

Рис. 1. Структура обобщенного инструмента фаззинга
Fig. 1. Generic fuzzer structure

3.2 Техники, используемые фаззерами

Часто используется классификация фаззеров на фаззеры черного ящика, белого ящика и серого ящика. *Фаззером черного ящика* (black box fuzzer) обычно называют инструмент фаззинга, не имеющий монитора и обработчика прохода, а также не имеющий передачи данных о структуре кода SUT из препроцессора в конфигурацию. При этом информация о найденных ошибках может использоваться для приоретизации входных данных для фаззинга. *Фаззером белого ящика* (white box fuzzer) называют инструмент, активно использующий информацию о структуре кода SUT (получаемую из монитора или из статического анализе в препроцессоре) для построения тестовых данных, чаще всего с помощью ресурсоемких методов, таких как динамическая символьная интерпретация. *Фаззером серого ящика* (grey box fuzzer) обычно называют фаззер, в котором монитор собирает лишь небольшую часть информации о коде SUT, например, только данные о достигаемом покрытии.

К сожалению, из приведенных выше определений (которым следует большинство источников) границы между этими типами фаззеров не выявляются достаточно четко.

Неясно, как именно различать фаззеры белого и серого ящиков, когда речь идет не об информации, указанной в определениях выше, а занимающей промежуточное положение между данными об элементах покрытия и полной информацией об условиях всех ветвлений. Например, фаззеры, использующие анализ помеченных данных, поскольку он менее ресурсоемкий и требует лишь часть информации о структуре кода, разные авторы относят иногда к белому ящику, иногда к серому. Более того, можно представить фаззер, основанный на случайной генерации и мутациях входных данных, как стандартные фаззеры черного ящика, но, использующий дополнительно информацию о коде разбора данных в SUT (и только ее), чтобы эффективно преодолевать места, где используются проверочные суммы, хеш-коды, магические числа (для контроля форматов и целостности входных данных). Такой инструмент не относится к фаззерам черного ящика, так как использует статический анализ кода SUT, однако почти все решаемые им задачи и соответствующие проектные решения, будут полностью аналогичны задачам и решениям в рамках обычных фаззеров черного ящика (поскольку он использует дополнительную информацию только чтобы успешно пройти код разбора входных данных в SUT).

В силу указанных причин далее эта классификация используется только для тех случаев, где имеется консенсус относительно ее применения. В спорных и промежуточных случаях она не приносит хорошего понимания возникающих проблем, там нужны более информативные описания.

Далее мы рассматриваем различные методы решения задач, возникающих при работе компонентов обобщенного фаззера.

3.2.1 Предобработка

Предобработка выполняется препроцессором. Она может использовать следующие техники.

- Инструментация исходного или исполнимого кода SUT.
  Инструментация исходного кода (source code instrumentation) используется при желании иметь меньше накладных расходов во время проведения фаззинга, однако, она не позволяет задействовать информацию из динамически загружаемых библиотек. *Инструментация исполнимого кода* (binary code instrumentation) обычно рассматривается как более ресурсоемкая, но более гибкая. Она может быть как статической, так и динамической, т.е. использующей специфическую среду выполнения для мониторинга определенных инструкций, вызовов или событий в исполняемом коде. Значительное число фаззеров используют в качестве средств динамической инструментации Pin [27] или QEMU [28], часто также используются Dyninst [29,30] или DynamoRIO [31,32]. Один фаззер может поддерживать разные техники. Например, AFL [33,34] может использовать статическую инструментацию исходного кода или динамическую с помощью QEMU, которая также может быть нацелена только на код SUT или на этот код вместе с используемыми библиотеками.
- Построение оберток.
  В некоторых случаях (библиотеки, драйвера устройств и пр.) непосредственный фаззинг самой SUT существенно затруднен, и для его выполнения необходим некоторый оберточный код, который обращается к функциям SUT, являющимся целью фаззинга. Такой код называется *оберткой* (английский термин — driver). Достаточно часто обертка разрабатывается вручную, поскольку ее можно дальше использовать без изменений, пока неизменным остается интерфейс SUT. Ряд фаззеров (IOCTL Fuzzer [35], IoTFuzzer [36]) могут генерировать простые обертки. Имеется ряд дополнительных инструментов (FUDGE [37], FuzzGen [38], IntelliGen [39]), позволяющих генерировать иногда достаточно сложные обертки, учитывающие ограничения на обращения к целевым функциям. Результаты их работы могут использоваться совместно со многими разными фаззерами.

- Сокращение исполняемого кода.
  При тестировании объемного ПО иногда имеет смысл подвергать фаззингу только небольшую важную его часть и, по возможности, экономить ресурсы на инициализации остальной части системы при каждом проходе фаазинга. Такая техника — восстановление процесса, в котором работает SUT, для очередного прохода фаззинга в некотором промежуточном состоянии, где сам фаззинг может быть повторен, но не нужно повторять действия по инициализации процесса и проходу в это состояние, называется *сокращением исполняемого кода* (соответствующий английский термин — in-memory fuzzing). Поскольку эта техника связана с существенным преобразованием исполняемой части SUT, она отнесена к предобработке.
  Инструмент Cyberdyne [11], наряду с фаззером, содержит вспомогательный инструмент для сброса и восстановления состояния процесса, GRR [40], который может быть использован и другими фаззерами. AFL [33,34], libFuzzer [41] и honggfuzz [42] сами поддерживают такой режим работы. В AFL он назван persistent mode, при нем проходы фаззинга выполняются, за счет вызова fork, начиная с некоторого состояния процесса SUT, без остановки и реинициализации. При этом за то же время можно выполнить больше проходов фаззинга, однако возможны ситуации, где на возникающие ошибки (или выполняемые ветвления в коде) оказывают влияние побочные эффекты от предшествующих проходов, из-за чего эти ошибки (или покрытие соответствующих ветвей) крайне тяжело воспроизводить.
- Инструментация для (де)рандомизации параллелизма.
  В нескольких работах [43-47] рассматривается инструментация (и использующие ее фаззеры), позволяющая сделать детерминированным выполнение параллельных нитей (threads) в SUT или, наоборот, внести больше случайности в переключения между ними. Первое позволяет сделать воспроизводимыми ошибки, связанные с различным порядком выполнения нитей, второе обеспечивает покрытие при фаззинге большего количества таких порядков, что позволяет найти больше ошибок.
- Статический анализ кода SUT на этапе предобработки.
  Статический анализ кода SUT (все равно, исходного или исполнимого) на этапе предобработки может быть нацелен на выявление элементов покрытия и условий их достижения, выявление возможных мест возникновения ошибок, определение несущественных ветвлений, уводящих от основного пути выполнения SUT (вычисление хеш-кодов, проверка магических чисел и пр.), а также определение влияния элементов входных данных на исполняемые пути в коде.
  BuzzFuzz [48] анализирует исходный код SUT и на основе указанного ему списка функций, вызов которых может привести к ошибке, выявляет места их вызова и проводит анализ помеченных данных, нацеленный на построение входных данных, приводящих к вызову этих функций с некорректными значениями параметров.
  Dowser [49] с помощью статического анализа выявляет места возможных переполнений буфера – обращения к указателям и массивам внутри циклов, – чтобы затем с помощью динамического анализа помеченных данных и символического исполнения подобрать данные, приводящие к некорректному обращению.
  GRT [50] и VUzzer [51] оба используют как статический, так и динамический анализ (последний - при обработке данных мониторинга на проходах фаззинга, а не в рамках предобработки). Статический анализ, в частности, выявляет константы, которые используются как элементы входных данных, и, в случае GRT, информацию о возможной модификации данных внутри методов.
- Динамический анализ кода SUT на этапе предобработки.
  Этот анализ может быть нацелен на решение тех же задач, что и статический. Например, TaintScope [7] помощью бинарной инструментации выполняет два вида анализа: эвристическое выделение несущественных ветвлений, которые отделяют

некорректные входные данные, и анализ помеченных данных, нацеленный на выявление элементов входных данных, влияющих на попадание в место возможной ошибки в коде.

- Модификация SUT для облегчения нахождения входных данных, позволяющих добраться до выполнения ее основной функциональности. То есть, для обхода несущественных для основной функциональности ветвлений, связанные с проверкой целостности данных, магических чисел, вычислением хеш-кодов и пр. В литературе пути исполнения кода, связанные с основной функциональностью, иногда называются hot paths, а несущественные ветвления и связанные с ними пути — cold paths.
  Примером является используемая в TaintScope [7] и T-Fuzz [52] инструментация. В первом случае несущественные ветвления определяются препроцессором на основе эвристик и обходятся в динамике, пока не обнаруживаются (частичные) входные данные, приводящие к ошибке. Далее эти данные дополняются до полного набора входных данных, обеспечивающего проход до места проявления ошибки. Во втором случае несущественные ветви обнаруживаются во время выполнения фаззинга за счет того, что они тормозят рост покрытия ветвей на генерируемых за счет простых мутаций данных, и модификация SUT происходит прямо во время фаззинга.

3.2.2 Упорядочивание и выбор исходных данных

В рамках обработчика конфигураций выполняется выбор конфигурации (scheduling) для текущего прохода фаззинга, а также упорядочивание и модификация данных конфигураций для повышения эффективности дальнейших проходов. При этом используются следующие техники.

- Выбор пула исходных данных.
  Исходные данные обычно выбираются из набора тестов, сопровождающих SUT. Достаточно часто для получения необходимого разнообразия используют поиск данных в нужном формате в Интернет, рассматривая отдельно имеющиеся известные открытые наборы данных и репозитории открытых проектов (например, для формата видео mpeg есть репозиторий проекта FFmpeg [53]). Иногда применяется конвертация данных в необходимый формат из других форматов.
- Упорядочивание и выбор исходных данных на основе информации о ранее найденных ошибках и времени выполнения фаззинга. Такие методы используются в фаззерах черного ящика, не имеющих доступа к другой информации о выбираемых данных.
  Одним из наиболее развитых примеров фаззеров, использующих подобные техники, является CERT BFF [54], сами техники описаны в работах [55,56].
  Наиболее простая техника состоит в том, что выбор исходных данных для генерации из них новых выполняется чаще, если ранее выбор этих же данных чаще приводил к обнаружению ошибки. Также вероятность выбора может определяться ранее измеренным временем прохода фаззинга на этих данных, так, чтобы фаззер, по возможности, тратил некоторое фиксированное время на обработку заданного набора данных (учитывая, что обработка одного набора может содержать несколько проходов). Еще один учитываемый фактор — количество использований набора.
- Упорядочивание и выбор исходных данных на основе информации о достигаемом на них покрытии.
  AFL [33,34] является одним из первых примеров, использующих подобные техники. Он поддерживает пул исходных данных, который при генерации новых подвергается модификации в соответствии с эволюционным алгоритмом внесения мутаций, скрещивания и отбора. При отборе функция приспособленности учитывает покрытие ветвей в коде, размер самих данных и время прохождения фаззинга для них. Далее AFL отбирает некоторое количество наиболее приспособленных наборов, и для каждого из них выполняется некоторое фиксированное количество проходов фаззинга.
  AFLFast [57] добавляет еще несколько эвристик: среди наборов данных, покрывающих

одну и ту же ветвь, выбирается использовавшийся реже, среди одинаково редко используемых — тот, который покрывает набор ветвей, использовавшийся реже. Кроме того, количество проходов, использующих один набор может изменяться, так, чтобы реже использовавшиеся наборы имели больше проходов фаззинга на них.

- Повышение эффективности пула данных (повышения вероятности обнаружения ошибок или повышения покрытия) на основе эволюционной стратегии.
  Несколько фаззеров используют эволюционные алгоритмы, чтобы в ходе фаззинга модифицировать пул исходных данных. Помимо ранее указанных AFL и AFLFast, это syzkaller [58] и рассматриваемые далее. libFuzzer [41], honggfuzz [42], go-fuzz [59] и Steelix [60] учитывают в функции приспособленности количество операций сравнения, которое смог пройти набор данных. Angora [61] вносит в функцию приспособленности на основе покрытия ветвей в коде зависимости от контекста (места, из которого была вызвана функция, содержащая эти ветви). В VUzzer [51] функция приспособленности учитывает покрываемые ветви с весами, отражающими вероятность того, что ветвь является ответвлением от пути, на котором выполняется основная функциональность, на обработку какой-либо ошибки, чтобы снизить вес несущественных ветвей.
- Сокращение пула данных на основе получаемого покрытия.
  AFL может выбрасывать некоторые наборы данных из пула, если на них достигаются те же ветви, что и на меньшем количестве из оставшихся наборов. Cyberdyne [11] пытается поддерживать минимальный пул, обеспечивающий максимальное покрытие.
- Нацеливание на более частое использование данных, покрывающих определенные ветви или участки кода с помощью оптимизационных алгоритмов. Такие техники применены, например, в AFLGo [62] и QTEP [63]. В последнем для определения проблемных участков кода, которые нужно покрывать чаще, используется статический анализ и статистическая модель, оценивающая вероятность ошибки на участке кода. Для итеративного приближения к покрытию заданных элементов кода используется, например, алгоритм симуляции отжига.

3.2.3 Генерация входных данных

В данном разделе рассматриваются методы генерации данных для очередного прохода фаззинга. Обычно все такие методы делят на *генеративные*, каким-либо образом строящие данные заданной структуры без опоры на имеющиеся примеры таких же данных, и *мутационные*, строящие данные из некоторых заданных при помощи внесения в них небольших модификаций, или при помощи разбиения их на блоки и комбинирования этих блоков. При генерации очередных тестовых данных в фаззерах применяются следующие методы.

- Использование моделей входных данных или сценариев обращений к SUT, определяющих структуру и правила их построения, вместе с возможными связями и ограничениями.
  - Использование явно описанной модели.
    Такая модель может быть задана в виде грамматики (Peach [64], Nautilus [65]), может быть описана в виде структур данных на основе некоторого API (SPIKE [66,67], Sulley [68]). Фаззеры, предназначенные для тестирования протоколов, могут использовать спецификации протоколов в виде описания структуры сообщений и автоматной модели, задающей их возможные последовательности (PROTOS [69], SNOOZE [70], KiF [71], T-Fuzz [72], *не путать с другим* T-Fuzz [52]!). Фаззеры для API ядра ОС (Trinity [73], KernelFuzzer [74], syzkaller [58]) используют шаблоны обращений к системным вызовам.
    Большинство использующих модели входных данных инструментов относятся к фаззерам черного ящика, однако, есть фаззеры [75-77], совмещающие использование

грамматик и разрешение ограничений на основе результатов символической интерпретации.

Некоторые фаззеры, нацеленные на тестирование обработчиков специфических языков, имеют встроенные модели в виде грамматик этих языков (DOMFuzz [78] для тестирования, работы с DOM, jsfunfuzz [79] для тестирования обработки JavaScript). Также встроенные модели протоколов иногда используются в фаззерах, нацеленных на специфические протоколы ([80], tlsfuzzer [81], TLS-Attacker [82]).

- Построение модели в ходе работы.

  Модель структуры входных данных или сценариев вызовов может строиться в самом начале работы, до фаззинга, или достаиваться и модифицироваться в процессе фаззинга. Обработка входных данных в начале работы относится к задачам обработчика конфигурации, но из-за близости с другими техниками использования модели входных данных мы рассматриваем ее в этом разделе.

  Техника первого типа реализована в следующих инструментах. Skyfire [83] получает на вход исходную грамматику и корпус данных, удовлетворяющих ей, после чего строит вероятностную контекстно-зависимую грамматику, учитывающую как синтаксис данных, так и выявляемые на корпусе ограничения на них. После этого уже эта модель используется для генерации данных фаззинга. TestMiner [84] анализирует большой корпус данных для выделения литералов, с использованием которых затем генерируются новые тестовые данные. CodeAlchemist [85] выделяет из кода на JavaScript такие блоки, чтобы затем, комбинируя их, получать семантически корректные программы. IMF [86] с помощью анализа логов системных вызовов строит модель возможных последовательностей обращений к ним. Learn&Fuzz [87] строит модель входных данных с помощью нейросети. Еще одна техника использования нейросетей для моделирования текстовых входных данных описана в [88].

  Построение модели входных данных в виде грамматики во время работы реализовано в [89] и в GLADE [90]. Техники построения модели протокола в процессе работы использованы в работе [91] и в фаззере PULSAR [92].

- Техники построения входных данных
  - Псевдослучайная генерация данных заданного типа.

    Эта техника обычно используется лишь на числовых данных, небольших строках и байтовых массивах.
  - Использование экстремальных данных заданного типа.

    Экстремальными могут считаться самые маленькие или самые большие представимые в рамках типа числа, байтовые массивы, целиком заполненные нулями или единицами, даты и времена, связанные с особенностями календарей (например, 29.02 и 31.12 в високосные года) или полным исчерпанием используемой для хранения даты памяти, и пр.
  - Получение новых данных с помощью небольших модификаций (мутаций) имеющихся.

    Одна из наиболее широко используемых техник мутации — обращение некоторого количества бит в исходных данных. Она используется, например, в AFL [33,34], honggfuzz [42], BFF [54], radamsa [93], zzuf [94]. В некоторых случаях количество обращаемых бит выбирается случайно. В SymFuzz [95] некоторые обращаемые биты вычисляются на основе анализа кода, общее их количество выбирается в зависимости от SUT и изменяемых данных.

    Другая техника внесения мутаций — интерпретация блока данных как целого числа и прибавление к нему или вычитание из него небольшого числа. Использована в AFL и honggfuzz.

    Часто используются блоковые мутации — вставка или замена случайно выбранного

блока случайно сгенерированным или выбранным из другого набора данных, удаление случайно выбранного блока, перестановка местами двух случайно выбранных блоков. Такие операции используются в AFL, radamsa, honggfuzz, libFuzzer [41].

- Внесение мутаций в программу генерации данных и генерация входных данных с помощью полученных мутантов. Такая техника использована в MutaGen [96].
- Разрешение ограничений (constraint solving).
  Символьная интерпретация (DSE) позволяет выявлять ограничения на входные данные, удовлетворение которых приводит к выполнению определенного элемента покрытия (инструкции или ветви) или к нарушению условия корректности в некотором месте кода, которое соответствует определенной ошибке. Чтобы получить входные данные, удовлетворяющие выявленному набору ограничений, фаззер, использующий DSE, обычно обращается к SMT-решателю [97]. SMT-решатели (SMT-solvers) способны достаточно эффективно находить данные, удовлетворяющие систему ограничений над некоторой теорией (обычно, набор поддерживаемых теорий включает целочисленную арифметику, сравнения чисел, операции над битовыми векторами, возможность указывать объекты по ссылкам и пр.) или показывать, что заданная система неразрешима. В некоторых фаззерах для ограничений специфических видов используются внутренние реализации решателей.

3.2.4 Обработка данных о проходе

Рассматриваемые в этом разделе техники выполняются при обработке данных из монитора о текущем проходе фаззинга.

- Сбор данных о покрытии.
  Фаззер, использующий данные о достигнутом покрытии для его расширения, в рамках монитора отслеживает, какие элементы целевого критерия покрытия достигаются на данном проходе, и сохраняет эту информацию в конфигурации.
  AFL [33,34] имеет в качестве целевого покрытие ветвей в коде, однако, для идентификации ветвей использует некоторые случайно сгенерированные числа, из-за чего время от времени могут происходит коллизии (идентификаторы разных ветвей могут совпадать). В рамках CollAFL [98] предложено решение, позволяющее избавиться от коллизий без значительного дополнительного расхода ресурсов.
  libFuzzer [41] и syzkaller [58] в качестве целевого используют покрытие базовых блоков (что эквивалентно покрытию инструкций). honggfuzz [42] позволяет выбирать между покрытиями базовых блоков и ветвлений.
  Angora [61] может использовать в качестве критерия покрытия покрытие ветвей с учетом стека (т.е. дополнительной информации о вызывающих функциях).
  Обычно, более сложные и детальные критерии покрытия позволяют выявлять более «хитрые» ошибки, однако их отслеживание и хранение информации о покрытых ситуациях существенно усложняется при росте детализации.
- Динамическая символьная интерпретация (DSE).
  Динамическая символьная интерпретация используется в некоторых фаззерах на этапе обработки прохода для сбора информации о покрытых ветвлениях, их условиях, и об условиях возможного проявления ошибок, которые далее используются для генерации тестовых данных, обеспечивающих покрытие непокрытых ветвлений или попадание в ситуацию возможной ошибки. Иногда фаззинг, использующий комбинацию из DSE и других техник, называют гибридным (hybrid fuzzing) [99].
  Одним из первых фаззеров, использующих DSE, стал CUTE [100], лежащая в его основе идея применения DSE для генерации тестов описана немного раньше в работе [101].
  Наиболее важные примеры фаззеров, использующих DSE, включают KLEE [102], SAGE [103,104], GWF [75], S^2E [105], Mayhem [106], Dowser [49], BORG [107], MoWF [76],

Driller [10], QSYM [108] (хотя разработчики части из них не используют термин «фаззинг» при описании своих инструментов). Можно также упомянуть ИСП Crusher [109,110], способный использовать DSE с помощью среды Sydr [25,111]. TaintScope [7] использует DSE только для того, чтобы выявить полный набор входных данных, необходимый для достижения ошибки, после того как был выявлен частичный набор данных, приводящей к ней при игнорировании несущественных ветвлений.

Более подробно различные проблемы и техники реализации DSE рассмотрены в посвященном ей разделе ниже.

- Динамический анализ помеченных данных.

  Динамический анализ помеченных данных используется на этапе обработки прохода для выявления элементов входных данных, влияющих на выполнение ветвления в коде или на возникновение ошибки.

  Выше в разделе о предобработке есть примеры использования DTA в препроцессоре.

  Dowser [49] использует DTA для выявления элементов входных данных, которые влияют на возможные операции разыменования указателей, затем применяет DSE для вычисления ограничений на эти элементы и их разрешение для получения соответствующих входных данных.

  GRT [50], SYMFuzz [95], BORG [107], VUzzer [51] и Angora [61] используют DTA для выявления элементов входных данных, влияющих на выполнение ветвей в коде.

  REDQUEEN [112] с помощью DTA выявляет константы (магические числа и др.), использование которых во входных данных позволяет избежать выполнения несущественных ветвей.

3.2.5 Обработка ошибок

В этом разделе рассматриваются техники построения тестовых оракулов и техники обработки ошибок, используемые в инструментах фаззинга.

- Обнаружение ошибок.

  Обнаружение ошибок часто происходит за счет фиксации падения тестируемого ПО. Однако, для многих ошибок само исполнение кода, их содержащего, может не приводить к разрушению выполняющегося процесса, поэтому часто прибегают к дополнительной инструментации (исходного или бинарного кода), позволяющей выявить некорректное поведение. Такая инструментация внедряет в код разного рода утверждения (assertions) и проверки, которые демонстрируют ошибочную ситуацию.

  Более подробно разновидности проверок и средств для выявления ошибок при выполнении рассмотрены ниже в разделе о верификационном мониторинге.

- Обработка ошибок.

  В рамках фаззинга обработка данных об ошибках (англоязычный термин, triage, означает также сортировку раненых или пострадавших по степени полученного ими ущерба) иногда включает модификацию разметки в конфигурации (чтобы отразить риск возникновения ошибки для использованных входных данных — см. выше раздел об упорядочении и выборе исходных данных), выявление дубликатов (deduplication), минимизацию данных для выявления ошибки, оценку возможности использования выявленной уязвимости (exploitability assessment). Для решения этих задач фаззеры также могут использовать специализированные инструменты, такие как CASR [113,114].

  - Выявление дубликатов.

    Для выявления дубликатов ошибок, т.е. для определения того, что на разных проходах фаззинга фиксируется одна и та же ошибка, используются 3 техники: хеширование стека (stack backtrace hashing), выявление дубликатов на основе покрытия (coverage-based deduplication) и семантическое выявление дубликатов (semantic-aware deduplication).

    Хеширование стека [115] использует эвристику, утверждающую, что если несколько

последних вызванных функций в стеке перед возникновением двух ошибок одинаковы, то это одна и та же ошибка. Глубина обрезания стека (сколько последних вызванных функций сравнивать) может быть разной: в [56,115] используется 3, CERT BFF [54] использует 5, а MutaGen [96] — разные числа в разных случаях. Ясно, однако, что эта эвристика не аккуратна, она может определять разные ошибки как одну и наоборот, одну как различные.

Выявление дубликатов на основе покрытия используется в AFL [33,34]. Он считает ошибку уникальной, если при ее возникновении была покрыта ветвь, которая не покрывается в других случаях, или, наоборот, не покрывается ветвь, покрытая в других случаях.

Семантическое выявление дубликатов реализовано в специализированном инструменте RETracer [116], который использует обратный анализ помеченных данных от дампа памяти, полученного при возникновении ошибки.

- Минимизация данных для выявления ошибки.
  Такая техника реализована в AFL и BFF. В первом случае фаззер пытается уменьшить объем исходных данных и обнулить как можно большую их часть, при этом сохраняя выявленную ошибку. Во втором случае минимизируется разница в битах между выявляющими ошибку данными и теми валидными исходными данными, из которых была получен набор, выявивший ее.
  Иногда можно использовать специализированный инструмент, например, [117], для минимизации данных, выявляющих ошибку, после ее получения.
- Оценка возможности использования выявленной уязвимости.
  Такая оценка может выносится на основе эвристик [118], считающих что деление на 0 трудно превратить в программу, исполняющую посторонний код, а переполнение буфера — достаточно легко. Mayhem [106] в некоторых случаях позволяет по найденной ошибке сгенерировать такую программу (эксплойт).

3.3 Примеры фаззеров по областям применения

В данном разделе рассматриваются важные примеры фаззеров, сгруппированные по областям их применения.

- Фаззеры общего назначения. Они не имеют выраженной специфичной области использования, чаще используются для тестирования общих приложений.
  - Peach [64] (2004). Один из первых широко известных фаззеров черного ящика. Применялся для тестирования общего ПО, протоколов, встроенных устройств. Использует грамматики в специализированном формате (Peach Pits) для описания структуры корректных входных данных. Для генерации данных используются псевдослучайные генераторы и мутации. Алгоритмы генерации данных и внесения мутаций могут быть настроены. Является основой для нескольких других фаззеров, включая honggfuzz [42].
    Код открыт, с 2013 г. не развивается.
  - KLEE [102] (2008). Один из первых промышленно применимых фаззеров на основе DSE. Разработан в университете Стэнфорда, является развитием более раннего инструмента EXE [119]. Использует инструментацию исходного кода на уровне промежуточного представления LLVM. Последовательно симулирует выполнение обнаруживаемых путей в коде, используя fork для снижения накладных расходов при переключении на выполнение нового пути.
    Код открыт [120], продолжает развиваться.
  - SAGE [103,104] (2008). Тоже один из первых промышленно применимых фаззеров на основе динамической символьной интерпретации. Использует инструментацию бинарного кода платформы x86. Разработан в Microsoft, стал стандартной частью

процесса разработки для ряда приложений этой компании. Комбинирует случайную генерацию входных данных и эвристики поиска данных, нацеленные на повышение покрытия.

- American Fuzzy Lop, AFL [33,34] (2013). Широко известный фаззер, использующий покрытие ветвей для нацеливания, эволюционный алгоритм и мутации исходных данных для генерации новых данных на базе стартового пула. Реализует сокращение исполняемого кода за счет запуска новых проходов фаззинга с помощью fork. Является основой для большого числа других фаззеров.
  Код открыт [33], с 2017 г. не развивается, поддержка передана в Google [34], там нет развития с 2021 г. Активно развивается более продвинутая версия AFL++ [121,122], тоже с открытым кодом.
  Часто используемыми фаззерами, развивающими идеи AFL, являются также libFuzzer [41] и honggfuzz [42].

- Фаззеры для компонентов ядер операционных систем.
  - Trinity [73] (2004). Первый фаззер черного ящика для системных вызовов ядра Linux. Использует генерацию данных по спецификации их типов (можно указать базовые типы C, числовые интервалы, наборы флагов и пр.).
    Код открыт, продолжает развиваться, поддерживая совместимость с текущим состоянием ядра Linux.
  - IOCTL fuzzer [36] (2009). Фаззер для библиотек ядра и драйверов ОС Windows.
    Код открыт, с 2011 г. не развивается.
  - syzkaller [58] (2016). Реализовал возможность нацеливания на покрытие при фаззинге ядра ОС. Использует шаблоны обращений к системным вызовам в качестве исходного набора данных.
    Код открыт, продолжает развиваться.
  - kAFL [123] (2017). Фаззер для компонентов ядра ОС, в определенной степени независимый от ОС. Использует виртуализацию и встроенную технику трассировки инструкций процессоров Intel.
  - CAB-Fuzz [77] (2017). Фаззер для компонентов ядра ОС, использующий динамическое выполнение.

- Фаззеры для реализаций телекоммуникационных протоколов.
  - SPIKE [66,67] (2001). Фреймворк для построения генераторов данных. Содержит набор API (application program interface) для описания структур генерируемых данных, включая разнообразные зависимости и вычислимые поля.
    Код открыт, с 2017 г. не развивается.
  - Sulley [68] (2016). Фреймворк для фаззинга протоколов. Поддерживает использование случайных генераторов и мутаций, а также обработку ошибок в виде их классификации и генерации трасс, повторяющих ошибку.
    Код открыт, с 2017 г. не развивается (в 2019 г. изменен README.md).
    Развитием Sulley является boofuzz [124], его код также открыт и продолжает развиваться.
  - Можно также отметить фреймворк Defensics [125], также предназначенный для фаззинга реализаций протоколов и содержащий значительный корпус тестов для более чем 150 разных протоколов.
  - Имеется достаточно много фаззеров, способных генерировать тесты для одного заданного протокола. Можно отметить среди них SNOOZE [70] и KiF [71] для VOIP/SIP, TLS-Attacker [82] для TLS, Secfuzz [126] для IKE.
  - Обзор специфики фаззинга реализаций протоколов можно найти в [127].

- Фаззеры для компиляторов и интерпретаторов.
  - jsfunfuzz [79] (2007). Основанный на грамматике фаззер черного ящика для обработчиков JavaScript.
    Код открыт, продолжает развиваться.
  - Csmith [128,129] (2011). Поддерживает генерацию случайных программ на языке C, удовлетворяющих стандарту C99.
    Код открыт, продолжает развиваться.
  - LangFuzz [130] (2012). Развитие jsfunfuzz, может поддерживать несколько языков, но использовался только для JavaScript и PHP. Использует случайную генерацию программ по блокам и мутации.
  - Обзор специфики фаззинга для компиляторов можно найти в [131].
- Фаззеры для встроенных систем и устройств.
  - Можно отметить VDF [132] (2017), основанный на эволюционной стратегии фаззер для виртуальных устройств, а также IoTFuzzer [36] (2018).
  - Область фаззинга встроенных систем и устройств активно развивается в последние годы, обзоры специфичных для этой области инструментов и решений можно найти в [133-135].

3.4 Техники снижения эффективности фаззинга

Бурное развитие инструментов фаззинга привело к тому, что во второй половине 2010-х годов они стали активно использоваться хакерами для выявления ранее неизвестных уязвимостей в широко используемом ПО с целью дальнейшего проведения атак. В связи с этим встала задача борьбы с подобным использованием фаззеров одновременно с сохранением, по возможности, эффективности фаззинга при его использовании разработчиками. Эффективность фаззинга при этом понимается как количество обнаруживаемых ошибок и достигаемых элементов покрытия за единицу времени выполнения и/или на определенном числе его прогонов.

Обычно предполагается, что злоумышленнику, использующему фаззер для создания атак, целевое ПО доступно только в виде предназначенного для эксплуатации исполнимого кода, в то время как разработчики имеют доступ к исходному коду и могут собирать специализированные исполнимые файлы, используемые для «хорошего» фаззинга с целью поиска и дальнейшего исправления ошибок и уязвимостей. Во втором случае эффективность фаззинга не должна заметно снижаться, поэтому изменения, вносимые в код рассматриваемыми техниками решения этой задачи, обычно отключаются. Использование же предназначенного для эксплуатации исполнимого кода должно существенным образом снижать эффективность фаззинга, незначительно влияя при этом на эффективность обычного выполнения, чтобы не сказаться на производительности работы пользователей.

Также, использование техник противодействия фаззингу должно отслеживаться (и отключаться) при проведении независимого фаззинга, например, при сертификации безопасности ПО.

Одними из первых работ, описывающих практически значимые техники противодействия фаззингу, являются [136,137]. Многие такие техники реализованы в виде инструментов трансформации кода или библиотек-фреймворков, подключаемых при сборке.

Методы снижения эффективности фаззинга обычно используют комбинации следующих техник.

- Обфускация и сокрытие кода.
  Обычные техники обфускации кода в виде рандомизированных перестроений потока данных, замены констант, шифрования части данных и пр. [138,139], сами по себе не слишком подходят для решения данной задачи. Они, во-первых, значительно снижают

производительность кода при обычной работе, а, во-вторых, не слишком влияют на количество достигаемых элементов покрытия на заданном числе прогонов фаззинга (хотя и значительно повышают время их выполнения). Однако обфускация иногда может использоваться в комбинации с другими техниками для решения отдельных подзадач.

К этой же группе можно отнести и различные техники сокрытия кода (шифрование, динамическая загрузка, динамическая модификация и пр.), которые предназначены для затруднения его анализа, и часто используются в злонамеренном ПО, чтобы избежать его обнаружения. Например, SAFTE [140] использует для противодействия фаззингу динамическую загрузку кода программы из данных.

- Детектирование работы в режиме фаззинга.
  Некоторые авторы предлагают определять использование ПО в режиме фаззинга, чтобы включить техники противодействия ему. Для этого используются техники определения режима отладки [137], например, детектирование использования ptrace на Linux, а также техники, основанные на статистических шаблонах использования функций и создания событий [141].
  No-Fuzz [142,143] использует создание временных файлов в несущественном коде, чтобы по большому числу таких файлов детектировать применение фаззинга.

- Замедление выполнения несущественных ветвлений.
  Основано на автоматическом выделении несущественных ветвлений (см. выше, раздел про предобработку). Реализуется в виде добавления значительно замедляющих выполнение инструкций на несущественных ветвлениях, не связанных с выполнением основных функций. При этом обычная работа, в ходе которой эти ветвления почти никогда не используются, не замедляется, а фаззинг, при котором эти ветвления исполняются довольно часто, замедляется существенно. На количество прогонов фаззинга для достижения нужного покрытия эта техника не сильно влияет.
  В рамках [144] несущественные блоки кода выделяются на основе статистического анализа частоты их выполнения. ANTIFUZZ [145,146] предлагает вставлять вручную замедляющие вызовы в код обработки некорректных данных.

- Вставка многочисленных ветвлений, зависящих от входных данных.
  Реализуется за счет генерации большого числа ветвлений, зависящих от входных данных (вплоть до того, что почти каждый байт входных данных влияет на исполнение какого-нибудь ветвления), но не добавляющих никакой функциональности. Могут также использоваться многочисленные рандомизированные переходы на сгенерированный компилятором несущественный код. При анализе такого кода фаззер тратит значительные усилия на то, чтобы попасть в каждое ветвление. Дополнительно, само количество возникающих ветвей (могут добавляться десятки тысяч) может переполнять используемые фаззером хранилища для данных о покрытии. Используется в [142-146], а также в VALL-NUT [147].

- Вставка ложных ошибок или маскировка ошибок.
  Некоторые методы усложняют выполнение фаззинга с помощью затруднения определения ошибок. Для этого используется вставка многочисленных «ложных» ошибок, которые, однако, нельзя использовать для построения атак, в коде, связанном с выполнением несущественных ветвлений [145,146,148] или маскировка ошибок с помощью перехвата сигналов об ошибочном завершении работы ПО и обычного завершения работы в рамках кода-перехватчика [137,145,146]. Для маскировки может использоваться и перехват работы ptrace, поскольку некоторые фаззеры (например, honggfuzz) используют и этот механизм.

- Усложнение предикатов ветвлений и потоков данных.
  Для снижения эффективности фаззинга, использующего символическую интерпретацию, выполняется усложнение предикатов ветвлений, за счет чего значительная их часть

становится трудноразрешимой. Для противодействия анализу помеченных данных, аналогично, используется запутывание потоков данных, введение в них дополнительных (часто ложных) зависимостей.

В качестве такого усложнения используется замена сравнения строк или байтовых массивов на сравнение их хеш-кодов [144-146], шифрование, а затем расшифрование части входных данных [145,146], замена некоторых строк или массивов данных на их копии, полученные с помощью нетривиального копирования буферов (каждый элемент получается равным соответствующему элементу исходного буфера, но при этом он проходит через несколько преобразований) [144]. Могут использоваться замены в выражениях арифметических операций на гомоморфно преобразованные результаты операций над обратно преобразованными аргументами.

No-Fuzz [142,143] использует замену использования переменных на обращения к функциям, вычисляющим значения этих переменных сложным для анализа способом (например, числовая переменная *y*, имеющая значение больше 1, заменяется на вызов функции $f(1-1/y)$, где функция $f(x) = 1/(1-x)$ вычисляется при помощи неполного суммирования ряда $\sum_{i=0}^{N} x^i$). Также No-Fuzz использует результаты итеративных преобразований, приводящих к неподвижным точкам, несмотря на исходные данные (постоянная Капрекара [149]), в качестве констант. Такие константы выглядят для символического анализа как числа, сложным образом зависящие от (произвольно выбираемых) исходных данных.

4. Верификационный мониторинг

Верификационный мониторинг [150] (runtime verification) или ***мониторинг утверждений/свойств*** является методом верификации ПО, при котором исходный или бинарный код проверяемого ПО подвергается инструментации, вставляющей в некоторые места код проверки некоторых заданных свойства (в виде утверждений, assertions, или в другом), либо записывающий обнаруженные нарушения в некоторый журнал/трассу, либо просто прерывающий работу ПО при нарушении проверяемого свойства, после чего инструментированный код исполняется, чаще всего в обычном эксплуатационном режиме. Такой инструментированный код может выполняться в рамках тестов, при этом тесты могут не проверять отдельно требования, связанные с записанными в инструментированный код свойствами.

Верификационный мониторинг стал полноценной областью исследований в последние 25 лет, до этого спорадически выполнялись работы, близкие по тематике, но их авторы были разрознены и не пытались обобщить полученный опыт. Первые работы по инструментам верификационного мониторинга появились в 2000-2001 гг. [151,152]. Сейчас это достаточно развитая область, есть книги по верификационному мониторингу [150], несколько достаточно аккуратных обзоров [153-155].

Есть работы схожей тематики, которые относят себя к области *пассивного тестирования* [156,157] (passive testing). Они являются переносом техник формального тестирования протоколов, использующих автоматные модели, на случай мониторинга, т.е. пытаются на основе некоторого набора выполнений SUT оценить, насколько аккуратно проверены разные элементы поведения, описываемого моделью.

Основные темы, рассматриваемые в этой области, таковы.

- С помощью каких формальных теорий и нотаций (языков спецификаций) можно выразить нужные свойства так, чтобы их проверка во время работы проверяемого ПО (system under test, SUT) была возможна и достаточно эффективна.
- Как обеспечить корректную инструментацию и оценку работы SUT. Какие техники инструментации (исходного или бинарного кода, предварительные или во время выполнения, с привлечением аппаратной инструментации или без нее, и пр.) и оценки

корректности (во время выполнения, это так называемая *онлайн верификация*, или по трассе, уже после выполнения, что называется *офлайн верификацией*) использовать.

- Как добиться приемлемого снижения производительности при работе инструментированной системы.

В связи с автоматизированной верификацией свойств защищенности три вида работ вносят наиболее заметный вклад в данную область.

- Работы, посвященные автоматизированному мониторингу свойств защищенности отдельных систем, описанных в виде выделенной формальной модели.
  В работах такого типа описывается некоторая техника инструментированного мониторинга (обычно на основе трассы событий, происходивших в SUT) свойств, описанных в целостной формализованной модели безопасности [158-161].
  Такая верификация весьма ресурсоемка и часто не может производиться непосредственно во время работы SUT, из-за чего выполняется офлайн, по трассе событий, связанных с вызовами определенных функций или обращением к определенным данным. Однако предлагаемые в таких работах техники дают возможность строго проверить единовременно большое количество значимых требований к безопасности SUT, которые иными путями тяжело верифицировать.
- Работы, посвященные инструментам мониторинга, проверяющим настраиваемые или описываемые в виде дополнительных входных данных свойства безопасности.
  Часто это тоже достаточно ресурсоемкие инструменты, требующие предварительной разработки конфигурации/описания проверяемых свойств, поскольку они используют языки спецификаций, не всегда близкие по синтаксису и набору базовых понятий к языку SUT. Они отличаются друг от друга формализмами и нотациями, используемыми для описания проверяемых правил, поддерживаемыми языками SUT, техниками внедрения мониторов в проверяемый код и пр.
- Работы, посвященные инструментам мониторинга, проверяющим небольшой фиксированный набор специфических свойств, которые можно верифицировать достаточно эффективно, чтобы производительность инструментированной SUT не слишком отличалась от исходной (обычно приемлемым считается снижение производительности в 2-5 раз).
  Инструменты этого типа находят широкое применение в фаззинге в качестве тестовых оракулов, компонентов фаззера, выявляющих некорректное поведение SUT.

Работы первого типа обычно жестко связаны с верифицируемой системой и требуют значительной модификации средств мониторинга для переноса на другие системы. В следующих подразделах рассматриваются инструменты второго и третьего типов.

4.1 Инструменты с настраиваемыми проверками

Для таких инструментов в 2014-2016 проводились соревнования [162-164], сравнивающие эффективность инструментов как с точки зрения выявления ошибок, так и по снижению производительности SUT.

Стоит отметить следующие инструменты верификационного мониторинга с настраиваемыми проверяемыми свойствами.

- E-ACSL [165-167] (Executable ANSI/ISO C Specification Language, 2013).
  Реализует мониторинг программ на C, проверяемые свойства описываются на подмножестве ACSL [168]. ACSL представляет собой расширение C при помощи спецификационных комментариев, позволяющих указывать инварианты для типов данных и циклов, свойства, которые должны выполняться для глобальной или локальной переменной, предусловия и постусловия функций в виде логических выражений, почти всегда имеющих C-подобный синтаксис.
  Код открыт, последние изменения в 2017 г.

- RiTHM [169] (Runtime Time-triggered Heterogeneous Monitoring, 2013).
  Реализует мониторинг программ на C, проверяемые свойства описываются на расширении LTL (Linear Temporal Logic), используемое расширение и алгоритмы мониторинга описаны в [170]. Для проведения оценки может использовать распараллеливание как на обычных процессорах, так и на графических ускорителях (GPU).
- Larva [171-173] (2009).
  Инструмент мониторинга Java программ, языком спецификаций служит DATEs [174], разновидность событийных автоматов с таймерами. Фрагменты мониторингового кода вставляются в код SUT при помощи описания аспектов и точек вставки на AcpectJ.
  Код открыт, последние изменения в 2022 г.
- RV-Monitor [175,176] (2014).
  Независимый от языка инструмент мониторинга, оптимизированный для одновременного мониторинга многих свойств. На его основе построены расширения для Java и для C. Также имеет несколько плагинов для описания спецификаций в разных формализмах (временные логики, расширенные конечные автоматы, системы переписывания термов и пр.). Еще одним его расширением является RV-Android [177], фреймворк для контроля безопасности приложений на платформе Android.
  Код открыт, последние изменения в 2020 г.
- MarQ [178] (Monitoring at runtime with Quantified Event Automata, 2015).
  Инструмент мониторинга Java программ, поддерживает как онлайн, так и офлайн верификацию. Спецификации описываются в нотации событийных автоматов с числовыми метками (QEA, Quantified Event Automata), интегрируются в SUT при помощи AspectJ.
- Mufin [179,180] (Monitoring with Union-Find, 2016).
  Инструмент мониторинга Java программ, мониторы в нем определяются при помощи специализированного API. Использует специализированные алгоритмы для мониторинга большого количества объектов.

4.2 Инструменты с фиксированными проверками

Далее рассматриваются наиболее широко используемые инструменты мониторинга, выполняющие проверку жестко зафиксированного небольшого множества свойств.

- Мониторинг ошибок использования памяти.
  Эти инструменты нацелены на выявление некорректной работы с памятью: обращений к памяти за пределами объекта или буфера (пространственная корректность, spatial safety), или обращений к неинициализированной/ освобожденной памяти (временная корректность, temporal safety).
  - AddressSanitizer [181,182] (ASan).
    Инструментирует исходный код для выявления некорректных обращений к памяти, используя теневую память, которая хранит разметку текущей памяти процесса на безопасную/опасную для обращений. Снижение производительности при его использовании обычно остается в пределах 2-3 раз.
    Имеется интегрированный в QEMU вариант, QASan [183].
    Код открыт [182], последние изменения в 2019 г.
  - MEDS [184].
    Нацелен на эффективное обнаружение некорректных обращений к памяти в крупномасштабных приложениях, с большими объемами рабочей памяти. Немного больше снижает производительность, чем ASan, но позволяет обнаруживать больше ошибок при фаззинге с помощью AFL.

- SoftBound/CETS [185,186].
  Инструментирует исходный код. Связывает границы и признак занятости с каждым указателем, поэтому теоретически способен обнаружить все ошибки некорректного использования памяти. Это обеспечивается за счет большего снижения производительности, которое, однако, в большинстве случаев на превосходит 3-х раз.

- Мониторинг ошибок приведения типов.
  Подобные ошибки возникают в C++ в связи с неаккуратным использованием конструкций static_cast и dynamic_cast для приведения типов указателя на объект к дочернему типу. В некоторых случаях могут приводить к появлению уязвимостей. К инструментам, нацеленным на обнаружение ошибок этого вида, относятся CAVER [187], TypeSan [188], HexType [189]. Они дополняют код информацией о реальных типах объектов и указателей и проверками корректности приведения типов. Приводят к снижению производительности от 1.7 до 5 раз. TypeSan и HexType позволяют выявлять подобные проблемы для локальных и глобальных переменных, и для объектов на стеке.

- Мониторинг случаев неопределенного поведения.
  В языках C/C++ в достаточно многих ситуациях стандарт не фиксирует точную семантику, определяющую поведение тех или иных конструкций языка, позволяя разработчикам компиляторов использовать такие места для более эффективной оптимизации. Однако в ряде случаев конструкции с неопределенной семантикой могут становиться источниками ошибок и уязвимостей [190].

  - Valgrind [191,192].
    Наиболее известный инструмент этого типа. Использует статическую (до начала выполнения) трансформацию бинарного кода. Одна из основных функций связана с контролем использования памяти (Memcheck) — вся память при инструментации получает метки о ее занятости или свободе, все выделяемые блоки памяти окаймляются дополнительными блоками, попадание в которые служит индикатором выхода за границы корректно используемой памяти. Помимо проверки обращений к памяти может проверять работу со стеком, искать состояния гонок (data races). Снижение производительности при использовании Valgrind достаточно велико (бывает 10-50 раз, иногда больше), поэтому он редко используется с фаззерами.
  - Dr. Memory [193].
    Выявляет ошибки, связанные с использованием неинициализированной или освобожденной памяти. По сравнению с Memcheck в Valgrind инструментированный Dr. Memory код работает в 2-3 раза быстрее, но для фаззинга это тоже оказывается часто недостаточно эффективно.
  - MemorySanitizer [194,195] (MSan).
    Еще один монитор ошибок, связанных с использованием неинициализированной памяти в C и C++. За счет оптимизированного использования теневой памяти снижает производительность в 2-4 раза. Достаточно широко используется с фаззерами.
    Код открыт, последние изменения в 2020 г.
  - UndefinedBehaviorSanitizer [196,197] (UBSan).
    Монитор большого количества типов ошибок, связанных с неопределенным поведением в C/C++. Снижение производительности при полном наборе проверок может быть большим, каждый отдельный вид проверок обычно дает снижение не более чем в 2-3 раза, но использование совместно проверок нескольких видов снижает производительность более значительно.
    Код открыт, последние изменения в 2020 г.

- ThreadSanitizer [198,199].
  Монитор условий гонок (data races) в многопоточных программах.
  Код открыт, последние изменения в 2020 г.

5. Динамическая символьная интерпретация

Символическая интерпретация или ***символическое выполнение*** (symbolic execution) состоит в том, что код (исходный или бинарный) подвергается анализу, при котором участвующие в нем данные (переменные, объекты, регистры процессора или участки памяти) рассматриваются как символьные переменные, между которыми устанавливаются связи в виде символьных выражений (выражающих одни данных через другие) в ходе интерпретации инструкций кода как трансформаций этих символьных выражений. ***Динамическая символьная интерпретация*** (dynamic symbolic execution, DSE) является символической интерпретацией, выполняемой в динамике, при исполнении анализируемого ПО, и поэтому, в качестве исходной точки проводимого анализа обычно имеет некоторое конкретное выполнение, с конкретными входными данными. Другим термином для динамической символьной интерпретации является ***конколическое выполнение*** (concolic execution) [100] или ***конколическая интерпретация*** (concolic interpretation), здесь англоязычный термин образован из склейки слов конкретный (concrete) и символический (symbolic). При таком исполнении часть данных представляется одновременно конкретными значениями и символическими выражениями, что позволяет упрощать ряд получаемых ограничений и проводить анализ с меньшими накладными расходами.

Динамическая символьная интерпретация используется при фаззинге или обычной генерации тестов для извлечения символьных ограничений, решая которые можно получить тестовые данные, обеспечивающие попадание в нужную ситуацию или проявление некоторой ошибки. Она может применяться и при других видах анализа: выявлении дубликатов кода, выявлении способов обхода механизмов защиты, формальной верификации кода и др.

Сама идея использовать символическое выполнение при тестировании для получения ограничений на входные данные, которые затем разрешаются, давая исходные данные для тестов, известна уже почти 50 лет [200-203].

Практически реализуемое применение динамической символьной интерпретации (DSE) для генерации тестов описано в работе [101], в 2005 г. Далее идеи этой работы привели к созданию фаззера, использующего DSE, которым стал CUTE [100], хотя он еще оставался на уровне исследовательского прототипа. Сам термин «конколическое выполнение» был введен в описывающей его статье. Первые промышленно применимые инструменты фаззинга на базе DSE [102,103] появились позже, в 2008 г.

Далее рассматриваются основные техники, используемые при реализации инструментов DSE, большая часть материала взята из обзоров [99,204,205] и обзора в работе [25].

Прямое символическое выполнение некоторого кода выглядит как последовательная трансформация его инструкций в символические преобразования их входных данных в результаты и последовательное же раскрытие всех встречающихся разветвлений в вычисление различных выражений на разных ветках. Применение его в таком виде к достаточно объемному реалистичному коду сталкивается со многими проблемами.

- Необходимость сокращения количества анализируемых путей.
  Простое размножение путей при выборе ветвлений даже на простых циклах быстро приводит к комбинаторному взрыву.
- Необходимость символической обработки указателей и массивов.
  В программах часто данные берутся из некоторой памяти по адресам, причем сами эти адреса вычисляются динамически. Необходимо каким-то образом описывать и анализировать возникающие при этом зависимости между данными, для этого

используется так называемая модель памяти, основная задача которой — моделирование операций с адресами.

- Необходимость обработки обращений во внешние библиотеки и системы.
  Прямая символическая интерпретация вызовов внешних функций невозможна без полного доступа к их коду, поэтому необходимо как-то моделировать работу с окружением анализируемой системы.
- Чтобы провести анализ нового пути в рамках инструмента DSE, чаще всего требуется решить соответствующие ему ограничения. Также разрешение полученных ограничений нужно и при использовании DSE в рамках фаззинга или какой-то другой деятельности. Обычно при этом возникает необходимость дополнительной обработки выявленных символических ограничений при их передаче решателю.
  При этом важно, какие именно теории и операции используются решателем, как обрабатываются инструкции, не укладывающиеся в поддерживаемые решателем теории. Может использоваться внешний решатель, собственная реализация разрешающего алгоритма, или какая-то смешанная техника. Перед передачей для обработки решателем могут выполняться оптимизирующие преобразования и сокращения ограничений.
  В инструментах поиска ошибок ограничения могут использоваться не только для описания путей, но и задания условий возникновения ошибок.

Помимо техник решения этих задач, инструменты, использующие DSE, могут различаться то применяемой инструментации кода SUT — на уровне исходного или бинарного кода, с помощью вставок в код иди за счет внешнего мониторинга выполнения определенных инструкций.

5.1 Сокращение количества и выбор анализируемых путей

Для сокращения множеств анализируемых путей и выбора очередного пути анализа применяются следующие техники.

- Выбор путей
  - Выбор путей для символического выполнения.
    В первых инструментах DSE (CUTE [100]) символически исполнялись все находимые пути. Далее были использованы разные стратегии выбора. KLEE [102,119] и SAGE [103,104] в первую очередь выполняют пути, позволяющие покрыть больше новых ветвей. S^2E [105] выбирает пути, наибольшим образом задействующие код целевых компонентов (несущественные компоненты и функции выполняются только конкретно). Driller [10] и QSYM [108] используют фаззер AFL [33] в качестве дополнительного источника для выбора путей, причем Driller использует символьное выполнение тогда, когда применение AFL уже не дает нового покрытия, QSYM использует символьное выполнение параллельно конкретному, в первую очередь для путей, обеспечивающих покрытие новых ветвлений. AEG [206] и Mayhem [106] в первую очередь обрабатывают пути, содержащие элементы кода с высокой вероятностью ошибок (циклы с обращением по указателям в них и пр.).
  - Выполнение сразу нескольких путей.
    KLEE, AEG, S^2E иногда пытаются выполнять новые пути с меньшими затратами за счет переключения на них с помощью fork или использования механизмов копирования при записи, чтобы снизить затраты на хранение состоянии анализируемой программы. Mayhem не пытается исполнять два пути одновременно, но при выполнении одного сохраняет контрольные точки для более легкого возвращения к новым путям.
- Упрощение и сокращение множества анализируемых путей
  - Сокращение путей с помощью использования условных выражений.
    Если решатель позволяет использовать выражения ite(c,x,y,), означающие, что если

выполнено условие c, нужно взять значение x, а иначе значение y, инструменты DSE активно пользуются этим. Условные выражения позволяют сокращать количество анализируемых путей и компактно представлять символически многие функции.

- Упрощение анализа с помощью анализа помеченных данных.
  Некоторые инструменты, например, LeanSym [207], используют динамический анализ помеченных данных, чтобы выделить только влияющую на проход по определенной ветви часть входных данных и использовать ограничения только на нее.
- Использование символических моделей (summary) для функций, циклов или отдельных участков кода.
  Некоторые техники DSE [208-210] позволяют во время анализа строить символические модели часто вызываемых функций и выполняемых циклов. [211,212] представляют аналогичные техники, позволяющие строить символические аннотации для отдельных ветвей. Эти аннотации и модели затем можно использовать, не выполняя повторный анализ этих элементов кода.
- Использование ограничений на входные данные.
  AEG [206] позволяет определять предусловия на входные данные, которые сужают множество анализируемых путей.
- Отождествление символических состояний.
  Для сокращения количества путей, подлежащих анализу, иногда используется отождествление состояний [213].

5.2 Моделирование работы с адресами

В инструментах DSE используются следующие техники моделирования работы с указателями, массивами и сложными структурами данных.

- В CUTE и SAGE использовались конкретные адреса, получаемые в текущем конкретном выполнении.
- Полное символическое моделирование памяти.
  Предлагалось еще в статье Кинга [202], поддерживалось во фреймворках BitBlaze [214,215] и BAP [216]. В KLEE и SAGE моделирование части массивов перекладывается на SMT решатели [97], в предположении, что они поддерживают соответствующие теории. Аналогично, S^2E [105] использует теорию массивов для моделирования участков памяти, на которые может ссылаться некоторый указатель. В Sydr [111,217] используется аппроксимация возможных адресов линейными выражениями, которые затем моделируются битовыми векторами.
- В Mayhem и angr [218] используется смешанное моделирование, для чтения адреса моделируются символически, а запись идет по конкретным адресам. В angr запись тоже может моделироваться символически, если используются небольшие буфера.

5.3 Моделирование работы с окружением

Для моделирования окружения анализируемой программы используются следующие техники.

- Пропуск функций.
  Некоторые функции (например, управления памятью, печати и пр.) можно полностью игнорировать при символическом анализе, поскольку они не оказывают влияния на его результаты.
- Моделирование семантики функций с помощью символических выражений.
  S^2E [105] и angr [218] используют символические модели для библиотечных функций, которые используются каждый раз, когда в коде эти функции вызываются. angr содержит

символические модели для довольно большого числа функций из стандартных библиотек libc и POSIX.

- Использование упрощенных реализаций или упрощенных символических моделей.
  В KLEE [102] и AEG [206] обращения к функциям стандартной библиотеки C заменяются на вызов встроенной упрощенной реализации, что упрощает их дальнейший анализ.
  S^2E [105] использует при анализе только некоторые пути из реализаций функций.
  SymCC [219] содержит для части функций стандартной библиотеки C упрощенные символические модели, которые добавляют возможные разветвления с их условиями в набор ограничений пути.
  Fuzzolic [220] может использовать три режима анализа функции: пропуск, анализ без инвертирования условий путей и полную символическую интерпретацию.

5.4 Дополнительная обработка ограничений

Следующие техники применяются для обработки ограничений перед передачей их решателю.

- Устранение избыточных ограничений с помощью выделения независимых подмножеств всего набора ограничений.
  Используется в EXE [119] и KLEE [102].
- Выбрасывание части ограничений для упрощения условия прохода по пути.
  QSYM [108] иногда выбрасывает из условий прохода по новому пути все ограничения, кроме последнего, полученного отрицанием последнего условия для только что пройденного пути. Часто это помогает решателю найти подходящие данные из-за простоты полученного ограничения (и несмотря на игнорирование сложной истории ранее пройденных условий).
  QSYM также может выкидывать из анализа некоторые блоки, которые часто исполняются на разных путях и порождают новые ограничения, усложняя тем самым условия прохождения других путей.
- Добавление ограничений, обеспечивающих проявление ошибок.
  Дополнительные ограничения представляют собой условия проявления ошибок, что при их успешном разрешении позволяет получить входные данные, демонстрирующие данную ошибку.
  KLEE добавляет ограничения, приводящие к делению на 0 в арифметических выражениях. IntScope [221] добавляет условия возникновения переполнения в операциях над целочисленными типами данных, Sydr [111] тоже может выполнять такие действия. Mayhem [106] может добавлять ограничения, приводящие к переполнению буферов или некорректной обработке форматных строк во входных данных.
  Некоторые фаззеры (SAVIOR [222], ParmeSan [223]) используют код, добавляемый санитайзерами, как источник условий, которые могут привести к возникновению ошибки.

6. Заключение

В данном документе проведен обзор техник и инструментов динамического анализа программ, нацеленных, в первую очередь, на проверку свойств безопасности и защищенности. Подробно рассмотрены методы фаззинга, верификационного мониторинга, отдельно представлены техники динамической символьной интерпретации, которые активно используются в рамках инструментов фаззинга. Методы и средства динамического анализа помеченных данных остались за рамками обзора из-за трудностей сбора технической информации о них. При рассмотрении фаззинга и динамической символьной интерпретации больше внимания уделено не отдельным инструментам, из-за их очень большого количества,

а техникам решения различных задач, возникающих при их работе. Также представлен обзор техник снижения эффективности фаззинга.

Отдельного упоминания заслуживает линейка инструментов динамического анализа, разрабатываемая и активно развиваемая на протяжении уже долгого времени в Институте системного программирования им. В. П. Иванникова РАН, покрывающая почти все виды динамического анализа [109-111,160,161,224-226].

Список литературы / References

[1]. M. Ozkan-Okay, R. Samet, Ö. Aslan, and D. Gupta. A Comprehensive Systematic Literature Review on Intrusion Detection Systems. IEEE Access, vol. 9, pp. 157727-157760, 2021, doi: 10.1109/ACCESS.2021.3129336

[2]. L. Santos, C. Rabadao, and R. Gonçalves. Intrusion Detection Systems in Internet of Things: A literature review. Proc. of 13-th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, Spain, 2018, pp. 1-7, doi: 10.23919/CISTI.2018.8399291

[3]. H. Zhu, P. A. V. Hall, and J. H. R. May. Software Unit Test Coverage and Adequacy. ACM Computing Surveys, 29(4):366-427, 1997. doi: 10.1145/267580.267590

[4]. M. Sutton, A. Greene, and P. Amini. Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley, 2007. ISBN: 9780321446114

[5]. J. Newsome and D. Song. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. Proc. of Network and Distributed System Security Simposium, 2005. doi: 10.1184/R1/6468716.v1

[6]. E. J. Schwartz, T. Avgerinos, and D. Brumley. All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask). Proc. of IEEE Symposium on Security and Privacy, pp. 317-331, 2010. doi: 10.1109/SP.2010.26

[7]. T. Wang, T. Wei, G. Gu, and W. Zou. TaintScope: a Checksum-aware Directed Fuzzing Tool for Automatic Software Vulnerability Detection. Proc. of IEEE Symposium on Security and Privacy, pp. 497-512, 2010. doi: 10.1109/SP.2010.37

[8]. B. P. Miller, L. Fredriksen, and B. So. An Empirical Study of the Reliability of UNIX Utilities. Communications of the ACM, 33(12):32-44, 1990. doi: 10.1145/96267.96279

[9]. The Cyber Grand Challenge. URL: https://blogs.grammatech.com/the-cyber-grand-challenge (доступ 13.06.2023)

[10]. N. Stephens, J. Grosen, C. Salls, A. Dutcher, R. Wang, J. Corbetta, Y. Shoshitaishvili, C. Krügel, and G. Vigna. Driller: Augmenting Fuzzing Through Selective Symbolic Execution. Proc. of Network and Distributed System Security Symposium. 2016. doi: 10.14722/NDSS.2016.23368

[11]. P. Goodman and A. Dinaburg. The Past, Present, and Future of Cyberdyne. IEEE Security & Privacy, 16(2):61-69, 2018. doi: 10.1109/MSP.2018.1870859

[12]. Cisco Secure Development Lifecycle. URL: https://www.cisco.com/c/en/us/about/trust-center/technology-built-in-security.html#~trustworthysolutionsfeatures (доступ 13.06.2023)

[13]. Chromium Security. URL: https://www.chromium.org/Home/chromium-security/bugs/ (доступ 13.06.2023)

[14]. Clusterfuzz. Chrome Fuzzing Infrastructure. URL: https://code.google.com/archive/p/clusterfuzz/ (доступ 13.06.2023)

[15]. M. Aizatsky, K. Serebryany, O. Chang, A. Arya, and M. Whittaker. Announcing OSS-Fuzz: Continuous fuzzing for open source software. Google Open Source Blog, 2016. URL: https://opensource.googleblog.com/2016/12/announcing-oss-fuzz-continuous-fuzzing.html (доступ 13.06.2023)

[16]. Microsoft Security Development Lifecycle. URL: https://www.microsoft.com/en-us/securityengineering/sdl/practices (доступ 13.06.2023)

[17]. E. Bounimova, P. Godefroid, and D. Molnar. Billions and billions of constraints: Whitebox fuzz testing in production. Proc. of 35-th International Conference on Software Engineering (ICSE), San Francisco, USA, 2013, pp. 122-131, doi: 10.1109/ICSE.2013.6606558

[18]. Fuzzing Survey URL: https://fuzzing-survey.org/ (доступ 15.06.2023)

[19]. N. Rathaus, G. Evron. Open Source Fuzzing Tools. Syngress, 2007. ISBN: 9781597491952

[20]. A. Takanen, J. D. DeMott, C. Miller, and A. Kettunen. Fuzzing for Software Security Testing and Quality Assurance. 2-nd ed. Artech House, 2018. ISBN: 9781608078509

[21]. J. Li, B. Zhao, and C. Zhang. Fuzzing: a Survey. Cybersecurity 1, 6, 2018. doi: 10.1186/s42400-018-0002-y

[22]. C. Chen, B. Cui, J. Ma, R. Wu, J. Guo, and W. Liu. A Systematic Review of Fuzzing Techniques. Computers & Security, 75:118-137, 2018. doi: 10.1016/j.cose.2018.02.002

[23]. V. J. M. Manes, H. Han, C. Han, S. K. Cha, M. Egele, E. J. Schwartz, and M. Woo. The Art, Science, and Engineering of Fuzzing: A Survey. IEEE Transactions on Software Engineering, 47(11):2312-2331, 2021. doi: 10.1109/TSE.2019.2946563. URL: http://arxiv.org/abs/1812.00140

[24]. H. Liang, X. Pei, X. Jia, W. Shen, and J. Zhang. Fuzzing: State of the Art. IEEE Transactions on Reliability, 67(3):1199-1218, 2018. doi: 10.1109/TR.2018.2834476

[25]. А. В. Вишняков. Поиск ошибок в бинарном коде методами динамической символьной интерпретации. Диссертация на соискание учёной степени к. ф.-м.н., ИСП РАН, Москва, 2022.

[26]. A. Fioraldi, D. C. Maier, D. Zhang, and D. Balzarotti. LibAFL: a Framework to Build Modular and Reusable Fuzzers. Proc of ACM SIGSAC Conference on Computer and Communication Security, pp. 1051-1065, 2022. doi: 10.1145/3548606.3560602

[27]. C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. ACM SIGPLAN Notices, 40(6):190-200, 2005. doi: 10.1145/1064978.1065034

[28]. F. Bellard. QEMU, a Fast and Portable Dynamic Translator. Proc. of ATEC'05, USENIX Annual Technical Conference, pp. 41-46, 2005. doi: 10.5555/1247360.1247401

[29]. Dyninst. URL: https://dyninst.org/dyninst (доступ 05.12.2023)

[30]. Dyninst GitHub. URL: https://github.com/dyninst/dyninst (доступ 05.12.2023)

[31]. D. L. Bruening. Efficient, Transparent, and Comprehensive Runtime Code Manipulation. Ph.D. thesis, Massachusetts Institute of Technology, 2004.

[32]. DynamoRIO. URL: https://github.com/DynamoRIO/dynamorio (доступ 05.12.2023)

[33]. M. Zalewski. American Fuzzy Lop. URL: https://github.com/mirrorer/afl (доступ 14.06.2023)

[34]. AFL, supported by Google. URL: https://github.com/google/AFL (доступ 19.06.2023)

[35]. D. Oleksiuk. IOCTL Fuzzer. URL: https://github.com/Cr4sh/ioctlfuzzer (доступ 14.06.2023)

[36]. J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. C. Lau, M. Sun, R. Yang, and K. Zhang. IoTFuzzer: Discovering Memory Corruptions in IoT through App-based Fuzzing. Proc. of the Network and Distributed System Security Symposium, 2018. doi:10.14722/ndss.2018.23159

[37]. D. Babić, S. Bucur, Y. Chen, F. Ivančić, T. King, M. Kusano, C. Lemieux, L. Szekeres, and W. Wang. FUDGE: Fuzz Driver Generation at Scale. Proc. of 27-th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 975-985, 2019. doi: 10.1145/3338906.3340456

[38]. K. K. Ispoglou, D. Austin, V. Mohan, and M. Payer. FuzzGen: Automatic Fuzzer Generation. Proc. of 29-th USENIX Security Symposium, pp. 2271-2287, 2020. doi: 10.5555/3489212.3489340

[39]. M. Zhang, J. Liu, F. Ma, H. Zhang, and Y. Jiang. IntelliGen: Automatic Driver Synthesis for Fuzz Testing. Proc. of IEEE/ACM 43-rd International Conference on Software Engineering: Software Engineering in Practice, pp. 318-327, 2021. doi: 10.1109/ICSE-SEIP52600.2021.00041. URL: https://arxiv.org/abs/2103.00862

[40]. GRR. URL: https://github.com/lifting-bits/grr (доступ 14.06.2023)

[41]. LibFuzzer – a Library for Coverage-guided Fuzz Testing. URL: https://llvm.org/docs/LibFuzzer.html (доступ 14.06.2023)

[42]. R. Swiecki and F. Gröbert. Honggfuzz. https://github.com/google/honggfuzz (доступ 16.06.2023)

[43]. K. Sen. Effective random testing of concurrent programs. Proc. of 22-th IEEE/ACM International Conference on Automated Software Engineering, pp. 323-332, 2007. doi: 10.1145/1321631.1321679

[44]. P. Joshi, C.-S. Park, K. Sen, and M. Naik. A Randomized Dynamic Program Analysis Technique for Detecting Real Deadlocks. ACM SIGPLAN Notices, 44(6):110-120, 2009. doi: 10.1145/1543135.1542489

[45]. Z. Lai, S. Cheung, and W. Chan. Detecting Atomic-set Serializability Violations in Multithreaded Programs through Active Randomized Testing. Proc. of 32-nd ACM/IEEE International Conference on Software Engineering, 1:235-244, 2010. doi: 10.1145/1806799.1806836

[46]. Y. Cai and W. K. Chan. MagicFuzzer: Scalable deadlock detection for large-scale applications. Proc. of 34-th International Conference on Software Engineering (ICSE), Zurich, Switzerland, pp. 606-616, 2012. doi: 10.1109/ICSE.2012.6227156

[47]. M. Samak, M. K. Ramanathan, and S. Jagannathan. Synthesizing racy tests. Proc. of 36-th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 175–185, 2015. doi: 10.1145/2737924.2737998

[48]. V. Ganesh, T. Leek, and M. Rinard. Taint-based Directed Whitebox Fuzzing. Proc. of 31-st International Conference on Software Engineering (ICSE'09), pp. 474-484, 2009. doi: 10.1109/ICSE.2009.5070546

[49]. I. Haller, A. Slowinska, M. Neugschwandtner, and H. Bos. Dowsing for Overflows: a Guided Fuzzer to Find Buffer Boundary Violations. Proc. of 22-nd USENIX Security Symposium, pp. 49-64, 2013. doi: 10.5555/2534766.2534772

[50]. L. Ma, C. Artho, C. Zhang, H. Sato, J. Gmeiner, and R. Ramler. GRT: Program-Analysis-Guided Random Testing. Proc. of 30-th IEEE/ACM International Conference on Automated Software Engineering, pp. 212-223, 2015. doi: 10.1109/ASE.2015.49

[51]. S. Rawat, V. Jain, A. Kumar, L. Cojocar, C. Giuffrida, and H. Bos. VUzzer: Application-aware Evolutionary Fuzzing. Proc. of Network and Distributed System Security Symposium, 2017. doi: 10.14722/NDSS.2017.23404

[52]. H. Peng, Y. Shoshitaishvili and M. Payer. T-Fuzz: Fuzzing by Program Transformation. Proc. of IEEE Symposium on Security and Privacy, pp. 697-710, 2018. doi: 10.1109/SP.2018.00056

[53]. Репозиторий FFmpeg. URL: http://samples.ffmpeg.org/ (доступ 16.06.2023)

[54]. CERT BFF. URL: https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=507974 (доступ 15.06.2023)

[55]. A. D. Householder and J. Foote. Probability-Based Parameter Selection for Black-Box Fuzz Testing. SEI Technical Note, CMU/SEI-2012-TN-019, 2012. doi:10.21236/ada610472

[56]. M. Woo, S. K. Cha, S. Gottlieb, and D. Brumley. Scheduling Black-box Mutational Fuzzing. Proc. of ACM SIGSAC Conference on Computer & Communications Security (CCS '13), pp. 511-522, 2013. doi: 10.1145/2508859.2516736

[57]. M. Böhme, V.-T. Pham, and A. Roychoudhury. Coverage-based Greybox Fuzzing as Markov Chain. Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS '16), pp. 1032-1043, 2016. doi: 10.1145/2976749.2978428

[58]. Syzkaller – kernel fuzzer. URL: https://github.com/google/syzkaller (доступ 15.06.2023)

[59]. D. Vyukov. go-fuzz. URL: https://github.com/dvyukov/go-fuzz (доступ 19.06.2023)

[60]. Y. Li, B. Chen, M. Chandramohan, S.-W. Lin, Y. Liu, and A. Tiu. Steelix: Program-State Based Binary Fuzzing. Proc. of 11-th Joint Meeting on Foundations of Software Engineering, pp. 627-637, 2017. doi: 10.1145/3106237.3106295

[61]. P. Chen and H. Chen. Angora: Efficient Fuzzing by Principled Search. Proc. of IEEE Symposium on Security and Privacy, pp. 711-725, 2018. doi: 10.1109/SP.2018.00046

[62]. M. Böhme, V.-T. Pham, M.-D. Nguyen, and A. Roychoudhury. Directed Greybox Fuzzing. Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS '17), pp. 2329-2344, 2017. doi: 10.1145/3133956.3134020

[63]. S. Wang, J. Nam, and L. Tan. QTEP: Quality-aware Test Case Prioritization. Proc. of 11-th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017), pp. 523-534, 2017. doi: 10.1145/3106237.3106258

[64]. M. Eddington. Peach Fuzzer. URL: https://peachtech.gitlab.io/peach-fuzzer-community/ (доступ 13.06.2023)

[65]. C. Aschermann, T. Frassetto, T. Holz, P. Jauernig, A. Sadeghi, and D. Teuchert. NAUTILUS: Fishing for Deep Bugs with Grammars. Proc. of Network and Distributed System Security Symposium, 2019. doi: 10.14722/ndss.2019.23412

[66]. S. Bradshaw. Fuzzer Automation with SPIKE. URL: https://resources.infosecinstitute.com/topic/fuzzer-automation-with-spike/ (доступ 13.06.2023)

[67]. SPIKE Protocol Fuzzer Creation Kit. URL: https://github.com/guilhermeferreira/spikepp (доступ 13.06.2023)

[68]. P. Amini, A. Portnoy, and R. Sears. Sulley. URL: https://github.com/OpenRCE/sulley (доступ 15.06.2023)

[69]. R. Kaksonen, M. Laakso, and A. Takanen. Software security assessment through specification mutations and fault injection. In: R. Steinmetz, J. Dittman, M. Steinebach (eds). Communications and Multimedia Security Issues of the New Century. IFIP — The International Federation for Information Processing, vol 64. Springer, pp. 173-183, 2001. doi: 10.1007/978-0-387-35413-2_16

[70]. G. Banks, M. Cova, V. Felmetsger, K. Almeroth, R. Kemmerer, and G. Vigna. SNOOZE: Toward a Stateful NetwOrk prOtocol fuzZEr. In: S. K. Katsikas, J. López, M. Backes, S. Gritzalis, and B. Preneel (eds). Information Security, ISC 2006. Lecture Notes in Computer Science, 4176, pp. 343-358. Springer, 2006. doi: 10.1007/11836810_25

[71]. H. J. Abdelnur, R. State, and O. Festor. KiF: a Stateful SIP Fuzzer. Principles, Systems and Applications of IP Telecommunications, 2007. doi: 10.1145/1326304.1326313

[72]. W. Johansson, M. Svensson, U. E. Larson, M. Almgren, and V. Gulisano. T-Fuzz: Model-Based Fuzzing for Robustness Testing of Telecommunication Protocols. Proc. of IEEE 7-th International Conference on Software Testing, Verification and Validation, pp. 323-332, 2014. doi: 10.1109/ICST.2014.45

[73]. Trinity: Linux System Call Fuzzer. URL: https://github.com/kernelslacker/trinity (доступ 13.06.2023)

[74]. KernelFuzzer. URL: https://github.com/FSecureLABS/KernelFuzzer (доступ 15.06.2023)

[75]. P. Godefroid, A. Kiezun, and M. Y. Levin. Grammar-based Whitebox Fuzzing. Proc. of 29-th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 206–215, 2008. doi: 10.1145/1375581.1375607

[76]. V.-T. Pham, M. Böhme, and A. Roychoudhury. Model-based Whitebox Fuzzing for Program Binaries. Proc. of 31-st IEEE/ACM International Conference on Automated Software Engineering, pp. 543-553, 2016. doi: 10.1145/2970276.2970316

[77]. S. Y. Kim, S. Lee, I. Yun, W. Xu, B. Lee, Y. Yun, and T. Kim. CAB-Fuzz: Practical Concolic Testing Techniques for COTS Operating Systems. Proc. of USENIX Annual Technical Conference, pp. 689-701, 2017. doi: 10.5555/3154690.3154755

[78]. DOMFuzz. URL: https://github.com/MozillaSecurity/domfuzz (доступ 16.06.2023)

[79]. Jzfunfuzz. URL: https://github.com/MozillaSecurity/funfuzz (доступ 16.06.2023)

[80]. C. Brubaker, S. Jana, B. Ray, S. Khurshid, V. Shmatikov. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations. Proc. of IEEE Symposium on Security and Privacy, pp. 114-129, 2014. doi: 10.1109/SP.2014.15

[81]. H. Kario. Tlfuzzer. URL: https://github.com/tlsfuzzer/tlsfuzzer (доступ 16.06.2023)

[82]. J. Somorovsky. Systematic Fuzzing and Testing of TLS Libraries. Proc. of ACM SIGSAC Conference on Computer and Communications Security, pp. 1492-1504, 2016. doi: 10.1145/2976749.2978411

[83]. J. Wang, B. Chen, L. Wei, and Y. Liu. Skyfire: Data-driven Seed Generation for Fuzzing. Proc. of the IEEE Symposium on Security and Privacy, pp. 579-594, 2017. doi: 10.1109/SP.2017.23

[84]. L. Della Toffola, C. A. Staicu, and M. Pradel. Saying 'hi!' is not Enough: Mining Inputs for Effective Test Generation. Proc. of 32-nd IEEE/ACM International Conference on Automated Software Engineering, pp. 44-49, 2017. doi: 10.5555/3155562.3155572

[85]. H. Han, D. Oh, and S. K. Cha. CodeAlchemist: Semantics-aware Code Generation to Find Vulnerabilities in Javascript Engines. Proc. of Network and Distributed System Security Symposium, 2019. doi: 10.14722/ndss.2019.23263

[86]. H. Han and S. K. Chaю IMF: Inferred Model-based Fuzzer. Proc. of ACM SIGSAC Conference on Computer and Communications Security, pp. 2345-2358, 2017. doi: 10.1145/3133956.3134103

[87]. P. Godefroid, H. Peleg, and R. Singh. Learn&Fuzz: Machine Learning for Input Fuzzing. Proc. of 32-nd IEEE/ACM International Conference on Automated Software Engineering, pp 50-59, 2017. doi: 10.48550/arXiv.1701.07232. URL: https://arxiv.org/abs/1701.07232

[88]. P. Liu, X. Zhang, M. Pistoia, Y. Zheng, M. Marques, and L. Zeng. Automatic Text Input Generation for Mobile Testing. Proc. of IEEE/ACM 39-th International Conference on Software Engineering (ICSE), pp. 643-653, 2017. doi: 10.1109/ICSE.2017.65

[89]. M. Höschele and A. Zeller. Mining Input Grammars from Dynamic Taints. Proc. of 31-st IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 720-725, 2016. doi: 10.1145/2970276.2970321

[90]. O. Bastani, R. Sharma, A. Aiken, and P. Liang. Synthesizing Program Input Grammars. Proc. of 38-th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 95-110, 2017. doi: 10.1145/3062341.3062349. URL: https://arxiv.org/abs/1608.01723

[91]. A. Doupé, L. Cavedon, C. Kruegel, and G. Vigna. Enemy of the State: a State-aware Black-box Web Vulnerability Scanner. Proc. of 21-st USENIX Security Symposium, pp. 523–538, 2012. doi: 10.5555/2362793.2362819

[92]. H. Gascon, C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck. PULSAR: Stateful Black-box Fuzzing of Proprietary Network Protocols. Proc. of International Conference on Security and Privacy in Communication Systems, pp. 330-347, 2015. doi: 10.1007/978-3-319-28865-9_18

[93]. A. Helin. Radamsa. URL: https://gitlab.com/akihe/radamsa (доступ 16.06.2023)

[94]. S. Hocevar. Zzuf. URL: https://github.com/samhocevar/zzuf (доступ 16.06.2023)

[95]. S. K. Cha, M. Woo, and D. Brumley. Program-Adaptive Mutational Fuzzing. Proc. of IEEE Symposium on Security and Privacy, pp. 725-741, 2015. doi: 10.1109/SP.2015.50

[96]. U. Kargén and N. Shahmehri. Turning Programs Against Each Other: High Coverage Fuzz Testing Using Binary-code Mutation and Dynamic Slicing. Proc. of 10-th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015), pp. 782-792, 2015. doi: 10.1145/2786805.2786844

[97]. L. D. Moura and N. Bjørner. Satisfiability Modulo Theories: Introduction and Applications. Communications of the ACM, 54(9): 69-77, 2011. doi: 10.1145/1995376.1995394

[98]. S. Gan, C. Zhang, X. Qin, X. Tu, K. Li, Z. Pei, and Z. Chen. CollAFL: Path Sensitive Fuzzing. Proc. of IEEE Symposium on Security and Privacy, pp. 679-69677, 2018. doi: 10.1109/SP.2018.00040

[99]. F. Rustamov, J. Kim, J. Yu, and J. Yun. Exploratory Review of Hybrid Fuzzing for Automated Vulnerability Detection. IEEE Access, 9:131166-131190, 2021. doi: 10.1109/ACCESS.2021.3114202

[100]. K. Sen, D. Marinov, and G. Agha. CUTE: a Concolic Unit Testing Engine for C. ACM SIGSOFT Software Engineering Notes, 30(5):263–72, 2005. doi: 10.1145/1095430.1081750

[101]. P. Godefroid, N. Klarlund, and K. Sen. DART: Directed Automated Random Testing. ACM SIGPLAN Notices, 40(6): 213-223, 2005. doi: 10.1145/1064978.1065036

[102]. C. Cadar, D. Dunbar, and D. Engler. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. Proc. of the 8-th USENIX conference on Operating System Design and Implementation, pp. 209–224, 2008. doi: 10.5555/1855741.1855756

[103]. P. Godefroid, M. Y. Levin, and D. A. Molnar. Automated Whitebox Fuzz Testing. Proc. of Network and Distributed System Security Symposium, pp. 151-166, 2008.

[104]. P. Godefroid, M. Y. Levin, and D. Molnar. SAGE: Whitebox Fuzzing for Security Testing. Communications of ACM, 55(3):40-44, 2012. doi: 10.1145/2093548.2093564

[105]. V. Chipounov, V. Kuznetsov, G. Candea. S2E: a Platform for In-Vivo Multi-path Analysis of Software Systems. ACM SIGARCH Computer Architecture News Notices, 46(3):265–278, 2011. doi: 10.1145/1961295.1950396

[106]. S. K. Cha, T. Avgerinos, A. Rebert, and D. Brumley. Unleashing Mayhem on Binary Code. Proc. of IEEE Symposium on Security and Privacy, pp. 380-394, 2012. doi: 10.1109/SP.2012.31

[107]. M. Neugschwandtner, P. M. Comparetti, I. Haller, and H. Bos. The BORG: Nanoprobing Binaries for Buffer Overreads. Proc. of 5-th ACM Conference on Data and Application Security and Privacy (CODASPY '15), pp. 87-97, 2015. doi: 10.1145/2699026.2699098

[108]. I. Yun, S. Lee, M. Xu, Y. Jang, and T. Kim. QSYM: a Practical Concolic Execution Engine Tailored for Hybrid Fuzzing. Proc. of 27-th USENIX Security Symposium, pp. 745-761, 2018. doi: 10.5555/3277203.3277260

[109]. S. Sargsyan, J. Hakobyan, M. Mehrabyan, M. Mishechkin, V. Akozin, and S. Kurmangaleev. ISP-Fuzzer: Extendable Fuzzing Framework. Proc. of 2019 Ivannikov Memorial Workshop (IVMEM), pp. 68-71, 2019. doi: 10.1109/IVMEM.2019.00017

[110]. М. В. Мишечкин, В. В. Акользин, Ш. Ф. Курмангалеев. Архитектура и функциональные возможности инструмента ИСП Фаззер. Открытая конференция ИСП РАН им. В.П. Иванникова, 2020.

[111]. A. Vishnyakov, A. Fedotov, D. Kuts, A. Novikov, D. Parygina, E. Kobrin, V. Logunova, P. Belecky, S. Kurmangaleev. Sydr: Cutting Edge Dynamic Symbolic Execution. Ivannikov ISPRAS Open Conference (ISPRAS), pp. 46-54, 2020. doi: 10.1109/ISPRAS51486.2020.00014

[112]. C. Aschermann, S. Schumilo, T. Blazytko, R. Gawlik, and T. Holz. REDQUEEN: Fuzzing with Input-to-state Correspondence. Proc. of Network and Distributed System Security Symposium, 2019. doi: 10.14722/ndss.2019.23371

[113]. G. Savidov, A. Fedotov. Casr-Cluster: Crash Clustering for Linux Applications. 2021 Ivannikov ISPRAS Open Conference (ISPRAS), pp. 47-51, 2021. doi: 10.1109/ISPRAS53967.2021.00012

[114]. CASR: Crash Analysis and Severity Report. URL: https://github.com/ispras/casr (доступ 05.12.2023)

[115]. D. Molnar, X. C. Li, and D. A. Wagner. Dynamic Test Generation to Find Integer Bugs in x86 Binary Linux Programs. Proc. of 18-th USENIX Security Symposium, pp. 67-82, 2009. doi: 10.5555/1855768.1855773

[116]. W. Cui, M. Peinado, S. K. Cha, Y. Fratantonio, and V. P. Kemerlis. RETracer: Triaging Crashes by Reverse Execution from Partial Memory Dumps. Proc. of 38-th International Conference on Software Engineering, pp. 820-831, 2016. doi: 10.1145/2884781.2884844

[117]. J. Regehr, Y. Chen, P. Cuoq, E. Eide, C. Ellison, and X. Yang. Test-case Reduction for C Compiler Bugs. Proc. of ACM SIGPLAN Notices, 47(6):335-346, 2012. doi: 10.1145/2345156.2254104

[118]. J. Foote. GDB exploitable plugin. URL: https://github.com/jfoote/exploitable (доступ 19.06.2023)

[119]. C. Cadar, V. Ganesh, P. M. Pawlowski, D. L. Dill, and D. Engler. EXE: Automatically Generating Inputs of Death. Proc. of 13-th ACM Conference on Computer and Communications Security, pp 322-335, 2006. doi: 10.1145/1180405.1180445

[120]. KLEE Symbolic Virtual Machine. URL: https://github.com/klee/klee

[121]. A. Fioraldi, D. Maier, H. Eißfeldt, and M. Heuse. AFL++: Combining Incremental Steps of Fuzzing Research. Proc. of 14-th USENIX Conference on Offensive Technologies (WOOT'20), article 10. USENIX Association, 2020. doi: 10.5555/3488877.3488887

[122]. AFL++. URL: https://github.com/AFLplusplus/AFLplusplus (доступ 05.12.2023)

[123]. S. Schumilo, C. Aschermann, R. Gawlik, S. Schinzel, and T. Holz. kAFL: Hardware-Assisted Feedback Fuzzing for OS Kernels. Proc. of 26-th USENIX Security Symposium, pp. 167-182, 2017. doi: 10.5555/3241189.3241204

[124]. Boofuzz. URL: https://github.com/jtpereyda/boofuzz (доступ 19.06.2023)

[125]. Defensics. URL: https://www.synopsys.com/software-integrity/security-testing/fuzz-testing.html (доступ 05.12.2023)

[126]. P. Tsankov, M. T. Dashti, and D. Basin. SecFuzz: Fuzz-Testing Security Protocols. Proc. of 7-th International Workshop on Automation of Software Test (AST), pp. 1-7, 2012. doi: 10.1109/IWAST.2012.6228985

[127]. T. L. Munea, H. Lim, and T. Shon. Network Protocol Fuzz Testing for Information Systems and Applications: a Survey and Taxonomy. Multimedia Tools and Applications, 75:14745-14757, 2016. doi: 10.1007/s11042-015-2763-6

[128]. X. Yang, Y. Chen, E. Eide, and J. Regehr. Finding and Understanding Bugs in C Compilers. ACM SIGPLAN Notices, 46(6):283-294, 2011. doi: 10.1145/1993316.1993532

[129]. Csmith. URL: https://github.com/csmith-project/csmith (доступ 20.06.2023)

[130]. C. Holler, K. Herzig, and A. Zeller. Fuzzing with Code Fragments. Proc. of 21-th USENIX Security Symposium, pp. 445-458, 2012. doi: 10.5555/2362793.2362831

[131]. H. Ma. A Survey of Modern Compiler Fuzzing. 2023. doi: 10.48550/arXiv.2306.06884. URL: https://arxiv.org/abs/2306.06884

[132]. A. Henderson, H. Yin, G. Jin, H. Han, and H. Deng. VDF: Targeted Evolutionary Fuzz Testing of Virtual Devices. In: M. Dacier, M. Bailey, M. Polychronakis, M. Antonakakis (eds). Research in Attacks, Intrusions, and Defenses (RAID 2017). LNCS, 10453:3-25, Springer, 2017. doi: 10.1007/978-3-319-66332-6_1

[133]. M. Eceiza, J. L. Flores and M. Iturbe. Fuzzing the Internet of Things: a Review on the Techniques and Challenges for Efficient Vulnerability Discovery in Embedded Systems. IEEE Internet of Things Journal, 8(13):10390-10411, 2021. doi: 10.1109/JIOT.2021.3056179

[134]. M. Eisele, M. Maugeri, R. Shriwas, C. Huth, and G. Bella. Embedded Fuzzing: a Review of Challenges, Tools, and Solutions. Cybersecurity, 5, article 18, 2022. doi: 10.1186/s42400-022-00123-y

[135]. J. Yun, F. Rustamov, J. Kim, and Y. Shin. Fuzzing of Embedded Systems: A Survey. ACM Computing Surveys, 55(7):1-33, article 137, 2023. doi: 10.1145/3538644

[136]. O. Whitehouse. Introduction to Anti-fuzzing: a Defence in Depth Aid. 2014. URL: http://research.nccgroup.com/2014/01/02/introduction-to-anti-fuzzing-a-defence-in-depth-aid (доступ 05.12.2023)

[137]. E. Edholm, D. Göransson. Escaping the Fuzz – Evaluating Fuzzing Techniques and Fooling Them with Anti-fuzzing. M.S. thesis, Chalmers University of Technology, 2016.

[138]. C. Collberg, C. Thomborson, and D. Low. Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs. Proc. of 25-th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 184-196, 1998. doi: 10.1145/268946.268962

[139]. P. Junod, J. Rinaldini, J. Wehrli and J. Michielin. Obfuscator-LLVM — Software Protection for the Masses. Proc. of 2015 IEEE/ACM 1-st International Workshop on Software Protection, pp. 3-9, 2015. doi: 10.1109/SPRO.2015.10

[140]. J. Zhang, Z. Li, Y. Liu, Z. Sun, and Z. Wang. SAFTE: a Self-injection Based Anti-fuzzing Technique. Computers and Electrical Enginerring, vol. 111, part B, 108980, 2023. doi: 10.1016/j.compeleceng.2023.108980

[141]. C. CC. Cheng, L. Lin, C. Shi, Y. Guan. An Anti-fuzzing Approach for Android Apps. In G. Peterson, S. Shenoi (eds), Digital Forensics 2023: Advances in Digital Forensics XIX, IFIP Advances in Information and communication Technology, Springer, vol. 687, pp. 37-53, 2023. doi: 10.1007/978-3-031-42991-0_3

[142]. Z. Zhou, C. Wang, and Q. Zhao. No-Fuzz: Efficient Anti-fuzzing Techniques. In: F. Li, K. Liang, Z. Lin, S. K. Katsikas. (eds). Security and Privacy in Communication Networks 2022. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 462, pp. 731-751. Springer, 2023. doi: 10.1007/978-3-031-25538-0_38

[143]. Z. Zhou, and C. Wang. Practical Anti-fuzzing Techniques with Performance Optimization. IEEE Open Journal of the Computer Society, vol. 4, pp. 206-217, 2023. doi: 10.1109/OJCS.2023.3301883

[144]. J. Jung, H. Hu, D. Solodukhin, D. Pagan, K. H. Lee, and T. Kim. FUZZIFICATION: Anti-fuzzing Techniques. Proc. of 28-th USENIX Conference on Security Symposium (SEC'19), pp. 1913–1930, 2019. doi: 10.5555/3361338.3361471

[145]. E. Güler, C. Aschermann, A. Abbasi, and T. Holz. ANTIFUZZ: Impeding Fuzzing Audits of Binary Executables. Proc. of 28-th USENIX Conference on Security Symposium (SEC'19), pp. 1931-1947, 2019. doi: 10.5555/3361338.3361472

[146]. ANTIFUZZ. URL: https://github.com/RUB-SysSec/antifuzz (доступ 05.12.2023)

[147]. Y. Li, G. Meng, J. Xu, C. Zhang, H. Chen, X. Xie, H. Wang, and Y. Liu. Vall-nut: Principled Anti-grey Box – Fuzzing. Proc. of IEEE 32-nd International Symposium on Software Reliability Engineering, pp. 288-299, 2021. doi: 10.1109/ISSRE52982.2021.00039

[148]. Z. Hu, Y. Hu, and B. Dolan-Gavitt. Chaff Bugs: Deterring Attackers by Making Software Buggier, 2018, arXiv:1808.0065. URL: https://arxiv.org/abs/1808.00659 (доступ 05.12.2023)

[149]. D. R. Kaprekar. On Kaprekar Numbers. Journal of Recreational Mathematics, 13(2):81-82, 1980.

[150]. E. Bartocci, Y. Falcone (eds). Lectures on Runtime Verification. Introductory and Advanced Topics. LNCS 10457, Springer, 2018. ISBN: 9783319756318

[151]. D. Drusinsky. The Temporal Rover and the ATG Rover. In: K. Havelund, J. Penix, W. Visser. (eds). SPIN Model Checking and Software Verification (SPIN 2000). LNCS 1885:323-330, 2000, Springer. doi: 10.1007/10722468_19

[152]. K. Havelund and G. Roşu. Java PathExplorer – A Runtime Verification Tool. Proc. of 6-th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'01), 2001.

[153]. M. Leucker and C. Schallhart. A Brief Account of Runtime Verification. Journal of Logic and Algebraic Programming, 78(5):293-303, 2009. doi: 10.1016/j.jlap.2008.08.004

[154]. Y. Falcone, S. Krstić, G. Reger, and D. Traytel. A Taxonomy for Classifying Runtime verification Tools. International Journal on Software Tools for Technology Transfer, 23:255-284, 2021. doi: 10.1007/s10009-021-00609-z

[155]. C. Sánchez, G. Schneider, W. Ahrendt, E. Bartocci, D. Bianculli, C. Colombo, Y. Falcone, A. Francalanza, S. Krstić, J. M. Lourenço, D. Nickovic, G. J. Pace, J. Rufino, J. Signoles, D. Traytel, and A. Weiss. A Survey of Challenges for Runtime Verification from Advanced Application Domains (beyond Software). Formal Methods in System Design, 54:279-335, 2019. doi: 10.1007/s10703-019-00337-w

[156]. A. R. Cavalli, T. Higashino, and M. Núñez. A Survey on Formal Active and Passive Testing with Applications to the Cloud. Annals of Telecommunications, 70:85-93, 2015. doi: 10.1007/s12243-015-0457-8

[157]. I. Itkin, R. Yavorskiy. Overview of Applications of Passive Testing Techniques. Modeling and Analysis of Complex Systems and Processes, 2019. URL: https://ceur-ws.org/Vol-2478/paper9.pdf (доступ 20.06.2023)

[158]. A. Edwards, T. Jaeger, and X. Zhang. Runtime Verification of Authorization Hook Placement for the Linux Security Modules Framework. Proc. of 9-th ACM Conference on Computer and Communications Security, pp. 225-234, 2002. doi: 10.1145/586110.586141

[159]. M. K. Sarrab. Policy-Based Runtime Verification of Information Flow. PhD Thesis, Software Technology Research Laboratory, De Monfort University, UK, 2011.

[160]. D. Efremov and I. Shchepetkov. Runtime Verification of Linux Kernel Security Module. Proc. of International Workshop on Formal Methods, LNCS 12233:185-199, Springer, 2020. doi: 10.1007/978-3-030-54997-8_12. URL: https://arxiv.org/pdf/2001.01442.pdf

[161]. Д. В. Ефремов, В. В. Копач, Е. В. Корныхин, В. В. Кулямин, А. К. Петренко, А. В. Хорошилов, И. В. Щепетков. Мониторинг и тестирование модулей операционных систем на основе абстрактных моделей поведения системы. Труды Института системного программирования РАН, 33(6):15-26б 2021. doi: 10.15514/ISPRAS-2021-33(6)-2

[162]. E. Bartocci, B. Bonakdarpour, and Y. Falcone. First International Competition on Runtime Verification. In: B. Bonakdarpour, S. A. Smolka (eds.). Runtime Verification 2014. LNCS 8734:1-9, Springer, 2014. doi: 10.1007/978-3-319-11164-3_1

[163]. Y. Falcone, D. Ničković, G. Reger, and D. Thoma. Second International Competition on Runtime Verification. In: E. Bartocci, R. Majumdar (eds). Runtime Verification 2015. LNCS 9333:405-422, Springer, 2015. doi: 10.1007/978-3-319-23820-3_27

[164]. G. Reger, S. Hallé, and Y. Falcone. Third International Competition on Runtime Verification. In: Y. Falcone, C. Sánchez (eds). Runtime Verification 2016. LNCS 10012:21-37, Springer, 2016. doi: 10.1007/978-3-319-46982-9_3

[165]. M. Delahaye, N. Kosmatov, and J. Signoles, Common Specification Language for Static and Dynamic Analysis of C Programs. Proc. of 28-th Annual ACM Symposium on Applied Computing, pp. 1230-1235, 2013. doi: 10.1145/2480362.2480593

[166]. E-ACSL. URL: https://frama-c.com/fc-plugins/e-acsl.html (доступ 21.06.2023)

[167]. Код E-ACSL. URL: https://github.com/evdenis/e-acsl (доступ 21.06.2023)

[168]. ANSI/ISO C Specification Language. https://frama-c.com/html/acsl.html (доступ 21.06.2023)

[169]. S. Navabpour, Y. Joshi, C. W. W. Wu, S. Berkovich, R. Medhat, B. Bonakdarpour, S. Fischmeister. RiTHM: a Tool for Enabling Time-Triggered Runtime Verification for C Programs. Proc. of 9-th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013), pp. 603-606, 2013. doi: 10.1145/2491411.2494596

[170]. R. Medhat, Y. Joshi, B. Bonakdarpour, and S. Fischmeister. Accelerated Runtime Verification of LTL Specifications with Counting Semantics. In Y. Falcone, C. Sánchez (eds). Runtime Verification 2016, LNCS 10012:251-267, Springer, 2016. doi: 10.1007/978-3-319-46982-9_16. URL: https://arxiv.org/abs/1411.2239

[171]. C. Colombo, G. J. Pace, and G. Schneider. LARVA — Safer Monitoring of Real-Time Java Programs. Proc. of 7-th IEEE International Conference on Software Engineering and Formal Methods, pp. 33-37, 2009. doi: 10.1109/SEFM.2009.13

[172]. LARVA. URL: http://www.cs.um.edu.mt/~svrg/Tools/LARVA/ (доступ 21.06.2023)

[173]. Код LARVA. URL: https://github.com/ccol002/larva-rv-tool (доступ 21.06.2023)

[174]. C. Colombo, G. J. Pace, and G. Schneider. Dynamic Event-Based Runtime Monitoring of Real-Time and Contextual Properties. Proc. of Formal Methods for Industrial Critical Systems (FMICS 2008), LNCS 5596:135-149, Springer, 2008. doi: 10.1007/978-3-642-03240-0_13

[175]. Q. Luo, Y. Zhang, C. Lee, D. Jin, P. O'Neil Meredith, T.-F. Serbanuta, and G. Roşu. RV-Monitor: Efficient Parametric Runtime Verification with Simultaneous Properties. In: B. Bonakdarpour and A. Smolka (eds). Runtime Verification 2014, LNCS 8734:285-300, Springer, 2014. doi: 10.1007/978-3-319-11164-3_24

[176]. Код RV-Monitor. URL: https://github.com/runtimeverification/rv-monitor (доступ 21.06.2023)

[177]. Y. Falcone, P. Meredith, T. F. Şerbănuţă, S. Shiriashi, A. Iwai, and G. Roşu. RV-Android: Efficient Parametric Android Runtime Verification, a Brief Tutorial. In: E. Bartocci, R. Majumdar (eds). Runtime Verification 2015. LNCS 9333:342-357, Springer, 2015. doi: 10.1007/978-3-319-23820-3_24

[178]. G. Reger, H. C. Cruz, and D. E. Rydeheard. MarQ: Monitoring at Runtime with QEA. Proc. of 21-st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015), LNCS 9035:596-610, Sringer, 2015. doi: 10.1007/978-3-662-46681-0_55

[179]. N. Decker, J. Harder, T. Scheffel, M. Schmitz, and D. Thoma. Runtime Monitoring with Union-Find Structures. Proc. of 22-nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2016), LNCS 9636:868-884, Springer, 2016. doi: 10.1007/978-3-662-49674-9_54

[180]. Mufin Project. URL: https://www.isp.uni-luebeck.de/mufin (доступ 21.06.2023)

[181]. K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov. AddressSanitizer: a Fast Address Sanity Checker. Proc. of USENIX Annual Technical Conference, pp. 309-318, 2012. doi: 10.5555/2342821.2342849

[182]. AddressSanitizer. URL: https://github.com/google/sanitizers/wiki/AddressSanitizer (доступ 22.06.2023)

[183]. QASan (QEMU-AddressSanitizer). URL: https://github.com/andreafioraldi/qasan (доступ 22.06.2023)

[184]. W. Han, B. Joe, B. Lee, C. Song, and I. Shin. Enhancing Memory Error Detection for Large-Scale Applications and Fuzz Testing. Proc. of Network and Distributed System Security Symposium, 2018. doi: 10.14722/ndss.2018.23318.

[185]. S. Nagarakatte, J. Zhao, M. M. K. Martin, and S. Zdancewic. SoftBound: Highly Compatible and Complete Spatial Memory Safety for C. ACM SIGPLAN Notices, 44(6):245-258, 2009. doi: 10.1145/1543135.1542504

[186]. S. Nagarakatte, J. Zhao, M. M. K. Martin, and S. Zdancewic. CETS: Compiler Enforced Temporal Safety for C. ACM SIGPLAN Notices, 45(8):31-40, 2010. doi: 10.1145/1837855.1806657

[187]. B. Lee, C. Song, T. Kim, and W. Lee. Type Casting Verification: Stopping an Emerging Attack Vector. Proc. of 24-th USENIX Security Symposium, pp. 81-96, 2015. doi: 10.5555/2831143.2831149

[188]. I. Haller, Y. Jeon, H. Peng, M. Payer, C. Giuffrida, H. Bos, and E. van der Kouwe. TypeSan: Practical Type Confusion Detection. Proc. of ACM SIGSAC Conference on Computer and Communications Security, pp. 517-528, 2016. doi: 10.1145/2976749.2978405

[189]. Y. Jeon, P. Biswas, S. Carr, B. Lee, and M. Payer. HexType: Efficient Detection of Type Confusion Errors for C++. Proc. of ACM SIGSAC Conference on Computer and Communications Security, pp. 2373-2387, 2017. doi: 10.1145/3133956.3134062

[190]. X. Wang, N. Zeldovich, M. F. Kaashoek, and A. Solar-Lezama. Towards Optimization-Safe Systems: Analyzing the Impact of Undefined Behavior. Proc. of 24-th ACM Symposium on Operating System Principles, pp. 260-275, 2013. doi: 10.1145/2517349.2522728

[191]. Valgrind. URL: https://valgrind.org/ (доступ 21.06.2023)

[192]. J. Seward and N. Nethercote. Using Valgrind to Detect Undefined Value Errors with Bit-Precision. Proc. of USENIX Annual Technical Conference, pp. 2, 2005. doi: 10.5555/1247360.1247362

[193]. D. Bruening and Q. Zhao. Practical Memory Checking with Dr. Memory. Proc. of International Symposium on Code Generation and Optimization, pp. 213-223, 2011. doi: 10.1109/CGO.2011.5764689

[194]. E. Stepanov and K. Serebryany. MemorySanitizer: Fast Detector of Uninitialized Memory Use in C++. Proc. of IEEE/ACM International Symposium on Code Generation and Optimization, pp. 46-55, 2015. doi: 10.1109/CGO.2015.7054186

[195]. MemorySanitizer in LLVM/Clang. URL: https://clang.llvm.org/docs/MemorySanitizer.html (доступ 22.06.2023)

[196]. W. Dietz, P. Li, J. Regehr, and V. Adve. Understanding Integer Overflow in C/C++. ACM Transactions on Software Engineering and Methodology, 25(1):1-29, 2015. doi: 10.1145/2743019

[197]. UndefinedBehaviorSanitizer in LLVM/Clang. URL: https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html (доступ 22.06.2023)

[198]. K. Serebryany and T. Iskhodzhanov. ThreadSanitizer: Data Race Detection in Practice. Proc. of Workshop on Binary Instrumentation and Applications, pp. 62-71, 2009. doi: 10.1145/1791194.1791203

[199]. ThreadSanitizer in LLVM/Clang. URL: https://clang.llvm.org/docs/ThreadSanitizer.html (доступ 22.06.2023)

[200]. R. S. Boyer, B. Elspas, and K. N. Levitt. SELECT — a Formal System for Testing and Debugging Programs by Symbolic Execution. ACM SIGPLAN Notices, 10(6):234-245, 1975. doi: 10.1145/390016.808445

[201]. W. E. Howden. Methodology for the Generation of Program Test Data. IEEE Transactions on Computers, C-24(5):554-560, 1975. doi: 10.1109/T-C.1975.224259
[202]. J. C. King. A New Approach to Program Testing. Proc. of International Conference on Reliable Software, pp. 228-233, 1975. doi: 10.1145/800027.808444
[203]. J. C. King. Symbolic Execution and Program Testing. Communications of the ACM, 19(7):385-394, 1976. doi: 10.1145/360248.360252
[204]. C. Cadar and K. Sen. Symbolic Execution for Software Testing: Three Decades Later. Communications of ACM, 56(2):82-90, 2013. doi: 10.1145/2408776.2408795
[205]. R. Baldoni, E. Coppa, D. Cono D'Elia, C. Demetrescu, and I. Finocchi. A Survey of Symbolic Execution Techniques. ACM Computing Surveys. 51:3(1-39), art. 50, 2018. doi: 10.1145/3182657. URL: https://arxiv.org/abs/1610.00502
[206]. T. Avgerinos, S. K. Cha, B.T.H. Lim, and D. Brumley. AEG: Automatic Exploit Generation. Proc. of Network and Distributed System Security Symposium, pp. 283-300, 2011.
[207]. X. Mi, S. Rawat, C. Giuffrida, and H. Bos. LeanSym: Efficient Hybrid Fuzzing Through Conservative Constraint Debloating. Proc. of 24-th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '21), pp. 62-77, 2012. doi: 10.1145/3471621.3471852
[208]. P. Godefroid. Compositional Dynamic Test Generation. ACM SIGPLAN Notices, 42(1):47-54, 2007. doi: 10.1145/1190215.1190226
[209]. P. Godefroid and D. Luchaup. Automatic Partial Loop Summarization in Dynamic Test Generation. Proc. of International Symposium on Software Testing and Analysis (ISSTA'11), pp. 23-33, 2011. doi: 10.1145/2001420.2001424
[210]. X. Xie, B. Chen, Y. Liu, W. Le, and X. Li. Proteus: Computing Disjunctive Loop Summary via Path Dependency Analysis. Proc. of 24-th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'16), pp. 61-72, 2016. doi: 10.1145/2950290.2950340
[211]. K. L. McMillan. Lazy Annotation for Program Testing and Verification. Proc. of 22-nd International Conference on Computer Aided Verification (CAV'10), LNCS 6174:104-118, 2010. doi: 10.1007/978-3-642-14295-6_10
[212]. Q. Yi, Z. Yang, S. Guo, C. Wang, J. Liu, and C. Zhao. Postconditioned Symbolic Execution. Proc. of IEEE 8-th International Conference on Software Testing, Verification and Validation (ICST), pp. 1-10, 2015. doi: 10.1109/ICST.2015.7102601
[213]. V. Kuznetsov, J. Kinder, S. Bucur, and G. Candea. Efficient State Merging in Symbolic Execution ACM SIGPLAN Notices, 47(6):193-204, 2012. doi: 10.1145/2345156.2254088
[214]. D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena. BitBlaze: a New Approach to Computer Security via Binary Analysis. Proc. of 4-th International Conference on Information Systems Security ((ICISS'08), LNCS 5352:1-25, 2008. doi: 10.1007/978-3-540-89862-7_1
[215]. BitBlaze: Binary Analysis for Computer Security. URL: http://bitblaze.cs.berkeley.edu/ (доступ 27.06.2023)
[216]. D. Brumley, I. Jager, T. Avgerinos, and E. J. Schwartz. BAP: A Binary Analysis Platform. Proc. of 23-rd International Conference on Computer Aided Verification (CAV'11), LNCS 6806:463-469, 2011. doi: 10.1007/978-3-642-22110-1_37
[217]. D. Kus. Towards Symbolic Pointers Reasoning in Dynamic Symbolic Execution. arXiv 2109.03698, 2022. URL: https://arxiv.org/abs/2109.03698 (доступ 05.12.2023)
[218]. Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel, and G. Vigna. SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis. Proc. of IEEE Symposium on Security and Privacy, pp. 138-157, 2016. doi: 10.1109/SP.2016.17
[219]. S. Poeplau and A. Francillon. Symbolic Execution with SymCC: Don't Interpret, Compile! Proc. of 29-th USENIX Security Symposium, pp. 181-198, 2020. doi: 10.5555/3489212.3489223
[220]. L. Borzacchiello, E. Coppa, C. Demetrescu. FUZZOLIC: Mixing Fuzzing and Concolic Execution. Computers and Security, 108(C), art 102368, 2021. doi: 10.1016/j.cose.2021.102368
[221]. T. Wang, T. Wei, Z. Lin, and W. Zhou. IntScope: Automatically Detecting Integer Overflow Vulnerability in x86 Binary using Symbolic Execution. Proc of Network and Distributed System Security Simposium, 2009.

[222]. Y. Chen, P. Li, J. Xu, S. Guo, R. Zhou, Y. Zhang, T. Wei, and L. Lu. SAVIOR: Towards Bug-Driven Hybrid Testing. Proc. of IEEE Symposium on Security and Privacy, pp. 1580-1596, 2020. doi: 10.1109/SP40000.2020.00002. URL: https://arxiv.org/abs/1906.07327

[223]. S. Österlund, K. Razavi, H. Bos, and C. Giuffrida. ParmeSan: Sanitizer-Guided Greybox Fuzzing. Proc. of 29-th USENIX Conference on Security (SEC'20), article 129, pp. 2289-2306. doi: 10.5555/3489212.3489341

[224]. П.М. Довгалюк, М.А. Климушенкова, Н.И. Фурсова, В.М. Степанов, И.А. Васильев, А.А. Иванов, А.В. Иванов, М.Г. Бакулин, Д.И. Егоров. Natch: Определение поверхности атаки программ с помощью отслеживания помеченных данных и интроспекции виртуальных машин. Труды Института системного программирования РАН, 34(5):89-110, 2022. doi: 10.15514/ISPRAS-2022-34(5)-6

[225]. I. K. Isaev, D. V. Sidorov. The Use of Dynamic Analysis for Generation of Input Data that Demonstrates Critical Bugs and Vulnerabilities in Programs. Programming and Computer Software, 36(40):225-236, 2010. doi: 10.1134/S0361768810040055

[226]. М.К. Ермаков, А.Ю. Герасимов. Avalanche: применение параллельного и распределенного динамического анализа программ для ускорения поиска дефектов и уязвимостей. Труды Института системного программирования РАН, 25:29-38, 2013.

Информация об авторах / Information about authors

Виктор Вячеславович КУЛЯМИН – кандидат физико-математических наук, доцент кафедры Системного программирования ВМК МГУ, ведущий научный сотрудник ИСП РАН. Сфера научных интересов: программная инженерия, тестирование на основе моделей, формальные методы программной инженерии.

Victor Viatcheslavovitch KULIAMIN – PhD, Associate Professor of System Programming Department, Faculty of Computational Mathematics and Cybernetics Moscow State University, Leading Researcher of the Institute for System Programming, Russian Academy of Sciences. Research interests: software engineering, model based testing, formal methods of software engineering.

DOI: 10.15514/ISPRAS-2023-35(4)-2

Вызовы в реализации систем глубокого анализа сетевого трафика методом полного протокольного декодирования

[1] *Р.Е. Пономаренко, ORCID: 0009-0009-5741-3627 <rerandom@ispras.ru>*
[1] *В.И. Егоров, ORCID: 0009-0001-5168-6474 <unclehook@ispras.ru>*
[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

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

Аннотация. В данной статье описываются проблемы, возникающие при реализации инструментов глубокого анализа сетевого трафика методом полного протокольного декодирования. Описываемые проблемы условно делятся на две группы. Первая группа проблем связана с основополагающими задачами, которые необходимо решить при реализации систем полного протокольного декодирования. В частности, важно обеспечить корректный разбор протоколов, что включает в себя правильное определение и интерпретацию заголовков и полей протоколов. Также требуется обеспечить обработку фрагментированных пакетов и сборку фрагментов в исходное сообщение. Важной задачей является также обработка и анализ зашифрованного трафика, что может потребовать использования специальных алгоритмов и инструментов. Вторая группа проблем связана с оптимизацией процесса полного протокольного декодирования для обеспечения высокой скорости обработки трафика, а также с поддержкой новых протоколов и возможностью добавления пользовательских расширений. Существуют системы с открытым исходным кодом, которые в некоторой мере решают базовые проблемы, связанные с полным протокольным декодированием. Однако, для эффективной работы и расширения функционала таких систем могут потребоваться дополнительные усилия и разработка специализированных решений.

Ключевые слова: глубокий анализ сетевого трафика; декодирование протоколов; параллельная обработка; управление памятью.

Для цитирования: Пономаренко Р.Е., Егоров В.И., Гетьман А.И. Вызовы в реализации систем глубокого анализа сетевого трафика методом полного протокольного декодирования. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 45–64. DOI: 10.15514/ISPRAS–2023–35(4)–2.

Challenges in the implementation of systems for deep packet inspection by the method of full protocol decoding

[1] *R.E. Ponomarenko ORCID: 0009-0009-5741-3627 <rerandom@ispras.ru>*
[1] *V.I. Egorov ORCID: 0009-0001-5168-6474 <unclehook@ispras.ru>*
[1,2,3,4] *A.I. Getman ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

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

Abstract. This paper presents a summary of experience in developing the deep packet inspection system using full protocol decoding. The paper reviews the challenges encountered during implementation and provides a high-level overview of the solutions to these issues. The challenges can be grouped into two groups. The first group is related to the fundamental tasks which must be addressed when implementing full protocol decoding systems. This includes ensuring correct protocol parsing, which involves identifying and interpreting protocol headers and fields correctly. Moreover, it is necessary to ensure the processing of fragmented packets and the assembly of fragments into the original message. Additionally, the processing and analysis of encrypted traffic is a crucial task that may require the use of specialized algorithms and tools. The second group of problems is related to optimizing the process of full protocol decoding to ensure high-speed traffic processing, as well as supporting new protocols and the ability to add user-defined extensions. While there are open-source systems that address some of the primary issues associated with full protocol decoding, there may be a need for additional effort and specialized solutions to efficiently operate and expand the functionality of such systems. Although implementing deep network traffic analysis tools using full protocol decoding requires the use of advanced hardware and software technologies, the benefits of such analysis are significant. This approach provides a more complete understanding of network traffic patterns and enables more effective detection and prevention of cyber-attacks. It also allows for more accurate monitoring of network performance and the identification of potential bottlenecks or other issues that may impact network efficiency. In this article, we also emphasize the importance of system architecture development and implementation to ensure the successful deployment of deep network traffic analysis tools using full protocol decoding. At last, we conducted an experiment where several advanced optimizations were implemented in the system that had already solved primary issues. These optimizations related to working with memory, based on the features of the traffic processing scheme. By results, we evaluated significant performance improvement in solving secondary tasks, described in this work.

Keywords: deep packet inspection; decoding protocols; parallel processing; memory management.

For citation: Ponomarenko R.E., Egorov V.I., Getman A.I. Challenges in the implementation of systems for deep packet inspection by the method of full protocol decoding. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 4, 2023. pp. 45-64 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-2.

1. Введение

В сфере информационных технологий постоянно возрастает потребность в создании высокопроизводительных систем для решения сложных задач. Не всегда это напрямую связано с увеличением скорости работы. Под таким углом можно взглянуть на развитие систем передачи данных в сети интернет. Физических параметров у таких систем не много – скорость передачи и задержки при передаче. Улучшение этих параметров влечёт за собой большие расходы на обновление оборудования, замену линий передачи и тому подобное. Но влиять на субъективное восприятие эффективности работы таких систем (QoE) [1] можно при помощи других параметров, в частности, за счёт приоритизации потоков данных. При помощи

этого механизма трафик, чувствительный к скорости или задержкам работы систем передачи, обрабатывается в первую очередь, и у пользователя возникает ощущение работы с более производительными системами, хотя выигрыш был достигнут за счёт ухудшения параметров для трафика, который оказался менее приоритетным, исходя из предположения, что пользователь этого не заметит.

Другой важной и актуальной темой в информационных технологиях остаются атаки на компьютерные сети. Целей для атак (устройств и систем) становится больше, число угроз растёт, вредоносное программное обеспечение также не стоит на месте, однако успешные атаки не становятся обыденностью для большинства пользователей. Для того, чтобы оградить пользователей от деструктивного воздействия со стороны сети без вмешательства в его работу, необходимо распознавать такие атаки до того, как они достигнут пользователя, то есть на стороне поставщика услуг доступа в интернет.

В обоих частных случаях, описанных выше может применяется технология глубокого анализа сетевого трафика (DPI).

Многие из всемирно известных разработчиков систем глубокого анализа сетевого трафика, такие как Enea, Allot и Sandvine [2], прекратили продажи своих разработок в России. Благодаря накопленной экспертизе в разработке таких систем, программным обеспечением упомянутых компаний пользуются по всему миру и альтернативные решения возникают достаточно редко. В качестве российских компаний среди таковых можно упомянуть VAS Experts и РДП.РУ.

Глубокий анализ сетевого трафика является базовым механизмом, необходимым для решения множества задач, в частности, задач обеспечения информационной безопасности, что уже упоминалось выше. Поэтому в этом году российские компании, такие как Positive Technologies, Ростелеком-Солар и ВТБ, заявили о разработке систем глубокого анализа сетевого трафика в составе комплексных средств обеспечения информационной безопасности, преимущественно межсетевых экранов следующего поколения (NGFW) [3-5].

Приведённые выше факты показывают актуальность задач проработки архитектуры систем глубокого анализа сетевого трафика.

Разработка таких систем – нетривиальный процесс, сопряженный с большим количеством трудностей. Не разработана единая архитектура программного обеспечения, позволяющая проектировать одинаково эффективные системы глубокого анализа трафика для различных целей. В последующих разделах этой работы произведён обзор трудностей, связанных с реализацией системы глубокого анализа трафика методом полного протокольного декодирования, который позволяет получить максимальное количество информации.

1.1 Применение анализа сетевого трафика

Анализ сетевого трафика можно поделить по степени “глубины” анализа: поверхностный, средний и глубокий (SPI, MPI и DPI) [6]. К глубокому относится анализ полезной нагрузки уровня приложений сетевой модели OSI. Такой тип анализа даёт самые широкие возможности, однако и является самым трудоёмким. Цели применения этой технологии могут быть самыми разными.

Можно выделить две схемы работы в зависимости от входных данных: *онлайн-анализ*, то есть анализ трафика, поступающего на вход сетевого интерфейса и *офлайн-анализ*, то есть анализ предварительно записанного трафика [7]. В зависимости от цели применения DPI можно использовать ту или иную схему.

Онлайн-анализ трафика:

- Классификация пользователей и приложений:
  - Приоритизация или тарификация

Для справедливого распределения канала между абонентами при максимальной утилизации канала, провайдерами может использоваться DPI. Применяя эту технологию, можно выделять чувствительные к задержкам приложения, и отдавать приоритет в обработке трафику таких приложений.

Также, при выделении категорий приложений, возможна отдельная тарификация таких категорий. Например, безлимитный трафик для мессенджеров.

- Управление доступом групп пользователей к группам приложений (и функциям в этих приложениях)

  Основной функционал, который предоставляют межсетевые экраны актуального поколения (NGFW). При помощи DPI решаются такие задачи как: выделение групп пользователей по трафику, выделение приложений, а также отдельных функций приложений.

- Выделение полезной нагрузки седьмого уровня модели OSI:
  - Предотвращение утечек информации (DLP)

    Чувствительные к утечкам данные располагаются в полезной нагрузке протоколов прикладного уровня, для их выделения так же используются DPI.
- Выявление особых признаков:
  - Анализ и выявление сетевых атак

    Для выделения атак на приложения, а также целевых, ранее неизвестных атак (APT) системами обнаружения и предотвращения (NTA/NDR) [8] также применяется DPI, для выделения признаков.
  - Балансировка нагрузки

    Не всегда балансировки потоков данных (L4) приложения достаточно для распределения нагрузки по узлам. В случаях, когда отдельный клиент может генерировать большие объёмы информации для обработки, балансировка по потокам направит все запросы такого клиента на один узел. Для более эффективного решения такой задачи применяются L7 балансировщики. В них уровень приложений дополнительно разбивается на подуровни и по ним также возможна балансировка [9].

    В качестве другого примера, масштабируемое приложение, для которого нужна балансировка потока данных, чувствительно к балансировке, иными словами, промах при балансировке приводит к серьёзным последствиям, то для балансировки имеет смысл использовать глубокий анализ, что позволит точнее выделять сессии. Кроме того, это позволяет корректно обрабатывать трафик большого количества протоколов, например, это актуально для балансировки трафика агрегированных каналов.

Анализ записанного трафика:

- Отладка

  Отладка программ, генерирующих и обрабатывающих сетевой трафик, отладка средств обеспечения информационной безопасности, исследование произошедших инцидентов.

- Сетевая криминалистика

  Применяется для сбора доказательств при расследовании инцидентов. Инциденты не обязательно связаны с информационной безопасностью. Но по механизму работы ближе всего к анализу сетевых атак, за тем исключением, что время на анализ не ограничено в общем случае.

1.2 Классификация систем глубокого анализа сетевого трафика

В [10] выделяют 4 метода реализации функций глубокого анализа трафика:

- Классификация пакетов по портам L4.
- Поиск совпадения шаблонов (в основном, на основе регулярных выражений).
- Техники, основанные на машинном обучении.
- Декодирование протокола.

Наиболее полную информацию даёт анализ результатов полного протокольного декодирования. Полное декодирование каждого пакета, в общем случае, осуществляется получателем. Максимальное количество данных, которые получатель может принять определяется шириной канала и вычислительными возможностями входного тракта получателя. Именно получателя, а не отправителя, потому что обработка при отправке и получении может отличаться.

Для полного протокольного декодирования в сетевом исполнении и последующего анализа на скорости канала связи необходимы вычислительные мощности, превосходящие суммарную мощность входных трактов всех получателей, обрабатывающих исследуемый трафик, что в общем случае недостижимо.

Поэтому при решении различных задач происходит уменьшение числа анализируемых данных, вкупе с разными методами ускорения, до достижения приемлемой скорости обработки. С ростом возможностей вычислительной техники усложнялись и методы ускорения обработки. От чтения данных по фиксированному смещению (вычислительно самый дешёвый вариант разбора, к нему относится классификация пакетов по портам L4) к обработке потока байт регулярными выражениями (например, в случае поиска признаков атак в текстовых протоколах, таких как HTTP). Но всегда оставался класс задач, которые решались только протокольным декодированием. Это задачи, в которых важна высокая точность распознавания протоколов или их признаков, с небольшим количеством ложных срабатываний [10].

Кроме того, с ростом возможностей вычислительной техники проводить более “глубокий” анализ стало проще, что позволило как решать задачи в большем числе случаев, так и решать всё более сложные задачи. Примерами первого может быть избавление от ограничений на протоколы низкого уровня при анализе высокого, то есть анализ любого стека протоколов или любой вложенности. Если средство анализа не поддерживает какой-то из протоколов низкого уровня, использующийся для передачи или инкапсуляции (GRE, например), то анализ высокого уровня становится невозможен. Примерами второго – классификация трафика внутри отдельного приложения, например, выделение отдельных сервисов предоставляемыми социальными сетями (чаты, видео, музыка и тому подобное).

Все это приводит к тому, что задача анализа трафика методом полного протокольного декодирования, а также ускорение этого анализа, остаётся актуальной.

2. Базовые проблемы реализации протокольного декодирования и подходы к их решению

В данном разделе описываются проблемы, являющиеся основополагающими при реализации систем глубокого анализа трафика методом полного протокольного декодирования. Обобщённое представление процесса разбора сетевого трафика представлено на рис. 1.

2.1 Захват трафика с интерфейса

Прежде чем анализировать трафик, его необходимо получить с сетевого интерфейса. С ростом скоростей передачи данных делать это средствами, предоставляемыми ОС становится всё сложнее.

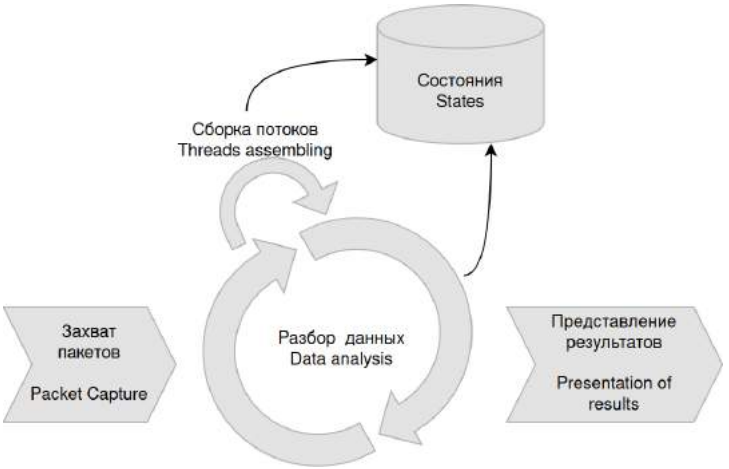

Рис. 1. Обобщённое представление процесса разбора сетевого трафика
Fig. 1. Generalized representation of the process of network traffic analysis

Можно выделить несколько проблем, которые влияют на скорость захвата трафика:

- Переключение контекста

  Взаимодействие с сетевым интерфейсом производится драйвером ОС, который, как правило, работает в привилегированном режиме. В то время как логика приложения, анализирующего трафик, выполняется в непривилегированном режиме. Переключение между этими режимами – очень затратная операция. Для сокращения количества таких переключений применяется отказ от попакетной обработки, тогда переключение будет происходить не на каждый пакет. Полный перенос обработки трафика в ядро, как и вынесение функций драйвера за пределы ядра, позволяет полностью отказаться от таких переключений [11].

- Копирование данных

  Аналогично переключению контекста, уменьшить число копирований позволяет отказ от попакетной обработки. Лучшее решение предоставляют библиотеки, позволяющие задействовать функции прямого обращения к памяти сетевой карты (DMA), а также копирование напрямую из буфера сетевой карты в память процесса, осуществляющего обработку. Такие библиотеки содержат в своём названии “zero copy” [11].

- Задействование возможностей аппаратного уровня

  Ускорить копирование из буфера сетевой карты позволяет уже упомянутый механизм DMA. Но современные сетевые карты и сопроцессоры могут брать на себя гораздо больше функций, причём выполнять их аппаратно, что снимает нагрузку с программной составляющей: ядра или библиотеки, осуществляющие обработку. К таковым функциям можно отнести как простую проверку контрольной суммы, так и полное управление сервисными сообщениями (TCP offloading) [12].

2.2 Выделение и сборка потоков

Данную проблему можно рассмотреть с двух сторон. На низком уровне – это задача обработки переупорядоченных, повторяющихся и повреждённых данных. Базовым примером тут является обработка протокола TCP, которым предусмотрены такие механизмы [7]. Дополнительно можно упомянуть QUIC (на базе UDP), в котором реализовано большое количество подобных механизмов, с которыми должен работать DPI. На более высоком уровне – это задача о реализации единого интерфейса обработки для потоков данных.

“Сырым” потоком данных будем называть тот, который можно получить с сетевого интерфейса. “Логическим” потоком можно назвать поток, передающийся в качестве полезной нагрузки какого-либо протокола, если протоколом предусмотрена передача потока, а не отдельных сообщений. Хорошим примером тут также может послужить TCP, так как в качестве полезной нагрузки у него выступает именно поток данных. Другими примерами могут быть QUIC, HTTP, IPv4. У последних двух полезная нагрузка не является потоком, но несмотря на это, они поддерживают фрагментацию полезной нагрузки.

2.3 Организация конвейера обработки потоков данных

Процесс обработки сетевого трафика методом протокольного декодирования должен быть устроен итеративно, из-за инкапсуляции потоков данных друг в друга. Организовать процесс таким образом позволит единый интерфейс взаимодействия отдельных этапов обработки. Единый интерфейс обработки потоков данных позволит выделять обработчики в изолированные и взаимозаменяемые модули [7].

Для создания такого интерфейса можно создать новый непрерывный буфер, в который скопировать полезную нагрузку отдельных пакетов. Это позволит реализовать простой интерфейс анализа вложенных протоколов, так как на вход им будет подаваться непрерывный буфер. Недостатком такого решения являются затраты времени на копирование данных. Также нужно иметь в виду, что данные полезной нагрузки могут быть закодированы. К ним может применяться сжатие, шифрование и тому подобные процедуры.

Примером тут может стать протокол HTTP. В этом протоколе длинная последовательность байт полезной нагрузки (например, при передаче большого файла) разделяется на отдельные пакеты – чанки (куски, chunks) с добавлением небольших заголовков. Но, помимо этого, полезная нагрузка может быть сжата. Для последующего анализа нужно не только скопировать данные в новый буфер, но и предварительно распаковать их.

Другим подходом в реализации может стать отказ от копирований. В этом случае придётся реализовывать структуру для хранения адресов отрезков, имеющих отношение к разбираемому потоку данных. В таком случае вместо затрат времени на копирование данных, появятся затраты времени на обращение к данным, так как нужно корректно высчитывать смещение. Работать в такой схеме, с данными, подверженными кодированию, всё сложнее, так как для каждого отрезка нужно хранить параметры для декодирования, либо высчитывать их в момент обращения.

Какой бы подход не был выбран, система анализа должна в своём внутреннем представлении хранить информацию о выделенных потоках, и осуществлять разбор исходя из этой информации. Это обязательное требование к DPI [7].

2.4 Отслеживание и хранение состояния

Для каждого анализируемого потока необходимо сохранить информацию для его идентификации, так называемый “ключ”. При поступлении нового пакета на обработку, при помощи этой информации, необходимо однозначно отнести пакет к какому-либо потоку или создать новый.

По нашему опыту, идентификация потока сводится к двум случаям. Первый случай – когда протокол предусматривает передачу идентификатора потока. Зачастую, он представлен в виде пары идентификаторов в заголовках каждого пакета, для каждой стороны обмена. Например, это можно видеть в QUIC, TCP, UDP. Второй случай – когда поток определяется идентификатором потока нижестоящего протокола, то есть тем, в который он инкапсулирован. Например, “поток” HTTP (до HTTP/2) в рамках TCP-сессии может быть только один.

Однако при определении потока только по таким идентификаторам возможны коллизии.

Первый вид коллизий связан с возможностью инкапсуляции разных потоков одинаковых протоколов с одинаковыми идентификаторами друг в друга. Это может происходить при использовании туннелирования (GRE, протоколы, реализующие VPN и так далее). Пример подхода к разрешению такой коллизии будет подробнее описан в следующем подразделе, о представлении результатов.

Другим видом коллизии можно назвать коллизии во времени, происходящие с протоколами без явного признака начала и конца сессии, например, с протоколом UDP. При очередном соединении порт клиента может быть выбран случайно. Спустя время, после освобождения этого порта он может быть использован снова. Аналогичная ситуация может возникнуть при анализе протокола с явным признаком конца сессии, если этот признак по какой-либо причине не был захвачен системой анализа трафика. Для разрешения такого типа коллизии чаще всего используются метки времени, то есть вводят промежуток времени. Если новый пакет в рамках существующих идентификаторов был создан через время, большее заданного промежутка, то данный пакет относится к новому потоку данных [13].

Разрешение подобных коллизий кажется чем-то самим собой разумеющимся, однако достаточно часто в исследованиях можно встретить ошибки, связанные именно с такими коллизиями [14-15].

Помимо идентификаторов, каждый поток, в зависимости от его “типа”, может сохранять дополнительную информацию. Как минимум, к этой информации относится состояние, в котором пребывает обработчик потока в данный момент. При использовании кодирования – информация для дальнейшего декодирования. При клиент-серверном взаимодействии с различным форматом сообщений для каждого из них – информацию о том, какая сторона является сервером, а какая клиентом.

2.5 Представление результатов

При инкапсуляции потоки данных (сессии) вложены друг в друга. Например, в исходный поток данных (файл pcap или последовательность пакетов Ethernet), поданный на вход вложен поток пакетов IPv4 (возможно фрагментированный). В поток IPv4 вложен поток TCP, а в него прикладной протокол. Это базовый пример, который может быть усложнён при использовании туннелирования (например GRE, VPN и так далее).

При клиент-серверном взаимодействии есть два потока данных: от клиента к серверу и обратный. При одноранговом (P2P) число потоков возрастает вместе с числом участников обмена. Тем не менее, все эти потоки связаны между собой и результат работы полного протокольного декодирования должен это отражать [16].

Так как разбор происходит в рамках потоков данных (сессий), то на высоком уровне результат разбора представляет собой зависимость этих потоков данных (вложенность, логическая связь, и пр.).

В предыдущем подразделе описывалась коллизия при инкапсуляции. Для решения этой проблемы, “ключ” потока должен представлять из себя не только идентификатор данного потока, но также и структуру вложенности протокола, то есть быть составным и включать в себя идентификаторы всех протоколов, в которые он вложен.

При полном протокольном декодировании должен осуществляться разбор каждого поля каждого отдельного пакета. Это означает, что для каждого такого поля должно быть обозначены его начало и конец, должно быть извлечено значение, в соответствии с форматом, значение должно быть интерпретировано и влияние значения на состояние потока должно быть отображено в дополнительной информации потока. Так как поля чаще всего имеют вложенную структуру, то и представляется результат в виде дерева. Подробнее это описано в [17].

2.6 Расширяемость (модульность)

Затруднительно организовать монолитную структуру сетевых анализаторов в связи с тем, что постоянно появляются новые протоколы, а уже существующие всё время дополняются. Поэтому зачастую прибегают к подходу, когда реализация разбора конкретного протокола представляется в виде модуля (библиотеки) [18].

В таком случае для центрального компонента системы анализа возникает необходимость подсистемы для управления отдельными модулями разбора. В задачи такой подсистемы входит: обнаруживать такие модули, последовательно передавать им входные данные и получать результат. При этом нужно выдерживать слабую связность компонентов между собой, чтобы обновление одного не приводило к повторному проектированию других.

В качестве альтернативного подхода тут можно рассматривать некое “уведомление” всех модулей о появлении нового. В таком случае упрощается реализация центрального компонента и усложняется реализация модулей разбора отдельного протокола.

2.7 Неприменимость стандартных алгоритмов и библиотек для обработки трафика

Отдельного упоминания в данной работе заслуживает тот факт, что в системах глубокого анализа, которые захватывают сетевой трафик по пути его прохождения (а не на конечных узлах), почти не встречаются готовые библиотеки, в отличии от клиентских реализаций протоколов. Вместо этого, разбор отдельных полей протоколов, отслеживание состояния, и прочий функционал реализуется разработчиками систем анализа. Это связанно с вероятными отличиями трафика, который будет получен принимающей стороной, от трафика, который будет захвачен в одной отдельной точке пути его прохождения. Например, какие-то пакеты могут быть модифицированы или вообще отброшены в процессе передачи. На этом основаны многие атаки на сетевые системы обнаружения и предотвращения вторжений, в составе которых имеются системы глубокого анализа сетевого трафика [19].

3. Обзор архитектур открытых инструментов

В этом разделе представлен обзор существующих архитектурных решений и свойств открытых инструментов для глубокого анализа сетевого трафика. Инструменты были выбраны из-за использования метода протокольного декодирования или наличия релевантных механизмов.

3.1 nDPI

На данный момент, nDPI самый популярный инструмент для создания прототипов инструментов, содержащих глубокий анализ сетевого трафика. У него относительно простой для внедрения API, и он работает с приемлемой скоростью [20-21].

В связи с этим, многие разработчики внедрили этот инструмент в свою инфраструктуру, однако результаты, которые могут быть получены с помощью этого инструмента, не всегда хорошо соответствуют требованиям бизнеса. Запрос таких разработчиков – получить более глубокий разбор или большее количество метаинформации по большему числу протоколов, не изменяя механизмов взаимодействия, а также повысить скорость и реализовать распараллеливание обработки.

У nDPI очень гибкий прикладной интерфейс, используемый для получения трафика и его последующей обработки. Есть примеры использования с различными библиотеками, позволяющими производить высокоскоростной захват трафика с сетевого интерфейса. К таковым относятся PF_RING, DPDK, AF_XDP и прочие.

nDPI предполагает как расширение числа модулей, так и интерфейс для получения результатов обработки сетевого трафика.

Для добавление нового протокола или метрики (если речь только о сборе статистики) нужно изменять исходный код nDPI, после чего заново компилировать. Это обусловлено попыткой достичь максимальной скорости работы инструмента и монолитной архитектурой.

Функционал по сборке потоков транспортных протоколов TCP и UDP реализован как частный случай. Информация о собранных потоках располагается непосредственно в ядре системы разбора и не подлежит расширению или модификации без перекомпиляции всего проекта.

Несмотря на то, что nDPI выполняет лишь частичное протокольное декодирование, в данной работе он упомянут из-за схожего с полным протокольным декодированием механизма работы. Каждый протокол имеет свои, независимые функции распознавания и разбора. Хотя для разработчиков оставлена возможность объединить некоторые распознаватели, из-за организации кода в виде монолита.

Однако nDPI часто используется для категорирования потока, а не каждого отдельного пакета в нём. Поэтому в коде предусмотрены оптимизации, позволяющие не обрабатывать потоки, которые уже были категоризированы.

Есть несколько примеров использования nDPI, предоставляемых разработчиками библиотеки. Существуют модули ядра Linux, называемые ndpi-netfilter или xt_ndpi [22]. Эти модули позволяют классифицировать трафик по протоколам прикладного уровня и ограничивать прохождение трафика, исходя их этой информации.

В библиотеке nDPI существует программное API, которое помимо классификации трафика по прикладным протоколам и категориям, позволяет выявлять признаки некоторых сетевых атак (например, возможные XSS, SQL injection, RCE, слабую криптографию и так далее).

В библиотеку также включен демон (nDPId), пропускающий через себя трафик и генерирующий поток событий [23]. К событиям могут относиться создание и завершение потока, обработка очередного пакета, успешное или неуспешное детектирование протокола транспортного или прикладного уровня, а также сообщения об ошибках в процессе разбора.

Есть инструменты, например, proof of concepts (ntop) [24], реализующие полный функционал системы фильтрации контента, и системы предотвращения угроз, на сколько это позволяет библиотека nDPI.

3.2 Wireshark

Другим примером может служить Wireshark. Тем более, что в научных статьях можно встретить множество упоминаний сравнения результатов работы именно с этим инструментом [25-26].

Для захвата трафика может использоваться библиотека libpcap, что ставит под вопрос производительность при работе с трафиком, получаемым из сетевого интерфейса (*онлайн-анализ*). Хотя из-за модульности архитектуры возможны исключения, со своими ограничениями [27].

С точки зрения организации кода в Wireshark, центральным функциональным блоком, интересным для изучения в рамках данной работы, является Epan (Enhanced Packet ANalyzer) [28]. Он включает в себя функционал для хранения состояния разбора (дерева разбора), интерфейс для обращения к модулям, реализующим разбор отдельных протоколов и фильтрацию результатов исходя из запросов пользователя.

Разбор отдельных протоколов делится на два этапа, предварительный и этап запросов пользователей информации о конкретных пакетах. В некоторых модулях за оба прохода отвечает один и тот же код. Интерфейс ядра позволяет выделить поток из буфера пакета и произвести его сборку в отдельный буфер [29].

Хранение состояния модуля разбора происходит на стороне ядра, состояние передаётся каждый раз при разборе очередного пакета. Хранение дополнительных параметров также возможно.

Предполагается, что результаты обработки будут показаны человеку посредством графического интерфейса. Для автоматической же обработки предполагается либо использование плагинов, схожее с добавлением поддержки нового протокола, либо использование скриптов на языке Lua. Однако стоит упомянуть также про tshark – клиент командной строки Wireshark. После обработки трафика, tshark преобразует результаты в строковый формат и может передать на вход другому приложению или в файл. Для вывода доступно большое количество форматов [30].

Wireshark, помимо возможности добавления поддержки новых протоколов через изменение исходного кода, как в nDPI, также поддерживает добавление новых протоколов через разделяемые библиотеки [31].

Другими инструментами, позволяющими расширять базовый функционал Wireshark, можно назвать написание скриптов на языке Lua и декларативное описание разборщиков при помощи языка ASN.1 [32-33].

4. Актуальные проблемы в реализации полного протокольного декодирования и подходы к их решению

После реализации полного протокольного декодирования разработчики приходят к схожим проблемам, которые можно разбить на 3 группы:

- Скорость обработки
- Хранение внутреннего состояния
- Расширение функционала

В ИСП РАН ведётся разработка системы глубокого анализа пакетов “Протосфера” [34]. В процессе разработки были выявлены проблемы из этих групп, подробнее описанные далее.

Эти же проблемы наблюдаются и у ближайшей по возможностям открытой системы – Wireshark [35-36].

4.1 Подходы к общему ускорению работы системы анализа

Для этой группы проблем можно предложить два направления развития решений, потенциально приносящие ускорение:

- Сужение области применимости решаемой задачи, или, другими словами, отбрасывание всего “ненужного” из результатов.
- Распараллеливание алгоритмов работы.

Для первого направления показателен пример операции поиска по метаданным. Во время разбора пакетов часто выполняются операция поиска: поиск нужного потока, поиск встречного потока, поиск подходящего ключа и так далее. В зависимости от контекста можно применять тот или иной алгоритм, чтобы приблизиться к теоретическому минимуму времени работы. Но время всё равно будет зависеть от количества метаданных, по которым осуществляется поиск. Поэтому для ещё большего ускорения в абсолютных значениях необходимо снижать количество метаданных. Этого, в свою очередь, можно достичь, если сокращать время жизни метаданных. Ниже приведены такие эксперименты вкупе с другими методами оптимизации производительности, этот эксперимент будет описан в отдельном разделе.

Также к этому направлению решения можно отнести “насыщение” метаинформацией поверхностный разбор пакетов. Когда полное протокольное декодирование запускается на отдельном участке процесса разбора, в то время как уровни ниже и даже уровни выше могут быть уже распознаны и размечены. Такая схема показывает свою эффективность в системах, где стек протоколов, как правило низкого уровня, зафиксирован, в виду доменной специфики. В этом случае, система, реализующая полное протокольное декодирование, должна иметь интерфейс, для описания протоколов ниже, и смещения, с которого следует начать разбор.

В направлении распараллеливания алгоритмов работы можно предложить несколько подходов.

Исторически первый и самый простой – деление программы на части. Одни, “критические” – те, которые не поддаются распараллеливанию. Остальные же запускать параллельно допустимо. Данный подход позволяет ускорить обработку в рамках одного вычислительного узла. Ускорение в данном случае будет изменяться по закону Амдала, то есть если предположить, что для распараллеливания подходит только половина алгоритма, то ускорение не может быть более 2x раз. В качестве примера можно представить использование библиотеки OpenMP [37].

Другой подход заключается в распределённых вычислениях в рамках кластера:

- Использовать MapReduce для разделения DPI на модули и запускать модули распределённо, например, в таких системах, как Apache Hadoop [38].
- Использовать оркестраторы, например, Kubernetes. Что наиболее предпочтительно ввиду сложности модулей, особенно тех, которые требуют для эффективной работы большой объём контекстной информации.

Популярным подходом является пул “воркеров”, то есть специально заготовленные потоки и/или процессы, которые работают по принципу producer-consumer [37]. Наиболее эффективной реализацией видится деление по потокам данных.

Эффективность работы такой схемы зависит от эффективности балансирования потоков по обработчикам. Но исходя из специфики обрабатываемых данных, в начале работы всегда будет занят только один обработчик, так как входной поток может быть только один для одного источника данных. А этот входной поток позже породит новые потоки. Решением может стать предварительная балансировка, в частности, с привлечением аппаратных средств.

В попытках повысить утилизацию ресурсов можно прийти к концепции асинхронного программирования, при которой возможность распараллеливания сильно увеличивается. За счёт использования “обещаний” в асинхронном программировании можно параллельно обрабатывать даже данные внутри одного пакета.

У этого метода есть много оговорок, также он существенно усложняет поддержку проекта. Например, появляется необходимость синхронизации. Поэтому конкретное описание применения этого метода к задаче полного протокольного декодирования выходит за рамки этой работы и будет рассматриваться отдельно.

4.2 Подходы к управлению памятью

С точки зрения выделения памяти можно выделить два подхода: статическое распределение памяти и использование динамического выделения памяти.

Второй подход проще в реализации, однако имеет несколько значительных недостатков. Главный из которых в том, что нельзя заранее предсказать, в какой момент библиотека или ОС, на которую переложили ответственность за выделение памяти откажет в этой операции. Также критичной является скорость работы, при обращении к ОС.

Статическое распределение памяти решает главный недостаток динамического, количество доступной памяти в используемых структурах заранее известно. Однако с эффективной утилизацией памяти могут быть проблемы, когда память заполняется метаданными неравномерно. А при попытке удалить данные из заполненного буфера приходится также удалять и связанные с ними данные других, не заполненных буферов. Эта ситуация тесно связана со способом хранения связанной информации, что будет описано ниже.

Компромиссным решением в случае реализации системы полного протокольного декодирования сетевого трафика может быть комбинация этих подходов. Например, динамическое выделение больших регионов памяти и статическое распределение этих регионов между структурами разных видов.

Размер данных (и метаданных) рано или поздно превысит доступный объём памяти и встанет вопрос о том, какую информацию в данный момент можно больше не хранить. Существует большое количество подходов к реализации вытеснения информации, среди которых можно упомянуть кольцевой буфер и LRU кэш [39]. Анализ того, какие подходы в конкретных случаях показывают наилучшую эффективность выходят за рамки данной работы и будет рассмотрен отдельно.

Особняком стоит вопрос организации связности данных. Можно хранить связанные данные в виде одной (большой) структуры, а все данные представить в виде массива таких структур, как это показано на рис. 2. Другой же подход состоит в распределении данных по небольшим структурам, массивов в этом случае будет несколько, см. рис. 3.

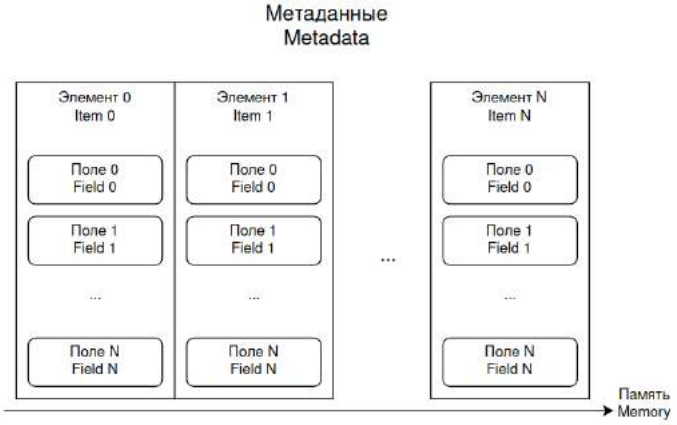

Рис. 2. Представление метаданных в виде массива больших структур
Fig. 2. Representation of metadata as an array of large structures

Показателен пример игровой индустрии, где повсеместно наивный подход хранения коллекции объектов в парадигме ООП заменяется на Data-oriented design. В последнем данные группируются в коллекции по своему типу, что позволяет производить их последовательную обработку, снижая количество “промахов” кэш памяти [40-42].

Выбор того или иного подхода обусловлен спецификой данных, целей анализа, реализацией конструкций языка программирования, размерами и иерархией кэш памяти процессора и прочими факторами, поэтому подробный разбор также выходит за рамки этой работы.

Среди базовых проблем была упомянута проблема сборки потоков. Влияние выбранных механизмов управления памятью на этот процесс очевидно. Однако после оптимизации этого процесса, заметнее проявляет себя проблема обращения к отдельным полям пакета, которые не расположены последовательно. Так как это большое количество обращений к небольшим участкам памяти, копирование на этом этапе оптимизаций является очень затратной операцией. Особенно актуальна эта проблема в Wireshark, где происходит двухэтапный разбор, а также повторный разбор каждый раз, когда пользователь обращается к конкретному пакету или потоку.

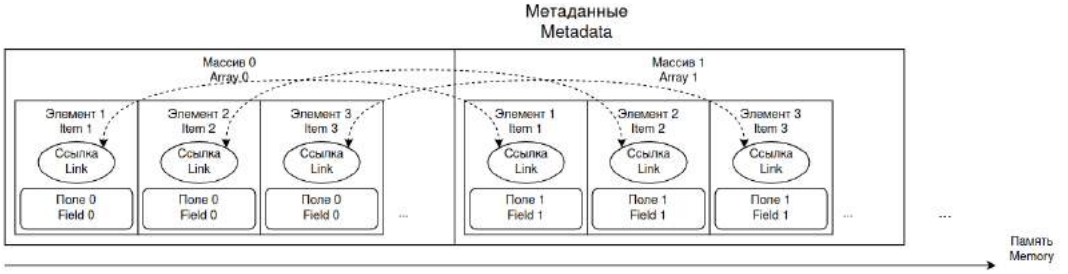

Рис. 3. Представление метаданных в виде массивов небольших связанных структур
Fig. 3. Representation of metadata in the form of arrays of small related structures

4.3 Проблемы с расширением функционала

Различные разработчики могут по-разному реализовывать и использовать различные протоколы. Эта проблема особенно актуальна для разработчиков сетевого оборудования, где можно видеть много расширений открытых и проприетарных протоколов, которые затрудняют согласованную работу оборудования различных производителей. Помимо намеренного внесения усложнений, также свои проблемы вносят допускаемые разработчиками ошибки.

Данная проблема частично была описана среди базовых и часть решения такого рода проблем существует у открытых инструментов, таких как Zeek и Wireshark. Оно состоит в упрощении разработки модулей разбора протоколов. В Wireshark для расширения функционала может использоваться скриптовый язык Lua, так как это императивный язык, то скрипты позволяют модифицировать процесс анализа [32]. В Zeek используется DSL язык генерации парсеров – Spicy [43]. Данный язык основной своей целью ставит именно разработку разборщиков, поэтому на первый план выходят декларативные свойства.

При анализе подобного недетерминированного сетевого трафика, система разбора должна либо по каким-либо косвенным признакам догадываться об используемых расширениях и протоколах (fingerprinting). Либо хранить состояние и производить разбор всех протоколов и расширений, которые возможны в конкретном контексте. На практике для получения наиболее полных результатов необходимо совместное использование этих подходов.

В различных случаях использования DPI требуется различная информация из внутреннего представления, а также различная дополнительная обработка этой информации (собственно, анализ). Например, в решении задач обеспечения информационной безопасности чаще используются анализ потока событий (например, в Zeek). В то же время в задачах отладки и криминалистики – задач, где чаще анализируются записанные данные, анализируется каждое отдельное поле в пакете.

Достаточно сложно организовать универсальный доступ ко всем этим данным, так как итеративно кодовая база системы анализа трафика превращается в набор “заплаток”, которые помогают получать информацию из различных мест в разных контекстах. Именно такую “архитектуру” можно наблюдать в nDPI. С течением времени расширение функционала и исправление ошибок в такой системе даётся всё труднее. Поэтому имеет смысл организовать структуру модулей (менеджеров, компонентов) таким образом, чтобы было возможно получить *всю* информацию. При этом, для того чтобы избежать наложения заплаток должен быть реализован интерфейс, который бы позволял универсально получать всю информацию, и в то же время позволял бы на раннем этапе обработки производить настройку разбора и фильтрацию результата. Последнее особенно важно в контексте полного протокольного декодирования для того, чтобы “иметь возможность не платить за то, что не используется”.

Дополнительно стоит упоминания подход, предложенный при создании eBPF в ядре Linux [44], и который используется в инструменте bpftrace [45]. В этом случае можно небольшие программы оформлять как обратные вызовы при обращениях к API. За счёт этого, например,

иметь возможность собирать (или не собирать) статистику во время выполнения без перезапуска программы.

4.4 Проблемы обработки «одностороннего» трафика

При прохождении трафика в глобальной сети интернет возможны ситуации, когда разные пакеты в рамках одной сессии могут передаваться по различным маршрутам, иногда даже по различным каналам. В качестве примера здесь можно привести спутниковый интернет, где зачастую непосредственно через спутник проходит только входящий для абонента трафик.

Для анализа ряда протоколов очень важно располагать доступом к обоим потокам данных: входящему и исходящему. В качестве примера можно привести протокол ISAKMP из IPSec [46], в котором после инициализации защищённого канала шифрование пакетов происходит последовательно по обоим направлениям. Без доступа к пакетам из одного из них невозможно получить вектор инициализации, необходимый для расшифровки пакета встречного направления. Приведённый пример показывает скорее невозможность полного анализа в такой схеме работы. Однако можно привести и примеры, когда анализ может быть затруднён, например, HTTP 1.1. При анализе только ответов нужно будет определять, должны ли после заголовка идти данные или нет (как в случае ответа на запрос HEAD) [47].

Для полного решения данной проблемы могут применяться системы, предполагающие распределённый захват трафика в различных точках и последующую их синхронизацию.

5. Эксперимент

В качестве оценки прироста производительности при решении актуальных задач, описываемых в данной работе, был проведён эксперимент, в ходе которого в систему, в которой были решены базовые проблемы, было внедрено несколько оптимизаций второго порядка. Оптимизации касались работы с памятью, исходя из особенностей схемы обработки трафика.

Метаинформация была поделена на два класса: “необходимую для дальнейшего анализа” и прочую. Примером первого класса можно привести информацию о “ключе” потока, описанную в предыдущих разделах. Примером второго может послужить информация о поле контрольной суммы отдельного пакета. Такая информация может быть полезна в момент анализа отдельного пакета, а в рамках анализа всего потока, данная информация не несёт ценности.

Время жизни метаинформации первого класса совпадало с временем жизни потока, то есть этот функционал не изменялся. Время жизни метаинформации второго класса зависело от скорости поступления (или скорости вытеснения) такого рода метаинформации, так как она хранилась в ограниченном по размеру буфере, то есть то, что такая информация будет удалена *после* окончания обработки не гарантируется. Но количество одновременно хранимой метаинформации было существенно сокращено, что в свою очередь, привело к росту скорости её обработки, в частности, при поиске связанной информации.

Для измерения производительности был подготовлен набор данных объёмом 5,8 Гб (6076000 пакетов), содержащий трафик реальной сети. Отдельным запуском системы анализа было посчитано количество генерируемой метаинформации (блоков из [7]): всего 164049139 структур, из них 138905530 помечены как метаинформация второго класса, то есть распределение по классам было 15% и 85%.

Система разбора запускалась с включением порядка 30 модулей разбора протоколов, которые детектировали и разбирали порядка 50 протоколов.

При этом стоит пояснить, что время жизни объекта было ограничено только в ядре разбора. Если в это время произошло копирование данных за пределы ядра разбора, то там объект принудительно не удалялся.

Для получения “лучшего времени” исходные данные для анализа считывались с диска, то есть захват трафика с сетевого интерфейса был исключён, но тестирование с захватом трафика с сетевого интерфейса также проводились.

Одновременно с этим проводилась оптимизация использования ресурсов памяти системы в целом. Размеры очередей разных типов данных, выравнивания данных, а также механизмы выделения и высвобождения данных (на куче) эмпирически подбирались таким образом, чтобы оптимизировать обращения к этим данным, в частности, сокращая количество “промахов” при обращении к кэшу памяти. Конкретные результаты в этой работе не приводятся ввиду большого объёма проделанной работы и сложности сравнения различных этапов оптимизаций, в частности, связанных с изменением структуры метаинформации.

Совокупно, данные изменения ограничили возможность обработки метаинформации второго класса, в обмен на ускорение обработки на общих примерах в 6 раз в среднем и 6000 раз в лучшем случае.

6. Заключение

В данной работе был собран и обобщён опыт в разработке системы глубокого анализа трафика (DPI) методом полного протокольного декодирования, который был представлен в виде обзора проблем, возникающих при реализации. Описанные проблемы были разделены на две группы: базовые и актуальные. Рассмотрены решения, представленные в системах с открытым исходным кодом nDPI и Wireshark (преимущественно базовых проблем). Описаны направления решения рассматриваемых проблем, подробное рассмотрение решений каждой будет представлено дополнительно. Приведены результаты эксперимента по применению описываемых оптимизаций в существующей системе глубокого анализа трафика, которые показывают большой потенциал в дальнейшем исследовании возможности применения оптимизаций в описываемой области.

Список литературы / References

[1]. Brunnström K., Beker S., De Moor K., Dooms A., Egger S., Garcia M., Hoßfeld T., Jumisko-Pyykkö S., Keimel C., Larabi C., Lawlor B., Le Callet P., Möller S., Pereira F., Pereira M., Perkis A., Pibernik J., Pinheiro A., Pibernik J., Raake A., Reichl P., Reiter U., Schatz R., Schelkens P., Skorin-Kapov L., Strohmeier D., Timmerer C., Varela M., Wechsung I., You J., Zgank A. Qualinet white paper on definitions of quality of experience. – 2013.

[2]. Gallagher, R. Sandvine Pulls Back From Russia as US, EU Tighten Control on Technology It Sells / R. Gallagher. – Текст: электронный // Bloomberg: [сайт]. – URL: https://www.bloomberg.com/news/articles/2022-06-03/sandvine-pulls-back-from-russia-as-us-eu-tighten-control-on-technology-it-sells 03.06.2022 (дата обращения: 29.06.2023).

[3]. Афанасьева, О. Positive Technologies разрабатывает собственный NGFW / О. Афанасьева. – Текст: электронный // Anti-Malware.ru: [сайт]. – URL: https://www.anti-malware.ru/news/2023-01-13-118537/40305 13.01.2023 (дата обращения: 28.06.2023).

[4]. Афанасьева, О. РТК-Солар показала импортонезависимый NGFW / О. Афанасьева. – Текст: электронный // Anti-Malware.ru: [сайт]. – URL: https://www.anti-malware.ru/news/2023-04-13-118537/40945 13.04.2023 (дата обращения: 28.06.2023).

[5]. Королёв, И. В России вложат более 3 миллиардов в разработку межсетевых экранов нового поколения / И. Королёв. – Текст: электронный // CNews: [сайт]. – URL: https://www.cnews.ru/news/top/2023-04-14_v_rossii_vlozhat_bolee_3_mlrd 14.04.2023 (дата обращения: 28.06.2023).

[6]. Christopher Parsons. Deep Packet Inspection in Perspective: Tracing its lineage and surveillance potentials // Working Paper, January 2009

[7]. Маркин Ю. В. Методы и средства углубленного анализа сетевого трафика //автореферат дис. кандидата технических наук/Ин-т систем. программирования. Москва. – 2017.

[8]. Ким Д., Рыжков В. NTA, IDS, UTM, NGFW – в чем разница? / Д. Ким, В. Рыжков – Текст: электронный // SecurityLab.ru: [сайт]. – URL: https://www.securitylab.ru/analytics/517592.php 19.03.2021 (дата обращения: 05.07.2023).

[9]. Klein, M. Introduction to modern network load balancing and proxying / M. Klein. – Текст: электронный // The official Envoy Proxy blog: [сайт]. – URL: https://blog.envoyproxy.io/introduction-to-modern-network-load-balancing-and-proxying-a57f6ff80236 28.12.2017 (дата обращения: 29.05.2023).
[10]. Çelebi M., Özbilen A., Yavanoğlu U. A comprehensive survey on deep packet inspection for advanced network traffic analysis: issues and challenges //Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi. – vol. 12. – №. 1. – pp. 1-29.
[11]. Ларин Д. В., Гетьман А. И. Средства захвата и обработки высокоскоростного сетевого трафика //Труды Института системного программирования РАН. – 2021. – т. 33. – №. 4. – с. 49-68.
[12]. Pismenny, B., Eran, H., Yehezkel, A., Liss, L., Morrison, A., Tsafrir, D. Autoomous NIC noffloads //Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. – 2021. – С. 18-35.
[13]. Borisov, N., Brumley, D., Wang, H. J., Dunagan, J., Joshi, P., Guo, C. Generic Application-Level Protocol Analyzer and its Language //NDSS. – 2007.
[14]. Engelen G., Rimmer V., Joosen W. Troubleshooting an intrusion detection dataset: the CICIDS2017 case study //2021 IEEE Security and Privacy Workshops (SPW). – IEEE, 2021. – с. 7-12.
[15]. Гетьман, А. И., Горюнов, М. Н., Мацкевич, А. Г., Рыболовлев, Д. А. Методика сбора обучающего набора данных для модели обнаружения компьютерных атак //Труды Института системного программирования РАН. – 2021. – т. 33. – №. 5. – с. 83-104.
[16]. Рекомендация МСЭ-Т Y.2770 - Требования к углубленной проверке пакетов в сетях последующих поколений.
[17]. Гетьман, А. И., Иванников, В. П., Маркин, Ю. В., Падарян, В. А., Тихонов, А. Ю. Модель представления данных при проведении глубокого анализа сетевого трафика //Труды Института системного программирования РАН. – 2015. – т. 27. – №. 4. – с. 5-22.
[18]. Andrew Moore, James Hall, Christian Kreibich, Euan Harris, and Ian Pratt. Architecture of a Network Monitor // International Workshop on Passive and Active Network Measurement, PAM 2003
[19]. Bukac V. IDS system evasion techniques //Master. Masarykova Univerzita. – 2010.
[20]. Bujlow T., Carela-Espanol V. Comparison of Deep Packet Inspection (DPI) Tools for Traffic Classification. – 2013.
[21]. Satrya G. B., Nugroho F. E., Brotoharsono T. Improving network security-a comparison between ndpi and l7-filter //International Journal on Information and Communication Technology (IJoICT). – 2016. – vol. 2. – №. 2. – pp. 11-11.
[22]. ndpi-netfilter – Текст: электронный // github.com: [сайт]. – URL: https://github.com/betolj/ndpi-netfilter (дата обращения: 10.07.2023).
[23]. nDPId: Tiny nDPI based deep packet inspection daemons / toolkit – Текст: электронный // github.com: [сайт]. – URL: https://github.com/utoni/nDPId (дата обращения: 10.07.2023).
[24]. ntopng – Текст: электронный // ntop: [сайт]. – URL: https://www.ntop.org/products/traffic-analysis/ntop/ (дата обращения: 10.07.2023).
[25]. C. Shen, L. Huang, On detection accuracy of L7-filter and OpenDPI, in: 2012 Third International Conference on Networking and Distributed Computing (ICNDC), IEEE, Hangzhou, China, 2012, pp. 119–123, dOI: 10.1109/ICNDC.2012.36.
[26]. R. Goss, R. Botha, Deep Packet Inspection – Fear of the Unknown, in: Information Security for South Africa (ISSA), 2010, IEEE, Sandton, Johannesburg, South Africa, 2010, pp. 1–5, dOI: 10.1109/ISSA.2010.5588278
[27]. Capture, Filter, Extract Traffic using Wireshark and PF_RING. – Текст: электронный // ntop: [сайт]. – URL: https://www.ntop.org/pf_ring/capture-filter-extract-traffic-using-wireshark-and-pf_ring/ 04.04.2017 (дата обращения: 10.07.2023).
[28]. Chapter 7. How Wireshark Works – Текст: электронный // ntWiresharkop: [сайт]. – URL: https://www.wireshark.org/docs/wsdg_html_chunked/ChWorksOverview.html (дата обращения: 24.07.2023).
[29]. 9.5. How to reassemble split packets. – Текст: электронный // wireshark: [сайт]. – URL: https://www.wireshark.org/docs/wsdg_html_chunked/ChDissectReassemble.html (дата обращения: 28.06.2023).
[30]. tshark(1) Manual Page. – Текст: электронный // Gitlab: [сайт]. – URL: https://gitlab.com/wireshark/wireshark/-/blob/master/doc/tshark.adoc (дата обращения: 28.06.2023).
[31]. Chapter 9. Packet Dissection. – Текст: электронный // Wireshark: [сайт]. – URL: https://www.wireshark.org/docs/wsdg_html_chunked/ChapterDissection.html (дата обращения: 28.06.2023).

[32]. Chapter 11. Wireshark's Lua API Reference Manual. – Текст: электронный // Wireshark: [сайт]. – URL: https://www.wireshark.org/docs/wsdg_html_chunked/wsluarm_modules.html (дата обращения: 28.06.2023).
[33]. Chapter 14. Creating ASN.1 Dissectors. – Текст: электронный // Wireshark: [сайт]. – URL: https://www.wireshark.org/docs/wsdg_html_chunked/CreatingAsn1Dissectors.html (дата обращения: 28.06.2023).
[34]. Свид. 2019614453 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Ядро системы глубокого разбора пакетов «ПРОТОСФЕРА» / А. И. Аветисян, С. С. Гайсарян, А. И. Гетьман, Ю. В. Маркин, Д. О. Обыденков, В. А. Падарян, А. Ю. Тихонов; Правообладатель: Федеральное государственное бюджетное учреждение науки Институт системного программирования им. В.П. Иванникова Российской академии наук (RU). – №2019613262; заявл. 28.03.2019; опубл. 04.04.2019, Реестр программ для ЭВМ. – 1 с.
[35]. KnownBugs - OutOfMemory. – Текст: электронный // Wireshark Wiki: [сайт]. – URL: https://wiki.wireshark.org/KnownBugs/OutOfMemory.md (дата обращения: 28.06.2023).
[36]. Multithreading. – Текст: электронный // Wireshark Wiki: [сайт]. – URL: https://wiki.wireshark.org/Development/multithreading (дата обращения: 28.06.2023).
[37]. Garg R. P., Sharapov I. A. Techniques for optimizing applications: high performance computing. – Palo Alto: Sun Microsystems Press, 2002. – С. 394.
[38]. MapReduce Tutorial. – Текст: электронный // Apache Hadoop: [сайт]. – URL: http://apache.github.io/hadoop/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (дата обращения: 29.06.2023).
[39]. Larin D. V., Get'man A. I. Tools for Capturing and Processing High-Speed Network Traffic //Programming and Computer Software. – 2022. – vol. 48. – №. 8. – pp. 756-769.
[40]. Llopis N. Data-Oriented Design (Or Why You Might Be Shooting Yourself in The Foot With OOP) / Llopis N. – Текст: электронный // Games from Within: [сайт]. – URL: https://gamesfromwithin.com/data-oriented-design 04.12.2009 (дата обращения: 29.05.2023).
[41]. Llopis N., Touch S. High-performance programming with data-oriented design //Game Engine Gems. – 2011. – vol. 2. – pp. 251-261.
[42]. Fabian R. Data-oriented design //framework. – 2018. – vol. 21. – pp. 1.7.
[43]. Spicy – Generating Robust Parsers for Protocols & File Formats. – Текст: электронный // Spicy: [сайт]. – URL: https://docs.zeek.org/projects/spicy/en/latest/index.html (дата обращения: 29.06.2023).
[44]. BPF: the universal in-kernel virtual machine. – Текст: электронный // Linux Weekly News: [сайт]. – URL: https://lwn.net/Articles/599755/ (дата обращения: 28.06.2023).
[45]. bpftrace. – Текст: электронный // github.com: [сайт]. – URL: https://github.com/iovisor/bpftrace (дата обращения: 28.06.2023).
[46]. Maughan D., Schertler M., Schneider M., Turner J. Internet Security Association and Key Management Protocol (ISAKMP) IETF RFC № 2408 // IETF. - 1998 г. - № 2408
[47]. Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T. Hypertext Transfer Protocol - HTTP/1.1 IETF RFC № 2616 // IETF. - 1999 г. - № 2616

Информация об авторах / Information about authors

Роман Евгеньевич ПОНОМАРЕНКО – аспирант, стажёр-исследователь ИСП РАН. Научные интересы: архитектура программного обеспечения, оптимизация программ, глубокий анализ сетевого трафика.

Roman Evgenevich PONOMARENKO – PhD student, intern researcher at ISP RAS. Research interests: software architecture, program optimization, deep packet inspection.

Владислав Игоревич ЕГОРОВ – аспирант, стажёр-исследователь ИСП РАН. Научные интересы: обработка, анализ и хранение результатов анализа сетевого трафика.

Vladislav Igorevich EGOROV – PhD student, intern researcher at ISP RAS. Research interests: processing, analysis and storage of network traffic analysis results.

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

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

DOI: 10.15514/ISPRAS-2023-35(4)-3

Применение глубокого обучения для обнаружения компьютерных атак в сетевом трафике

[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*
[5] *М.Н. Горюнов, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>*
[5] *А.Г. Мацкевич, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>*
[5] *Д.А. Рыболовлев, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>*
[5] *А.Г. Никольская, ORCID: 0000-0001-5965-4664 <nikolskaya.a.g@yandex.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
[2] *Московский физико-технический институт,*
141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9
[3] *Национальный исследовательский университет «Высшая школа экономики»,*
101978, Россия, г. Москва, ул. Мясницкая, д. 20
[4] *Московский государственный университет имени М.В. Ломоносова*
119991, Россия, г. Москва, Ленинские горы, д. 1
[5] *Академия ФСО России*
302015, Россия, г. Орел, ул. Приборостроительная, д. 35

Аннотация. В работе рассмотрены вопросы применения методов глубокого обучения для обнаружения компьютерных атак в сетевом трафике. Представлены результаты анализа релевантных исследований и обзоров в области применения глубокого обучения для обнаружения вторжений. Произведено описание и сравнение наиболее используемых методов глубокого обучения, предложена система их классификации. Определены существующие тенденции и проблемы применения методов глубокого обучения для обнаружения компьютерных атак в сетевом трафике. Для оценки применимости методов глубокого обучения для обнаружения вторжений синтезирована нейронная сеть CNN-BiLSTM и представлены результаты её сравнения с разработанной ранее моделью, основанной на использовании классификатора типа «случайный лес». Использование метода глубокого обучения позволило упростить этап конструирования признаков, что вместе с близостью полученных значений метрик для сравниваемых моделей подтверждает перспективность применения методов глубокого обучения для обнаружения вторжений.

Ключевые слова: информационная безопасность; система обнаружения атак; обнаружение вторжений; машинное обучение; глубокое обучение; нейронная сеть; свёрточная нейронная сеть; случайный лес; сетевой трафик; компьютерная атака.

Для цитирования: Гетьман А.И., Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А., Никольская А.Г. Применение глубокого обучения для обнаружения компьютерных атак в сетевом трафике. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 65–92. DOI: 10.15514/ISPRAS–2023–35(4)–3.

Deep Learning Applications for Intrusion Detection in Network Traffic

[1,2,3,4] *A.I. Getman, ORCID: 0000-0002-6562-9008<ever@ispras.ru>*
[5] *M.N. Goryunov, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>*
[5] *A.G. Matskevich, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>*
[5] *D.A. Rybolovlev, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>*
[5] *A.G. Nikolskaya, ORCID: 0000-0001-5965-4664 <nikolskaya.a.g@yandex.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Moscow Institute of Physics and Technology (National Research University)*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.
[3] *National Research University «Higher School of Economics»*
20, Myasnitskaya ulitsa, Moscow 101978, Russia
[4] *Lomonosov Moscow State University*
1, Leninskie Gory, Moscow, 119991, Russia
[5] *The Academy of Federal Security Guard Service of the Russian Federation,*
35, Priborostroitelnaya st., Oryol, 302015, Russia

Abstract. The paper discusses the issues of applying deep learning methods for detecting computer attacks in network traffic. The results of the analysis of relevant studies and reviews of deep learning applications for intrusion detection are presented. The most used deep learning methods are discussed and compared. The classification system of deep learning methods for intrusion detection is proposed. Current trends and challenges of applying deep learning methods for detecting computer attacks in network traffic are identified. The CNN-BiLSTM neural network is synthesized to assess the applicability of deep learning methods for intrusion detection. The synthesized neural network is compared to the previously developed model based on the use of the Random Forest classifier. The usage of the deep learning method enabled to simplify the feature engineering stage, and evaluation metrics of Random Forest and CNN-BiLSTM models are close. This confirms the prospects for the application of deep learning methods for intrusion detection.

Keywords: information security; network intrusion detection system; intrusion detection; machine learning; deep learning; neural network; convolutional neural network; random forest; network traffic; computer attack.

For citation: Getman A.I., Goryunov M.N., Matskevich A.G., Rybolovlev D.A., Nikolskaya A.G. Deep Learning Applications for Intrusion Detection in Network Traffic. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 4, 2023. pp. 65-92 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-3.

1. Введение

В последние десятилетия наблюдается активное развитие информационно-коммуникационных технологий и оказываемых с их помощью услуг, неизбежный рост объёмов передаваемых данных. Соответственно увеличивается количество угроз и факторов, которые приводят к нарушению функционирования информационных систем и компьютерных сетей. Вследствие этого актуальными являются проблемы обеспечения информационной безопасности в целом и вопросы развития средств обнаружения компьютерных атак в частности.

Для выявления компьютерных атак в сетевом трафике в настоящее время в основном используются сигнатурные анализаторы, являющиеся частью сетевой системы обнаружения атак (СОА, IDS). Анализ опубликованных исследований в области построения современных систем обнаружения атак свидетельствует о возможности применения в них методов искусственного интеллекта и машинного обучения. Одним из основных преимуществ эвристических анализаторов СОА, основанных на использовании методов машинного обучения, по сравнению с традиционными сигнатурными анализаторами является способность выявлять новые виды атак. Вместе с тем при разработке таких эвристических анализаторов исследователи сталкиваются со множеством объективных трудностей:

отсутствие доверенных размеченных данных для обучения, необходимость разработки и/или поддержания актуальными генераторов атак, сложность формирования оптимального признакового пространства для решения задачи переноса обучения, необходимость защиты модели машинного обучения против состязательных атак и др.

В настоящей работе предпринята попытка провести анализ релевантных исследований и обзоров в области применения глубокого обучения для обнаружения вторжений. Новизна исследования заключается в описании и сравнении наиболее используемых методов глубокого обучения, предложенной системе их классификации. Представленные результаты отражают современное состояние научных исследований в рассматриваемой предметной области.

В исследовании [1] отмечается, что для использования методов машинного обучения в СОА обычно требуется преобразование данных в набор признаков, то есть формирование признакового пространства, и подчеркивается важность отбора подходящих признаков для точности данных методов. Для решения задач обработки признакового пространства большой мощности, выбора оптимального признакового пространства и ряда других всё чаще используют так называемые методы глубокого обучения нейронных сетей. Данные методы позволяют повысить степень автоматизации обработки данных большой размерности и добиться лучшего качества обнаружения атак по сравнению с другими методами машинного обучения.

Сегодня не существует чётко определённых стандартов в применении методов глубокого обучения для решения задачи обнаружения компьютерных атак. Рассматриваемая предметная область активно развивается, что обусловливает актуальность и подтверждает необходимость проведения системного анализа применяемых методов, их классификации и сравнения, исследования имеющихся проблем и возможных способов их решения. Многообразие и вариативность характеристик методов глубокого обучения при решении задачи обнаружения компьютерных атак также обусловливает актуальность задачи их классификации.

Для оценки применимости методов глубокого обучения для обнаружения вторжений в исследовании синтезирована нейронная сеть CNN-BiLSTM [2] и представлены результаты её сравнения с разработанной ранее моделью [3], основанной на использовании классификатора типа «случайный лес» (Random Forest), на публичном наборе данных обучения CICIDS2017 [4]. Разработанная модель нейронной сети предназначена для использования в сетевой СОА.

2. Глубокое обучение и архитектуры нейронных сетей

Глубокое обучение применяется к искусственным нейронным сетям, состоящим из объединённых в слои искусственных нейронов. В такой сети входные данные обрабатываются в процессе прохождения от первого слоя, называющегося входным, через промежуточные (скрытые) слои до последнего (выходного) слоя. Если промежуточных слоёв больше одного, такая искусственная нейронная сеть называется глубокой.

Функция, в виде которой представляются нейроны, называется функцией активации. Значение функции активации зависит от взвешенной суммы входов нейрона и порогового значения, при этом выходом нейрона является результат применения функции активации к скалярному произведению входного вектора и вектора весов нейрона, смещенное на заданное расстояние [5]. Примерами нелинейных функций, используемых в качестве функции активации, являются сигмоида, функция softmax, линейный выпрямитель (Rectified linear unit, ReLU), гиперболический тангенс.

В обучении нейронной сети используется функция потерь (ошибки), которая характеризует разницу между правильным значением целевой переменной и значением, предсказанным нейронной сетью.

Различные конфигурации нейронов, слоёв, их связей между собой порождают различные архитектуры нейронных сетей. Методы глубокого обучения, таким образом, можно классифицировать по используемой архитектуре нейронной сети.

Кроме того, как и традиционные методы машинного обучения, методы глубокого обучения можно разделить на основные группы: методы глубокого обучения с учителем, с частичным привлечением учителя, без учителя, с подкреплением. Возможность создавать сложные архитектуры нейронных сетей, объединяя разные архитектуры в одной сети, приводит к существованию гибридных методов.

Далее приведено краткое описание основных методов глубокого обучения, которые фигурируют в проведённом анализе релевантных работ.

Искусственная нейронная сеть (Artificial Neural Network, ANN), как упоминалось выше, может быть как *глубокой (Deep Neural Network, DNN)*, так и *неглубокой (Shallow Neural Network, S-NN)* в зависимости от числа скрытых слоёв. Базовый вариант таких сетей представляет собой сеть прямого распространения (Feed forward neural network), в которой сигнал распространяется строго от входа к выходу. Обучение обычно осуществляется методом обратного распространения ошибки.

Многослойный перцептрон (Multilayer Perceptron, MLP) представляет собой разновидность перцептрона – простейшего вида нейронных сетей, основанного на математической модели восприятия информации мозгом, предложенной Ф. Розенблаттом [6]. Перцептроны состоят из сенсоров, ассоциативных и реагирующих элементов. MLP является полносвзяной ANN, при этом между входным и выходным слоями может быть один или несколько скрытых слоев. Многослойный перцептрон обучается с учителем при помощи алгоритма обратного распространения ошибки. Более современной разновидностью MLP по Розенблатту является многослойный перцептрон по Румельхарту [7].

Свёрточная нейронная сеть (Convolutional Neural Network, CNN) представляет собой однонаправленную многослойную сеть с чередующимися свёрточными (convolution layers) и субдискретизирующими (subsampling layers или pooling layers) слоями. Свёрточный слой формирует карту признаков путём поэлементного умножения матрицы весов (ядра свёртки) на каждый фрагмент входного слоя и суммирования результата, который записывается в аналогичную позицию выходного слоя [5]. Субдискретизирующий слой уменьшает размерность карт признаков на отдельных слоях путём выбора отдельного нейрона карты среди соседних. CNN позволяет выделять в данных иерархию абстрактных признаков. Изначально CNN применялись для обработки изображений, но в настоящий момент используются и в других задачах. Использование CNN накладывает ограничение на входные данные: они должны быть представлены в виде нормализованного «изображения». В плане применения к задаче обнаружения вторжений это значит, что каждый вектор признаков из набора данных должен быть преобразован в условное «изображение», имеющее формат «сетки» или таблицы, и нормализован. CNN может обучаться с учителем или без учителя, чаще всего для обучения используется алгоритм обратного распространения ошибки [8].

Рекуррентная нейронная сеть (Recurrent Neural Network, RNN) представляет собой сеть, где связи между нейронами образуют направленную последовательность, при этом нейроны имеют внутреннюю память и могут передавать данные самим себе. Такая архитектура сети позволяет RNN иметь динамическое поведение во времени и обуславливает способность обрабатывать последовательные данные произвольной длины. Однако такая способность приводит к высоким требованиям к ресурсам и проблеме исчезающего (или взрывного) градиента: долгосрочная информация должна последовательно проходить через все ячейки и может быть повреждена многократным умножением на слишком малые (или большие) числа, происходящем при вычислении весов сети методом обратного распространения ошибки во времени (Backpropagation through time, BPTT) [9]. Рекуррентная нейронная сеть может обучаться с различной степенью привлечения учителя, как и её разновидности – LSTM и

GRU, описанные ниже. RNN и её разновидности чаще всего обучаются при помощи BPTT или метода рекуррентного обучения в реальном времени (Real-time recurrent learning, RTRL).

Длинная цепь элементов краткосрочной памяти (Long Short-Term Memory, LSTM) предназначена для решения проблемы исчезающего градиента, свойственной RNN. Сохранение данных в LSTM обеспечивается рекуррентным LSTM-модулем, представляющим собой ячейку памяти, содержащую специальные структуры, реализующие функцию активации – «вентили» или «гейты» (gates). LSTM-модуль позволяет запоминать значения как на короткие, так и на длинные промежутки времени.

Управляемый рекуррентный блок или нейрон (Gated Recurrent Unit, GRU) использует похожий на LSTM вариант ячейки, но с меньшим количеством «вентилей». Таким образом, GRU имеет меньше параметров, чем LSTM, и требует меньше ресурсов для обучения.

Двунаправленная GRU (Bidirectional GRU, BGRU) и *двунаправленная LSTM (Bidirectional LSTM, BiLSTM)* являются разновидностями GRU и LSTM соответственно, позволяющими сети предсказывать результат на основании не только уже обработанных данных, но и всей последовательности целиком. Такие сети имеют два направления вычислений: выходные блоки нейронов вычисляют представление, зависящее как от прошлого, так и от будущего.

Автокодировщик или автоэнкодер (Autoencoder, AE) состоит из входного слоя, кодирующего данные (энкодер), скрытого слоя и выходного слоя, декодирующего данные (декодер). Автокодировщики применяются для автоматического кодирования информации: нейросеть ищет обобщения и корреляцию в поступающих на вход данных и выполняет их сжатие, при этом на выходном слое должен получаться отклик, наиболее близкий к входным данным. Необходимы некоторые ограничения, чтобы AE не обучался простому копированию результатов [8]: например, в стандартном автокодировщике промежуточный слой имеет меньшую размерность, чем входной и выходной слои. Ограничение количества одновременно активных нейронов промежуточного слоя позволяет получить разновидность автокодировщика – *разряженный автокодировщик (Sparse Autoencoder)*. В разряженном автокодировщике размерность промежуточного слоя превышает размерность входного и выходного слоёв. Автокодировщики обучаются без учителя и используют метод обратного распространения ошибки.

Последовательное объединение автокодировщиков позволяет получить *многослойный автокодировщик (Stacked Autoencoder, SAE)*.

Глубокая сеть доверия (Deep Belief Network, DBN) представляет собой композицию подсетей, в которой скрытый слой каждой подсети служит видимым слоем для следующей подсети, и при обучении каждая подсеть должна научиться кодировать предыдущую. Подсетью в данном случае является автокодировщик или ограниченная машина Больцмана. При этом нейроны внутри скрытых слоев не связаны друг с другом, но связаны с нейронами соседнего слоя. DBN обучаются без учителя при помощи жадного послойного обучения.

3. Анализ релевантных работ

В данном разделе приведён анализ релевантных работ и сравнение описанных в них методов глубокого обучения.

Большинство проанализированных в данной работе исследований были отобраны при помощи программы «Publish or Perish» [10] из базы Google Scholar [11]. По запросам «"intrusion detection system" AND "neural network"» и «"intrusion detection system" AND "neural networks"» были отобраны по 50 работ с 2017 по 2021 годы. После объединения полученных списков работ и удаления дубликатов, было получено 94 работы, из которых были выбраны 12 релевантных работ с количеством цитирований не менее 10.

В работе [12] для решения задачи обнаружения вторжений используется CNN-BiLSTM: CNN выявляет пространственные признаки, BiLSTM – временны́е. Применяется гибридный сэмплинг (OSS вместе со SMOTE) для уменьшения времени обучения и балансировки

наборов данных. После гибридного сэмплинга время обучения сократилось для всех сравниваемых моделей, при этом CNN-BiLSTM уступила в скорости обучения LeNet-5, но показала лучшие результаты по всем остальным метрикам. Сравнение проводилось не на самых современных наборах данных: NSL-KDD и UNSW-NB15.

В статье [13] при решении задачи обнаружения вторжений глубокая нейронная сеть (DNN) с 3 скрытыми слоями показала наилучшие результаты по сравнению с классическими алгоритмами машинного обучения. Сравнение проводилось на наборе данных KDD Cup 99 для DNN с 1-5 скрытыми слоями и алгоритмами Ada Boost, Decision Tree, K-Nearest Neighbour, Linear Regression, Navie Bayes, Random Forest, SVM*-Linear, SVM*-rbf. Однако авторами отмечается, что необходимо проводить исследования на более современных наборах данных и в реальных условиях, в том числе в состязательных средах (adversarial environment). Для данной статьи имеется общедоступный репозиторий с кодом и используемым набором данных [14], ссылка на который, однако, не указана в самом исследовании.

В работе [15] для задачи обнаружения вторжений разработана гибридная IDS на основе свёрточно-рекуррентной нейронной сети: CNN выявляет пространственные признаки, RNN – временны́е. Для балансировки набора данных использовалось дублирование примеров миноритарного класса (oversampling). Для оценки производительности в нейросеть перед слоями CNN и RNN были добавлены слои с гауссовским шумом для улучшения обобщающей способности и уменьшения переобучения нейросети. Оценка эффективности проводилась на наборе данных CSE-CIC-IDS2018: использовалось как проведённое авторами сравнение методов, так и данные из других исследований. В проведённом в данном исследовании эксперименте предложенная модель показала лучшие результаты по сравнению с деревом решений, логистической регрессией и алгоритмом XGBoost. Стоит отметить, что, хотя в данном исследовании приводятся два разных сравнения, используются разные наборы оценок и указанные в них оценки не совпадают. Кроме того, авторы указывают на важность тестирования разработанной IDS на более современных данных. Для указанного исследования имеется общедоступный репозиторий с кодом [16], который, однако, на момент написания настоящей работы пуст.

В исследовании [17] для решения задачи обнаружения вторжений используется CNN с предварительным преобразованием данных из векторного формата в «изображение» (матрицу). Для уменьшения размерности пространства признаков использовались метод главных компонент (PCA) и автокодировщик (AE), для оптимизации обучения использовалась пакетная нормализация (batch normalization, BN). По сравнению с традиционными алгоритмами машинного обучения (Naive Bayes, Logistic Regression, Decision Tree, Random Forest, SVM, Adaboost), RNN и трёхслойной DNN, предложенная модель позволяет значительно сократить время обнаружения и показывает лучшие результаты на наборе данных KDD Cup 99. Однако модель показывает низкий уровень обнаружения для атак типов User to Root (U2R) и Remote to Local (R2L) – 20.61% и 18.96% соответственно, в силу малого количества примеров этих атак в используемом наборе данных. В дальнейших исследованиях авторы планируют решать данную проблему путём генерации примеров атак при помощи генеративно-состязательной сети (GAN).

В работе [18] для решения задачи обнаружения вторжений предложена IDS на иерархических пространственно-временных признаках (HAST-IDS), которая сначала «выучивает» низкоуровневые пространственные признаки при помощи CNN, а затем – высокоуровневые временные признаки при помощи двунаправленной LSTM. Исследовались 2 варианта предложенной IDS: HAST-I, которая обучалась только на пространственных признаках, и HAST-II, которая использует и CNN, и LSTM. Выделение признаков происходило автоматически из необработанных данных трафика, производительность оценивалась на наборах данных DARPA 1998 и ISCX 2012. Исследование также определяет оптимальные

длины обрабатываемых данных сетевого соединения и отдельного пакета, которые позволяют провести классификацию. HAST-IDS уменьшает уровень ложных срабатываний (FAR) с помощью улучшения набора признаков, которые не требуется конструировать вручную. Авторы отмечают необходимость усовершенствования предложенного решения для применения к несбалансированным наборам данных, где есть атаки с малым количеством образцов. Также отмечается возможность повышения эффективности решения при обогащении данных признаками, сконструированными вручную.

В статье [19] для решения задачи обнаружения вторжений предложен метод конвертации данных из набора данных NSL-KDD в бинарные векторы, из которых строят «изображение» (матрицу) для классификации при помощи CNN, что позволяет избежать этапа отбора признаков. На подмножествах конвертированного набора данных NSL-KDD протестированные CNN-сети (ResNet 50 и GoogLeNet) показали результаты лучше стандартных классификаторов, но не сильно лучше state-of-the-art решений (сравнение проводилось с J48, Naive bayes, NB Tree, Random forest, Random tree, Multi-layer perceptron, SVM). В статье отмечается необходимость усовершенствования техник репрезентации данных в виде «изображения» для сохранения структурных характеристик данных, к которым чувствительны CNN.

В исследовании [20] при решении задачи обнаружения вторжений применяется метод, позволяющий обнаруживать вторжения на прикладном (application) уровне. Данный метод использует SAE или DBN для конструирования признаков из биграмм символов HTTP-запросов нормального трафика, поступающего на межсетевой экран веб-приложений (web application firewall, WAF). Выполняется параллельное слияние 100 или 30 сконструированных признаков и признаков, полученных из униграмм символов HTTP-запросов нормального трафика (parallel-feature-fusion). Всего в статье проверяется три сценария: простое конструирование признаков при помощи N-грамм, извлечение признаков при помощи нейронных сетей (SAE и DBN) или методов понижения размерности (PCA, KPCA, FICA), параллельное слияние признаков. Эффективность вариантов в данных сценариях сравнивается при помощи одноклассовых классификаторов: One-Class SVM, изоляционный лес, эллиптический конверт (Elliptic Envelope). Сравнение проводится на наборах данных CSIC 2010 и ECML/PKDD 2007. Предложенный метод на основе модели глубокого обучения и слияния признаков показал лучший результат точности и обобщения при разумном времени обнаружения. Варианты с DNN оказались наиболее сбалансированы по точности, обобщению и скорости.

В работе [21] предлагается использовать рекуррентные нейронные сети (RNN) для решения задачи обнаружения вторжений. И для бинарной, и для многоклассовой классификации RNN показывает производительность лучше, чем традиционные алгоритмы (J48, ANN, RF, SVM и др.) на наборе данных NSL-KDD, хоть и требует больше времени на обучение. Предложенный метод также показывает лучшие результаты, чем RNN уменьшенного размера [22] на наборе данных KDD CUP 1999. В статье также изучают выбор гиперпараметров: влияние количества нейронов и скорости обучения на точность метода. Авторы данной работы отмечают имеющиеся у данного метода проблемы исчезающего и взрывающегося градиентов и собираются в дальнейшем исследовать LSTM и Bidirectional RNN для решения данных проблем.

В исследовании [23] протестировали различные архитектуры RNN-сетей (RNN, LSTM, GRU) при решении задачи обнаружения вторжений. Подробно рассматриваются процесс выбора гиперпараметров и проведённые на наборах данных KDD Cup 99 и UNSW-NB15 эксперименты, в том числе эксперименты с малым количеством признаков (4, 8, 11). По сравнению с не-рекуррентными сетями, RNN-сети показали меньший уровень ошибок первого рода (false positives). LSTM давала лучшие результаты, GRU – близкие к ней. На наборе данных UNSW-NB15 результаты были хуже, чем на KDD Cup 99, поскольку в UNSW-NB15 содержится больше различных типов атак. Авторы исследования отмечают, что RNN-

сети хорошо выделяли динамические паттерны, но тестирования только на синтетических наборах данных недостаточно.

В статье [24] предложена модель обнаружения вторжений, объединяющая искусственные нейронные сети с отбором признаков, основанном на корреляции (Correlation based Feature Selection, CFS). Модель была реализована при помощи инструмента RapidMiner и проверена на наборах данных NSL-KDD и UNSW-NB15. Использование CFS позволило за счёт сокращения размерности данных повысить точность, специфичность и чувствительность модели, сократить вычислительное время. Было проведено сравнение реализованной модели с другими современными подходами: данная модель показывает лучшие результаты, однако требует больше вычислительного времени. Предложенный подход может применяться в различных сетях связи, для защиты серверов интернета вещей (Internet of Things, IoT).

В статье [25] для решения задачи обнаружения вторжений предложена модель, состоящая из двух нейронных сетей: Shallow Neural Network (S-NN) и Deep-Optimized Neural Network (D-ONN). Сеть S-NN более простая и быстрая, D-ONN – более сложная и медленная. Отбор признаков осуществлялся методом корреляционного анализа и с применением энтропийного подхода. Данная модель показала наилучший результат на наборе данных KDD Cup 99. Авторы отмечают возможность использования данного метода для защиты беспроводных сетей и IoT.

В статье [26] предложена модель BGRU+MLP для решения задачи обнаружения вторжений. В экспериментах использовались наборы данных KDD Cup 99 и NSL-KDD. По результатам экспериментов, GRU показывает результаты лучше, чем LSTM, BGRU – лучше, чем GRU в отдельности, а сочетание BGRU и MLP даёт лучшие результаты по сравнению с отдельным использованием RNN (GRU или LSTM) или MLP. BGRU+MLP показывает лучшие результаты по точности, количеству обнаруженных инцидентов и доле ложноположительных примеров (FPR), однако имеются проблемы с выявлением атак типа R2L и U2R. Авторы отмечают, что данная проблема свойственна и системам других исследователей в силу малой доли данных атак в используемых наборах данных. Кроме того, используемая RNN-сеть имеет больше преимуществ при работе с временными рядами, а у атак R2L и U2R меньше характеристик, очевидно связанных со временем.

В исследовании [27] для решения задачи обнаружения вторжений предложена модель SFSDT+RNN. Данная модель предназначена для улучшения точности выявления атак, в том числе отдельных типов атак – в частности, упомянутых ранее R2L и U2R. Для отбора признаков используется гибридный алгоритм SFSDT: при помощи алгоритма последовательного прямого выбора (SFS) отбираются наиболее релевантные наборы признаков, среди которых определяется лучший набор признаков при помощи дерева принятий решений (DT). Эксперименты проводились на наборах данных NSL-KDD и ISCX 2012. Модель с использованием LSTM показала лучшую точность среди трёх видов RNN (RNN, LSTM, GRU). Благодаря отбору признаков с помощью SFSDT уменьшились время вычисления и использование памяти. Стоит отметить, что в данном исследовании не указаны подробности конкретных реализаций использованных в экспериментах архитектур (RNN, LSTM, GRU).

Отдельно стоит остановиться на исследованиях, представляющих собой аналитические обзоры в области применения глубокого обучения в IDS.

В статье [28] представлен обзор литературы по использованию нейронных сетей в IDS за 2015-2019 годы. В исследование вошли обзоры литературы, предложения новых методов и обучающие статьи. Рассматриваются наиболее используемые в системах обнаружения атак архитектуры нейронных сетей, распространённые и частные наборы данных, особенности их использования. Поднимается проблема последствий для безопасности при использовании нейронных сетей в IDS.

В работе [29] представлен обзор литературы по использованию машинного обучения и нейронных сетей в IDS с точки зрения предложенной авторами таксономии IDS. В исследование вошли обзоры классификаций IDS, часто используемых в IDS алгоритмов машинного обучения, метрик, наборов данных. Отмечены имеющиеся проблемы предметной области и будущие направления исследований.

В исследованиях [5, 30] представлен аналитический обзор в области глубокого обучения для задач кибербезопасности. Рассматривается применение различных методов глубокого обучения в зависимости от конкретного приложения кибербезопасности. Авторами сделаны выводы по применимости и особенностям применения методов глубокого обучения в задачах кибербезопасности.

4. Классификация методов глубокого обучения

Для осуществления анализа современных методов глубокого обучения в области обнаружения вторжений было выполнено сравнение рассмотренных выше методов. Стоит отметить, что исследователи в своих работах фокусируются на различных аспектах, не всегда предоставляя полную информацию о реализации предложенных ими методов и проведённых экспериментах.

На основании проведённого анализа релевантных работ была разработана классификация методов глубокого обучения в области обнаружения вторжений. Выделенные классификационные признаки условно могут быть разделены на две группы: отражающие основные характеристики и особенности метода и отражающие результаты использования данного метода в практическом применении (в экспериментах).

Таким образом, предлагается следующая классификация методов глубокого обучения.

1) Основные характеристики метода:
   - предложенный метод;
   - уровень обнаружения атак (сетевой, хостовой, прикладной);
   - сочетание с другими методами (метод глубокого обучения используется вместе с не DL-методами);
   - особенности архитектуры предложенного метода;
   - предобработка данных;
   - конструирование признаков;
   - вспомогательные приёмы (методы оптимизации, алгоритмы сэмплирования, техники регуляризации и т.п.).
2) Результаты экспериментальных исследований:
   - методы, с которыми производилось сравнение в исследовании;
   - набор данных;
   - вид классификации (бинарная, многоклассовая);
   - оценка;
   - оборудование и программное обеспечение;
   - время обучения и/или выполнения.

Предложенная классификация позволяет систематизировать знания в рассматриваемой предметной области и может быть использована для проведения дальнейших исследований и проведения сравнительного анализа методов глубокого обучения согласно предложенной классификации, результаты которого представлены в табл. 1 и табл. 2. Как этот анализ показывает, обнаружение атак на прикладном уровне или уровне хоста встречается всего в

двух примерах, а подавляющее большинство реализованных исследователями методов глубокого обучения (11 из 13) обнаруживают атаки на сетевом уровне.

Табл. 1. Сравнение методов глубокого обучения в области обнаружения вторжений по основным характеристикам

Table 1. Comparison of deep learning methods for intrusion detection by main features

Работа, год, ссылка	Уровень обнаружения	Сочетание с другими методами	Предложенный метод	Особенности архитектуры	Предобработка данных	Конструирование признаков	Вспомогательные приёмы
K. Jiang et al, 2020 [12]	Сетевой (network)	+	CNN-BiLSTM	2 свёрточных слоя CNN, 2 скрытых слоя BiLSTM	- OHE; - min-max нормализация; - данные преобразуются в матрицу (ч/б изображение)	- CNN: пространственные признаки; - BiLSTM: временны́е признаки	- гибридный сэмплинг (OSS + SMOTE)
Rahul Vigneswaran et al, 2018	Сетевой (network)	-	DNN	От 1 до 5 скрытых слоёв (оптимально: 3)			- Back propagation; - Dropout
M.A. Khan, 2021 [15]	Сетевой (network), на уровне хоста (host)	+	HCRNN (CNN-RNN)	2 свёрточных слоя CNN, 2 скрытых слоя RNN	данные преобразуются в матрицу (ч/б изображение)	- удаление части признаков (IP-адреса и временные метки); - CNN: пространственные признаки; - RNN: временны́е признаки	- Oversampling; - слои с Гауссовским шумом; - оптимизация гипер-параметров при помощи случайного поиска
Yihan Xiao et al, 2019 [17]	Сетевой (network)	+	CNN	Lenet-5 + слой Dropout	- OHE; - min-max нормализация; - данные преобразуются в матрицу (ч/б изображение)	PCA/AE для уменьшения размерности пространства признаков	- пакетная нормализация (Batch normalization); - Dropout
Wei Wang et al, 2017 [18]	Сетевой (network)	-	HAST-IDS (CNN-BiLSTM)	*HAST-I:* 2 свёрточных слоя CNN *HAST-II:* 4 свёрточных слоя CNN, 2 скрытых слоя BiLSTM	- OHE; - данные преобразуются в матрицу (ч/б изображение)	- CNN: пространственные признаки; - BiLSTM: временны́е признаки	
Zhipeng Li et al, 2017 [19]	Сетевой (network)	-	CNN	ResNet50, GoogLeNet	- OHE; - min-max нормализация; - данные преобразуются в матрицу (ч/б изображение)		- градиентный спуск; - кросс-энтропия как функция потерь

Работа, год, ссылка	Уровень обнаружения	Сочетание с другими методами	Предложенный метод	Особенности архитектуры	Предобработка данных	Конструирование признаков	Вспомогательные приёмы
Ali Moradi Vartouni et al, 2019 [20]	Прикладной (application)	+	parallel-feature-fusion + SAE/DBN + 1-SVM/IF/Elliptic	От 3 до 4 скрытых слоёв (оптимально: 4)		SAE/DBN + parallel-feature-fusion	
C. Yin et al, 2017 [21]	Сетевой (network)	-	RNN	От 20 до 240 скрытых узлов (оптимально: 80 с NSL-KDD, 20 – с KDD CUP 1999)	- OHE; - min-max нормализация		- кросс-энтропия; - оптимизация коэффициента скорости обучения
Vinayakumar R. et al, 2017 [23]	Сетевой (network)	-	RNN (RNN, LSTM, GRU)	От 1 до 4 скрытых слоёв с 32 ячейками памяти			- BPTT; - ADAM; - кросс-энтропия
Sumaiya Thaseen et al, 2020 [24]	Сетевой (network)	+	CFS + ANN	6 скрытых слоёв	min-max нормализация	CFS	
Mangayarkarasi Ramaiah et al, 2021 [25]	Сетевой (network)	+	S-NN, D-ONN	S-NN без скрытых слоёв, D-ONN с 2 скрытыми слоями	- LabelEn-coder; - стандарти-зация; - диаграмма размаха для выявления выбросов	- корреляция; - RF	ADAM
Congyuan Xu et al, 2018 [26]	Сетевой (network)	-	BGRU + MLP	MLP: 3 слоя, 48 скрытых узлов; BGRU: 128 скрытых блоков	- 1-to-N encoding; - min-max нормализация		- SGD; - Backpropa-gation; - BPTT; - кросс-энтропия
Thi-Thu-Huong Le et al, 2019 [27]	Сетевой (network)	+	SFSDT + RNN (RNN, LSTM, GRU)	RNN, LSTM, GRU	восстанов-ление пропущенных значений	SFSDT: SFS + DT	BPTT

Табл. 2 Сравнение методов глубокого обучения в области обнаружения вторжений по результатам экспериментальных исследований

Table 2. Comparison of deep learning methods for intrusion detection by experimental results

Работа, год, ссылка	Предложенный метод	Сравниваемые методы	Набор данных	Вид классификации	Оценка, %	Оборудование и ПО	Время обучения (применения), с
K. Jiang et al, 2020 [12]	CNN-BiLSTM	RF, AlexNet, LeNet-5, CNN, BiLSTM	NSL-KDD	много-классовая	ACC=83.58, Precision= 85.82, Recall=84.49, F1=85.14	Intel i3-7100U, 12 GB RAM; Windows 10, TensorFlow, Keras 2.2, Python	341.69 (-)
			UNSW-NB15	много-классовая	ACC=77.16, Precision=82.63 , Recall=79.91, F1=81.25		2750.47 (-)
Rahul Vigneswaran et al, 2018 [13]	DNN	Ada Boost, Decision Tree, K-NN, Linear Regression, Naive Bayes, Random Forest, SVM*-Linear, SVM*-rbf	KDD CUP 1999	бинарная	*DNN (3 слоя):* ACC=93, Precision=99.7, Recall=91.5, F1=95.5	Nvidia GK110BGL Tesla k40; Keras, TensorFlow	
M.A. Khan, 2021 [15]	HCRNN (CNN-RNN)	LR, XGB, Decision Tree, HCRNN	CSE-CIC-IDS 2018	бинарная	Precision= 0.9633, Recall=0.9712, F1=0.976, DR=0.97, FAR=2.5	NVIDIA GTX 1080ti; 64-bit, 32 GB RAM, 32-core processor, desktop computer Core I7; Java (JDK) 12, Deeplearning4j 1.0.0. alpha, Spark v2.3.0	
		DBN, DNN, Deep learning, LSTM, IDS using DL, CNN IDS			ACC=97.75, FAR=1.4		
Yihan Xiao et al, 2019 [17]	CNN	Naive Bayes, LR, Decision Tree, Random Forest, SVM, Adaboost, RNN, DNN (3 слоя)	KDD CUP 1999	много-классовая	ACC= 94, DR=93, FAR=0.5	NVIDIA GTX 1080ti; Intel i7-8700k, 32 GB RAM; Windows 10, Keras 1.2	20 (11)

Работа, год, ссылка	Предложенный метод	Сравниваемые методы	Набор данных	Вид классификации	Оценка, %	Оборудование и ПО	Время обучения (применения), с
Wei Wang et al, 2017 [18]	HAST-IDS (CNN-BiLSTM)	PLSSVM, Multi-Classifier, Random Forest, Bayes Net, JRip, SVM, Naïve Bayes, ID3, EID3, MARK-ELM	DARPA1998	много-классовая	*HAST-I:* ACC=99.68, FAR=0.07, DR=97.78	NVIDIA Tesla K40m; Ubuntu 16.04 64-bit OS с Keras и TensorFlow, сервер DELL R720, 16 CPU ядер, 16GB RAM	58 мин (1.7)
		MHCVF, ALL-AGL, KMC + NBC, AMGA2-NB	ISCX 2012	много-классовая	*HAST-I:* ACC=99.69, FAR=0.22, DR=96.91 *HAST-II:* ACC=99.89, FAR=0.02, DR=96.96		
Zhipeng Li et al, 2017 [19]	CNN	J48, Naive Bayes, NB Tree, Random Forest, Random Tree, MLP, SVM	NSL-KDD Test^{+}	бинарная	*ResNet50:* ACC=79.14, Precision= 91.97, Recall=69.41, F1=79.12 *GoogLeNet:* ACC=77.04, Precision= 91.66, Recall=65.64, F1=76.5	TITAN X Pascal; Dell 7910, TensorFlow	
			NSL-KDD Test^{-21}	бинарная	*ResNet50:* ACC=81.57, Precision= 81.81, Recall=99.63, F1=89.85 *GoogLeNet:* ACC=81.84, Precision= 81.84, Recall=100, F1=90.01		
Ali Moradi Vartouni et al, 2019 [20]	parallel-feature-fusion + SAE/DBN + 1-SVM/ IF/Elliptic	1-gram/2-gram + 1-SVM/IF/ Elliptic, PCA/KPCA/ FICA + 1-SVM/IF/ Elliptic, SAE/DBN + 1-SVM/IF/ Elliptic, SAE100/ DBN100 + Fusion + 1-SVM/IF/ Elliptic, SAE30/DBN 30 + Fusion + IF/Elliptic	CSIC 2010	бинарная	*SAE30 + Fusion + IF:* ACC=89.24, DR=89.48, PR=81.58, Specificity= 89.11, F1= 85.35	Intel Xeon 2.2 GHz (2 Processors), 64.0 GB RAM, Windows 7 (64-bit), Python 3.6, Tensorflow	14.14 (14.05)
			ECML/PKDD 2007	бинарная	*DBN30 + Fusion + Elliptic:* ACC=84.02, DR=89.75, PR=80.61, Specificity= 78.25, F1= 84.93		284.54 (0.25)

Работа, год, ссылка	Предложенный метод	Сравниваемые методы	Набор данных	Вид классификации	Оценка, %	Оборудование и ПО	Время обучения (применения), с
C. Yin et al, 2017 [21]	RNN	J48, Naive Bayesian, NB Tree, Random Forest, Random Tree, MLP, SVM	NSL-KDD Test^{+}	бинарная	ACC=83.28	ThinkPad E450, Intel Core i5-5200U CPU @ 2.20 GHz, 8 GB RAM, Theano	5516 (-)
				много-классовая	ACC=81.29		11444 (-)
			NSL-KDD Test^{-21}	бинарная	ACC=68.55		5516 (-)
				много-классовая	ACC=64.67		11444 (-)
		reduced-size RNN	KDD CUP 1999	много-классовая	DR=97.09		1765 (-)
Vinayakumar R. et al, 2017 [23]	RNN (RNN, LSTM, GRU)	RNN (4, 8, 11); LSTM (4, 8, 11); GRU (4, 8, 11); LR; NB; KNN; DT; RF; AB; RNN, LSTM, GRU	KDD CUP 1999	бинарная	*RNN:* ACC=94.2, Precision=100, Recall=92.8, F1=96.2, Loss=0.22 *LSTM:* ACC=99.9, Precision=100, Recall=99.9, F1=99, Loss=0.01 *GRU:* ACC=99.7, Precision=100, Recall=99.7, F1=99.8, Loss=0.01	Nvidia GK110BGL Tesla k40, Tensorflow, Scikit-learn	
				много-классовая	*RNN:* ACC=95.7, Loss=0.33 *LSTM:* ACC=96.98, Loss=0.31 *GRU:* ACC=95.37, Loss=0.63		
		RNN, LSTM, GRU	UNSW-NB15	бинарная	*RNN:* ACC=88.3, Precision=87.6, Recall=96.5, F1=91.8, Loss=0.25 *LSTM:* ACC=89.9, Precision=88.9, Recall=97.3, F1=92.9, Loss=0.22 *GRU:* ACC=89.7, Precision=88.6, Recall=97.3, F1=92.8, Loss=0.23		
				много-классовая	*RNN:* ACC=58.5, Loss=0.89 *LSTM:* ACC=67.5, Loss=0.79 *GRU:* ACC=64.8, Loss=0.84		

Работа, год, ссылка	Предложенный метод	Сравниваемые методы	Набор данных	Вид классификации	Оценка, %	Оборудование и ПО	Время обучения (применения), с
Sumaiya Thaseen et al, 2020 [24]	CFS + ANN	Naïve Bayes, REP Tree, Decision Tree, SVM, Random Tree, RF, Bagging, Randomizable, AlexNet, BiLSTM, CNN, CNN+BiLSTM, Stacking Ensemble, Pelican, TSDL, Neural Network with reduced feature	UNSW-NB15	много-классовая	ACC=96.44, Specificity=98.4		660 (-)
		SVM, CNN, MLP, Bi-LSTM, Naïve Bayes, C4.5, CART, Random Forest, CNN-BiLSTM, Ensemble, SVM-RBF, SAE-SVM-RBF	NSL-KDD	много-классовая	ACC=97.49, Specificity=99.31		500 (-)
Mangayarkarasi Ramaiah et al, 2021 [25]	Shallow Neural Network Model (S-NN), Deep-Optimized Neural Network Model (D-ONN)	SVM, PIO-SVM, GA-SVM, PSO-SVM, DNN-1, DNN-5, Naïve Bayes, KNN, RF, CNN 3 layer, CNN 1/3 layer.LSTM CNN 1-3 layer.GRU CNN 3 layer.RNN DNN 2-5 layer	KDD CUP 1999	много-классовая	*S-NN:* ACC=91, Precision=93, Recall=93 *D-ONN:* ACC=98, Precision=93, Recall=93, F1=98	Google Colab, Keras Tensorflow from Python 3.7	

Работа, год, ссылка	Предложенный метод	Сравниваемые методы	Набор данных	Вид классификации	Оценка, %	Оборудование и ПО	Время обучения (применения), с
Congyuan Xu et al, 2018 [26]	BGRU + MLP	LSTM, LSSVM-FMIFS, LSTM-RNN, Pruning VELM, GA+FLN, PSO+FLN	KDD CUP 1999	много-классовая	ACC=99.84, DR=99.42, FPR=0.05	Intel Core i7 @ 3.4 GHz, 64 GB RAM, NVIDIA TESLA K40. Ubuntu 16.04 LTS, CUDA 8.0, cuDNN 6.0, TensorFlow 1.4.1	
		OS-ELM, LSSVM-FMIFS, TVCPSO-MCLP, VELM	NSL-KDD	много-классовая	ACC=99.24, DR=99.31, FPR=0.84		
Thi-Thu-Huong Le et al, 2019 [27]	SFSDT + RNN (RNN, LSTM, GRU)	SCDNN, STL, DNN, Gaussian-Bernoulli RBM, Naive Bayes, J48, ANN, CART, MDPCA-DBN, Zscore + Kmeans, RNN	NSL-KDD	много-классовая	*RNN:* ACC=89.6 *LSTM:* ACC=92 *GRU:* ACC=91.8	Windows 10, Python	63 (-)
		NB, Bagged-NB, Boosted-NB, AMGA2-NB, TCM-KNN, Zscore + Kmeans	ISCX 2012	бинарная	*RNN:* ACC=94.75 *LSTM:* ACC=97.5 *GRU:* ACC=97.08		120.83 (-)

Практически в половине (7 из 13) работ нейронная сеть используется в сочетании с другими, не DL-методами. Данные методы применяются для конструирования признаков (предназначены для выделения или сокращения признакового пространства), оптимизации процесса обучения (например, методы для балансировки набора данных или уменьшения переобучения), классификации.

Наиболее часто встречаются RNN с разновидностями и CNN (7 и 5 работ соответственно). При этом в 4 из 13 рассмотренных работ используется не один вид нейронной сети, а их сочетание (например, CNN-RNN). Сочетания различных архитектур в одном методе призваны устранить недостатки конкретных методов или в целом улучшить степень автоматизации всего процесса выявления атак.

Стоит отметить, что не во всех исследованиях указаны подробности архитектуры реализованных методов глубокого обучения. Также заметно отсутствие общепринятой

краткой нотации описания слоёв нейронной сети: в различных работах используются визуализации, словесные описания, математическая нотация или принятые в конкретных фреймворках машинного обучения обозначения. В связи с этим, для каждого рассмотренного метода были отмечены конкретные особенности, указанные в исследовании, например, используемые в работе вариации архитектуры или число скрытых слоёв.

В качестве метода предобработки данных чаще всего встречается быстрое кодирование (One-Hot Encoding, OHE), минимаксная (min-max) нормализация и преобразование данных в изображение (в случае использования CNN).

Преобладание работ, в которых исследователи обращают внимание на методы конструирования признаков, предобработки данных или различные вспомогательные приёмы (например, методы оптимизации), позволяют говорить о важности данных этапов для достижения нейронными сетями полученных в исследованиях высоких результатов. Стоит отметить, что не каждое исследование касается такого важного вопроса, как выбор гиперпараметров.

В экспериментах, проводимых в рассмотренных исследованиях, сравнение предложенных методов глубокого обучения производится в основном с другими методами глубокого обучения (в том числе с вариациями самого предложенного метода) или с различными методами машинного обучения.

Чаще всего сравнение методов проводилось на наборах данных KDD Cup 1999 и NSL-KDD 2009 (по 6 работ соответственно), UNSW-NB15 (3 работы) и ISCX 2012 (2 работы). Самый современный набор данных в проанализированных исследованиях – CSE-CIC-IDS2018; потерявший актуальность – DARPA 1998. Стоит отметить проблему методов глубокого обучения, свойственную также классическим методам машинного обучения: использование для их обучения устаревших несбалансированных наборов данных, не соответствующих современным данным.

Применение методов глубокого обучения для задачи обнаружения вторжения в рассмотренных исследованиях чаще всего проводится для задачи многоклассовой классификации (9 работ). Указанные в исследованиях проблемы с недостаточно качественными результатами в основном наблюдаются для задачи многоклассовой классификации атак и решаются аккуратной подготовкой набора данных (например, балансировкой набора данных) и/или тщательным отбором признаков.

Использованное оборудование и программное обеспечение указано практически в каждом изученном исследовании, однако время обучения оказалось указано примерно в половине проанализированных статей (7 из 13 работ), а время применения – только в 3 исследованиях. При этом для программного обеспечения не всегда указывается его версия. Данные особенности не позволяют провести качественное сравнение рассмотренных методов глубокого обучения для задачи обнаружения вторжения по критерию времени.

Для оценки качества во всех исследованиях используется доля правильных ответов (accuracy, ACC). Другие часто встречающиеся метрики оценки качества – это точность (precision), полнота (recall), F-мера (F1) и частота обнаружения (DR), встречающиеся примерно в половине работ. Следует отметить разнообразие используемых метрик оценки качества.

Отдельно стоит отметить, что только для одной проанализированной работы из 13 имеется общедоступный репозиторий с кодом, и ещё для одной работы наличие такого кода объявлено, однако репозиторий на настоящий момент пуст. Отсутствие общедоступного кода не позволяет однозначно верифицировать результаты экспериментов, изложенные в исследованиях.

5. Экспериментальные данные

Как было указано ранее, в большинстве проанализированных исследований используются устаревшие наборы данных и отсутствуют общедоступные репозитории с кодом. Устранение

данных недостатков позволит обеспечить возможность работы с современными реальными данными и повысить верифицируемость результатов экспериментов.

С учётом данных замечаний был проведён эксперимент, в котором для оценки применимости методов глубокого обучения для обнаружения вторжений была синтезирована нейронная сеть, состоящая из CNN и двунаправленной LSTM. Выбранная архитектура нейронной сети была предложена в исследованиях [31, 32]. Разработанная модель нейронной сети предназначена для использования в сетевой СОА.

Для проверки синтезированной модели был выбран один из наиболее актуальных публичных наборов данных обучения – Intrusion Detection Evaluation Dataset (CICIDS2017). Набор данных CICIDS2017 подготовлен Канадским институтом кибербезопасности по результатам анализа сетевого трафика в изолированной среде, в которой были смоделированы действия 25 легальных пользователей, а также вредоносные действия нарушителей. Набор объединяет более 50 Гб «сырых» данных в формате PCAP и включает 8 предварительно обработанных файлов в формате CSV, содержащих размеченные сетевые сессии с выделенными признаками в разные дни наблюдения. Сетевые сессии относятся к одному из 15 классов: классу нормального трафика («BENIGN») или одному из 14 классов атак («DoS Hulk», «PortScan», «DDoS», «DoS GoldenEye», «FTP-Patator», «SSH-Patator», «DoS slowloris», «DoS Slowhttptest», «Bot», «Infiltration», «Heartbleed», «Web Attack – Brute Force», «Web Attack – XSS», «Web Attack – SQL Injection»).

Синтезированная модель нейронной сети решает задачу бинарной классификации: определяет, относится ли конкретная сетевая сессия к классу нормального трафика или к классам атак, без определения конкретного класса атаки.

Оценка качества классификаторов производилась на разработанной ранее [3] сбалансированной и предварительно обработанной подвыборке веб-атак WebAttacks набора данных CICIDS2017 с 7267 записями и соотношением нормального и аномального трафика 70% / 30%. В подвыборке содержатся 4 класса: «BENIGN» (5087 записей), «Web Attack - Brute Force» (1507 записей), «Web Attack - Sql Injection» (21 запись), «Web Attack - XSS» (652 записи). Каждая запись в наборе данных WebAttacks представляет собой сетевую сессию и характеризуется 84 признаками, например, IP-адресами источника и приёмника потока данных («Source IP» и «Destination IP»), скоростью потока данных («Flow Bytes/s») и др.

Для обучения нейронной сети использовались 76 из 84 признаков сессий, содержащихся в подвыборке. Из признакового пространства были исключены признаки «Flow ID», «Source IP», «Source Port», «Destination IP», «Destination Port», «Protocol», «Timestamp» в силу относительной лёгкости их подделки злоумышленником и предположении, что признаки «формы» (соответствующие статистикам сетевого трафика) являются более значимыми для общего случая [33].

Данные прошли следующую предварительную обработку:

- минимаксная нормализация признаков: данные, поступающие на вход CNN, лежат в пределах [0, 1];
- преобразование категориальных значений меток при помощи быстрого кодирования (OHE).

На рис. 1 представлено описание слоёв модели синтезированной нейронной сети. Модель является последовательной, входным слоем нейросети является одномерный свёрточный слой, принимающий на вход 76 признаков сессии, выходным – полносвязный слой с 2 нейронами, к выходам которых применяется функция softmax. Между свёрточным и полносвязным слоем имеются два слоя BiLSTM, предварённые слоями пакетной нормализации. Между слоями BiLSTM расположен слой, изменяющий размерность выхода для корректной работы слоёв. После данного слоя, а также свёрточного слоя, применяется операция субдискретизации по максимальному значению. Перед полносвязным слоем

выполняется операция прореживания для предотвращения переобучения. В качестве функции потерь используется категориальная перекрёстная энтропия, в качестве оптимизатора нейронной сети – алгоритм ADAM [34].

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 76, 64)            2112
_________________________________________________________________
max_pooling1d_2 (MaxPooling1D)  (None, 15, 64)         0
_________________________________________________________________
batch_normalization_2 (BatchNormalization)  (None, 15, 64)  256
_________________________________________________________________
bidirectional_2 (Bidirectional)  (None, 128)           66048
_________________________________________________________________
reshape_1 (Reshape)          (None, 128, 1)            0
_________________________________________________________________
max_pooling1d_3 (MaxPooling1D)  (None, 25, 1)          0
_________________________________________________________________
batch_normalization_3 (BatchNormalization)  (None, 25, 1)  4
_________________________________________________________________
bidirectional_3 (Bidirectional)  (None, 256)           133120
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
_________________________________________________________________
activation_1 (Activation)    (None, 2)                 0
=================================================================
Total params: 202,054
Trainable params: 201,924
Non-trainable params: 130
```

Рис. 1. Краткое описание модели CNN-BiLSTM
Fig. 1. Summary of the CNN-BiLSTM model

Тестирование модели нейронной сети было проведено в сравнении с разработанной ранее моделью, использующей классификатор Random Forest [3, 35], также предназначенной для решения задачи бинарной классификации. При этом для модели Random Forest использовались только 10 признаков, на которых данная модель была обучена:

- «Average Packet Size», средняя длина поля данных пакета TCP/IP;
- «Flow Bytes/s», скорость потока данных;
- «Max Packet Length», максимальная длина пакета;
- «Fwd Packet Length Mean», средняя длина переданных в прямом направлении пакетов;
- «Fwd IAT Min», минимальное значение межпакетного интервала (IAT, inter-arrival time) в прямом направлении;
- «Total Length of Fwd Packets», суммарная длина переданных в прямом направлении пакетов;
- «Flow IAT Mean», среднее значение межпакетного интервала;
- «Fwd Packet Length Max», максимальная длина переданного в прямом направлении пакета;
- «Fwd IAT Std», среднеквадратическое отклонение значения межпакетного интервала в прямом направлении пакетов;
- «Fwd Header Length», суммарная длина заголовков переданных в прямом направлении пакетов.

Тестирование модели проводилось в экспериментальной среде со следующими характеристиками: Windows 8.1 64 bit, 2-ядерный CPU Intel Core i5-3317U 1.7 GHz, ОЗУ 4 GB, Python 3.9.7, Tensorflow 2.6.0. Время обучения составило около 801 с.

Качество ответов модели оценивалось с использованием следующих метрик:

- доля правильных ответов (accuracy);
- точность (precision, насколько можно доверять классификатору);
- полнота (recall, как много объектов класса «есть атака» определяет классификатор);
- F1-мера (F1-measure, гармоническое среднее между точностью и полнотой).

В табл. 3 представлены полученные значения метрик моделей CNN-BiLSTM и Random Forest.

Табл. 3. Результаты оценки качества классификаторов
Table 3. Results of evaluation of quality of classifiers

Модель (алгоритм)	Accuracy	Precision	Recall	F1
CNN-BiLSTM	0.948	0.959	0.862	0.908
Random Forest	0.983	0.975	0.968	0.971

На рис. 2 представлен график сравнения метрик качества моделей CNN-BiLSTM и Random Forest.

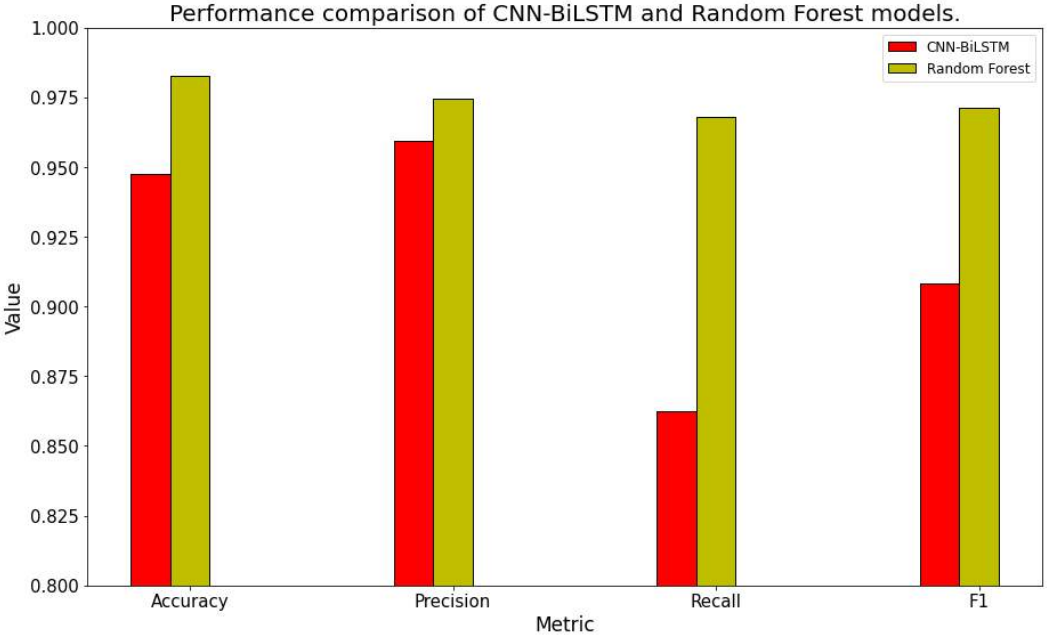

Рис. 2. Сравнение производительности моделей CNN-BiLSTM и Random Forest
Fig. 2. Performance comparison of CNN-BiLSTM and Random Forest models

Как видно из полученных результатов, модель CNN-BiLSTM имеет достаточно высокое качество, однако по всем метрикам уступает модели Random Forest.

На рис. 3 представлена матрица ошибок модели CNN-BiLSTM, а на рис. 4 – такая же матрица модели Random Forest.

Несмотря на то, что модель CNN-BiLSTM показала себя менее эффективной по сравнению с моделью Random Forest, использование метода глубокого обучения позволило исключить этап явного, «ручного» конструирования признаков. Близость полученных значений метрик для сравниваемых моделей подтверждает перспективность применения методов глубокого обучения для обнаружения вторжений. Для достижения необходимого уровня эффективности модели требуются проведение дальнейших исследований и более точная оптимизация нейронной сети для решения поставленной задачи.

Исходный код проекта доступен для выполнения в среде Google Colaboratory: https://colab.research.google.com/github/fisher85/ml-cybersecurity/blob/master/python-web-attack-detection/web-attack-detection-using-CNN-BiLSTM.ipynb.

6. Выводы

В настоящем исследовании проанализированы наиболее используемые методы глубокого обучения в области обнаружения вторжений и предложена система их классификации, основанная на двух группах классификационных признаков: отражающих основные характеристики и особенности метода и отражающих результаты использования данного метода в практическом применении, то есть экспериментах. На основании предложенной классификации было проведено сравнение методов глубокого обучения в области обнаружения вторжений. По результатам проведённого анализа методов глубокого обучения определены существующие тенденции и проблемы рассматриваемой предметной области.

Как видно из анализа релевантных работ, большинство используемых в исследованиях последних лет архитектур показывают хорошие результаты, независимо от того, используется какой-то один вид нейронной сети (например, RNN) или их сочетание (например, CNN-RNN). Сочетания призваны устранить недостатки конкретных методов или в целом улучшить степень автоматизации всего процесса выявления атак.

Отметим, что не столько применение методов глубокого обучения даёт преимущество над классическими ML-методами, сколько именно грамотное их использование со всеми подготовительными и вспомогательными приёмами. Применение только нейронных сетей или нейронных сетей в сочетании с другими, не DL-методами, даёт, в целом, сопоставимо хорошие результаты.

Из собранной выборки релевантных работ можно сделать вывод, что значительной популярностью у исследователей пользуются RNN с разновидностями, CNN и сочетания указанных архитектур. В настоящий момент можно говорить, что данные архитектуры хорошо себя зарекомендовали. Возможно, имеется проблема в том, что в охватываемый рассмотренными исследованиями временной период данные архитектуры пользовались повышенным интересом по сравнению с другими архитектурами, которым уделялось меньше внимания со стороны исследователей. Например, более поздние статьи, но ещё мало цитируемые и потому не попавшие в обзор, широко используют автокодировщики, графовые нейронные сети, трансформеры.

При описании полученных результатов авторы столкнулись с проблемой отсутствия общепринятой краткой нотации описания слоёв нейронной сети: в различных работах используются визуализации, словесные описания, математическая нотация или принятые в конкретных фреймворках машинного обучения обозначения.

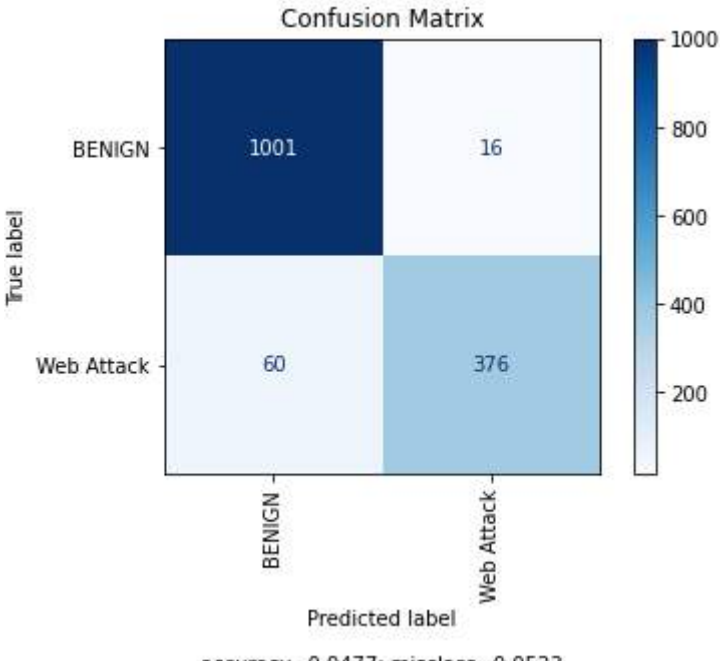

Рис. 3. Матрица ошибок модели CNN-BiLSTM
Fig. 3. Confusion matrix of the CNN-BiLSTM model

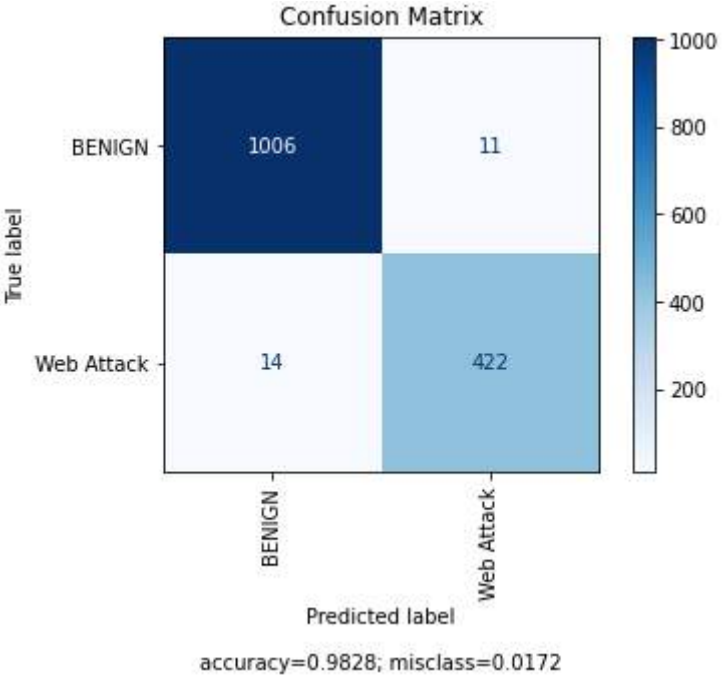

Рис. 4. Матрица ошибок модели Random Forest
Fig. 4. Confusion matrix of the Random Forest model

Стоит отметить, что подавляющее большинство реализованных исследователями методов глубокого обучения обнаруживают атаки на сетевом уровне, а методы обнаружения атак на прикладном уровне или уровне хоста изучены недостаточно.

Кроме того, не в каждом исследовании уделяется достаточное внимание вопросу выбора гиперпараметров. Либо подробные рассуждения просто оставались за рамками статей, либо им не всегда уделялось должное внимание.

Проблемы в основном наблюдаются с многоклассовой классификацией атак, что решается аккуратной подготовкой набора данных – балансировкой, тщательным отбором признаков. В целом, можно отметить наличие тех же проблем, которые имеются и у классических методов машинного обучения: устаревшие несбалансированные наборы данных, не соответствующие реальным данным, что делает сравнение методов проблематичным. Выбор различных метрик для оценки также не способствует качественному сравнению.

Использованное оборудование указано практически в каждом изученном исследовании, однако время обучения или применения оказалось указано только в части проанализированных исследований. Это привело к невозможности качественного сравнения рассмотренных архитектур по критерию времени. Также необходимо отметить, что отсутствие общедоступного кода в большинстве работ не позволяет однозначно верифицировать результаты экспериментов, изложенные в исследованиях.

Для оценки применимости использования методов глубокого обучения для обнаружения компьютерных атак была синтезирована нейронная сеть CNN-BiLSTM, обученная и протестированная на разработанной ранее сбалансированной и предварительно обработанной подвыборке веб-атак WebAttacks набора данных CICIDS2017. Исходный код проекта опубликован для общего доступа, что позволяет обеспечить верифицируемость результатов. При решении задачи бинарной классификации модель CNN-BiLSTM имеет достаточно высокое качество, однако по всем метрикам уступает разработанной ранее модели, использующей классификатор Random Forest. Несмотря на меньшую эффективность разработанной модели CNN-BiLSTM по сравнению с моделью Random Forest, использование метода глубокого обучения позволило исключить этап «ручного» конструирования признаков, что вместе с близостью полученных значений метрик для сравниваемых моделей подтверждает перспективность применения методов глубокого обучения для обнаружения вторжений. Для достижения необходимого уровня эффективности модели требуются проведение дальнейших исследований и настройка параметров нейронной сети для решения поставленной задачи.

7. Список сокращений

ОЗУ – оперативное запоминающее устройство

СОА – система обнаружения атак

ACC – Accuracy, доля правильных ответов

ADAM – Adaptive Moment Estimation, метод адаптивной оценки моментов

AE – Autoencoder, автокодировщик (автоэнкодер)

ANN – Artificial Neural Network, искусственная нейронная сеть

BGRU – Bidirectional GRU, двунаправленная GRU

BiLSTM – Bidirectional LSTM, двунаправленная LSTM

BN – Batch normalization, пакетная нормализация

BPTT – Backpropogation Through Time, метод обратного распространения ошибки во времени

CFS – Correlation Based Feature Selection, основанный на корреляции отбор признаков

CNN – Convolutional Neural Network, свёрточная нейронная сеть

CPU – Central Processing Unit, центральный процессор

CSV – Comma-Separated Values, разделённые запятыми значения

DBN – Deep Belief Network, глубокая сеть доверия
DL – Deep Learning, глубокое обучение
DNN – Deep Neural Network, глубокая нейронная сеть
D-ONN – Deep-Optimized Neural Network, глубоко оптимизированная нейронная сеть
DR – Detection Rate, частота обнаружения
DT – Decision Tree, дерево принятия решений
FAR – False Alarm Rate, уровень ложных срабатываний
FICA – Fast Independent Component Analysis, быстрый метод независимых компонент
FPR – False Positive Rate, доля ложноположительных примеров
GAN – Generative Adversarial Network, генеративно-состязательная сеть
GPU – Graphics Processing Unit, графический процессор
GRU – Gated Recurrent Unit, управляемый рекуррентный блок (нейрон)
HTTP – HyperText Transfer Protocol, протокол передачи гипертекста
IAT – Inter-Arrival Time, межпакетный интервал
IDS – Intrusion Detection System, система обнаружения вторжений
IF – Isolation Forest, изоляционный лес
IoT – Internet of Things, Интернет вещей
IP – Internet Protocol, Интернет-протокол
K-NN – k-Nearest Neighbors Algorithm, метод k-ближайших соседей
KPCA – Kernel Principal Component Analysis, ядерный метод главных компонент
LR – Logistic Regression, логистическая регрессия
LSTM – Long Short-Term Memory, длинная цепь элементов краткосрочной памяти
ML – Machine Learning, машинное обучение
MLP - Multilayer Perceptron, многослойный перцептрон
NB – Naive Bayes Classifier, Наивный байесовский классификатор
NB Tree – Naïve Bayes Tree, наивное байесовское дерево
OHE – One-Hot Encoding, быстрое кодирование
OSS – One-Sided Selection, односторонний сэмплинг
PCA – Principal Component Analysis, метод главных компонент
R2L – Remote to Local, атака типа Remote to Local
ReLU – Rectified Linear Unit, усеченное линейное преобразование
RF – Random Forest, случайный лес
RNN – Recurrent Neural Network, рекуррентная нейронная сеть
RTRL – Real-time recurrent learning, метод рекуррентного обучения в реальном времени
SAE – Stacked Autoencoder, многослойный автокодировщик
SFS – Sequence Forward Selection, последовательный прямой выбор
SGD – Stochastic Gradient Descent, стохастический градиентный спуск
SMOTE – Synthetic Minority Over-sampling Technique, техника передискретизации синтетического меньшинства
S-NN – Shallow Neural Network, неглубокая нейронная сеть
SQL – Structured Query Language, язык структурированных запросов
SVM – Support Vector Machine, метод опорных векторов
U2R – User to Root, атака типа User to Root
WAF – Web Application Firewall, файрвол веб-приложений
XSS – Cross-Site Scripting, межсайтовый скриптинг

Список литературы / References

[1]. Mohammadi S., Namadchian A. Anomaly-based Web Attack Detection: The Application of Deep Neural Network Seq2Seq With Attention Mechanism. The ISC International Journal of Information Security, vol. 12, issue 1, 2020, pp. 44-54. DOI: 10.22042/isecure.2020.199009.479.

[2]. Web attack detection using CNN-BiLSTM neural network and CICIDS2017 dataset. Доступно по ссылке: https://github.com/fisher85/ml-cybersecurity/blob/master/python-web-attack-detection/web-attack-detection-using-CNN-BiLSTM.ipynb, 04.10.2023.

[3]. Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. Синтез модели машинного обучения для обнаружения компьютерных атак на основе набора данных CICIDS2017. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 81-94 / Goryunov M.N., Matskevich A.G., Rybolovlev D.A. Synthesis of a machine learning model for detecting computer attacks based on the CICIDS2017 dataset. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 81-94 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–6.

[4]. Intrusion Detection Evaluation Dataset (CICIDS2017). Available at: https://www.unb.ca/cic/datasets/ids-2017.html, accessed 04.10.2023.

[5]. Гайфулина Д.А., Котенко И.В. Применение методов глубокого обучения в задачах кибербезопасности. Часть 1 // Вопросы кибербезопасности, вып. №3 (37), 2020 г., стр. 76-86. DOI: 10.21681/2311-3456-2020-03-76-86.

[6]. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, vol. 65, issue 6, 1958, pp. 386-408. DOI: 10.1037/H0042519.

[7]. Rumelhart D.E., Hinton G.E., Williams R.J. Learning Internal Representations by Error Propagation. In: Rumelhart, D.E. and McClelland, J.L., The PDP Group, Eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1, Foundations, MIT Press, Cambridge, 1985, pp. 318-362.

[8]. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. Available at: http://www.deeplearningbook.org.

[9]. Culurciello E. The fall of RNN / LSTM (2018). Available at: https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0.

[10]. Harzing A.W. Publish or Perish (2007). Available at: https://harzing.com/resources/publish-or-perish.

[11]. Google Scholar. Available at: https://scholar.google.com, accessed 04.10.2023.

[12]. Jiang K., Wang W., Wang A., Wu H. Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network. IEEE Access, vol. 8, 2020, pp. 32464-32476. DOI: 10.1109/ACCESS.2020.2973730.

[13]. Vigneswaran R.K., Vinayakumar R., Soman K.P., Poornachandran P. Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018, pp. 1-6. DOI: 10.1109/ICCCNT.2018.8494096.

[14]. Intrusion-Detection-Systems. Available at: https://github.com/rahulvigneswaran/Intrusion-Detection-Systems, accessed 04.10.2023.

[15]. Khan M.A. HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System. Processes, vol. 9, issue 5: 834, 2021, 14 p. DOI: 10.3390/pr9050834.

[16]. Hybrid-Convolutional-Recurrent-Neural-Network-Based-Network-IDS. Available at: https://github.com/Ashfaqjiskani/Hybrid-Convolutional-Recurrent-Neural-Network-Based-Network-IDS, accessed 04.10.2023.

[17]. Xiao Y., Xing C., Zhang T., Zhao Z. An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks. IEEE Access, vol. 7, 2019, pp. 42210-42219. DOI: 10.1109/ACCESS.2019.2904620.

[18]. Wang W., Sheng Y., Wang J., Zeng X., Ye X., Huang Y., Zhu M. HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection. IEEE Access, vol. 6, 2018, pp. 1792-1806. DOI: 10.1109/ACCESS.2017.2780250.

[19]. Li Z., Qin Z., Huang K., Yang X., Ye S. Intrusion Detection Using Convolutional Neural Networks for Representation Learning. In: Liu D., Xie S., Li Y., Zhao D., El-Alfy ES. (eds) Neural Information Processing. ICONIP 2017. Lecture Notes in Computer Science, vol. 10638. Springer, Cham, 2017, pp. 858-866. DOI: 10.1007/978-3-319-70139-4_87.

[20]. Vartouni A.M., Teshnehlab M., Kashi S.S. Leveraging Deep Neural Networks for Anomaly-Based Web Application Firewall. IET Information Security, vol. 13, issue 4, 2019, pp. 352-361. DOI: 10.1049/iet-ifs.2018.5404.

[21]. Yin C., Zhu Y., Fei J., He X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access, vol. 5, 2017, pp. 21954-21961. DOI: 10.1109/ACCESS.2017.2762418.
[22]. Sheikhan M., Jadidi Z., Farrokhi A. Intrusion detection using reduced-size RNN based on feature grouping. Neural Computing and Applications - NCA, vol. 21, no. 6, 2012, pp. 1185–1190. DOI: 10.1007/s00521-010-0487-0.
[23]. Vinayakumar R., Soman K.P., Poornachandran P. Evaluation of Recurrent Neural Network and its Variants for Intrusion Detection System (IDS). International Journal of Information System Modeling and Design, vol. 8, no. 3, 2017, pp. 43-63. DOI: 10.4018/IJISMD.2017070103.
[24]. Sumaiya Thaseen I., Saira Banu J., Lavanya K., Rukunuddin Ghalib M., Abhishek K. An integrated intrusion detection system using correlation-based attribute selection and artificial neural network. Transactions on Emerging Telecommunications Technologies, vol. 32, issue 2: e4014, 2021, 15 p. DOI: 10.1002/ett.4014.
[25]. Ramaiah M., Chandrasekaran V., Ravi V., Kumar N. An intrusion detection system using optimized deep neural network architecture. Transactions on Emerging Telecommunications Technologies, vol. 32, issue 4: e4221, 2021, 17 p. DOI: 10.1002/ett.4221.
[26]. Xu C., Shen J., Du X., Zhang F. An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units. IEEE Access, vol. 6, 2018, pp. 48697-48707. DOI: 10.1109/ACCESS.2018.2867564.
[27]. Le T.-T.-H., Kim Y., Kim H. Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks. Applied Sciences, vol. 9, no. 7: 1392, 2019, 29 p. DOI: 10.3390/app9071392.
[28]. Drewek-Ossowicka A., Pietrołaj M., Rumiński J. A survey of neural networks usage for intrusion detection systems. Journal of Ambient Intelligence and Humanized Computing, vol. 12, 2021, pp. 497–514. DOI: 10.1007/s12652-020-02014-x.
[29]. Liu H., Lang B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. Applied Sciences, vol. 9, no. 20: 4396, 2019, 28 p. DOI: 10.3390/app9204396.
[30]. Гайфулина Д.А., Котенко И.В. Применение методов глубокого обучения в задачах кибербезопасности. Часть 2 // Вопросы кибербезопасности, вып. №4 (38), 2020 г., стр. 11-21. DOI: 10.21681/2311-3456-2020-04-11-21
[31]. Sinha J., Manollas M. Efficient Deep CNN-BiLSTM Model for Network Intrusion Detection. Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition (AIPR 2020), Association for Computing Machinery, New York, NY, USA, 2020, pp. 223–231. DOI: 10.1145/3430199.3430224.
[32]. Efficient-CNN-BiLSTM-for-Network-IDS. Available at: https://github.com/razor08/Efficient-CNN-BiLSTM-for-Network-IDS/blob/master/NSL_KDD_Final.ipynb, accessed 04.10.2023.
[33]. Kostas K. Anomaly Detection in Networks Using Machine Learning. Master's Thesis. University of Essex, 2018, 70 p.
[34]. Kingma D.P., Ba J. Adam: A Method for Stochastic Optimization. The International Conference on Learning Representations (ICLR), San Diego, 2015, 15 p. DOI: 10.48550/arXiv.1412.6980.
[35]. Web attack detection using CICIDS2017 dataset. Доступно по ссылке: https://github.com/fisher85/ml-cybersecurity/blob/master/python-web-attack-detection/web-attack-detection.ipynb, 04.10.2023.

Информация об авторах / Information about authors

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

Aleksandr Igorevich GETMAN – Cand. Sci. (Phys.-Math.), senior researcher at ISP RAS, associate professor at HSE. Research interests: binary code analysis, data format recovery, network traffic analysis and classification.

Максим Николаевич ГОРЮНОВ – кандидат технических наук. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы анализа защищенности, машинное обучение, безопасная разработка программного обеспечения.

Maxim Nikolaevich GORYUNOV – Cand. Sci. (Tech.). Research interests: information security, intrusion detection systems, security analysis systems, machine learning.

Андрей Георгиевич МАЦКЕВИЧ – кандидат технических наук, доцент. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы антивирусной защиты, машинное обучение, криптографические методы защиты информации.

Andrey Georgievich MATSKEVICH – Cand. Sci. (Tech.), associate professor. Research interests: information security, intrusion detection systems, anti-virus protection systems, machine learning, cryptographic methods for protecting information.

Дмитрий Александрович РЫБОЛОВЛЕВ – кандидат технических наук. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, машинное обучение, криптографические методы защиты информации.

Dmitry Aleksandrovich RYBOLOVLEV – Cand. Sci. (Tech.). Research interests: information security, intrusion detection systems, machine learning, cryptographic methods for protecting information.

Анастасия Григорьевна НИКОЛЬСКАЯ - Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, машинное обучение, искусственные нейронные сети.

Anastasiya Grigorevna NIKOLSKAYA. Research interests: information security, intrusion detection systems, machine learning, artificial neural networks.

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

Анализ системы контроля доступа в гетерогенных системах больших данных

М.А. Полтавцева, ORCID: 0000-0001-9659-1244 <*poltavtseva@ibks.spbstu.ru*>
М.О. Калинин, ORCID: 0000-0002-9732-0099 <*max@ibks.spbstu.ru*>

Институт кибербезопасности и защиты информации,
Санкт-Петербургский политехнический университет Петра Великого
195251, Россия, г. Санкт-Петербург, ул. Политехническая, д. 29.

Аннотация. Системы управления большими данными являются сегодня востребованными практически во всех отраслях, они же являются фундаментом для обучения искусственного интеллекта. Использование в системах больших данных гетерогенных полихранилищ привело к тому, что инструменты в рамках одной системы имеют различную грануляцию данных и модели контроля доступа. Согласование таких компонентов администратором безопасности и реализация общей политики доступа сегодня выполняются вручную. Это приводит к росту числа уязвимостей настройки, что, в свою очередь, служит частой причиной утечек данных. Анализ работ в области автоматизации и анализа контроля доступа в системах больших данных показывает отсутствие решений автоматизации для систем на основе полихранилищ. В данной работе ставится задача автоматизации анализа контроля доступа в системах управления большими данными. Авторы формулируют основное противоречие, заключающееся, с одной стороны, в требовании масштабируемости и гибкости контроля доступа, а с другой – в росте нагрузки на администратора безопасности, усугубленное использованием различных моделей данных и контроля доступа в компонентах системы. Для решения этой проблемы предлагается новый автоматизированный метод анализа политик безопасности, основанный на графовой модели обработки данных и позволяющий снизить число возможных уязвимостей, возникающих в результате некорректного администрирования систем big data. При проведении анализа в рамках предложенного метода используется модель жизненного цикла данных в системе, текущие настройки и желаемая политика безопасности. Использование двухпроходного анализа (от источников данных к получателям и обратно) позволяет решить две задачи: анализ системы контроля доступа на возможные уязвимости и проверку соблюдения корректности бизнес – логики. В работе приводится пример анализа политик безопасности системы управления большими данными с использованием разработанного программного прототипа, анализируются полученные результаты.

Ключевые слова: информационная безопасность; большие данные; полихранилища; полибазы данных; контроль доступа; жизненный цикл данных; моделирование обработки данных; политика безопасности.

Для цитирования: Полтавцева М.А., Калинин М.О. Анализ системы контроля доступа в гетерогенных системах больших данных. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 93–108. DOI: 10.15514/ISPRAS–2023–35(4)–4.

Благодарности: Исследование выполнено за счет гранта Российского научного фонда № 23-11-20003, https://rscf.ru/project/23-11-20003/, грант Санкт-Петербургского научного фонда (Соглашение №23-11-20003 о предоставлении регионального гранта).

Access control system analysis in heterogeneous Big Data management systems

M.A. Poltavtseva ORCID: 0000-0001-9659-1244 <*poltavtseva@ibks.spbstu.ru*>
M.O. Kalinin ORCID: 0000-0002-9732-0099 <*max@ibks.spbstu.ru*>

Institute of Cyber Security and Information Protection,
Peter the Great St. Petersburg Polytechnic University
29, Polytechnicheskaya st., St. Petersburg, 195251, Russia.

Abstract. Big data management systems are in demand today in practically all industries, and they are also the foundation for artificial intelligence training. The use of heterogeneous poly-stores in big data systems has led to the fact that tools within the same system have different data granularity and access control models. Harmonization of such components by the security administrator and implementation of common access-policy is now done manually. This leads to an increasing number of customization vulnerabilities, which in turn serves as a frequent cause of data leaks. Analysis of works in the area of automation and analysis of access control in big data systems shows the lack of automation solutions for poly-store based systems. This paper poses the problem of automating the analysis of access control analysis in big data management systems. The authors formulate the main contradiction, which consists, on the one hand, in the requirement of scalability and flexibility of access control, and on the other hand – in the growth of the burden on the security administrator, aggravated by the use of different data models and access control in the system components. To solve this problem, we propose a new automated method for analyzing security policies based on a graph model of data processing, which reduces the number of possible vulnerabilities resulting from incorrect administration of big data systems. The proposed method uses the data life cycle model of the system, current settings and the desired security policy. The use of two-pass analysis (from data sources to recipients and back) allows to solve two tasks: analyzing the access control system for possible vulnerabilities and checking compliance with correctness of business logic. The paper gives an example of analysis of security policies of the big data management system using the developed software prototype and analyzes the obtained results.

Keywords: information security; big data; polystore; poly-databases; access control; data life cycle; data processing modeling; security policy.

For citation: Poltavtseva M.A., Kalinin M.O. Access control system analysis in heterogeneous Big Data management systems. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 4, 2023. pp. 93-108 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-4.

Acknowledgements. The study was supported by the grant of Russian Science Foundation No.23-11-20003, https://rscf.ru/project/23-11-20003/; grant of St. Petersburg Science Foundation (Agreement No.23-11-20003 on the regional grant).

1. Введение

Понятие больших данных за последние годы стало неотъемлемой частью современной цифровой экономики. Массивы разнородной, динамически меняющейся информации в своей деятельности используют не только поисковые системы и социальные сети, но и торговые интернет – площадки, операторы связи, банковский сектор, сервисы и службы электронного правительства и даже промышленные предприятия.

Экосистема больших данных включает, в общем случае, три архитектурных уровня, на каждом из которых существует свое понимание что такое "большие данные", своя терминология, специалисты, методы и технологии, а также свои угрозы безопасности массивам информации и свои методы защиты.

Эти уровни можно назвать инфраструктурным, логическим (инженерным) и концептуальным (прикладным). На инфраструктурном уровне под большими данными понимаются как правило системы ЦОД – центров обработки данных. В него входят соответствующее оборудование, сетевая инфраструктура, средства виртуализации [1]. На логическом уровне существует термин – системы управления большими данными,

аналогично хорошо известному понятию систем управления базами данных (СУБД). По аналогии с обработкой данных в СУБД этот уровень и называют иногда инженерным – Data engineering. Логический уровень включает в себя новый класс хранилищ данных – полихранилища или полибазы данных, представляющие собой объединение нескольких разнородных СУБД в единую архитектуру обработки информации [2]. Полихранилища также дополняют инструменты потоковой обработки информации и связанные сервисы: брокеры очередей, балансировщики нагрузки и др. На верхнем, концептуальном или прикладном уровне речь идет о ценности данных для бизнеса, организационных кейсах их использования, передаче на аутсорсинг, совместной обработке и других высокоуровневых задачах управления данными и знаниями в организации [3, 4].

Большое число утечек конфиденциальных данных при их обработке в экосистемах больших данных приводит к необходимости ускорения разработки методов и средств обеспечения безопасности для данного класса систем [5, 6]. В данной работе речь пойдет в первую очередь о безопасности логического уровня обработки данных, или безопасности полихранилищ и, в частности, о решении задачи автоматизации анализа контроля доступа.

2. Анализ контроля доступа в системах полихранилищ больших данных

Современные исследования в области повышения эффективности контроля доступа в гетерогенных системах больших данных сконцентрированы в области поиска новых моделей контроля доступа и использования технологий распределенного реестра для защиты от утечек информации.

2.1 Обзор методов средств анализа и анализа контроля доступа в системах полихранилищ больших данных

На сегодняшний день контроль доступа в системах больших данных автоматизирован для инструментов и экосистем, построенных на основе одной модели данных (например, экосистемы Hadoop и модели вида “ключ-значение”). Для систем больших данных, построенных на основе полихранилищ, задача настройки и анализа целостной системы контроля доступа с учетом всех компонентов и грануляции данных в них реализуется вручную. Последние несколько лет исследователями ведется работа в рамках этой задачи по нескольким направлениям.

Во-первых, в области методов и средств контроля доступа на основе одной универсальной модели. Несмотря на использование различных подходов: ролевой модели [7-9], атрибутивной модели [10] и ее модификаций [11], подхода на основе знаний [12] исследователям не удалось преодолеть проблему различной грануляции данных в полихранилищах [13] и проблема несовершенства гранулированного контроля доступа для этого подкласса решений остается открытой.

Только в этом году исследователями в принципе показано, что адаптация современных методов на основе атрибутивного контроля (ABAC) позволяет обеспечить согласованность доступа на уровне системы в целом хотя бы для гомогенной инфраструктуры [14]. Однако предложенное решение подходит только для инфраструктуры Hadoop, основанной на модели данных ключ-значение. При его переносе на другие классы решений и, тем более, распространении на полихранилища уязвимости переноса правил доступа между компонентами с различной грануляцией данных, описанные, в частности, в [15, 13], сохранятся.

Исследователями также предлагаются универсальные средства автоматизации на основе технологии блокчейн [16, 17]. Основная проблема таких средств – также отсутствие аналитического механизма, позволяющего выявить уязвимости реализации контроля

доступа, так как необходимость перехода к внутренним системам разграничения доступа в структурированных хранилищах сохраняется [15, 18].

Таким образом, существующие решения в области автоматизации и анализа контроля доступа в системах больших данных обеспечивают приемлемую безопасность только в рамках гомогенных систем, а в полихранилищах согласование настроек контроля доступа между гетерогенными компонентами остается выполняемым вручную, что приводит к большим временным затратам, высоким требованиям квалификации аналитика безопасности и значимому числу ошибок.

2.2 Проблемы применения методов контроля доступа в системах управления большими данными

Подсистемы контроля доступа в гетерогенных системах управления большими данными сегодня складываются из совместной работы модулей контроля доступа отдельных инструментов обработки информации. В них доминируют традиционные модели доступа, и настройка реализации политик безопасности между гетерогенными компонентами производится целиком вручную. Инструменты автоматизации существуют только в гомогенных решениях, то есть над отдельными сочетаниями компонентов в рамках одного семейства от одного разработчика. Тем не менее, такие особенности больших данных как объем, разнообразие, динамичность во времени приводят к необходимости поиска новых методов контроля доступа в этой области, которые в ближайшей перспективе могут быть интегрированы в промышленные продукты.

Проанализируем анализа современные методы разграничения доступа в системах управления большими данными с точки зрения проблемы автоматизации и анализа контроля доступа в гетерогенных системах больших данных. Ниже приведен ряд методов, предлагаемых различными исследователями или включенных в современные инструменты.

Основным методом контроля доступа в инструментах работы с большими данными сегодня все еще является ролевой контроль доступа (Role-based access control, RBAC) [7] и его модификации [8]. Он же применяется и в промышленных СУБД, интегрированных в системы больших данных на производстве и в корпорациях. Модель RBAC получила широкое распространение среди различных фреймворков (например, Apache Ranger и Apache Sentry [9]). Несмотря на такие достоинства этой модели, как простота и гибкость, ее основным недостатком является плохая масштабируемость и сложность корректного администрирования на большом числе пользователей (ролей).

Вторым направлением в области контроля доступа в системах управления большими данными является атрибутивный контроль доступа (Atribute-based access control, ABAC). Основной единицей данного метода является атрибут – некоторая характеристика объекта, субъекта или среды исполнения. Суть данного метода заключается в написании политик, состоящих из правил сравнения атрибутов субъектов, объектов и др. (например, среды или подключения) [10]. Хорошая масштабируемость, гибкость и универсальность ABAC компенсируется сложностью внедрения и необходимостью отдельной поддержки неструктурированных данных, выраженной в приведенных ниже его расширениях.

Проблему неструктурированных данных стремятся решить авторы таких подходов, как контроль доступа на основе содержимого (Content-Based Access Control, CBAC) [11] и контроль доступа на основе знаний (Knowledge-based access control, KBAC) [12]. Это еще более гибкие методы, чем RBAC и ABAC. Первый использует семантический анализ объектов для принятия решения о доступе, второй – фактически расширяет CBAC автоматизированным процессом нахождения ключевых слов (атрибутов для объектов по мере добавления их в систему и ведением базы знаний.

Несмотря на то, что последние два метода представлены только в теоретических работах, они явно иллюстрируют направление развития систем контроля доступа больших данных в

сторону масштабируемости и, при этом упрощения работы администратора безопасности путем автоматизации. Стоит отметить, что ни один из представленных методов не обладает универсальностью, то есть каждое решение ограничено определенным набором инструментов и предметной областью. В целом обеспечению контроля доступа в системах управления большими данными посвящено немало работ. Помимо упомянутых, отметим работы по интеграции контроля доступа с компонентами шифрования данных с применением современного подхода на основе иммунизации [16] и реализации на основе технологии распределенного реестра (blockchain) [17].

Такое разнообразие говорит о том, что сегодня в системах больших данных стоит ожидать применения различных моделей контроля доступа как на уровне отдельных инструментов, так и на уровне системы в целом. Поэтому на первый план выходит задача согласования этих моделей и политик в автоматизированном (а не ручном, как это проводится сейчас) режиме для минимизации утечек данных и обеспечения корректности бизнес-логики.

Основная проблематика, акцентированная в исследованиях [13, 15, 18], позволяет обозначить общие особенности систем управления большими данными, затрудняющие сегодня внедрение комплексного согласованного контроля доступа на уровне всей системы в целом. Это:

- Большое количество пользователей с доступом различного характера на всем множестве узлов системы, от обслуживающего персонала и администраторов баз данных, до источников данных и их потребителей.
- Сложность и нелинейность жизненного цикла фрагментов информации в системах больших данных, затрудняющая отслеживание фрагментов и согласованное управление доступом.
- Различная структурированность данных в процессе их обработки в системе, в силу использования различных инструментов обработки и множества операций извлечения частей данных и объединения их в новые наборы.

Эти черты приводят к основному противоречию контроля доступа в рассматриваемых системах. С одной стороны, контроль доступа для выполнения бизнес-задач в условиях большого объема и разнородной грануляции данных требует гибкости и масштабируемости, что приводит к большой сложности ее настройки, неточностям и утечкам данных. С другой, при ужесточении контроля доступа на уровне отдельных компонентов или источников данных, снижается вероятность утечек, но такая ситуация может привести к проблемам в реализации бизнес-логики и недоступности части данных для потребителей. Решение данного противоречия заключается в поиске оптимальной (или, по крайней мере, рациональной – если не формулировать задачу как оптимизационную в силу разнородности критериев и оценок для различных организаций) политики безопасности и автоматизации ее выполнения в рамках системы контроля доступа полихранилищ.

Определенную сложность в таком решении представляет собой задача его практической реализации. Как показано выше, при том, что исследователи уже предлагают интеллектуальные системы, основанные на знаниях, сами инструменты еще реализуют RBAC или другие виды контроля доступа, присущие традиционным системам управления базами данных: дистрибутивный, мандатный доступ. Именно этот факт не позволяет применить атрибутивные системы с использованием знаний [14] без дополнительных аналитических компонентов.

Таким образом, в основе контроля доступа в системах управления большими данными должен лежать компонент анализа, позволяющий в автоматизированном режиме выполнить поиск рациональных параметров доступа в конкретной системе с учетом ее компонентов, особенностей работы и бизнес-логики. Отдельным требованием является возможность проведения такого анализа как на этапе проектирования информационной системы,

включающей обработку больших данных, так и уже в процессе ее функционирования, для оценки защищенности и коррекции работы.

3. Метод автоматизированного анализа контроля доступа в гетерогенных системах больших данных

Анализ контроля доступа в системах управления большими данными фактически решает две основные задачи: снижение числа утечек данных из-за ошибок в настройке политик безопасности [19] и обеспечение достаточной информированности потребителей данных для выполнения их бизнес-функций [20]. Для снижения числа утечек данных необходимо проведение оценки всего жизненного цикла данных, так как полученные на более поздних этапах фрагменты в системах Больших данных не только могут быть семантически связаны с исходными, но и содержать их без изменений, а на доступ к исходным данным потребители могут иметь сложные ограничения, обусловленные политикой безопасности [21]. Поэтому при анализе системы контроля доступа требуется как оценка доступа получателей к исходным данным, при заданном доступе к выходной информации, так и оценка доступа получателей к выходным данным, при заданном желаемом доступе к исходным. В результате такого анализа должна быть сформирована общая политика безопасности системы.

Таким образом, в основе автоматизированного анализа контроля доступа находятся:

- Политика доступа к выходным данным, в терминах одной из моделей безопасности (в общем случае – любой). Этот компонент обусловлен бизнес-логикой.
- Желаемая политика доступа к входным данным, в терминах одной из моделей безопасности (в общем случае – любой). Этот компонент обусловлен требованием минимизации доверия.
- Модель обработки данных в системе управления большими данными, представляющую собой описание процесса получения выходного фрагмента из входного.

В модели процесса обработки данных в системе управления большими данными от исходных фрагментов к результирующим должны учитываться права доступа, которые получает каждый производный фрагмент. Для оценки уже функционирующих систем в данном случае могут быть получены настройки доступа напрямую из инструментов обработки автоматическим путем. Решение этой задачи на этапе проектирования систем управления большими данными заключается в формализации и анализе переноса прав доступа в гетерогенном полихранилище [22-24].

Для моделирования процессов обработки данных авторами предлагается использовать графовую модель жизненного цикла данных, предложенную в [25, 26], вершинами которой являются структурированные фрагменты данных, а ребрами – операции над ними. Такой граф описывает процесс обработки больших данных полхранилища и может быть построен в автоматизированном режиме [26], что является значительным преимуществом.

По результатам анализа было принято решение в качестве базового механизма разграничения доступа использовать атрибутивный подход (ABAC), поскольку он обладает достаточной степенью гранулярности и точностью доступа, к тому же, остальные политики могут быть описаны в терминах и правилах ABAC [27, 28]. Надстройка в виде KBAC также может быть приведена к данному виду [29]. Внутри системы анализа исходные правила контроля доступа представляются в терминах следующей атрибутивной модели безопасности:

- A_O, A_S – множество возможных атрибутов объектов и субъектов соответственно;
- $O = \{o_1, o_2, \ldots, o_k\}$ – множество вершин графа обработки информации в системе, представляющие собой фрагменты данных, включая множество $S = \{s_1, s_2, \ldots, s_n\}$ – множество субъектов системы обработки Больших данных.

Для $\forall o_i \in O$, i = 1, …, k, а также для $\forall s_t \in S$, t = 1, …, n определено множество пар «атрибут-значение»: $\{(a_1, v_1), (a_2, v_2), \ldots, (a_m, v_m)\}$, где $a_j \in A_O \cup A_S$, v_j – значение атрибута.

- $D_{вх} \subseteq O$ – множество входных фрагментов данных;
- $D_{вых} \subseteq O$ – множество выходных фрагментов данных;
- E – множество ребер графа обработки информации в системе, представляющие собой операции над фрагментами данных (агрегация и разделение);
- P – политика безопасности – задается как множество правил $\{P_1, P_2, \ldots, P_z\}$, таких что P_i = {conditionsubj, conditionobj, action, access}, где conditionsubj – множество условий для субъекта, conditionobj – множество условий для объекта (фрагмента), action – действие субъекта по отношению к объекту (чтение, запись, чтение и запись), access – разрешение/запрет на выполнение субъектом действия.

Схема метода анализа контроля доступа при анализе на предмет соблюдения бизнес-логики, от входных данных к выходным, приведена на рис. 1.

Проверка графа жизненного цикла данных в процессе обработки проводится на непротиворечивость, наличие ошибок и циклов. Этот шаг необходим в первую очередь для применения анализа на этапе проектирования, когда граф обработки данных строится на основе проектной документации и загружается в систему. Для анализа действующей системы контроля доступа этот шаг с точки зрения безопасности можно опустить, однако он дополнительно позволяет выявить технические проблемы построения процесса обработки данных и может использоваться как технологический инструмент инженером данных.

Далее последовательно рассматриваются все входные фрагменты, и, порожденные ими промежуточные. Последние также добавляются в список для оценки прав, пока не будут получены итоговые, выходные фрагменты данных, не порождающие новых элементов для рассмотрения. Дополнительным преимуществом такого анализа является возможность выявления ошибок в проектировании системы обработки данных, например, наличие неиспользуемых (излишних) данных в промежуточных хранилищах и др.

Аналогичным образом проводится анализ в обратном направлении, от выходных фрагментов данных к входным. Общий порядок проведения обратного анализа таков:

- Формируется список выходных фрагментов данных и устанавливаются права доступа к ним субъектов, на основе настроек системы или предполагаемой плотики безопасности. Формируется список O_{in}.
- Для каждого результирующего фрагмента из списка O_{in} устанавливается, из каких фрагментов данных он сформирован, формируется список O_{temp}. То есть, на основе графа жизненного цикла данных определяются связи по порождению.
- Для каждого фрагмента данных списка O_{temp} устанавливаются возможные права доступа, на основе обратного анализа правил доступа из политики P.
- Элементы O_{temp} переносятся в список O_{in}, отработанные элементы исключаются из рассмотрения (списка O_{in}) и снова запускается шаг 2.
- Формируются итоговые возможные матрицы доступа для входных данных, производится их сравнение с политикой безопасности и настройками контроля доступа.

В результате данный метод анализа позволяет выявить возможные утечки данных, за счет обнаружения доступа потребителей информации и пользователей системы (субъектов) к исходным данным, являющимся для них конфиденциальными.

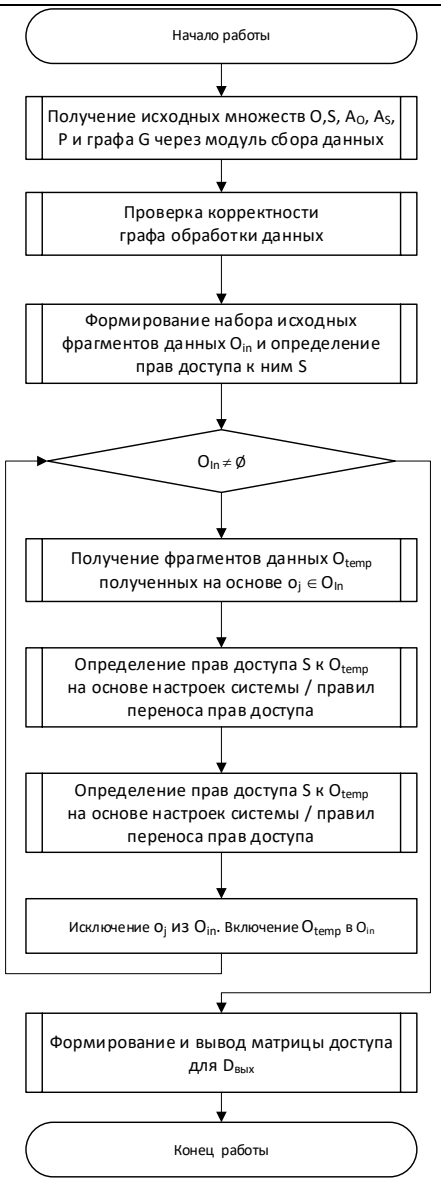

Рис. 1. Метод анализа системы контроля доступа (прямое направление)
Fig. 1. Access control system analysis mechanism (forward direction)

Дополнительно анализ позволяет установить несоответствия требуемым настройкам контроля доступа, и, что является важным, и предложить администратору безопасности возможные варианты иных настроек контроля доступа в инструментах обработки данных полихранилища, отвечающие текущей политике использования выходных данных.

Полученные варианты матриц доступа также могут быть использованы для совершенствования системы.

В то же время, если ни одного удовлетворительного решения не получено, это означает невозможность достичь выполнения требуемой политики безопасности на текущем наборе инструментов с текущим порядком обработки данных. При внесении изменений в эти

компоненты (граф жизненного цикла данных, грануляция и настройка доступа с правилами переноса прав) анализ необходимо повторить.

4. Применение метода анализа системы контроля доступа для гетерогенных систем больших данных

Общая архитектура фреймворка автоматизированного анализа контроля доступа в системах управления большими данными приведена на рис. 2.

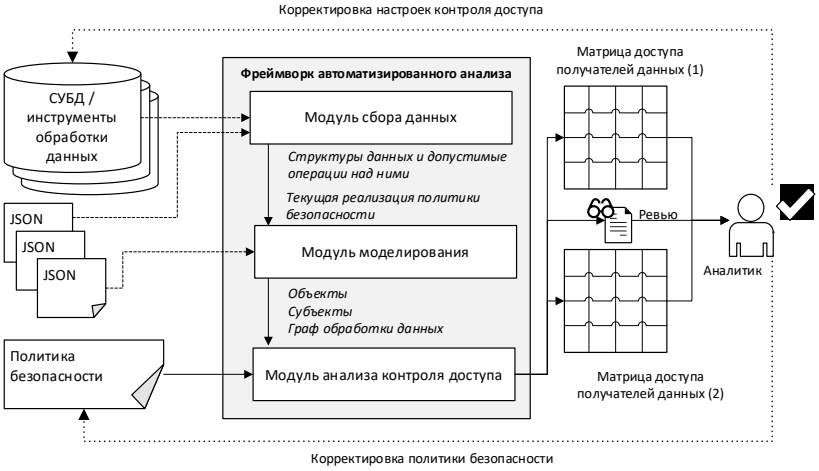

Рис. 2. Архитектура фреймворка автоматизированного анализа контроля доступа в системе больших данных

Fig. 2. Framework architecture for the access control system automation analysis in big data system

Входными данными модуля анализа могут являться как характеристики существующей системы, собранные в автоматическом режиме, так и параметры проектируемой. Выходные данные представляют собой матрицы доступа, полученные на основе имеющейся и желаемой политик безопасности.

Покажем проведение автоматизированного анализа согласно описанному ранее методу на примере. Так как информация в системах больших данных, в том числе та, которая касается контроля доступа и маршрутов обработки данных является конфиденциальной и может быть использована злоумышленниками [30], в статье описан анонимизированный пример системы управления большими данными на основе полихранилища с сокращенным числом пользовательских ролей.

В качестве экспериментального стенда была предложена система обработки данных некоторой организации, состоящей из 4 распределенных филиалов, каждый из которых имеет 4 подразделения: лаборатория, менеджмент, руководство и администратор. Программными компонентами стенда были инструмент потоковой обработки Apache Spark, хранилища данных на основе MongoDB и PostgreSQL. Таким образом, моделировалась гетерогенная среда на с использованием различных моделей данных. Сбор данных о реализованных в СУБД политиках безопасности проводился при помощи интерфейсов запросов и модуля на языке Python. В рамках эксперимента в инструментах обработки использовалась реализованная в них модель RBAC с грануляцией данных в соответствии с возможностями каждого инструмента. Для PostgreSQL – на уровне ячеек (Cell level), для MongoDB – на уровне коллекций (Collection-Level). Отметим, что для MongoDB это наиболее детализованный уровень грануляции доступа. Далее были сформированы таблицы входных фрагментов и субъектов системы вместе с наборами их атрибутов и определена общая

действующая политика безопасности уже в терминах атрибутивной модели. В политику вошли более десяти различных правил разграничения доступа, включая правила на базе предполагаемых настроек доступа конечных пользователей и правила, построенные на основе данных из инструментов обработки. Также сформированы желаемые правила разграничения доступа: желаемые политики безопасности в отношении входных и выходных данных, также в терминах атрибутивной модели. По умолчанию было принято, что если в политике безопасности нет подходящего правила, то доступ запрещен. Кроме того, запрещающие правила находятся в начале политики и имеют приоритет над разрешающими, поэтому если для фрагмента и субъекта на разных этапах жизненного цикла одновременно нашлось запрещающее и разрешающее правила, доступ не предоставляется.

В ходе обработки данных они видоизменяются – объединяются, разделяются и преобразуются. Соответственно, при использовании атрибутивной модели, возникает вопрос наследования не только прав доступа, но и атрибутов в результате этих операций. Если в случае разделения в большинстве случаев достаточно передать атрибуты всем дочерним фрагментам без изменений, то в случае объединения и преобразования все не так просто. В экспериментальной установке в качестве решения для операции объединения предлагается добавлять к атрибуту дополнительную характеристику – тип наследования. Реализованная программа имеет следующие типы наследования атрибутов при порождении новых фрагментов данных:

- max – взятие максимального значения атрибута среди всех родительских;
- min – взятие минимального значения атрибута среди всех родительских;
- concatenate – объединение значений в один список, если они разные.

Правила наследования не относятся к атрибутам субъектов, имеющих доступ к данным, так как наследование прав субъектов в принятой модели безопасности не рассматривается. В итоге была сформирована структура для описания каждого атрибута субъекта или объекта в системе и включения его в решающие правила контроля и переноса доступа. Был сформирован граф жизненного цикла фрагментов данных перехода фрагментов данных (рис. 3). В системе такой граф представлен как матрица смежности с дополнительным хранением весов вершин – типов фрагментов данных и весов ребер – операций над данными.

Пример матрицы доступа, полученной в результате прямого анализа контроля доступа в экспериментальной системе, приведена в табл. 1. Столбцами таблицы являются субъекты системы, а строками – доступ субъектов по отношению к конкретным фрагментам данных. В этой таблице приведены выходные фрагменты 14, 16, 20 и 21.

Полученная в результате анализа матрица доступа прямо указывает на возможное существование нескольких значимых проблем контроля доступа в рассматриваемой системе. Во-первых, если речь идет о полном представлении системы и ее пользователей (а не ее части) отсутствие доступа у всех пользователей к фрагменту 14 говорит о его недоступности для дальнейшего использования. Такой фрагмент (набор данных) должен быть вынесен за пределы системы в архивное хранение или удален. Или же должны быть изменены права доступа кого-либо пользователей в его отношении. Это, очевидно, повлечет фактическое расширение их доступа в отношении входных фрагментов, так как они получат данные на их основе. Такое расширение в свою очередь может нарушить политику безопасности и провести к утечке. Также отметим, что пользователь User 1 не имеет доступа к выходным данным вообще, что для полноценной системы также является не нормальным.

Рассмотрим теперь результат обратного анализа, приведенный в виде примера матрицы доступа из табл. 2. В таблице приведена матрица с минимальными правами доступа, которые должны иметь пользователи к входным фрагментам для получения назначенных им выходных без нарушения политики безопасности.

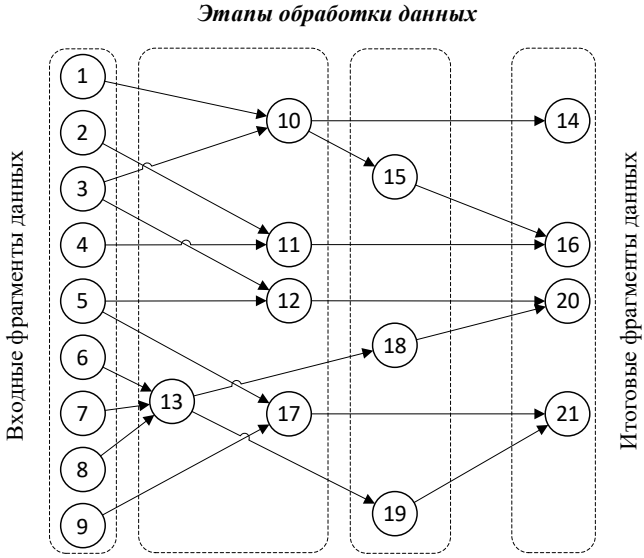

Рис. 3. Граф жизненного цикла фрагментов данных
Fig. 3. Data fragment lifecycle graph

Табл. 1. Матрица доступа для выходных фрагментов (прямой анализ)
Table 1. Access matrix for the output data (forward direction analysis)

ID фрагмента	User 1	User 2	User 3	User 4	User 5
14	Denied	Denied	Denied	Denied	Denied
16	Denied	Read	Read, Write	Read	Denied
20	Denied	Read	Read, Write	Read	Read
21	Denied	Read	Read, Write	Read	Read

Табл. 2. Матрица доступа для входных фрагментов (обратный анализ)
Table 2. Access matrix for the output data (reverse analysis)

ID фрагмента	User 1	User 2	User 3	User 4	User 5
1	Denied	Denied	Denied	Denied	Denied
2	Denied	Read	Denied	Denied	Denied
3	Denied	Denied	Denied	Denied	Denied
4	Denied	Denied	Denied	Read	Denied
5	Denied	Denied	Denied	Denied	Denied
6	Read, Write	Denied	Denied	Denied	Denied
7	Read, Write	Denied	Read, Write	Denied	Denied
8	Denied	Denied	Denied	Denied	Read
9	Denied	Denied	Read, Write	Denied	Denied

Дальнейшей задачей аналитика безопасности является сопоставление полученной матрицы с политикой безопасности организации для обнаружения возможных утечек. Например, действительно ли (User 5) должен иметь доступ на чтение документа 8.

Таким образом, представленный способ автоматизированного анализа контроля доступа в системах управления большими данными позволяет проводить их оценку как на предмет возможных утечек, так и на предмет нарушения бизнес-логики. Анализ может быть проведен на разных этапах жизненного цикла системы, от проектирования до эксплуатации.

5. Анализ полученных результатов

В результате работы авторами выделены основные проблемы применения отдельных методов контроля доступа в системах больших данных. На их основе впервые явно сформулировано основное противоречие между гибкостью и динамичностью настроек контроля доступа в системах больших данных и ростом нагрузки на администратора безопасности, вызванной, в том числе, наличием гетерогенных компонентов полихранилищ в системе обработки информации, не позволяющих эффективно автоматизировать анализ корректности разграничения доступа. Как показано в приведенном обзоре, сегодня эта задача является достаточно новой и даже на уровне исследовательских работ решена только частично: для гомогенных хранилищ экосистемы Hadoop в рамках одной модели контроля доступа. Решения для полихранилищ, кроме приведенного авторами в данной работе, пока не представлено.

Для решения выявленного противоречия авторами предложен новый метод автоматизированного анализа контроля доступа на основе исходной и желаемой политик безопасности и графа обработки данных в системе. Метод включает двухпроходный анализ правил доступа в прямом и обратном направлении над графом обработки данных. Отметим, что результате работы одного цикла разработанного метода для дальнейшей оценки администратору безопасности предоставляются матрицы доступа, позволяющие оценить несколько характеристик системы контроля доступа. При этом

- прямой анализ системы контроля доступа от входных данных к выходным формирует матрицу доступа, позволяющую оценить соблюдение бизнес-логики в системе на основе желаемой политики безопасности;
- обратный анализ от выходных данных к входным позволяет оценить возможность утечек данных в результате логического вывода на основе семантической связности фрагментов данных, проанализировать выполнение принципа минимизации прав доступа.

Итерационное согласование результатов прямого и обратного анализа путем корректировки политик безопасности либо настроек переноса прав доступа отдельных инструментов позволяет администратору (аналитику) безопасности получить рациональную политику контроля доступа для системы в целом, которая может быть также автоматически перенесена в правила конкретных инструментов. Тот факт, что итерационный процесс анализа, за исключением этапа сбора данных, проводится на модели, позволяет снизить нагрузку и избежать излишнего вмешательства в функционирование системы. Отметим, что сбор данных для проведения анализа над действующей системой больших данных, тем не менее, не может проводиться без нагрузки на базовую систему, так как требуется выполнить построение графовой модели обработки информации на основе перемещения фрагментов данных.

Представленный метод и фреймворк автоматизации анализа контроля доступа в системах больших данных в первую очередь позволяет снизить затраты администратора безопасности на сбор информации о контроле доступа в системе больших данных и ручной анализ переноса прав доступа между инструментами обработки. Точный объем времени, сэкономленного в результате автоматизации, к сожалению, не может быть рассчитан, так как на него в значительной степени влияет сложность конкретной системы больших данных и жизненного цикла информации в ней, число типов пользователей системы, число типов разнородных фрагментов данных. Тем не менее, можно говорить о сокращении времени на сбор данных и, главное, на поиск рациональной политики безопасности. Для анонимизированной системы, приведенной в примере данной работы, это время составило несколько часов.

Важно сказать, что представленный метод и фреймворк может быть интегрирован в систему больших данных вместе с другими решениями, обеспечивающими контроль доступа гомогенных компонентов и подсистем обработки информации, описанными ранее в обзоре.

Возможность работы с гетерогенными полихранилищами, в компонентах которых реализованы собственные системы контроля доступа, является его ключевым отличием и преимуществом. В рамках представленного эксперимента использовались СУБД с различной структуризацией данных (пост-реляционная и документоориентированная) относящиеся к разным производителям. Расширение решения на другие классы и конкретные экземпляры систем возможно только путем добавления соответствующих функциональных компонентов к модулю сбора данных, так как организация системных каталогов и диалекты языков запросов в различных решениях существенно отличаются. Также может потребоваться корректировка отображения правил разграничения доступа из инструмента обработки данных в атрибутивную модель. С научной точки зрения некоторую сложность все еще представляет задача построения максимально детализированного графа обработки данных для произвольных хранилищ информации. Ее решение требует интеграции компонентов аудита узлов обработки данных (например, аудита на основе распределенного реестра [26]) с внутренними преобразованиями данных в СУБД при выполнении запросов. Последние могут быть получены на основе анализа планов выполнения запросов реляционных систем и Map-Reduce конвейеров в не реляционных решениях. Сбор таких данных в современных СУБД также хорошо поддается автоматизации, однако требует индивидуальных модулей интеграции с каждым отдельным типом или даже версией системы. «Бесшовное» совмещение таких планов с результатами аудита движения данных между узлами – обработчиками представляется актуальной задачей для дальнейшей работы.

Представленный метод помимо научной новизны обладает следующими преимуществами с точки зрения практической реализации:

- Метод применим для автоматизированного анализа контроля доступа систем управления большими данными на разных стадиях жизненного цикла: как на этапе проектирования, так и на этапе эксплуатации.
- Реализации отельных шагов метода могут быть применены для решения смежных задач, например, анализа качества системы управления большими данными.

Определенными ограничениями применения представленного решения на сегодняшний день является необходимость разработки автоматического модуля сбора данных для каждого инструмента обработки данных, применяемого в целевой системе; все еще относительно высокие требования к квалификации администратора безопасности, и, отчасти, относительно высокая нагрузка прим сборе информации с действующей системы обработки больших данных.

Поэтому дальнейшее развитие представленной работы заключается в повышении степени автоматизации работы путем интеграции с различными типами хранилищ данных, совершенствование алгоритмов и методов сбора и анализа информации. Авторам также представляется рациональной разработка методов поддержки принятия решений при анализе политик безопасности и автоматизация процесса поиска рационального решения на основе методов многокритериальной дискретной оптимизации и алгебры логики.

6. Заключение

Обеспечение безопасности больших данных, даже на одном из уровней рассмотрения, представляет собой сложную и комплексную техническую задачу. Для систем управления большими данными, на уровне обработки информации, она заключается в первую очередь в проблеме преодоления неоднородности и несогласованности программных инструментов, которая усугубляется требованиями бизнес-логики (например, гибкость), масштабом систем и объемов данных, отсутствием стандартов и готовых подходов.

Задача автоматизации анализа контроля доступа является составной частью данной общей проблемы. Получение настроек доступа из инструментов обработки данных и автоматическая их установка в заданные значения – хорошо автоматизированная задача,

тогда как поиск рациональных настроек доступа, в свою очередь, крайне сложен и трудозатратен. Ключевое требование для его реализации – обеспечение согласованности контроля доступа между всеми инструментами, задействованными в процессе обработки данных, в данной работе достигается через модель жизненного цикла фрагментов данных в системе.

Предложенный двухпроходный метод анализа позволяет итерационно достигать рациональных настроек контроля доступа и облегчает проверку соответствия разграничения доступа в реальной системе и требуемой политики безопасности. Анализ в обратном направлении – от получателей к источникам – позволяет установить несоответствия и возможные утечки данных из-за нарушения принципа минимизации привилегий. Анализ в прямом направлении – от источников к получателям в свою очередь дает возможность ускорить поверку соответствия бизнес-логике использования данных.

Предложенный метод и программный прототип средства автоматизации анализа контроля доступа в системах управления большими данными может быть использован в дальнейшем как базис для построения целого ряда научно-технических решений, включая аудит данного класса систем и оценку их защищенности, поиск нарушителя и других.

Список литературы / References

[1]. Mushtaq M. S. et al. Security, integrity, and privacy of cloud computing and big data //Security and Privacy Trends in Cloud Computing and Big Data. – 2022. – pp. 19-51.

[2]. Yung L. R. B., Ströele V., Dantas M. A. R. A Polystore Proposed Environment Supported by an Edge-Fog Infrastructure //International Conference on Advanced Information Networking and Applications. – Cham : Springer International Publishing, 2023. – pp. 292-302. doi: 10.1007/978-3-031-28451-9_26.

[3]. Gao J. Analysis of enterprise financial accounting information manage-ment from the perspective of big data //International Journal of Science and Research (IJSR). – 2022. – vol. 11. – №. 5. – pp. 1272-1276.

[4]. Vasa J., Thakkar A. Deep learning: Differential privacy preservation in the era of big data //Journal of Computer Information Systems. – 2023. – vol. 63. – №. 3. – pp. 608-631. doi: 10.1080/08874417.2022.2089775.

[5]. Dhiman G. et al. Federated learning approach to protect healthcare data over big data scenario //Sustainability. – 2022. – vol. 14. – №. 5. – pp. 1-14. doi: 10.3390/su14052500.

[6]. Strzelecki A., Rizun M. Consumers' Change in Trust and Security after a Personal Data Breach in Online Shopping //Sustainability. – 2022. – vol. 14. – №. 10. – pp. 1-17. doi: 10.3390/su14105866.

[7]. Zhuang Y. et al. Research on big data access control mechanism //International Journal of Computational Science and Engineering. – 2023. – vol. 26. – №. 2. – pp. 192-198. doi: 10.1504/IJCSE.2023.129738.

[8]. Jiang R. et al. T-RBAC Model Based on Two-Dimensional Dynamic Trust Evaluation under Medical Big Data //Wireless Communications and Mobile Computing. – 2021. – vol. 2021. – pp. 1-17. doi: 10.1155/2021/9957214.

[9]. Gupta M., Patwa F., Sandhu R. Object-tagged RBAC model for the Hadoop ecosystem //IFIP Annual Conference on Data and Applications Security and Privacy. – Cham : Springer International Publishing, 2017. – pp. 63-81. doi: 10.1007/978-3-319-61176-1_4.

[10]. Servos D., Osborn S. L. Current research and open problems in attribute-based access control //ACM Computing Surveys (CSUR). – 2017. – vol. 49. – №. 4. – pp. 1-45. doi: 10.1145/3007204.

[11]. Zeng W., Yang Y., Luo B. Content-based access control: Use data content to assist access control for large-scale content-centric databases //2014 IEEE International Conference on Big Data (Big Data). – IEEE, 2014. – pp. 701-710. doi: 10.1109/BigData.2014.7004294.

[12]. El Haourani L., Elkalam A. A., Ouahman A. A. Knowledge Based Access Control a model for security and privacy in the Big Data //Proceedings of the 3rd International Conference on Smart City Applications. – 2018. – pp. 1-8. doi: 10.1145/3286606.3286793.

[13]. Anisetti M. et al. Dynamic and scalable enforcement of access control policies for big data //Proceedings of the 13th International Conference on Management of Digital EcoSystems. – 2021. – pp. 71-78. doi: 10.1145/3444757.3485107.

[14]. Tall A. M., Zou C. C. A Framework for Attribute-Based Access Control in Processing Big Data with Multiple Sensitivities //Applied Sciences. – 2023. – vol. 13. – №. 2. – pp. 1-28. doi: 10.3390/app13021183.

[15]. Colombo P., Ferrari E. Access control technologies for Big Data management systems: literature review and future trends //Cybersecurity. – 2019. – vol. 2. – №. 1. – pp. 1-13. doi: 10.1186/s42400-018-0020-9.
[16]. Muneeshwari P., Athisha G. Extended artificial immune system–based optimized access control for big data on a cloud environment //International Journal of Communication Systems. – 2020. – vol. 33. – №. 13. – p. e3947. pp. 1-15. doi: 10.1002/dac.3947.
[17]. Mounnan O., Abou El Kalam A., El Haourani L. Decentralized access control infrastructure using blockchain for big data //2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA). – IEEE, 2019. – pp. 1-8. doi: 10.1109/AICCSA47632.2019.9035221.
[18]. Vijayalakshmi K., Jayalakshmi V. Shared access control models for big data: a perspective study and analysis //Proceedings of International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2020. – Springer Singapore, 2021. – pp. 397-410. doi: 10.1007/978-981-15-8443-5_33
[19]. Hu V. C. et al. An access control scheme for big data processing //10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing. – IEEE, 2014. – pp. 1-7. doi: 10.4108/icst.collaboratecom.2014.257649.
[20]. Oussous A. et al. Big Data technologies: A survey //Journal of King Saud University – Computer and Information Sciences. – 2018. – vol. 30. – №. 4. – pp. 431-448. doi: 10.1016/j.jksuci.2017.06.001.
[21]. Centonze P. Security and Privacy Frameworks for Access Control Big Data Systems //Computers, Materials & Continua. – 2019. – vol. 59. – №. 2. – pp. 361-374.
[22]. Dziedzic A., Elmore A. J., Stonebraker M. Data transformation and migration in polystores //2016 IEEE High Performance Extreme Computing Conference (HPEC). – IEEE, 2016. – pp. 1-6. doi: 10.1109/HPEC.2016.7761594.
[23]. Kroll J. A., Kohli N., Laskowski P. Privacy and policy in polystores: a data management research agenda //Heterogeneous Data Management, Polystores, and Analytics for Healthcare: VLDB 2019 Workshops, Poly and DMAH, Los Angeles, CA, USA, August 30, 2019, Revised Selected Papers 5. – Springer International Publishing, 2019. – pp. 68-81. doi: 10.1007/978-3-030-33752-0_5.
[24]. Poudel M. et al. Processing analytical queries over polystore system for a large astronomy data repository //Applied Sciences. – 2022. – vol. 12. – №. 5. – pp. 1-23. doi: 10.3390/app12052663.
[25]. Poltavtseva, M. A. Modeling Big Data Management Systems in Information Security / M. A. Poltavtseva, M. O. Kalinin // Automatic Control and Computer Sciences. – 2019. – vol. 53, No. 8. – pp. 895-902. doi: 10.3103/S014641161908025X.
[26]. Poltavtseva M. A. et al. Data protection in heterogeneous big data systems //Journal of Computer Virology and Hacking Techniques. – 2023. – pp. 1-8. doi: 10.1007/s11416-023-00472-3.
[27]. Sahani G., Thaker C., Shah S. Supervised Learning-Based Approach Mining ABAC Rules from Existing RBAC Enabled Systems //EAI Endorsed Transactions on Scalable Information Systems. – 2022. – vol. 10. – №. 1. – pp. 1-8. doi: 10.4108/eetsis.v5i16.1560.
[28]. Talegaon S. et al. Contemporaneous Update and Enforcement of ABAC Policies //Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies. – 2022. – pp. 31-42. doi: 10.1145/3532105.3535021.
[29]. Gupta T., Sural S. Ontology-based Evaluation of ABAC Policies for Inter-Organizational Resource Sharing //Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics. – 2023. – pp. 85-94. doi: 10.1145/3579987.3586572.
[30]. Yang K. et al. An Efficient and Fine-Grained Big Data Access Control Scheme With Privacy-Preserving Policy // IEEE Internet of Things Journal. – vol. 4. – № 2. – p. 563-571. doi: 10.1109/JIOT.2016.2571718.

Информация об авторах / Information about authors

Мария Анатольевна ПОЛТАВЦЕВА – доктор технических наук, доцент, профессор института кибербезопасности и защиты информации, федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого». Область научных интересов: информационная безопасность систем хранения данных и СУБД, безопасность больших данных, сбор, обработка и анализ данных в кибербезопасности, моделирование данных и процессов, мониторинг информационной безопасности крупномасштабных систем.

Maria Anatolyevna POLTAVTSEVA – Dr. Sci. (Tech.), Associate Professor, Professor of the Institute of Cyber Security and Information Protection, Federal State Autonomous Educational Institution of Higher Education "Peter the Great St. Petersburg Polytechnic University". Research

interests: information security of data storage systems and DBMS, big data security, data collection, processing and analysis in cybersecurity, modeling of data and processes, monitoring of information security of large-scale systems.

Максим Олегович КАЛИНИН – доктор технических наук, профессор, профессор института кибербезопасности и защиты информации, федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого». Область научных интересов: информационная безопасность операционных систем, гипервизоров, сред виртуализации и облачных вычислений; методы искусственного интеллекта в кибербезопасности и безопасность интеллектуальных систем

Maxim Olegovych KALININ – Dr. Sci. (Tech.), Professor, Professor of the Institute of Cyber Security and Information Protection, Federal State Autonomous Educational Institution of Higher Education "Peter the Great St. Petersburg Polytechnic University". Research interests: information security of operating systems, hypervisors, virtualization environments and cloud computing; methods of artificial intelligence in cybersecurity and security of intelligent systems.

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

Подходы к разработке системы обнаружения дефектов печатных плат на основе технологии АОИ

Т.С. Ходатаева, ORCID: 0000-0002-6284-2292, <khodataeva_ts@marsu.ru>
Н.В. Каширин, ORCID: 0000-0002-3268-254X, <kachnic@mail.ru>
А.И. Аверина, ORCID: 0000-0002-3412-9641, <irene75@inbox.ru>
А.Е. Гурьянов, ORCID: 0000-0003-4138-5640, <artem-guryanov-00@mail.ru>
ФГБОУ ВО Марийский государственный университет,
109004, Россия, г. Йошкар-Ола, пл. Ленина, д. 1.

Аннотация. Рассматриваются некоторые современные подходы обнаружения дефектов печатных плат на основе автоматической оптической инспекции с целью проектирования собственной системы контроля производства. Важность процесса контроля растет в связи с ужесточением требований, предъявляемых современными производственными процессами. На предприятиях массового производства электроники предпринимаются попытки добиться высокого качества всех деталей, узлов и готовой продукции. Система оптической инспекции является одним из наиболее важных инструментов автоматизации визуального контроля печатных схем. Помимо обеспечения экономической эффективности и контроля качества продукции, автоматизированная система контроля также может собирать статистическую информацию для осуществления обратной связи с производственным процессом. В обзоре рассматриваются алгоритмы и методы автоматизированного оптического контроля проводящего рисунка на поверхности печатных плат с целью нахождения оптимального метода обнаружения дефектов.

Ключевые слова: автоматическая оптическая инспекция; обработка изображений; сверточные нейронные сети.

Для цитирования: Ходатаева Т.С., Каширин Н.В., Аверина А.И., Гурьянов А.Е. Подходы к разработке системы обнаружения дефектов печатных плат на основе технологии АОИ. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 109–120. DOI: 10.15514/ISPRAS–2023–35(4)–5.

Благодарности: Работа выполняется в рамках государственного задания на оказание государственных услуг (выполнение работ) № 075-01252-22-03 от 26.10.2022 на базе ФГБОУ ВО Марийского государственного университета в сотрудничестве с ведущим предприятием по производству печатных плат «ТЕХНОТЕХ» (сайт https://tehnoteh.ru/) в г. Йошкар-Ола.

Approaches to the Development of a Printed Circuit Board Defect Detection System Based on AOI Technology

T.S. Khodataeva, ORCID: 0000-0002-6284-2292<*khodataeva_ts@marsu.ru*>
N.V. Kashirin, ORCID: 0000-0002-3268-254X <*kachnic@mail.ru*>
A.I. Averina, ORCID: 0000-0002-3412-9641 <*irene75@inbox.ru*>
A.E. Guryanov, ORCID: 0000-0003-4138-5640 <*artem-guryanov-00@mail.ru*>
Mari State University, 1, pl. Lenina, Yoshkar-Ola, 424000, Russia.

Abstract. Some modern approaches to detecting defects in printed circuit boards based on automatic optical inspection are considered in order to design their own control system. The importance of the control process is growing in connection with the tightening of the requirements imposed by modern production processes. At the enterprises of mass production of electronics, attempts are being made to achieve high quality of all parts, assemblies and finished products. The optical inspection system is one of the most important tools for automating the visual inspection of printed circuits. In addition to ensuring cost efficiency and product quality control, an automated control system can also collect statistical information to provide feedback to the production process. The review considers algorithms and methods for automated optical control of the conductive pattern on the surface of printed circuit boards in order to find the optimal method for detecting defects.

Keywords: automatic optical inspection; image processing; convolutional neural networks.

For citation: Khodataeva T.S., Kashirin N.V., Averina A.I., Guryanov A.E. Approaches to the development of a printed circuit board defect detection system based on AOI technology**.** *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 4, 2023. pp. 109-120 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-5.

Acknowledgements. The work is carried out within the framework of the state task for the provision of public services (performance of work) No. 075-01252-22-03 dated 10/26/2022 on the basis of the Federal State Budgetary Educational Institution of Higher Education of the Mari State University in cooperation with the leading enterprise for the production of printed circuit boards "TECHNOTECH" (website https://tehnoteh.ru/) in Yoshkar-Ola.

1. Введение

Исходя из современного развития геополитической ситуации, усиления санкционного давления, распространения вызовов и угроз глобальной экономики, нашим государством был взят курс на создание собственных современных производств, которые смогут конкурировать с иностранными компаниями и выпускать товары, которые вытеснят зарубежные аналоги. Государство объявило о масштабных мерах господдержки технологического суверенитета.

В связи возникшей проблемой импортозамещения в Марийском государственном университете проводится исследование, ориентированное на поиск методов автоматической оптической инспекции (АОИ) для практического применения. Работа над проектом проводится в сотрудничестве с ведущим предприятием по производству печатных плат «ТЕХНОТЕХ» (https://tehnoteh.ru/) в г. Йошкар-Ола.

Производство печатных плат включает в себя множество технологических процессов. Сложность соблюдения технологического процесса при изготовлении печатных плат иногда приводит к браку продукции. Производители электроники придают большое значение раннему и точному обнаружению дефектов [1]. В настоящее время методы обнаружения дефектов можно разделить на три основных типа: визуальный контроль, тестирование электрических характеристик и АОИ.

В настоящее время тенденции к миниатюризации радиоэлектронных изделий приводят к необходимости создания все более малогабаритных электронных схем. Одним из способов

уменьшения размеров печатных плат является более плотное расположение её токопроводящих частей.

Простой визуальный контроль, выполняемый персоналом производственных предприятий, не может справиться с задачей быстрого и точного обнаружения дефектов, поэтому приоритетным направлением контроля при производстве печатных плат является технология АОИ. Это быстрый метод обнаружения дефектов, удовлетворяющий потребности высокоскоростной производственной линии.

С наступлением эры информационных технологий большинство дорогостоящих программных продуктов автоматического оптического контроля становятся дешевле и доступнее. Компьютерное зрение, обработка изображений, глубокое обучение и технологии искусственного интеллекта с открытым исходным кодом стали главными инструментами разработчиков.

Конечной целью развиваемого проекта является выработка требований, проведение технического проектирования и создание системы контроля проводящего рисунка на поверхности печатных плат с использованием относительно недорогого оборудования.

2. Обзор литературы

К настоящему времени в области обработки изображений было проведено и опубликовано значительное количество фундаментальных и прикладных исследований [2, 3] с целью использования технологии обработки изображений для обнаружения дефектов печатных плат.

В работе [4] представлен всесторонний обзор различных систем АОИ, используемых в электронике, микроэлектронике и оптоэлектронике. Алгоритмы проверки, используемые для обнаружения дефектов в электронных компонентах, обсуждаются с точки зрения инструментов предварительной обработки, выделения признаков и классификации. В работе так же рассматриваются недавние статьи, в которых использовались алгоритмы глубокого обучения. Статья завершается выделением текущих тенденций и будущих направлений исследований.

Методы проверки печатных плат в целом можно разделить на три категории [5, 6]: метод сравнения с эталоном, метод проверки правил проектирования (не эталонный) и гибридный подход. Возможности этих методов различаются. В следующем разделе будет сделан краткий обзор обозначенных методов. Особого внимания заслуживают исследования и реализация методов АОИ, усовершенствованных за счет их интеграции с методами машинного обучения.

Во всех рассмотренных публикациях для идентификации выбирались в основном следующие восемь категорий дефектов (Рис. 1): короткие замыкания, обрывы, царапины, выступы, медные включения (островки), микроотверстия (проколы), отклонение толщины проводников и смещение отверстий. На указанные восемь категорий дефектов приходится более 80% поверхностных дефектов, фиксируемых на заводах по производству печатных плат.

2.1 Метод сравнения с эталоном

Метод сравнения с эталоном кажется наиболее простым для реализации. В его основе лежит сравнение изображений эталонной и тестируемой плат. Для этого используется вычитание изображений для выявления их попиксельного различия. Данный метод с разными модификациями представлен во многих работах [7, 8, 9].

В работе [7] автор, применив метод вычитания изображения для обнаружения дефектов печатной платы, добился обнаружения типичных дефектов, таких как чрезмерное травление (например, разрывы), недостаточное травление (например, короткие замыкания) и

отсутствие отверстия. С помощью последующих процедур обработки изображений также были определены точные положения и размеры дефектов.

В публикации [8] авторы также использовали метод вычитания изображения для обнаружения дефектов и доработали алгоритм классификации дефектов, применив алгоритм заливки однородных областей. Они смогли классифицировать различные дефекты, такие как дефекты травления, дефекты отсутствующих отверстий, неправильный размер отверстий, отсутствующие элементы и линии разрыва.

Автор статьи [9] предложил аналогичный метод обнаружения дефектов на печатных платах путем сравнения эталонного изображения с проверенным изображением. Модификация заключалась в том, что стандартное изображение представляло собой среднее значение серии изображений эталонных печатных плат вместо использования только одного изображения. Этот подход к получению эталонного изображения дал возможность распознавать различные типы дефектов, а не только чрезмерное или недостаточное травление.

Рис. 1. Категории дефектов
Fig. 1. Categories of defects

В работе [10] предложен метод сравнения на основе кластеризации для обнаружения дефектов печатных платах с медным покрытием. Небольшое смещение проводящих дорожек и контактов считается дефектом в традиционном методе вычитания. В статье введен допуск для контроля разницы между эталонной печатной платой и образцом.

В публикации [11] авторы реализовали идею о том, что ряд дефектов возникают только на определенных участках тестового изображения, например, отверстие неправильного размера или отсутствие отверстия для сегмента с отверстием, отсутствие проводника и обрыв цепи для сегмента тонкой линии. При помощи математической морфологии эталонное и тестовое изображение было сегментировано на четыре основных области содержащие либо квадратные элементы, либо круги, либо тонкие, либо толстые линии. Квадратный сегмент содержал изображение квадратных контактных площадок, сегмент с кругами – изображение контактных площадок с отверстиями, сегмент с толстой линией содержал изображение толстых проводников, а сегмент с тонкой линией содержал изображения тонких проводников. Для вычитания изображений использовалась операция исключающего ИЛИ (XOR). Для каждого сегмента проводилась идентификация и классификация дефектов. Предложенный метод позволил идентифицировать 13 видов дефектов из 14.

Операция сравнения изображений имеет существенные ограничения. Поскольку методология в основном реализует попиксельное сравнение, любое смещение, разница в разрешении, разница в размерах изображения и разница в условиях освещения между шаблонным изображением и дефектным изображением влияет на точность и производительность системы. Более того, получение действительно качественного эталонного изображения является сложной задачей и требует больших затрат рабочего времени.

2.2 Метод проверки правил проектирования (не эталонный)

Методы проверки по правилам проектирования, представляет собой метод не требующий изображения эталонной платы для помощи в обнаружении дефектов. Такой подход работает либо на допущении, что элементы представляют собой простые геометрические формы, а дефекты представляют собой неожиданные неправильные элементы, либо на прямой проверке правил проектирования. В системах автоматизированного проектирования (САПР) радиоэлектронных средств по завершению проектирования печатной платы предусмотрено формирование gerber-файлов [12]. Формат Gerber предназначен для преобразования в него электронной модели печатной платы и передачи на производство.

Информация о топологии проводящего рисунка печатной платы – ширина проводника, расстояние между проводниками, размер и форма контактных площадок, диаметр монтажных отверстий и т. д. – это некоторые из правил проектирования, которые извлекаются из gerber-файла и используются при реализации этого подхода.

Авторы работы [13] предложили метод который позволяет обнаруживать дефекты без рассмотрения эталонной платы. В качестве основного метода проверки для поиска дефектов печатных плат был выбран метод сравнения изображений методом вычитания, причем эталонное изображение восстанавливается из файла используемой САПР. Информация о компонентах, таких как контактные площадки и дорожки, также извлекается из этого файла. Полученное эталонное изображение модифицируется с учётом правил проектирования печатных плат, при этом задается разумный допуск для процесса вычитания. К изображению тестируемой печатной платы авторы применяли методы бинаризации и математической морфологии, сохраняя все полученные варианты изображения. Затем ими подсчитывалось количество объектов на разных вариантах изображения. В случае выявления отличий тестируемое изображение подготавливали к локализации дефекта. Авторам удалось определять дефекты связанные с технологией нанесения рисунка методом травления меди.

В работе [14] представлен метод сравнения без эталона, основанный на предопределенном правиле проектирования печатной платы для определения наличия дефекта в изображении с использованием математической морфологии. Авторы предложили концепцию таблицы с информацией о связях между элементами рисунка печатной платы. Для предварительной обработки изображения тестируемой платы был выбран метод медианной фильтрации, для сегментации изображения использовали многопороговую операцию сегментации. Прореживание изображения до ширины в один пиксель вдоль центральной оси набора пикселей, также называемой скелетированием, осуществляли с помощью бинарной морфологической операции «hit-and-miss». Важно, что прореживание поддерживает связность объекта и сохраняет его отверстия (ни одно из них не удаляется и не добавляется). После предварительной обработки на основании данных таблицы анализировали особенности структуры изображения такие как элементы положения (контактной площадки, положение конечной точки проводника и т. д.) и элементы формы (диаметр сквозного отверстия, состояние соединения контактной площадки и т. д.). Авторам удалось выявить обычные, часто встречающиеся дефекты – короткое замыкание, обрыв цепи, потеря сквозного отверстия, нарушение ширины проводника.

Этот подход не требует точного выравнивания, но может пропустить большие дефекты и искаженные элементы.

2.3 Гибридные методы

Поскольку метод сравнения с эталоном и метод проверки правил проектирования могут дополнять друг друга, гибридные методы, как правило, обеспечивают лучшие результаты обнаружения среди существующих подходов, но требуют больше рабочего времени.

Система контроля, предложенная в публикации [15] использует гибридный метод обнаружения дефектов, основанный на методах сопоставления с шаблоном и методом граничного анализа. Для нахождения дефектов проводников используется алгоритм анализа границ. Сначала определяются области, которые могут иметь потенциальные дефекты. Эти области помечаются как нестандартные края и сопоставляются с шаблоном для измерения ширины проводника. Таким образом, этот метод позволяет значительно увеличить скорость алгоритма обнаружения по шаблону, проводя измерения проводника только в тех местах, которые могут быть дефектами. Точно так же алгоритм обнаружения шаблонов измеряет ширину площадок для дефектов отверстий после определения их центров с помощью метода вычитания изображения.

Авторы статьи [16] поставили цель использовать преимущества вейвлет-преобразований и изображения с несколькими разрешениями для сокращения времени проверки в приложениях для промышленного контроля печатных плат.

2.4 Методы машинного обучения и свёрточные нейронные сети

В дополнение к традиционным методам обработки изображений в системы АОИ для повышения точности и скорости обнаружения дефектов интегрируются алгоритмы машинного обучения, такие как метод опорных векторов (SVM) [17], нейронные сети (NN) [18, 19], генетический алгоритм (GA) [20].

Начиная с 2012 года, когда была предложена архитектура нейронной сети AlexNet [21], в алгоритмы, основанные на глубоком обучении [22], были внесены значительные улучшения в обнаружении объектов. В 2015 был предложен подход, существенно повлиявший на последующие исследования и разработки в этой области – модель R-CNN [23]. R-CNN модель представляет собой двухэтапный алгоритм классификации и обнаружения объектов. Последующие улучшения в виде Fast R-CNN [24] и Faster R-CNN [25] сделали его одним из самых точных подходов, что стало причиной его активного использования. В 2016 году был предложен алгоритм обнаружения объектов YOLO [26]. В том же году появился алгоритм обнаружения объектов SSD (Single Shot MultiBox Detector) [27]. Алгоритм в нейронных сетях архитектуры YOLO и SSD — это одношаговый алгоритм обнаружения объектов, который напрямую обнаруживает объекты с помощью регрессии. Эти идеи нашли своё применение в дефектоскопии печатных плат.

В работе [28] авторы выдвинули новую идею архитектуры глубокой сверточной нейронной сети под названием PartsNet, объединяющей традиционную обработку признаков и глубокое обучение для обнаружения дефектов деталей автомобильных двигателей. Они также построили уточняющую сеть, состоящую из нескольких типичных традиционных методов для улучшения способности к адаптации и достижения сквозного обучения. Таким образом, PartsNet использовал сильные стороны типичных методов обработки признаков, таких как срезы по плотности, сегментация областей и фильтрация областей, для преодоления слабости глубоких сверточных сетей при обнаружении небольших дефектных областей.

В исследовании [29] авторы разработали «крошечную» сеть обнаружения дефектов для печатных плат, названную TDD-net (tiny defect detection network), основанную на подходе Faster R-CNN. Традиционная сеть вида Faster R-CNN, которая создает якоря с использованием трех масштабов и трех различных соотношений, не подходит для обнаружения мелких дефектов. Для решения этой проблемы авторы применили подход, реализованный в системе YOLOv2 [30] – кластеризацию k-средних на обучающей выборке

окаймляющих прямоугольников для автоматического поиска хороших начальных приближений.

Нейронная сеть ResNet-101 [31] была использована в качестве опорной сети для извлечения признаков. Для этого она прошла предварительное обучение на наборе классификации ImageNet [32] и настроена на наборе данных дефектных печатных плат. Как правило, нейронная сеть, инициализированная весами из сети, предварительно обученной на большом наборе данных, таком как ImageNet, показывает лучшую производительность, чем обученные с нуля сети на небольшом наборе данных. Такой подход был предложен в работе [33] и активно используется для обучения нейронных сетей с ограниченным набором данных. В этой работе было достигнуто хорошее обнаружение дефектов по метрике mAP, благодаря использованию архитектуры FPN (Feature Pyramid Networks) [34] – объединению карт признаков разного масштаба.

В исследовании [35] для обнаружения дефектов на компонентах печатной платы была выбрана сеть tiny-YOLOv2, так как сети с архитектурой YOLO обеспечивают большие точность и скорость детектирования объектов на изображениях. Tiny-YOLOv2, основана на эталонной сети Darknet-19, в которой некоторые свёрточные слои удалены, и, кроме того, за каждым сверточным слоем добавлен слои пакетной нормализации для увеличения скорости сходимости модели. Тем самым было достигнуто увеличение скорости работы tiny-YOLOv2 при детектировании и классификации объектов. В работе была проведена идентификация 11 типов дефектов. Обучающая выборка содержала 11000 изображений, предоставленных инженерами по контролю качества. Достигнута точность обнаружения косметических дефектов печатных плат в 98,82%. Общепринятые метрики, которые используются для оценки качества работы нейронных сетей такие как F-score и mAP в статье отсутствуют. Авторами отмечены сложности при обучении сети несбалансированными данными.

Авторы публикации [36] предложили переделать сетевые структуры с учётом особенностей изображений дефектов на печатных платах. Все усилия были направлены на то, чтобы не потерять мелкие объекты, которыми являются дефекты печатных плат. Авторы предложили новую архитектуру сети и использовали различные модули: Faster R-CNN в качестве детектора, ResNet-50 в качестве магистрали. Они применили структуру пирамидальной сети признаков FPN в части извлечения признаков для объединения глубоких и мелких признаков. Для прогнозирования более точных привязок RPN (Region Proposal Networks) заменили на GARPN (Guided Anchoring Region Proposal Network) [37], уменьшение вычислений всей сети достигли за счёт добавления ShuffleNetV2 [38]. Для классификации и выполнения регрессии при определении ограничивающей рамки области дефекта использовались полносвязные слои сети. Проблему недостатка изображений для обучения сети авторы решили аугментацией данных. Сеть была обучена на распознавании шести распространенных типов дефектов печатных плат: обрыв цепи, короткое замыкание, перетравление, шпора, точечное отверстие, шарик припоя. Разработанный подход позволил достичь точности mAP на уровне 94,2% при низкой скорости обнаружения.

В этой статье [39] авторы работали над проблемой обнаружения дефектов печатных плат. Они поставили перед собой задачу создания детектора с высокой точностью обнаружения, высокой скоростью обнаружения, низким потреблением памяти и малым количеством операций умножения-сложения. В исследовании был предложен экономичный детектор на основе глубокой нейронной сети под названием YOLOv4-MN3, основанный на передовой облегченной сети YOLOv4 [40] и MobileNetV3[41].

В ходе экспериментов с целью выбора подходящей магистральной сети, которая могла бы снизить потребление памяти и вычислительные затраты, исходная магистральная сеть CSPDarknet53 из YOLOv4 заменялась на VGG16[42], VGG19, Resnet50, Darknet53, MobileNetV2 [43] и MobileNetV3. Для лучшего соответствия индивидуальному набору данных о дефектах, функции активации сетей в области «шеи» (neck) и прогнозирования в YOLOv4 были заменены различными функциями активации – Sigmoid, Tanh, ReLU, Leaky

ReLU и Mish. На основании полученных результатов, после сравнения метрик mAP авторы остановились на MobileNetV3 с показателем 97.26% mAP. Функция активации Mish [44] получила наименьшие потери при обучении, то есть в целом у неё был лучший результат обучения.

Для уменьшения нагрузки на модель размер всех изображений из набора данных 3018×4096 пикселей был изменен до 416×416 пикселей. Для увеличения наборов данных авторы использовали аугментацию. Изображения на обучающей и тестовой выборках содержали только один дефект. YOLOv4-MN3 была предварительно обучена на наборе PASCAL VOC2007 [45].

Экспериментальные результаты с настроенным набором данных показали, что YOLOv4-MN3 достигает хорошей точности обнаружения – 98,64% mAP, что выше чем у Faster R-CNN, RetinaNet [46], SSD, YOLOv3 [47] и YOLOv4 [48] на том же наборе данных, и высокой скорости обнаружения – 56,98 кадра в секунду с помощью графического процессора RTX3080. Эксперименты по обучению YOLOv4-MN3 показали, что она может адаптироваться к различным категориям поверхностных дефектов и решать сложную проблему разнообразия морфологии дефектов.

В работе [46] авторы применили YOLOv5 [49] без модификации или обновления алгоритма для обнаружения дефектов печатных плат. Усилия авторов были направлены на создание большого набора изображений для обучения и тестирования свёрточной нейронной сети. Исследователи подготовили 23000 изображений печатных плат содержащих дефекты и привели их к небольшому размеру 400 × 400 пикселей. Авторы использовали предварительно обученную модель YOLOv5. Так как алгоритм YOLOv5 превосходит другие алгоритмы обнаружения объектов, благодаря своим уникальным функциям, таким как улучшение данных алгоритмом «Mosaic», адаптивному расчету опорного кадра на разных обучающих наборах и равномерному масштабированию исходных изображений до стандартного размера, авторы добились хороших результатов. В этом эксперименте использовались три модели YOLOv5 разного размера (маленькая, средняя и большая). Большая модель YOLOv5 имела наилучшую точность обнаружения 99,74% mAP.

В работе [50] предпринята попытка снизить количество ложных срабатываний при обнаружении дефектов печатных плат. Авторами была разработана модель свёрточной нейронной сети, состоящая из двух подмоделей со схожей архитектурой и проведены исследования вопроса, как предварительная обработка изображений влияет на повышение качества модели при обучении. Модель2 обучали на предварительно обработанных изображениях. Для обработки изображений применялись [50]: нормализация; перевод изображения в градации серого цвета; выравнивание гистограммы для повышения контрастности изображения; двухканальное изображение объединением результатов выравнивания гистограммы и метода; трехканальное изображение объединением выравнивания гистограммы и методов Лапласиан и Собель; трехканальное изображение объединением выравнивания гистограммы и методов Лапласиан и Канни; четырехканальное изображение объединением выравнивания гистограммы и методов Лапласиан, Собель и Канни; трехканальное изображение объединением нормализациии и методов Лапласиан и Собель. Наилучшую точность модели на тестовых данных показало применение сочетания выравнивания гистограммы и методов Лапласиан и Собель.

В подмоделях для извлечения карты признаков каждого канала изображения использовалась свёртка с тремя различными масштабами 3×3, 4×4 и 5×5 и операция конкатенации для их объединения. Этот метод извлечения признаков помог включить в карту признаков больше информации о признаках на разных масштабах за одну операцию свертки, что способствовало повышению точности обучения и тестирования модели. Вместо традиционной свертки в операции свертки используется свертка с разделением по глубине, что позволяет значительно сократить количество параметров. Использование такого подхода действительно позволило повысить точность модели без увеличения количества параметров.

Модель1 и Модель2 обучались отдельно и затем объединялись в основную модель. При обучении основной модели веса извлечения признаков Модели1 и Модели2 были зафиксированы, а обучалась только часть классификатора (полносвязанный слой), что позволило обучать основную модель использовав признаки, извлеченные моделями 1 и 2. При этом точность основной модели увеличилась до 91%. Для замеров точности при обучении модели использовались метрики accuracy, precision, recall [51].

В исследовании [52] авторы взялись за решение проблемы выявления сразу нескольких дефектов на небольшом участке печатной платы. Это так называемая задача классификации с несколькими метками. Авторы предложили модель свёрточной нейронной сети и обучение с несколькими метками преобразовали в несколько задач бинарной классификации отдельно для каждой метки путем настройки функции потерь.

В задачах АОИ модели глубокого обучения могут выполнять предварительную обработку, извлечение, выбор признаков и классификацию дефектов. Используя глубокие свёрточные нейронные сети можно отследить множество сложных дефектов, которые традиционные алгоритмы АОИ не могут распознать. Точность обнаружения также можно повысить с помощью увеличения количества параметров, используемых при обучении моделей.

3. Заключение

На основании изученных статей можно сделать вывод о том, что на данный момент глубокие нейронные сети, которые демонстрируют хорошие результаты детектирования дефектов печатных плат, являются модификациями следующих семейств моделей – R-CNN, YOLO, SSD и FPN. В настоящее время использование свёрточных нейронных сетей незаменимо для предварительного поиска местоположения и классификации дефектов. При проектировании новой нейронной сети в Марийском государственном университете внимание будет сконцентрировано на тех элементах архитектуры, которые отвечают за обнаружение мелких объектов на изображении.

Для минимизации ложных срабатываний предполагается использовать информацию gerber-файла, в котором находятся все характеристики рисунка печатной платы – ширина линий, интервалы между линиями, формы и размеры контактных площадок и данные о сверлении. Также для уменьшения ложных срабатываний будет необходимо учитывать требования ГОСТ Р 53429-2009, в котором даны допуски по искажению рисунка печатной платы. Представляется, что предобученные модели свёрточных нейронных сетей и использование Quadro P6000 фирмы NVIDIA позволят сократить время обучения. Планируется, что инженеры, осуществляющие оптический контроль на предприятии «ТЕХНОТЕХ», предоставят изображения дефектов для формирования набора данных. Это поможет решить проблему отсутствия в открытом доступе больших наборов данных для обучения сети и несбалансированность данных. Особое внимание будет уделено подбору правильного освещения и обработке полученных изображений, так как необходимо находить дефекты на платах, имеющих медные (блестящие и матовые) проводники после травления или гальванического нанесения, или проводники с золочёным покрытием.

По результатам изучения статей и анализу поставленной задачи ясно, что только гибридные методы позволят приблизиться к функциональности, которая реализована на импортных промышленных установках автоматической оптической инспекции, например Orbotech серии Fusion производства «Orbotech Ltd.» Израиль (https://www.orbotech.com).

Список литературы / References

[1]. Taha Eid M., Emary E., Moustafa K. Automatic Optical Inspection for PCB Manufacturing : a Survey. International Journal of Scientific and Engineering Research, vol. 5, no. 7, pp. 1095-1102, 2014.

[2]. Гонсалес Р. и Вудс. Р. Мир цифровой обработки. Цифровая обработка изображений. М., ТЕХНОСФЕРА, 2012, 1104 с.

[3]. Прэтт У. Цифровая обработка изображений. М., Мир, 1982, 312 с.

[4]. Abd Al Rahman M. Abu Ebayyeh, Mousavi A. A Review and Analysis of Automatic Optical Inspection and Quality Monitoring Methods in Electronics Industry. IEEE Access, vol. 8, 2020, 183192–183271. DOI:10.1109/access.2020.3029127.

[5]. Moganti M., Ercal F., Dagli C. H., and Tsunekawa S. Automatic PCB inspection algorithms: A survey. Computer Vision and Image Understanding, Vol. 63, No. 2, 1996, pp. 287-313, DOI:10.1006/cviu.1996.0020.

[6]. Kumar M. A Survey on Various Approaches of Automatic Optical Inspection for PCB Defect Detection. International Journal of Computer Science and Engineering vol. 7, no. 6, 2019, pp. 837-841. DOI:10.26438/ijcse/v7i6.837841.

[7]. Pal A., Chauhan S., and Bhardwaj S. Detection of Bare PCB Defects by Image Subtraction Method using Machine Vision. Proceedings of the World Congress on Engineering, vol. 2, no. 7, 2011. ISBN: 978-988-19251-4-5.

[8]. Kaur B., Kaur G., Kaur A. Detection and classification of Printed circuit board defects using image subtraction method. Recent Advances in Engineering and Computational Sciences, 2014, DOI:10.1109/raecs.2014.6799537.

[9]. Ma J. Defect detection and recognition of bare PCB based on computer vision. 36th Chinese Control Conference, 2017, DOI:10.23919/chicc.2017.8029117.

[10]. Melnyk R. A., Tushnytskyy R. B. Detection of Defects in Printed Circuit Boards by Clustering the Etalon and Defected Samples. IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, 2020, DOI:10.1109/tcset49122.2020.23558.

[11]. S. H. Indera Putera, Ibrahim Z. Printed circuit board defect detection using mathematical morphology and MATLAB image processing tools. 2nd International Conference on Education Technology and Computer, Shanghai, vol 5, 2010, pp. 359-363, DOI:10.1109/ICETC.2010.5530052.

[12]. Hideaki Doi, Yasuhiko Hara, Koichi Karasaki, Tadashi Iida, Takashi Furutani, Shigeki Kitamura, Norihiro Minatani, and Satoshi Shinada Automated Inspection of hinted Circuit Board Pattems Referenced to CAD Data. IAPR Workshop on Machine Vision Applications, 1992, pp. 419-423.

[13]. Borba J. F. and Facon J. A printed circuit board automated inspection system. Proceedings of the 38th Midwest Symposium on Circuits and Systems, 1996, pp. 69–72.

[14]. Lin, L., Zhou, L., Wan, J., Qian Z. Study of PCB Automatic Optical Inspection System Based on Mathematical Morphology. International Conference on Computer Technology and Development, 2009, DOI:10.1109/icctd.2009.35.

[15]. Benhabib B., Charette C. R., Smith K. C., Yip A. M. Automatic Visual Inspection of Printed Circuit Boards: An Experimental System. International Journal of Robotics and Automation, 1990, vol. 5, no. 2.

[16]. Ibrahim, Z., Rahman Al-Attas, S. A. Wavelet-based printed circuit board inspection algorithm // Integrated Computer-Aided Engineering, 12(2), 201–213. DOI:10.3233/ica-2005-12206.

[17]. Evgeniou, T., Pontil, M. Support Vector Machines: Theory and Applications. Conference: Machine Learning and Its Applications, 2001, pp. 249–257. DOI:10.1007/3-540-44673-7_12.

[18]. Uhrig R. E. Introduction to artificial neural networks. Proceedings of IECON'95 - 21st Annual Conference on IEEE Industrial Electronics, 1995, DOI:10.1109/iecon.1995.483329.

[19]. Хайкин С. Нейронные сети: полный курс. Москва-Санкт-Петербург-Киев, Вильямс, 2006, 1104 с.

[20]. Reeves C. R. , Wright C. Genetic Algorithms and The Design of Experiments, School of Mathematical and Information Sciences, 1998, DOI: 10.1007/978-1-4612-1542-4_12.

[21]. Krizhevsky A., Ilya Sutskever I., Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Communications of the ACM, vol. 60 no. 6, 2017, pp. 84–90, DOI:10.1145/3065386.

[22]. Li Deng, Dong Yu, Deep Learning: Methods and Applications, Microsoft Research, 2014.

[23]. Girshick, R., Donahue, J., Darrell, T., Malik, J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1), 2015, pp.142–158. DOI:10.1109/tpami.2015.2437384.

[24]. Girshick R. Fast R-CNN. IEEE International Conference on Computer Vision, 2015, DOI: 10.1109/ICCV.2015.169.

[25]. Ren S., He K., Girshick, R., Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 2016, pp.1137–1149, DOI:10.1109/tpami.2016.2577031.

[26]. Redmon, J., Divvala, S., Girshick, R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition. 2016, DOI:10.1109/cvpr.2016.91.

[27]. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A. C. SSD: Single Shot MultiBox Detector. Lecture Notes in Computer Science, 2016, pp. 21–37. DOI:10.1007/978-3-319-46448-0_2.

[28]. Zhenshen Qu, Jianxiong Shen, Ruikun Li, Junyu Liu, Qiuyu Guan, PartsNet: A Unified Deep Network for Automotive Engine Precision Parts Defect Detection. Preprint State Key Laboratory of advanced welding and joining, Harbin Institute of Technology, Harbin, China, 2018, DOI:10.1145/3297156.3297190.

[29]. Ding, R., Dai, L., Li, G., Liu, H. TDD-Net: A Tiny Defect Detection Network for Printed Circuit Boards. CAAI Transactions on Intelligence Technology, 2019, DOI:10.1049/trit.2019.0019.

[30]. Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. IEEE Conference on Computer Vision and Pattern Recognition, 2017, DOI:10.48550/arXiv.1612.08242.

[31]. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition, 2016, DOI:10.1109/cvpr.2016.90.

[32]. Deng J., Dong W., Socher R., Li K. Li, Li F.-F. ImageNet: A large-scale hierarchical image database. IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, DOI: 10.1109/CVPR.2009.5206848.

[33]. Iglovikov V., Shvets A. Ternausnet: U-net with VGG11 encoder pre-trained on ImageNet for image segmentation. 2018, Corpus ID: 1385457.

[34]. Lin T.-Y., Dollar P., Girshick R., He K., Hariharan B., Belongie S. Feature Pyramid Networks for Object Detection. IEEE Conference on Computer Vision and Pattern Recognition. 2017, DOI:10.1109/cvpr.2017.106.

[35]. Adibhatla, V. A., Chih, H.-C., Hsu, C.-C., Cheng, J., Abbod, M. F., Shieh, J.-S. Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks. Electronics, vol. 9, no.9, p.1547. DOI:10.3390/electronics9091547.

[36]. Hu, B., Wang, J. Detection of PCB Surface Defects with Improved Faster-RCNN and Feature Pyramid Network. IEEE Access, vol. 8, 2020, pp. 108335-108345, DOI:10.1109/access.2020.3001349.

[37]. Wang, J., Chen, K., Yang, S., Loy, C. C., Lin D. Region Proposal by Guided Anchoring. IEEE Conference on Computer Vision and Pattern Recognition. 2019, DOI:10.1109/cvpr.2019.00308.

[38]. Ma N., Zhang X., Zheng H.-T., Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. Lecture Notes in Computer Science, 2018, pp.122–138, DOI:10.1007/978-3-030-01264-9_8.

[39]. Xinting Liao, Shengping Lv, Denghui Li, YOLOv4-MN3 for PCB Surface Defect Detection. Applied Sciences vol.11, no. 24, 2021, pp. 11701, DOI: 10.3390/app112411701.

[40]. Bochkovskiy A., Chien-Yao Wang, Hong-Yuan Mark Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection. IEEE Conference on Computer Vision and Pattern Recognition. 2020, DOI:10.48550/arXiv.2004.10934.

[41]. Howard A., Sandler M., Chu G., Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, Hartwig A. Searching for MobileNetV3. Conference on Conference on Computer Vision and Pattern Recognition. 2019, DOI:10.48550/arXiv.1905.02244.

[42]. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. Conference on Computer Vision and Pattern Recognition. 2015, DOI:10.48550/arXiv.1409.1556.

[43]. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. Conference on Computer Vision and Pattern Recognition. 2018, DOI:10.1109/cvpr.2018.00474.

[44]. Diganta Misra Mish: A Self Regularized Non-Monotonic Activation Function. Conference on Computer Vision and Pattern Recognition, 2019, Corpus ID: 201645264.

[45]. Everingham M., Christopher K. I. Williams, Luc Van Gool, John M. Winn, The Pascal Visual Object Classes (VOC) challenge. International Journal of Computer Vision, vol. 88, no.2, pp.303-338, 2010, DOI: 10.1007/s11263-009-0275-4.

[46]. Yixing Li, Fengbo Ren Light-Weight RetinaNet for Object Detection. Conference on Computer Vision and Pattern Recognition, 2019, DOI:10.48550/arXiv.1905.10011.

[47]. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. Conference on Computer Vision and Pattern Recognition, Corpus ID: 4714433.

[48]. Venkat Anil Adibhatla, Huan-Chuang Chih, Chi-Chang Hsu, Joseph Cheng, Maysam F. Abbod, Jiann-Shing Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once. Mathematical Biosciences and Engineering, 2021, vol. 18, no. 4, pp. 4411-4428. DOI:10.3934/mbe.2021223.

[49]. Ultralytics Yolov5. Available at: https://github.com/ultralytics/yoloV5, accessed 20.11.2022.

[50]. I-Chun Chen, Rey-Chue Hwang, Huang-Chu Huang PCB Defect Detection Based on Deep Learning Algorithm. Processes, vol. 11, no. 3, 2023, DOI: 10.3390/pr11030775.
[51]. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. М., ДМК, 2018, 652с.
[52]. Linlin Zhang, Yongqing Jin, Xuesong Yang, Xia Li, Xiaodong Duan, Yuan Sun,Hong Liu Convolutional Neural Network Based Multi-label Classification of PCB Defects. The Journal of Engineering. 2018, vol. 16, DOI:10.1049/joe.2018.8279.

Информация об авторах / Information about authors

Татьяна Сергеевна ХОДАТАЕВА – программист научно-исследовательской лаборатории разработки, проектирования и технической инспекции печатных плат. Сфера научных интересов: распознавание образов, глубокое обучение, нейронные сети.

Tatiana Sergeevna KHODATAEVA is a programmer of the research laboratory for the development, design and technical inspection of printed circuit boards. Research interests: pattern recognition, deep learning, neural networks.

Николай Владимирович КАШИРИН – кандидат химических наук, доцент, заведующий базовой кафедрой конструирования и производства керамических изделий микроэлектроники, заведующий молодежной научно-исследовательской лабораторией разработки, проектирования и технической и инспекции печатных плат, начальник молодежного научно-инновационного конструкторского центра Spektrum. Сфера научных интересов: материаловедение; керамические и композиционные материалы в изделиях электронной техники; методы исследования микроструктуры; электроника и микропроцессорная техника; электротехника; коллоидные системы; сопротивление материалов.

Nikolai Vladimirovich KASHIRIN – Candidate of Chemical Sciences, Associate Professor, Head of the Basic Department of Design and Production of Ceramic Microelectronic Products, Head of the Youth Research Laboratory for the Development, Design and Technical and Inspection of Printed Circuit Boards, Head of the Spektrum Youth Research and Innovation Design Center. Research interests: materials science; ceramic and composite materials in electronic products; methods of microstructure research; electronics and microprocessor technology; electrical engineering; colloid systems; strength of materials.

Александра Ивановна АВЕРИНА – инженер научно-исследовательской лаборатории разработки, проектирования и технической инспекции печатных плат, техник базовой кафедрой конструирования и производства керамических изделий микроэлектроники. Сфера научных интересов: электроника, исследование адгезионных свойств, проектирование печатных плат, разработка методов и проведение научно-исследовательских экспериментов.

Alexandra Ivanovna AVERINA – engineer of the research laboratory for the development, design and technical inspection of printed circuit boards, technician of the basic department of design and production of microelectronics ceramic products. Research interests: electronics, adhesion research, printed circuit board design, method development and research experiments.

Артём Евгеньевич ГУРЬЯНОВ – инженер научно-исследовательской лаборатории разработки, проектирования и технической инспекции печатных плат. Сфера научных интересов: электроника, технологии программирования, моделирование приборов и системы управления, 3D-моделирование.

Artyom Evgenyevich GURYANOV – engineer of the research laboratory for the development, design and technical inspection of printed circuit boards. Research interests: electronics, programming technologies, modeling of devices and control systems, 3D modeling.

DOI: 10.15514/ISPRAS-2023-35(4)-6

Технология синтеза программных комплексов с гибридной визуализацией Vulkan-OpenGL

П.Ю. Тимохин, ORCID: 0000-0002-0718-1436 <*webpismo@yahoo.de*>
М.В. Михайлюк, ORCID: 0000-0002-7793-080X <*mix@niisi.ras.ru*>

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», 117218, Россия, г. Москва, Нахимовский просп., д. 36, к.1.

Аннотация. В данной работе рассматривается задача встраивания компьютерной визуализации, выполняемой с помощью API Vulkan, в программные комплексы, основанные на API OpenGL. Описывается низкоуровневый гибридный подход к реализации совместной работы двух API в рамках одного приложения, а также организация и синхронизация доступа к совместно используемым ресурсам. Предлагается технология «инкапсуляции» гибридного подхода в отдельном библиотечном модуле (VK-капсуле) с высокоуровневым интерфейсом, который динамически подключается к исполняемому модулю OpenGL-комплекса (GL-визуализатору). В работе описаны методы построения и подключения интерфейса VK-капсулы, обеспечивающие минимальное вмешательство в GL-визуализатор. На основе предложенных методов и технологии был разработан прототип модульного программного комплекса, реализующего гибридную визуализацию Vulkan-OpenGL. Была проведена апробация созданного комплекса, которая подтвердила адекватность предложенных решений поставленной задаче и возможность их использования для расширения возможностей систем визуализации, построенных на базе OpenGL.

Ключевые слова: визуализация; программирование; GPU; Vulkan; OpenGL; интерфейс; библиотека.

Для цитирования: Тимохин П.Ю., Михайлюк М.В. Технология синтеза программных комплексов с гибридной визуализацией Vulkan-OpenGL. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 121–128. DOI: 10.15514/ISPRAS–2023–35(4)–6.

Благодарности: Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН “Проведение фундаментальных научных исследований (47 ГП)” по теме № FNEF-2022-0012 “Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации”.

A technology to synthesize software complexes with hybrid visualization Vulkan-OpenGL

P.Yu. Timokhin, ORCID: 0000-0002-0718-1436 <*webpismo@yahoo.de*>
M.V. Mikhaylyuk, ORCID: 0000-0002-7793-080X <*mix@niisi.ras.ru*>

Scientific Research Institute for System Analysis of the Russian Academy of Sciences (SRISA), build. 1, 36, Nakhimovskiy Avenue, Moscow, 117218, Russia.

Abstract. In this paper, the task of embedding computer visualization, performed using the Vulkan API, into OpenGL-based software complexes, is considered. A low-level hybrid approach to implement the collaboration of two APIs within the same application is described, as well as, the organization and synchronization of access to shared resources. The technology is proposed, which "encapsulates" the hybrid approach in a separate library module (VK-capsule) with a high-level interface that is dynamically linked to the executable module of

OpenGL-complex (GL-visualizer). The paper describes methods for construction of the interface and connection of the VK-capsule, providing minimal intrusion into GL-visualizer. Based on the proposed methods and technology, a prototype of modular software complex implementing hybrid Vulkan-OpenGL visualization was developed. The approbation of the created complex was carried out, which confirmed the adequacy of the proposed solutions to the task assigned and the possibility of using them to expand the capabilities of visualization systems built on the OpenGL.

Keywords: visualization; programming; GPU; Vulkan; OpenGL; interface; library.

For citation: Timokhin P.Yu., Mikhaylyuk M.V. A technology to synthesize software complexes with hybrid visualization Vulkan-OpenGL. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 4, 2023. pp. 121-128 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-6.

Acknowledgements. The publication is made within the state task of Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" on "Carrying out basic scientific researches (47 GP)" on topic No. FNEF-2022-0012 "Virtual environment systems: technologies, methods and algorithms of mathematical modeling and visualization".

1. Введение

OpenGL [1] является одним из основных интерфейсов программирования (API) 3D компьютерной графики реального времени, востребованной в современных системах виртуального окружения [2], научной визуализации [3, 4], видеосимуляторах [5] и др. Являясь открытым и платформонезависимым стандартом, OpenGL имеет интуитивно понятный интерфейс и продуманную архитектуру, поддерживаемую всеми современными графическими картами, благодаря чему завоевал популярность среди разработчиков. Вместе с этим парадигма OpenGL имеет ряд особенностей, ограничивающих эффективность данного API и возможности реализации актуальных графических технологий, например, аппаратно-ускоренной трассировки лучей [6].

Ввиду этого еще в 2014 году промышленный консорциум Khronos Group (разработчик OpenGL) начал создание открытого, кроссплатформенного стандарта Vulkan [7] – графического и вычислительного API следующего поколения. Целью нового API стало снижение накладных расходов при выполнении графических операций, предоставление более полного контроля над работой GPU, а также уменьшение использования CPU. В этой связи API Vulkan стал более низкоуровневым (по сравнению с OpenGL), что в свою очередь привело к существенному росту трудоемкости программирования графики. Так, многие рутинные задачи, которые в OpenGL выполнял драйвер (формирование пула и очередей команд, буферизация кадров, управление памятью GPU и др.), в Vulkan разработчик должен явно прописывать в коде. Несколько облегчить разработку может применение библиотек-оберток Vulkan [8] или автоматизация на основе паттернов (шаблонов) [9]. Однако данные подходы ориентированы на то, что разработка приложения ведется полностью на Vulkan, что не всегда целесообразно. Зачастую, имея достаточно развитый функционал, основанный на OpenGL, в системе визуализации необходимо реализовать отдельные подзадачи, которые более эффективно решаются с помощью Vulkan (например, визуализация полей высот с помощью аппаратно-ускоренной трассировки лучей [10]). В этой ситуации возникает задача встраивания Vulkan-визуализации в программный комплекс, реализованный на OpenGL, и обеспечения их совместной работы на GPU.

Принципиальная возможность такой совместной работы (интероперабельность) показана в примерах, выпущенных компаниями NVidia [11, 12] и Khronos Group [13]. Недостатком данных материалов является формат «все в одном», характеризующийся тесным взаимным переплетением API OpenGL и Vulkan, обернутым в корпоративные фреймворки (системы классов), что существенно затрудняет понимание механизма практической реализации интероперабельности. Целью же данного исследования является разработка технологии, при которой Vulkan-визуализация выполняется в отдельном библиотечном модуле, подключаемом к OpenGL-приложению с минимальным вмешательством.

2. Подход к гибридной визуализации Vulkan-OpenGL

Несмотря на то, что Vulkan позиционируется как OpenGL следующего поколения, оба API имеют разные идеологии визуализации, в общем случае несовместимые друг с другом. В OpenGL передача команд на GPU и связь с окном приложения осуществляется через специальный объект (оболочку) - контекст отрисовки. В Vulkan понятия контекста отрисовки как такового нет, а его функции, по сути, распределены между рядом абстрактных объектов (экземпляр, физическое и логическое устройства, пулы, буферы и очереди команд и т.д.). И хотя принципиальные различия между контекстом OpenGL и «контекстом» Vulkan не позволяют проводить совместную визуализацию напрямую в окне приложения, это не исключает возможности работы обоих API в рамках одного программного комплекса [11]. Учитывая этот факт, обойти проблему совместной визуализации можно с помощью *гибридного подхода*, при котором Vulkan осуществляет отображение виртуального объекта (сцены) в текстуру (Render-to-texture, RTT-текстуру), а OpenGL визуализирует эту RTT-текстуру на весь экран. Реализация этого подхода включает решение следующих двух ключевых задач.

Во-первых, необходимо чтобы Vulkan и OpenGL могли «видеть» одну и ту же область видеопамяти и работать с ней, как с текстурой. Решение этой задачи основано на идее *разделяемого текстурного объекта*. Ее суть состоит в том, что на стороне Vulkan выделяется область видеопамяти, и к ней привязывается текстурный объект в «контексте» Vulkan, а на стороне OpenGL импортируется образ этой области видеопамяти (по сути, ее адрес и размер) и к нему привязывается текстурный объект в контексте OpenGL. Таким образом, мы фактически получаем два рабочих текстурных объекта, привязанные к одной области видеопамяти, в которой хранится RTT-текстура.

Во-вторых, необходимо синхронизировать доступ двух API к совместно используемым ресурсам (GPU и разделяемому текстурному объекту). Если этого не сделать, то API будут мешать друг другу (например, Vulkan еще досчитывает RTT-текстуру, а OpenGL уже выводит ее на экран), и результат будет непредсказуемым. Решение второй задачи основано на использовании *GL-семафоров* - специальных примитивов синхронизации, которые могут переключаться только посредством GPU в состояние сигнала или ожидания (значение по умолчанию). В рассматриваемой задаче используется пара GL-семафоров:

- *updateSemaphore* – готовность разделяемого текстурного объекта к обновлению;
- *synthesisSemaphore* – завершение синтеза RTT-текстуры на GPU.

Управление GL-семафорами осуществляется обоими API на стадии формирования кадра визуализации. Вначале сторона OpenGL переводит *updateSemaphore* в состояние сигнала, получив который сторона Vulkan начинает синтез RTT-текстуры в разделяемом текстурном объекте. По окончании синтеза RTT-текстуры сторона Vulkan переводит *synthesisSemaphore* в состояние сигнала, получив который сторона OpenGL начинает визуализацию RTT-текстуры на экране. Отметим, что после получения сигналов сторонами Vulkan и OpenGL *updateSemaphore* и *synthesisSemaphore* автоматически сбрасываются в состояние ожидания, что позволяет их снова использовать на следующем кадре визуализации. Чтобы реализовать описанное совместное управление синхронизацией, каждый GL-семафор создается *разделяемым*: на стороне Vulkan в видеопамяти создается сам объект семафора, а на стороне OpenGL импортируется образ этого объекта (по аналогии с разделяемым текстурным объектом).

Из описания гибридного подхода видно, что API Vulkan и OpenGL взаимодействуют на достаточно низком уровне, что существенно усложняет процесс встраивания Vulkan-визуализации в программные комплексы на базе OpenGL, имеющие свои сложившиеся экосистемы. В данной работе предлагается технология «инкапсуляции» гибридного подхода, позволяющая встраивать Vulkan-визуализацию в программные комплексы на базе OpenGL с минимальным вмешательством. Рассмотрим ее более подробно.

3. Технология «инкапсуляции» гибридного подхода

В данной работе мы будем рассматривать задачу встраивания программного блока, реализующего синтез RTT-текстуры с помощью API Vulkan (далее *VK-блок*), в исполняемый модуль, осуществляющий OpenGL-визуализацию (далее *GL-визуализатор*). Предлагаемое решение основано на построении *VK-капсулы* - программной оболочки, изолирующей работу VK-блока в отдельном библиотечном модуле, динамически связываемом с GL-визуализатором (см. рис. 1). VK-капсула является системой взаимосвязанных программных блоков, в которую кроме VK-блока входят блок гибридной визуализации (далее *HR-блок*) и интерфейсный блок (далее *I-блок*). HR-блок реализует отображение рассчитанной VK-блоком RTT-текстуры в буфере кадра GL-визуализатора с помощью гибридного подхода, описанного в разделе 2. I-блок реализует высокоуровневый программный интерфейс взаимодействия GL-визуализатора и VK-капсулы, и является ядром предлагаемой технологии «инкапсуляции» гибридного подхода. Рассмотрим ее этапы.

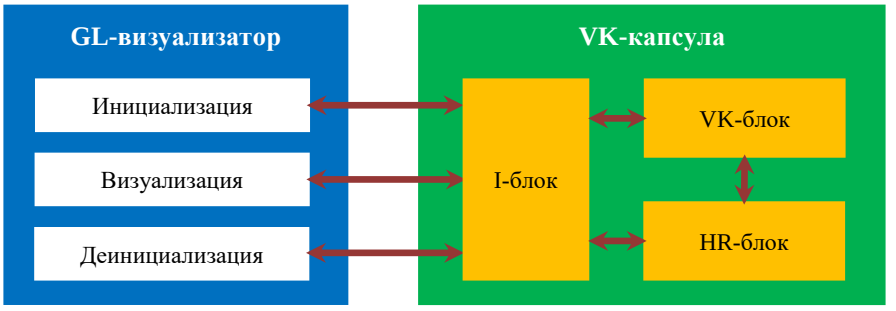

Рис. 1. Структура программного комплекса с гибридной визуализацией
Fig. 1. The structure of software complex with hybrid visualization

3.1 Построение интерфейса VK-капсулы

На данном этапе реализуется набор интерфейсных функций, посредством которых GL-визуализатор взаимодействует с VK- и HR-блоком. Набор состоит из базовой и пользовательской частей. В базовую часть входит минимальный список функций, необходимых для работы VK-капсулы:

- *init* - выделяет ресурсы, необходимые для VK- и HR-блока, инициализирует эти блоки, а также связывает их между собой через разделяемые GL-семафоры и текстурный объект (см. раздел 2);
- *render* - синтезирует RTT-текстуру с помощью VK-блока и отображает в буфере кадра GL-визуализатора с помощью HR-блока;
- *deinit* - освобождает ресурсы, выделенные для VK- и HR-блока и возвращает их в состояния до вызова функции *init*.

Пользовательская часть набора определяется, исходя из задачи, решаемой VK-блоком, и, в общем случае, может включать в себя достаточно большой список функций: задания позиций и ориентаций объектов сцены, управления виртуальной камерой и источниками освещения, обработки событий окна, клавиатуры, мыши и т.д.

Набор интерфейсных функций реализуется в I-блоке с помощью двух классов:

- *AVkCapsule* - абстрактный класс, состоящий из чистых виртуальных функций, описывающих набор интерфейсных функций VK-капсулы (см. рис. 2);
- *CVkCapsule* - наследуемый от *AVkCapsule* класс, который реализует его виртуальные функции (описанные выше связи с VK- и HR-блоком).

```
// Абстрактный класс интерфейса VK-капсулы.
class AVkCapsule
{
// == Набор интерфейсных функций VK-капсулы (чистых виртуальных функций) ==
// Базовая часть набора.
public:
    // Инициализирует VK-капсулу.
    virtual bool init(uint32_t _wndWidth, uint32_t _wndHeight, const char* _pScenePath) = 0;

    // Визуализирует VK-капсулу.
    virtual void render() = 0;

    // Деинициализирует VK-капсулу.
    virtual void deinit() = 0;

// Пользовательская часть набора.
public:
    // Обработчик события перемещения мыши.
    virtual void onMouseMove(float _xCoord, float _yCoord) = 0;

    // Обработчик события нажатия клавиши мыши.
    virtual void onMouseButton(int _buttonCode, int _actionType, int _mods) = 0;

    // Обработчик события нажатия клавиши клавиатуры.
    virtual void onKey(int _keyCode, int _scancode, int _actionType, int _mods) = 0;

    // Возвращает инфо-строку VK-капсулы.
    virtual const char* getInfoStr() = 0;
};
```

Рис. 2. Пример абстрактного класса AVkCapsule, включающего базовый и пользовательский набор интерфейсных функций

Fig. 2. An example of abstract AVkCapsule class including base and user sets of interface functions

3.2 Подключение VK-капсулы

На данном этапе реализуется *модифицированное* явное динамическое связывание библиотеки VK-капсулы и исполняемого модуля GL-визуализатора. В отличие от типового подхода, предполагающего экспорт всех интерфейсных функций библиотеки, в данной работе на стороне VK-капсулы (в I-блоке) экспортируется только пара функций:

- *create* - создает объект класса *CVkCapsule* в модуле VK-капсулы;
- *destroy* - удаляет объект класса *CVkCapsule*.

Ключевым аспектом является то, что при создании объекта *CVkCapsule* функция *create* возвращает на него указатель типа базового класса *AVkCapsule**:

$$AVkCapsule* \; pCapsule = new \; CVkCapsule(); \qquad (1)$$

Указатель *pCapsule*, по сути, представляет собой указатель на таблицу виртуальных функций класса *AVkCapsule*, созданную на этапе компиляции модуля VK-капсулы. После выполнения выражения (1), благодаря свойству полиморфизма программного кода, в эту таблицу будут записаны адреса соответствующих виртуальных функций объекта класса *CVkCapsule* (так называемое позднее связывание). Таким образом, имея лишь объявление базового класса *AVkCapsule* и указатель *pCapsule*, GL-визуализатор получает доступ ко всем интерфейсным функциям VK-капсулы.

Подключение VK-капсулы реализуется на трех типовых стадиях работы GL-визуализатора: инициализации, визуализации и деинициализации (см. рис. 1). Для Win-платформы алгоритм подключения VK-капсулы с базовым набором интерфейсных функций имеет следующий вид:

1. На стадии инициализации GL-визуализатора:

- загрузить модуль VK-капсулы с помощью функции *LoadLibrary*;

- получить адреса интерфейсных функций *create* и *destroy* модуля VK-капсулы с помощью функции *GetProcAddress*;
- получить указатель *pCapsule* на объект VK-капсулы с помощью функции *create*;
- вызвать через указатель *pCapsule* метод *init* объекта VK-капсулы.

2. На стадии визуализации (формирования кадра) GL-визуализатора вызвать через указатель *pCapsule* метод *render* объекта VK-капсулы.

3. На стадии деинициализации GL-визуализатора:

- вызвать через указатель *pCapsule* метод *deinit* объекта VK-капсулы;
- удалить объект VK-капсулы с помощью функции *destroy*;
- освободить модуль VK-капсулы от привязки к GL-визуализатору с помощью функции *FreeLibrary*.

4. Результаты

На основе предложенной технологии был разработан прототип модульного программного комплекса, реализующего гибридную визуализацию Vulkan-OpenGL. Программный комплекс написан на языке C++ с использованием языка GLSL программирования шейдеров. В комплекс входят GL-визуализатор, обеспечивающий создание окна и контекста OpenGL, и VK-капсула (построена на основе API Vulkan версии 1.3.204.1), реализующая гибридную визуализацию облаков точек с помощью аппаратно-ускоренной трассировки лучей. Была проведена апробация созданного комплекса на задаче моделирования и визуализации элементов рельефа с отрицательными уклонами (пещер, шахт, туннелей и др.). Для апробации использовался набор данных «NASA Planetary Pits and Caves Analog Dataset» [14], полученных на основе LIDAR-сканирования с высоким разрешением реальных наземных объектов. На рис. 3 показаны примеры кадров гибридной визуализации элемента рельефа «King's Bowl» («Королевская Чаша»), состоящего из 3740782 точек. Визуализация выполнялась со средней скоростью около 500 кадров в секунду на персональном компьютере, оборудованном видеокартой NVidia GeForce RTX 2080 (драйвер NVidia DCH версии 536.40), при разрешении экрана Full HD. Снижение производительности гибридной визуализации по сравнению с «чистой» Vulkan-визуализацией составило не более 8-10%.

5. Заключение

Проведенная апробация подтвердила адекватность предложенной технологии поставленной задаче и возможность ее эффективного применения в программных комплексах, выполняющих сложную интерактивную визуализацию, за счет следующих преимуществ.

Во-первых, VK-капсула и GL-визуализатор взаимодействуют напрямую, без использования промежуточной библиотеки импорта, создаваемой компоновщиком. Это позволяет разрабатывать модуль VK-капсулы в среде программирования, имеющей версию, отличную от используемой для GL-визуализатора. Это особенно актуально, когда в VK-капсуле (VK-блоке) необходимо реализовать современные графические технологии (например, аппаратно-ускоренную трассировку лучей), требующие использования сред программирования последних версий.

Во-вторых, благодаря явному типу связывания сохраняется возможность работы GL-визуализатора без библиотеки VK-капсулы (для задач, где необходима только OpenGL-визуализация). Это позволяет создавать эффективные конфигурации из модулей программного комплекса под конкретные задачи, а также вести разработку модулей независимо друг от друга (при наличии согласованного интерфейса VK-капсулы).

В-третьих, благодаря модификации явного связывания, на стороне GL-визуализатора устраняется необходимость реализации программной «обвязки» (создание указателей,

получение адресов) для каждой функции интерфейса VK-капсулы. Это особенно актуально, когда к GL-визуализатору необходимо подключить VK-капсулу с обширным набором интерфейсных функций. Также исключается конфликт имен интерфейсных функций при встраивании в GL-визуализатор нескольких VK-капсул с похожими интерфейсами. Все это делает процесс встраивания VK-капсулы комфортным и облегчает дальнейшее сопровождение и масштабирование программного комплекса.

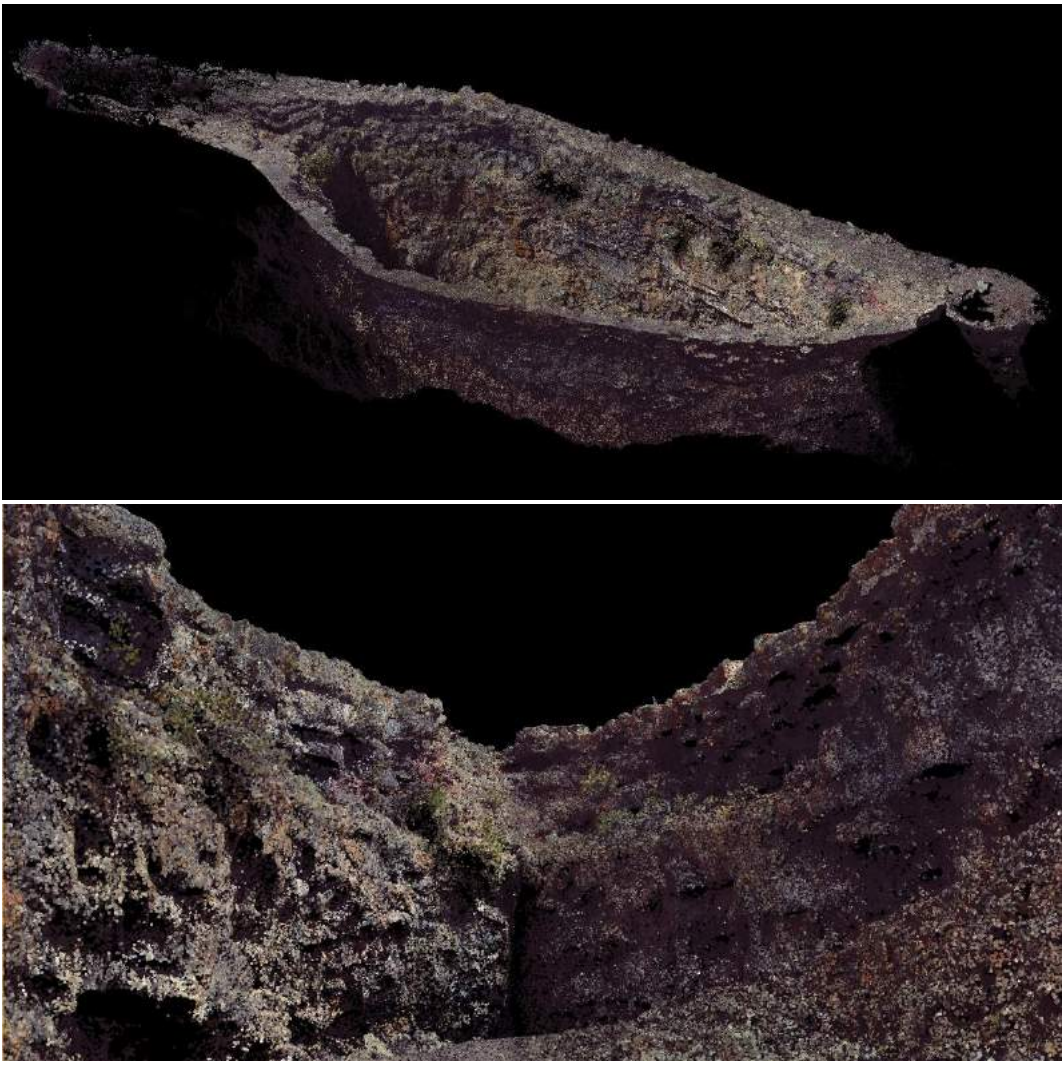

Рис. 3. Примеры Vulkan-визуализации элемента рельефа «King's Bowl» снаружи (вверху) и изнутри (внизу), выполняемой в OpenGL-окне с помощью разработанной технологии

Fig. 3. Examples of Vulkan-visualization of the "King's Bowl" terrain element from the outside (top) and from the inside (bottom), performed in OpenGL-window using the developed technology

Список литературы / References

[1]. OpenGL - The Industry Standard for High Performance Graphics. Available at: https://www.opengl.org/, accessed 13.07.2023.

[2]. Михайлюк М.В., Мальцев А.В., Тимохин П.Ю., Страшнов Е.В., Крючков Б.И., Усов В.М. Система виртуального окружения VirSim для имитационно-тренажерных комплексов подготовки

космонавтов. Пилотируемые полеты в космос, № 4(37), 2020 г., стр. 72-95. DOI: 10.34131/MSF.20.4.72-95. / Mikhaylyuk M.V., Maltsev A.V., Timokhin P.Yu., Strashnov E.V., Kryuchkov B.I., Usov V.M. The VirSim Virtual Environment System for the Simulation Complexes of Cosmonaut Training. Pilotiruemye polety v kosmos / Manned Spaceflight, № 4(37), 2020, pp. 72-95 (in Russian). DOI: 10.34131/MSF.20.4.72-95.

[3]. ParaView - Open-source, multi-platform data analysis and visualization application. Available at: https://www.paraview.org/, accessed 13.07.2023.

[4]. Avogadro - Free cross-platform molecule editor and visualizer. Available at: https://avogadro.cc/, accessed 13.07.2023.

[5]. UNIGINE: real-time 3D engine. Available at: https://unigine.com/, accessed 13.07.2023.

[6]. NVIDIA RTX platform. Available at: https://developer.nvidia.com/rtx, accessed 13.07.2023.

[7]. Vulkan - a cross-platform industry standard for 3D graphics and computing. Available at: https://www.vulkan.org/, accessed 13.07.2023.

[8]. VulkanSceneGraph (VSG), Vulkan & C++17 based Scene Graph Project. Available at: https://vsg-dev.github.io/vsg-dev.io/, accessed 13.07.2023.

[9]. Фролов В.А., Санжаров В.В., Галактионов В.А., Щербаков А.С. Автоматизация разработки на Vulkan: предметно-ориентированный подход. Труды ИСП РАН, том 33, вып. 5. 2021 г., стр. 181-204. DOI: 10.15514/ISPRAS-2021-33(5)-11 / Frolov V.A., Sanzharov V.V., Galaktionov V.A, Scherbakov A.S. Development in Vulkan: a domain-specific approach. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 181-204 (in Russian). DOI: 10.15514/ISPRAS-2021-33(5)-11.

[10]. Тимохин П.Ю., Михайлюк М.В. Рендеринг детализированных полей высот в реальном времени с использованием аппаратного ускорения трассировки лучей. Труды 32-й Международной конференции по компьютерной графике и машинному зрению (GraphiCon 2022), Рязань, 19-22 сентября 2022 г., стр. 124-135. DOI: 10.20948/graphicon-2022-124-135 / Timokhin P.Yu., Mikhaylyuk M.V. Real-time Rendering of Detailed Height Fields Using Hardware-based Ray Tracing Acceleration. In Proceedings of the 32th International Conference on Computer Graphics and Vision (GraphiCon 2022), Ryazan, Russia, September 19-22, 2022, pp. 124-135 (in Russian). DOI: 10.20948/graphicon-2022-124-135.

[11]. Lefrançois M.-K. OpenGL Interop. NVIDIA DesignWorks Samples. Available at: https://github.com/nvpro-samples/gl_vk_simple_interop, accessed 13.07.2023.

[12]. Lefrançois M.-K. OpenGL Interop – Raytracing. NVIDIA DesignWorks Samples. Available at: https://github.com/nvpro-samples/gl_vk_raytrace_interop, accessed 13.07.2023.

[13]. Vulkan Samples, OpenGL interoperability, The Khronos Group. 2020-2023. Available at: https://github.com/KhronosGroup/Vulkan-Samples/tree/main/samples/extensions/open_gl_interop, accessed 13.07.2023.

[14]. Wong U., Whittaker W., Jones H., Whittaker R. NASA Planetary Pits and Caves Analog Dataset. December 2014. Available at: https://ti.arc.nasa.gov/dataset/caves/, accessed 13.07.2023.

Информация об авторах / Information about authors

Петр Юрьевич ТИМОХИН – старший научный сотрудник ФГУ ФНЦ НИИСИ РАН. Сфера научных интересов: компьютерная графика, визуализация.

Petr Yurievich TIMOKHIN – Senior Researcher of SRISA RAS. Research interests: computer graphics, visualization.

Михаил Васильевич МИХАЙЛЮК – доктор физико-математических наук, профессор, главный научный сотрудник ФГУ ФНЦ НИИСИ РАН. Сфера научных интересов: компьютерная графика, визуализация, системы виртуального окружения.

Mikhail Vasilievich MIKHAYLYUK – Doctor of Physical and Mathematical Sciences, Professor, Chief Researcher of SRISA RAS. Research interests: computer graphics, visualization, virtual environment systems.

DOI: 10.15514/ISPRAS-2023-35(4)-7

Программа построения вполне интерпретируемых элементарных и неэлементарных квазилинейных регрессионных моделей

М.П. Базилевский, ORCID: 0000-0002-3253-5697 <*mik2178@yandex.ru*>

Иркутский государственный университет путей сообщения,
664074, Россия, г. Иркутск, ул. Чернышевского, д. 15.

Аннотация. Вполне интерпретируемая линейная регрессия удовлетворяет следующим условиям: знаки её коэффициентов соответствуют содержательному смыслу факторов; мультиколлинеарность незначительна; коэффициенты значимы; качество аппроксимации модели высокое. Ранее для построения таких моделей, оцениваемых с помощью метода наименьших квадратов, была разработана программа ВИнтер-1. В ней по заданным начальным параметрам автоматически формируется задача частично-булевого линейного программирования, в результате решения которой осуществляется отбор наиболее информативных регрессоров. Лежащий в основе этой программы математический аппарат со временем был существенно расширен: были разработаны неэлементарные линейные регрессии, для контроля мультиколлинеарности были предложены линейные ограничения на абсолютные величины интеркорреляций, появились предположения о возможности построения не только линейных, но и квазилинейных регрессий. Данная статья посвящена описанию разработанной второй версии программы построения вполне интерпретируемых регрессий ВИнтер-2. Программа ВИнтер-2 позволяет в зависимости от выбранных пользователем начальных параметров автоматически формулировать для решателя LPSolve задачи частично-булевого линейного программирования для построения как элементарных, так и неэлементарных вполне интерпретируемых квазилинейных регрессий. Предусмотрена возможность задания до девяти элементарных функций и контроля таких параметров, как число регрессоров в модели, число знаков в вещественных числах после запятой, абсолютные вклады переменных в общую детерминацию, число вхождений объясняющих переменных в модель и величины интеркорреляций. В процессе работы с программой также можно контролировать количество элементарно и неэлементарно преобразованных переменных, влияющих на скорость решения задачи частично-булевого линейного программирования. Программа ВИнтер-2 универсальна и может применяться для построения вполне интерпретируемых математических зависимостей в различных предметных областях.

Ключевые слова: линейная регрессия; вполне интерпретируемая регрессия; метод наименьших квадратов; мультиколлинеарность; интеркорреляция; квазилинейная регрессия; неэлементарная регрессия; отбор информативных регрессоров; задача частично-булевого линейного программирования; критерий нелинейности.

Для цитирования: Базилевский М.П. Программа построения вполне интерпретируемых элементарных и неэлементарных квазилинейных регрессионных моделей. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 129–144. DOI: 10.15514/ISPRAS–2023–35(4)–7.

Program for Constructing Quite Interpretable Elementary and Non-elementary Quasi-linear Regression Models

M.P. Bazilevskiy, ORCID: 0000-0002-3253-5697 <*mik2178@yandex.ru*>

Irkutsk State Transport University,
15, Chernyshevskogo st., Irkutsk, 664074, Russia.

Abstract. A quite interpretable linear regression satisfies the following conditions: the signs of its coefficients correspond to the meaningful meaning of the factors; multicollinearity is negligible; coefficients are significant; the quality of the model approximation is high. Previously, to construct such models, estimated using the ordinary least squares, the QInter-1 program was developed. In it, according to the given initial parameters, the mixed integer 0-1 linear programming task is automatically generated, as a result of which the most informative regressors are selected. The mathematical apparatus underlying this program was significantly expanded over time: non-elementary linear regressions were developed, linear restrictions on the absolute values of intercorrelations were proposed to control multicollinearity, assumptions appeared about the possibility of constructing not only linear, but also quasi-linear regressions. This article is devoted to the description of the developed second version of the program for constructing quite interpretable regressions QInter-2. The QInter-2 program allows, depending on the initial parameters selected by the user, to automatically formulate for the LPSolve solver the mixed integer 0-1 linear programming task for constructing both elementary and non-elementary quite interpretable quasi-linear regressions. It is possible to set up to nine elementary functions and control such parameters as the number of regressors in the model, the number of signs in real numbers after the decimal point, the absolute contributions of variables to the overall determination, the number of occurrences of explanatory variables in the model, and the magnitude of intercorrelations. In the process of working with the program, you can also control the number of elementary and non-elementarily transformed variables that affect the speed of solving the mixed integer 0-1 linear programming task. The QInter-2 program is universal and can be used to construct quite interpretable mathematical dependencies in various subject areas.

Keywords: linear regression; quite interpretable regression; ordinary least squares; multicollinearity; intercorrelation; quasi-linear regression; non-elementary regression; subset selection in regression; mixed integer 0-1 linear programming; non-linearity criterion.

For citation: Bazilevskiy M.P. Program for constructing quite interpretable elementary and non-elementary quasi-linear regression models. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 4, 2023. pp. 129-144 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-7.

1. Введение

Построение интерпретируемых моделей машинного обучения [1,2] в настоящее время является актуальной научной задачей. Наиболее просто интерпретируемыми моделями машинного обучения справедливо можно считать линейные регрессионные модели [3]. В результате оценивания линейной регрессии всегда можно объяснить любой её коэффициент, за исключение, быть может, свободного члена. Но, несмотря на это, не каждую оцененную линейную регрессию можно отнести к интерпретируемым. Причины этого состоят в следующем:

1) мультиколлинеарность [4], которая затрудняет оценку степени влияния входных переменных на выходную и может искажать знаки оценок линейной регрессии;
2) неверное направление влияния входных переменных на выходную даже при отсутствии мультиколлинеарности, что говорит об игнорировании исследователем предварительного анализа влияния факторов на отклик;
3) незначимость некоторых коэффициентов линейной регрессии, что делает бессмысленной интерпретацию влияния на зависимую переменную y соответствующих объясняющих переменных;
4) низкое качество аппроксимации построенной линейной регрессии, которое сводит на нет весь процесс интерпретации.

Как отмечено в [1], универсального математического определения интерпретируемости модели машинного обучения не существует. Поэтому для определенности автором был введен термин “вполне интерпретируемая линейная регрессия” [5]. Такая регрессия удовлетворяет следующим условиям:

1) знаки коэффициентов оцененной модели соответствуют содержательному смыслу входящих в уравнение факторов;
2) эффект мультиколлинеарности незначителен.

Это определение можно дополнить ещё двумя условиями: все коэффициенты должны быть значимы, например, по t-критерию Стьюдента; качество аппроксимации линейной регрессии должно быть высоким.

Естественным образом, для построения вполне интерпретируемых линейных регрессий требуется соответствующее математическое и программное обеспечение.

Существующих на сегодняшний день алгоритмов построения интерпретируемых регрессионных моделей мало. Можно выделить лишь алгоритм, предложенный в монографии [6]. Множество работ посвящено лишь отбору информативных регрессоров (ОИР) [7] в линейной регрессии, что не гарантирует возможность её интерпретации. В [6] для решения задачи ОИР применен метод “всех регрессий” [8], который заключается в переборе всех возможных альтернативных вариантов моделей и выборе лучшей из них со всеми значимыми коэффициентами, минимальной остаточной дисперсией отклика и наилучшей интерпретируемостью. При этом линейные регрессии представляются в стандартизованном масштабе, что, как известно [9], упрощает процесс их оценивания с помощью метода наименьших квадратов (МНК). Тем не менее, метод “всех регрессий” трудно назвать эффективным.

За последние десятилетия была существенно развита технология решения задач частично-целочисленного программирования. При этом задача ОИР в линейной регрессии, оцениваемой с помощью МНК, в зарубежной литературе часто сводится к задаче частично-булевого квадратичного программирования [10–19]. Автору такую задачу удалось свести к задаче частично-булевого линейного программирования (ЧБЛП) [5, 20–24]. Постепенно получилось расширить эту задачу линейными ограничениями так, чтобы она гарантировала построение вполне интерпретируемой линейной регрессии. Предложенный метод построения вполне интерпретируемых линейных регрессий был реализован в виде программы ВИнтер-1. После чего задача расширялась, и в [23,24] был предложен метод построения вполне интерпретируемых неэлементарных линейных регрессий (НЛР). Также параллельно возникла идея в возможности построения не только элементарных и неэлементарных линейных, но и квазилинейных регрессий.

Целью данной статьи является описание разработанной второй версии программы ВИнтер-1, предназначенной для построения вполне интерпретируемых элементарных и неэлементарных линейных и квазилинейных регрессионных моделей методом решения автоматически сформулированной задачи ЧБЛП.

2. Элементарные и неэлементарные квазилинейные регрессии

Рассмотрим модель множественной линейной регрессии с объясняющими переменными x_j, $j=\overline{1,l}$:

$$y_i = \alpha_0 + \sum_{j=1}^{l} \alpha_j x_{ij} + \varepsilon_i, \qquad i=\overline{1,n}, \tag{1}$$

где y – объясняемая переменная; n – объем выборки; ε – вектор ошибок аппроксимации; α_0, α_1, ..., α_l – неизвестные параметры.

Для увеличения скорости вычислений МНК-оценок линейной регрессии (1) проведем нормирование всех переменных по формулам

$$y_i^{\bullet} = \frac{y_i - \overline{y}}{\sigma_y},\ x_{i1}^{\bullet} = \frac{x_{i1} - \overline{x_1}}{\sigma_{x_1}},\ ...,\ x_{il}^{\bullet} = \frac{x_{il} - \overline{x_l}}{\sigma_{x_l}},\qquad i = \overline{1,n},$$

где $\overline{y}$, $\overline{x_1}$, ..., $\overline{x_l}$ – средние значения переменных; σ_y, σ_{x_1}, ..., σ_{x_l} – среднеквадратичные отклонения переменных.

С использованием нормированных переменных введем для модели (1) стандартизованную линейную регрессию:

$$y_i^{\bullet} = \beta_1 x_{i1}^{\bullet} + \beta_2 x_{i2}^{\bullet} + ... + \beta_l x_{il}^{\bullet} + \varepsilon_i^{\bullet},\qquad i = \overline{1,n}, \tag{2}$$

где β_1, β_2, ..., β_l – неизвестные параметры; $\varepsilon^{\bullet}$ – вектор ошибок аппроксимации.

В [5] на основе модели (2) предложена следующая задача ЧБЛП построения вполне интерпретируемой линейной регрессии:

$$R^2 = \sum_{j=1}^{l} r_{yx_j} \cdot \beta_j \to \max, \tag{3}$$

$$-(1-\delta_j)\cdot M \le \sum_{k=1}^{l} r_{x_j x_k} \cdot \beta_k - r_{yx_j} \le (1-\delta_j)\cdot M,\quad j = \overline{1,l}, \tag{4}$$

$$0 \le \beta_j \le \delta_j \cdot M,\qquad j \in J^+, \tag{5}$$

$$-\delta_j \cdot M \le \beta_j \le 0,\qquad j \in J^-, \tag{6}$$

$$\delta_j \in \{0,1\},\quad j = \overline{1,l}, \tag{7}$$

$$C_{x_j}^{\text{абс}} = r_{yx_j} \cdot \beta_j \ge \theta \cdot \delta_j,\quad j = \overline{1,l}, \tag{8}$$

где R^2 – коэффициент детерминации; $r_{x_i x_j}$ – коэффициент корреляции (интеркорреляция) между i-й и j-й объясняющей переменной; r_{yx_j} – коэффициент корреляции между объясняемой переменной y и j-й объясняющей переменной; M – большое положительное число; δ_j – бинарная переменная, принимающая значение 0, если j-я переменная не входит в модель, и значение 1, если входит; J^- и J^+ – множества объясняющих переменных, удовлетворяющих условиям $r_{yx_j} < 0$ и $r_{yx_j} > 0$ соответственно; $C_{x_j}^{\text{абс}} = r_{yx_j} \cdot \beta_j$ – абсолютный вклад j-й переменной в общую детерминацию R^2; $\theta \ge 0$ – нижняя граница абсолютных вкладов.

Заметим, что ограничения (4) предназначены для включения/исключения уравнений в систему линейных алгебраических уравнений $\sum_{k=1}^{l} r_{x_j x_k} \cdot \beta_k = r_{yx_j}$, $j = \overline{1,l}$, с помощью которой находятся МНК-оценки. Если $\delta_j = 0$, то из этой системы исключается j-е уравнение и коэффициент β_j принимает значение 0, то есть осуществляется МНК-оценивание без участия переменной x_j. Если $\delta_j = 1$, то из системы j-е уравнение не исключается и коэффициент β_j может принимать любое значение, согласованное по знаку с коэффициентом корреляции r_{yx_j}, то есть осуществляется МНК-оценивание с участием переменной x_j.

В [20] задача ЧБЛП (3) – (8) дополнена ограничениями на абсолютные величины интеркорреляций:

$$\left|r_{x_i x_j}\right|\left(\delta_i+\delta_j-1\right)\le r\,,\quad i=\overline{1,l-1}\,,\qquad j=\overline{i+1,l}\,, \tag{9}$$

где $0<r\le 1$ – верхняя граница интеркорреляций.

Решение задачи ЧБЛП (3) – (9) приводит к построению вполне интерпретируемой линейной регрессии с оптимальным по критерию R^2 количеством объясняющих переменных из множества Φ, в которой $\tilde{\beta}_j\cdot r_{yx_j}>0$, $j\in\Phi$, вклады $C_{x_j}^{\text{абс}}\ge\theta$, $j\in\Phi$, и интеркорреляции $\left|r_{x_i x_j}\right|\le r$, $i,j\in\Phi$.

В задаче (3) – (9) для контроля значимости используются абсолютные вклады переменных, а для контроля мультиколлинеарности – величины интеркорреляций. При желании исследователь может дополнить формализацию другими известными критериями. Например, в [21] обсуждается, как в задаче (3) – (9) можно контролировать факторы вздутия дисперсии (VIF), а в [22] – значимость оценок по t-критерию Стьюдента.

Отметим, что выбор в задаче (3) – (9) значения параметра M обсуждается в [5].

Аналогично можно сформулировать задачу ЧБЛП для построения вполне интерпретируемой квазилинейной регрессии. Пусть в распоряжении исследователя имеется elem элементарных функций: $f_1(x)$, $f_2(x)$, ..., $f_{\text{elem}}(x)$. Пусть $x_{ij}>0$, $i=\overline{1,n}$, $j=\overline{1,l}$. С помощью этих функций получим множество элементарно преобразованных переменных z_{jk}, $j=\overline{1,l}$, $k=\overline{1,\text{elem}}$, где z_{jk} – k-е преобразование j-й переменной. Тогда модель квазилинейной регрессии можно записать в виде:

$$y_i=\alpha_0+\sum_{j=1}^{l}\sum_{k=1}^{\text{elem}}\alpha_{jk}\cdot z_{ijk}+\varepsilon_i\,,\qquad i=\overline{1,n}\,. \tag{10}$$

Предварительно для построения модели (10) необходимо сократить список z-переменных по следующим причинам.

1. Иногда знак коэффициента корреляции преобразованной переменной с y может противоречить содержательному смыслу задачи.
2. Преобразование может быть в значительной степени нелинейным, что затрудняет объяснение соответствующего коэффициента.

Допустим, что все элементарные функции монотонны на отрезках $\left[x_{\min}^j,x_{\max}^j\right]$, $j=\overline{1,l}$. Тогда степень нелинейности z-переменных можно оценить с помощью следующих критериев нелинейности [25]:

$$NC_{z_{jk}}=\left|\frac{f_k(x_{\max}^j)+f_k(x_{\min}^j)}{f_k(x_{\max}^j)-f_k(x_{\min}^j)}-\frac{2\int_{x_{\min}^j}^{x_{\max}^j}f_k(x_j)dx_j}{\left(x_{\max}^j-x_{\min}^j\right)\left(f_k(x_{\max}^j)-f_k(x_{\min}^j)\right)}\right|,\ j=\overline{1,l}\,,\ k=\overline{1,\text{elem}}\,. \tag{11}$$

Если $NC_{z_{jk}}\le 0{,}2$, то вместо оценки $\tilde{\alpha}_{jk}$ при k-м преобразовании j-й переменной в модели (10) можно объяснить величину $\tilde{\alpha}_{jk}\dfrac{f_k\left(x_{\max}^j\right)-f_k\left(x_{\min}^j\right)}{x_{\max}^j-x_{\min}^j}$.

Таким образом, сократив список z -переменных, несложно по аналогии с задачей ЧБЛП (3) – (9) сформулировать задачу построения вполне интерпретируемой квазилинейной регрессии. Для возможности её интерпретации необходимо дополнить задачу ЧБЛП ограничением на единственность вхождения каждой объясняющей переменной в модель.

В [23] предложена неэлементарная линейная регрессия (НЛР) вида

$$y_i = \alpha_0 + \sum_{j=1}^{l} \alpha_j x_{ij} + \sum_{j=1}^{p} \alpha_j^{\min} \min\{x_{i,\mu_{j1}}, k_j^{\min} x_{i,\mu_{j2}}\} +$$

$$+ \sum_{j=1}^{p} \alpha_j^{\max} \max\{x_{i,\mu_{j1}}, k_j^{\max} x_{i,\mu_{j2}}\} + \varepsilon_i, \qquad i = \overline{1,n}, \tag{12}$$

где $p = C_l^2$ – количество комбинаций пар объясняющих переменных; min, max – бинарные операции, возвращающие минимум и максимум из двух чисел; μ_{j1}, μ_{j2}, $j = \overline{1,p}$ – элементы матрицы $\mathrm{M}_{p\times 2}$, содержащей в строках все возможные комбинации пар объясняющих переменных; $\alpha_j^{\min}$, $\alpha_j^{\max}$, $k_j^{\min}$, $k_j^{\max}$, $j = \overline{1,p}$ – неизвестные параметры.

Оптимальные оценки параметров $k_j^{\min}$, $k_j^{\max}$, $j = \overline{1,p}$, лежат внутри промежутков [23]

$$\left(k_{\text{нижн}}^j, k_{\text{верхн}}^j\right), \qquad j = \overline{1,p},$$

где $k_{\text{нижн}}^j = \min\left\{\frac{x_{1,\mu_{j1}}}{x_{1,\mu_{j2}}}, \frac{x_{2,\mu_{j1}}}{x_{2,\mu_{j2}}}, ..., \frac{x_{n,\mu_{j1}}}{x_{n,\mu_{j2}}}\right\}$, $k_{\text{верхн}}^j = \max\left\{\frac{x_{1,\mu_{j1}}}{x_{1,\mu_{j2}}}, \frac{x_{2,\mu_{j1}}}{x_{2,\mu_{j2}}}, ..., \frac{x_{n,\mu_{j1}}}{x_{n,\mu_{j2}}}\right\}$.

Разбивая эти промежутки на Razb точек, методом полного перебора можно получить приближенные МНК-оценки НЛР (12).

В [23] задача построения НЛР (12) сведена к задаче ЧБЛП. А в [24] эта задача расширена ограничениями, позволяющими строить вполне интерпретируемую НЛР. Для этого предварительно необходимо исключать неэлементарно преобразованные переменные, знак коэффициента корреляции которых с y не согласуется хотя бы с одним из знаков коэффициентов корреляции с y входящих в это преобразование переменных.

На основе (10) введем неэлементарную квазилинейную регрессию (НКР) вида

$$y_i = \alpha_0 + \sum_{j=1}^{l} \sum_{k=1}^{\text{elem}} \alpha_{jk} z_{ijk} + \sum_{j=1}^{p^*} \alpha_j^{\min} \min\{z_{i,\mu^*_{j,1,1},\mu^*_{j,1,2}}, k_j^{\min} z_{i,\mu^*_{j,2,1},\mu^*_{j,2,2}}\} +$$

$$+ \sum_{j=1}^{p^*} \alpha_j^{\max} \max\{z_{i,\mu^*_{j,1,1},\mu^*_{j,1,2}}, k_j^{\max} z_{i,\mu^*_{j,2,1},\mu^*_{j,2,2}}\} + \varepsilon_i, \qquad i = \overline{1,n}, \tag{13}$$

где $p^* = C_{l\cdot\text{elem}}^2$ – количество комбинаций пар элементарно преобразованных переменных; $\mu^*_{j,1,1}$, $\mu^*_{j,1,2}$, $\mu^*_{j,2,1}$, $\mu^*_{j,2,2}$, $j = \overline{1,p^*}$ – элементы трехмерного массива $\mathrm{M}^*_{p^*,2,2}$, содержащего на горизонтальных листах в первом столбце номера переменных, во втором – номера преобразований; $\alpha_j^{\min}$, $\alpha_j^{\max}$, $k_j^{\min}$, $k_j^{\max}$, $j = \overline{1,p^*}$ – неизвестные параметры.

Задача построения вполне интерпретируемой НКР (13) по аналогии с приёмами, рассмотренными в [24], сводится к задаче ЧБЛП. При этом для обеспечения интерпретируемости НКР требуется из трехмерного массива $\mathrm{M}^*_{p^*,2,2}$ исключать горизонтальные листы, содержащие одну и ту же объясняющую переменную.

3. Программа ВИнтер-2

Программа построения вполне интерпретируемых элементарных и неэлементарных квазилинейных регрессий (ВИнтер-2) была разработана в среде программирования Delphi.

Блок-схема алгоритма работы ВИнтер-2 представлена на рис. 1. После запуска программы сначала нужно загрузить текстовый файл формата ".txt" со статистическими данными. В первом столбце этого файла содержатся значения объясняемой переменной y, во втором – объясняющей переменной x_1 и так далее. Столбцы отделяются друг от друга табуляцией. Имена переменных в первой строке вводить не нужно. Целые и дробные части действительных чисел отделяются друг от друга символом ",". Важно, чтобы знаки коэффициентов корреляции r_{yx_j}, $j=\overline{1,l}$, были согласованы с содержательным смыслом всех расположенных в текстовом файле переменных, т.е. предварительно с факторами должны поработать эксперты из данной предметной области.

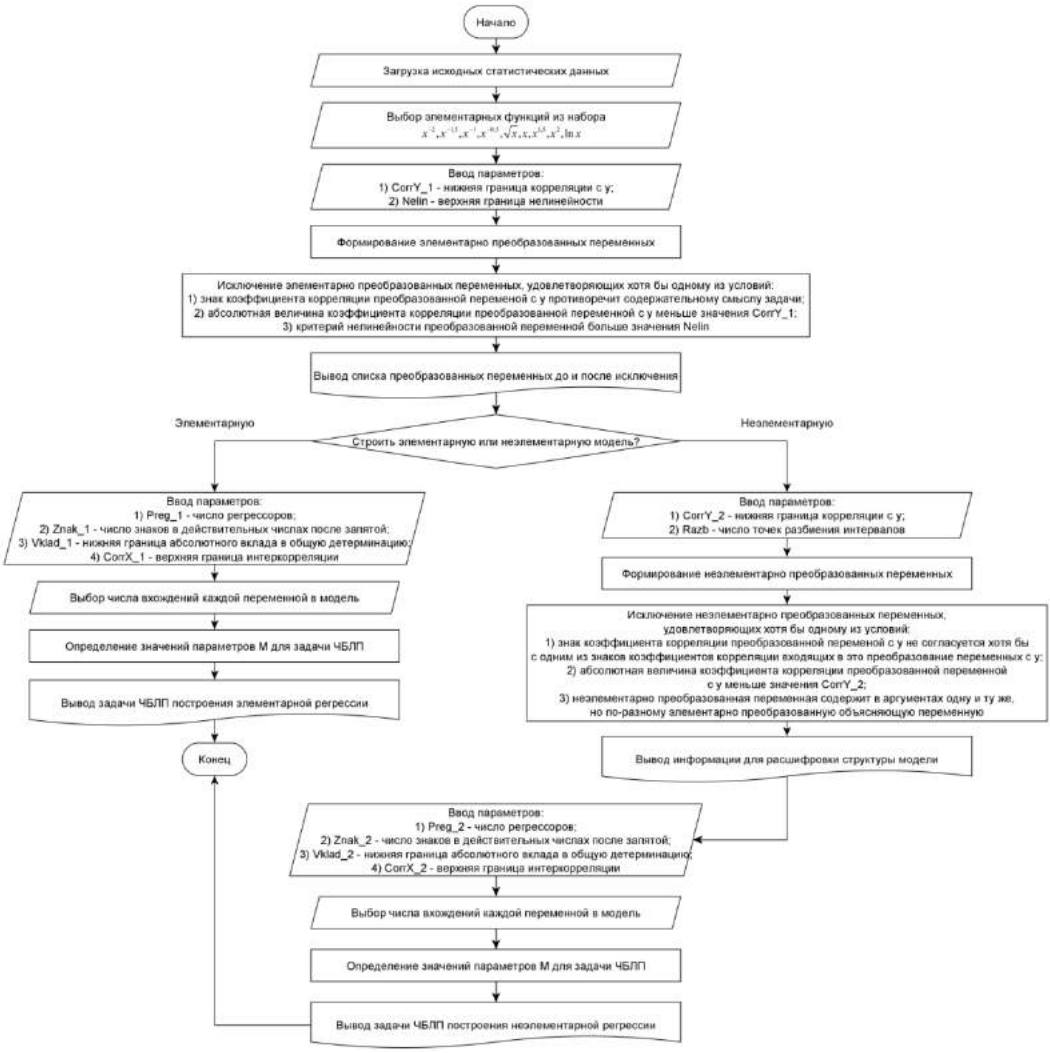

Рис. 1. Блок-схема алгоритма работы программы
Fig. 1. Block diagram of the program operation algorithm

Если всё сделано верно, то данные отобразятся в поле “Данные” главной формы системы (рис. 2). При этом будет сформирована матрица D, содержащая наблюдаемые значения переменных, корреляционная матрица KM для матрицы D, а также матрица MinMax, содержащая в первой и второй строке минимальные и максимальные значения всех переменных, и матрица MNC размера $l \times 9$, содержащая значения всех критериев нелинейности (11) переменных, преобразованных с помощью элементарных функций x^{-2}, $x^{-1,5}$, x^{-1}, $x^{-0,5}$, $x^{0,5}$, x, $x^{1,5}$, x^2 и $\ln x$.

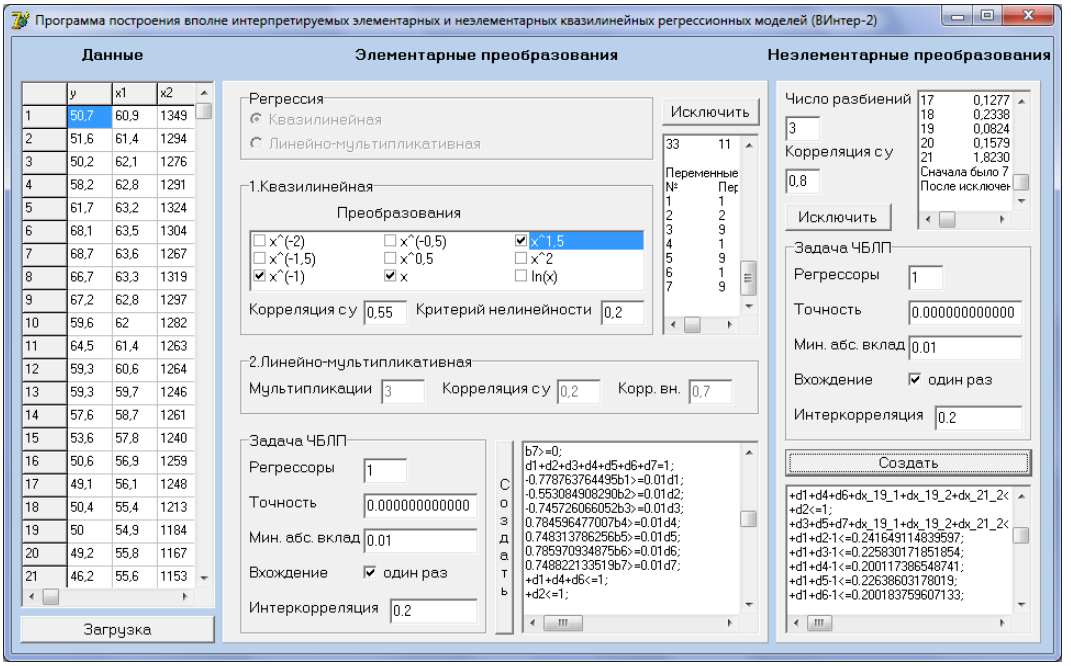

Рис. 2. Главная форма программы
Fig. 2. Main form of the program

Далее на панели “1.Квазилинейная” необходимо выбрать элементарные функции для преобразования переменных, кликнув на соответствующие флажки, а также ввести вручную нижнюю границу корреляции с y – CorrY_1, и верхнюю границу нелинейности – Nelin. Затем нажать кнопку “Исключить”. В результате нажатия программа выполняет следующую последовательность действий.

1. Формируется вектор номеров выбранных элементарных функций CheckPreobr из s элементов.
2. Формируется матрица данных D2 размера $n \times (1 + s \cdot l)$, содержащая значения всех элементарно преобразованных переменных.
3. Формируется корреляционная матрица KM2 по матрице данных D2.
4. Формируется матрица Shifr размера $(l \cdot s) \times 5$, в первом столбце которой содержатся номера исходных переменных, во втором – номера элементарных функций из вектора CheckPreobr. Остальные столбцы заполняются единицами и нулями по следующим правилам. Если знак коэффициента корреляции преобразованной переменной с y не противоречит содержательному смыслу, то в третий столбец вносится “0”, иначе “1”. Если абсолютная величина коэффициента корреляции преобразованной переменной с y не меньше значения CorrY_1, то в четвертый столбец вносится “0”, иначе “1”.

Если значения критерия нелинейности преобразованной переменной не больше значения Nelin, то в пятый столбец вносится “0”, иначе “1”. Например, если матрица Shifr имеет вид

$$\begin{pmatrix} 2 & 3 & 1 & 0 & 1 \\ 4 & 7 & 0 & 1 & 1 \end{pmatrix},$$

то это означает, что преобразование x_2^{-1} противоречит содержательному смыслу задачи, критерий нелинейности выше необходимого значения Nelin, но коэффициент корреляции с y по модулю выше требуемого значения CorrY_1; преобразование $x_4^{1,5}$ слабо коррелирует с y, имеет высокую степень нелинейности, но не противоречит содержательному смыслу задачи.

5. Осуществляется исключение тех преобразований, которые либо противоречат содержательному смыслу задачи, либо слабо коррелируют с y, либо имеют высокую степень нелинейности, т.е. из матрицы Shifr исключаются те строки, в которых сумма элементов последних трех столбцов больше нуля. В результате исключения из матрицы Shifr формируется новая матрица GShifr размера $\text{Lev1Per} \times 5$, где Lev1Per – число преобразований, для которых выполняются все 3 условия. Также формируется новая матрица данных D3.
6. В соответствующем поле главной формы ВИнтер-2 выводятся элементы матриц Shifr и GShifr. Тем самым, регулируя параметры CorrY_1 и Nelin, можно снижать количество участвующих в процессе моделирования элементарно преобразованных переменных, что повышает скорость построения регрессионной модели.

Заметим, во-первых, что пока в систему встроено только 9 элементарных функций. Причём, хотя бы одна из них обязательно должна быть выбрана. Для построения линейной регрессии нужно выбрать только один флажок с именем “x”. Во-вторых, если выбрать CorrY_1 = 0, Nelin = 1, то требования для сильной корреляции и слабой нелинейности выбранных преобразований будут игнорироваться. В-третьих, как видно по главной форме (рис. 2), неактивна панель “2.Линейно-мультипликативная”. Механизм построения такого вида регрессий пока находится на стадии тестирования.

После исключения элементарно преобразованных переменных нужно выбрать, какая модель будет строиться – элементарная (10) или неэлементарная (13). Если пользователь желает построить элементарную регрессию (10), то на панели “Задача ЧБЛП”, расположенной в нижней части главной формы, нужно задать следующие параметры: Preg_1, Znak_1, Vklad_1, CorrX_1. Описание этих параметров представлено на рис. 1. Также для обеспечения интерпретируемости модели нужно выбрать флажок “один раз”, означающий, что каждая объясняющая переменная должна входить в регрессию не более одного раза. Затем нужно нажать кнопку “Создать”. В результате нажатия программа выполняет следующую последовательность действий.

1. Формируется корреляционная матрица KM3 по матрице данных D3.
2. Определяются значения параметров M для линейных ограничений типа (5), (6) [5]. В результате формируется вектор Mb из Lev1Per элементов. Затем автоматически определяются значения параметров M для линейных ограничений типа (4) [5]. Для этого в цикле формируются необходимые задачи линейного программирования, которые последовательно передаются решателю LPSolve IDE, а результаты решения снова возвращаются в ВИнтер-2. В результате формируется матрица Mq размера $\text{Lev1Per} \times 2$, в строках которой содержатся нижние и верхние границы параметров M для ограничений типа (4).
3. Осуществляется формализация задачи ЧБЛП для пакета LPSolve. Формируется целевая функция типа (3) и линейные ограничения типа (4) – (9). Для обеспечения

единственности вхождения в модель каждой объясняющей переменной формируется матрица GVh размера $\text{Lev1Per} \times l$, состоящая из нулей и единиц. Эта матрица связывает преобразованные переменные с исходными факторами и заполняется по следующему правилу: если в k-й строке матрицы GShifr во втором столбце стоит номер h, то в матрице GVh на пересечении k-й строки и h-го столбца ставится “1”. В этой связи линейные ограничения на количество входящих в модель объясняющих переменных имеют вид:

$$\sum_{k=1}^{\text{Lev1Per}} \text{GVh}_{kj} \cdot \delta_k \le 1, \quad j = \overline{1, l}. \tag{14}$$

4. В соответствующем поле главной формы программы ВИнтер-2 выводится сформулированная задача ЧБЛП для построения элементарной регрессии (10).

Если же пользователь желает построить неэлементарную регрессию (13), то сначала необходимо из оставшихся на первом шаге элементарно преобразованных переменных сформировать неэлементарно преобразованные переменные. Для этого на панели, расположенной в правом верхнем углу главной формы, нужно задать следующие параметры: Razb – число точек разбиения интервалов $\left(k^j_{\text{нижн}}, k^j_{\text{верхн}}\right)$, $j = \overline{1, p^*}$, CorrY_2 – нижняя граница корреляции с y. С помощью этих параметров регулируется число неэлементарно преобразованных переменных. Затем нужно нажать кнопку “Исключить”. В результате нажатия система выполняет следующую последовательность действий.

1. Формируется матрица Ind размера $C^2_{\text{Lev1Per}} \times 2$, содержащая по строкам все возможные комбинации пар элементарно преобразованных переменных.
2. С помощью матрицы Ind формируется матрица LAM размера $C^2_{\text{Lev1Per}} \times 2$, содержащая в строках нижнюю и верхнюю границу параметров $k^{\min}_j$, $k^{\max}_j$, $j = \overline{1, p^*}$, для неэлементарных преобразований переменных.
3. С помощью матрицы LAM формируется матрица R размера $C^2_{\text{Lev1Per}} \times \text{Razb}$, содержащая в каждой строке Razb точек, равномерно разбивающих отрезки $\left(k^j_{\text{нижн}}, k^j_{\text{верхн}}\right)$, $j = \overline{1, p^*}$.
4. С помощью матриц Ind и R формируется матрица данных ND2 размера $n \times \left(1 + \text{Lev1Per} + C^2_{\text{Lev1Per}} \cdot \text{Razb} \cdot 2\right)$, содержащая в первом столбце значения переменной y, в последующих Lev1Per столбцах – значения элементарно преобразованных переменных, в последующих $C^2_{\text{Lev1Per}} \times \text{Razb}$ столбцах – значения переменных, преобразованных с помощью неэлементарной функции min, и в последующих $C^2_{\text{Lev1Per}} \times \text{Razb}$ столбцах – значения переменных, преобразованных с помощью неэлементарной функции max.
5. Формируется корреляционная матрица NKM по матрице данных ND2.
6. Формируется матрица NShifr размера $\left(\text{Lev1Per} + 2 \cdot C^2_{\text{Lev1Per}} \cdot \text{Razb}\right) \times 3$, содержащая информацию о всех преобразованных переменных. В первом её столбце содержится информация о типе преобразованной переменной (1 – если элементарная функция, 2 – если неэлементарная функция min, 3 – если неэлементарная функция max), во втором – для элементарной функции содержится номер объясняющей переменной, а для неэлементарной – номер пары объясняющих переменных в матрице Ind, в третьем – для элементарной функции запись отсутствует, а для неэлементарной функции указывается номер точки разбиения в матрице R.

7. Исключаются такие неэлементарно преобразованные переменные, для которых выполняется хотя бы одно из трех условий: знак коэффициента корреляции неэлементарно преобразованной переменной с y не согласуется со знаком хотя бы одного коэффициента корреляции входящей в данное преобразование переменной с y; значение коэффициента корреляции неэлементарно преобразованной переменной с y по модулю меньше величины CorrY_2; неэлементарная функция содержит в аргументах одну и ту же, но преобразованную разными элементарными функциями, объясняющую переменную. В результате исключения вместо матрицы NShifr формируется новая матрица GNShifr размера $(\text{Lev1Per}+\text{numU})\times 3$, где numU – количество неэлементарно преобразованных переменных, для которых не выполнено ни одно из трех перечисленных условий.
8. Из оставшихся после исключения переменных формируется матрица данных ND3 размера $n\times(1+\text{Lev1Per}+\text{numU})$.
9. В правом верхнем углу главной формы программы в соответствующем поле выводится матрица Ind и матрица R, с помощью которых после построения модели (13) можно расшифровать её структурную спецификацию. Также формируется информация о том, сколько неэлементарно преобразованных переменных было изначально, и сколько осталось после исключения.

После формирования неэлементарно преобразованных переменных на панели “Задача ЧБЛП”, расположенной в правой части главной формы ВИнтер-2, необходимо задать следующие параметры: Preg_2, Znak_2, Vklad_2, CorrX_2. Для обеспечения единственности вхождения объясняющих переменных в модель нужно выбрать флажок “один раз”. Затем нужно нажать кнопку “Создать”. В результате нажатия программа выполняет следующую последовательность действий.

1. Формируется корреляционная матрица KM по матрице данных ND3.
2. Определяются значения параметров M [23] для линейных ограничений.
3. Осуществляется формализация задачи ЧБЛП для пакета LPSolve. Имена МНК оценок делятся на 3 типа: “b” – для элементарных преобразований, “bm_” – для неэлементарных преобразований min, “bx_” – для неэлементарных преобразований max. Аналогично создаются имена бинарных переменных – “d”, “dm_” и “dx_”. Для обеспечения единственности вхождения в модель каждой объясняющей переменной с помощью матрицы GNShifr формируется матрица Vh размера $(\text{Lev1Per}+\text{numU})\times 1$, состоящая из нулей и единиц. Единица на пересечении в ней i-й строки и j-го столбца означает, что i-е преобразованием включает в себя j-ю объясняющую переменную. Линейные ограничения на количество вхождений объясняющих переменных в модель составляются аналогично (14). Также с помощью GNShifr формируется матрица Vh2 размера $(\text{Lev1Per}+\text{numU})\times\text{Lev1Per}$, в которой единица на пересечении i-й строки и j-го столбца означает, что i-е преобразование включает в себя j-ю элементарно преобразованную переменную. Помимо этого формируются ограничения на интеркорреляции регрессоров, входящих в НКР (13), и с помощью матрицы Vh2 – на интеркорреляции элементарно преобразованных переменных, входящих в НКР (13).
4. В соответствующем поле главной формы программы ВИнтер-2 выводится сформулированная задача ЧБЛП для построения неэлементарной регрессии (13).

4. Пример

Для демонстрации разработанной программы ВИнтер-2 проводилось моделирование по встроенным в эконометрических пакет Gretl данным о ценах и характеристиках произведенных в Америке автомобилей (data7-12.gdt). Эта выборка содержит одну объясняемую переменную и 10 независимых. Объем выборки составляет 82. Для возможности работы со всеми элементарными преобразованиями, из выборки были исключены фиктивные переменные hatch и trans. Оставшиеся переменные были переобозначены следующим образом: price – y, wbase – x_1, length – x_2, width – x_3, height – x_4, weight – x_5, cyl – x_6, liters – x_7, gasmpg – x_8. Корреляционная матрица для всех этих переменных представлена на рис. 3.

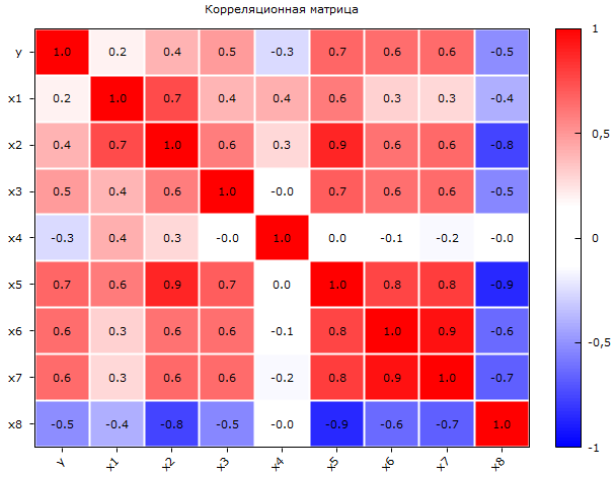

Рис. 3. Корреляционная матрица
Fig. 3. Correlation matrix

Как видно по рис. 3, есть переменные, которые очень слабо коррелируют, например, x_4 и x_5, а есть такие, которые коррелируют сильно, например, x_2 и x_5. Для уровня значимости $\alpha = 0{,}1$ было установлено, что коэффициент корреляции незначим, если его значение по модулю меньше 0,18, и значим в противном случае. Таким образом, все коэффициенты корреляции объясняющих переменных с y значимы на уровне $\alpha = 0{,}1$. Будем считать, что знаки этих коэффициентов соответствуют содержательному смыслу факторов, т.е. чем выше колесная база x_1, тем выше цена y, чем выше длина автомобиля x_2, тем выше цена y и т.д.

Сначала в ВИнтер-2 строилась элементарная линейная регрессия (1). Для исключения переменных были заданы параметры CorrY_1 = 0,18 и Nelin = 0,2. В результате ни одна объясняющая не была исключена. Для формирования задачи ЧБЛП были заданы следующие параметры: Preg_1 = 0 (означает, что нет ограничения на число регрессоров), Znak_1 = 12, Vklad_1 = 0,05, CorrX_1 = 0,18. Флажок “Вхождение” включен. Сформированная в результате задача ЧБЛП была решена в пакете LPSolve IDE. В итоге была построена следующая линейная регрессия:

$$\tilde{y} = 62{,}0819 - \underset{(-3{,}869)}{1{,}53052}\, x_4 + \underset{(8{,}610)}{1{,}26729}\, x_5, \qquad (15)$$

коэффициент детерминации которой $R^2 = 0{,}521219$. Полученное значение гораздо ниже идеального значения 1, но для выборки объема 82 это довольно неплохой результат, тем более, что модель (15) значима по критерию Фишера.

В уравнении (15) в скобках под коэффициентами указаны наблюдаемые значения t-критерия Стьюдента, подтверждающие значимость МНК-оценок для уровня $\alpha = 0,01$.

Знаки коэффициентов уравнения (15) совпадают со знаками соответствующих коэффициентов корреляции (рис. 3). А коэффициент корреляции между x_4 и x_5 незначим (рис. 3), что говорит от отсутствии мультиколлинеарности. Таким образом, выполняются все условия, чтобы можно было отнести линейную регрессию (15) ко вполне интерпретируемым.

Затем строилась элементарная квазилинейная регрессия (10). Для этого в поле "Преобразования" были выбраны все 9 элементарных функций. Для исключения переменных были заданы параметры CorrY_1 = 0,18 и Nelin = 0,2. В результате исключения осталось 50 элементарно преобразованных переменных. Для формирования задачи ЧБЛП были заданы следующие параметры: Preg_1 = 0, Znak_1 = 12, Vklad_1 = 0,05, CorrX_1 = 0,18. Флажок "Вхождение" включен. В результате была построена следующая квазилинейная регрессия:

$$\tilde{y} = -44,0643 + \underset{(4,728)}{118409}\, x_4^{-2} + \underset{(9,409)}{0,022611}\, x_5^2, \qquad (16)$$

для которой $R^2 = 0,577112$, что выше, чем у (15). Коэффициенты модели (16) значимы по t-критерию Стьюдента для уровня $\alpha = 0,01$, а их знаки не противоречат содержательному смыслу факторов. Коэффициент корреляции между регрессорами x_4^{-2} и x_5^2 равен -0,036, поэтому незначим. Из всего этого следует, что квазилинейная регрессия (16) является вполне интерпретируемой.

После чего строилась НЛР (12). Для этого использовались все 8 оставшихся после исключения на первом этапе объясняющих переменных. Для формирования и исключения неэлементарно преобразованных переменных были заданы параметры CorrY_2 = 0,18 и Razb = 3. В результате было сформировано 168 неэлементарно преобразованных переменных, которых после исключения осталось 96. Для формирования задачи ЧБЛП были заданы следующие параметры: Preg_2 = 0, Znak_2 = 12, Vklad_2 = 0,05, CorrX_2 = 0,18. Флажок "Вхождение" включен. В результате была построена следующая НЛР:

$$\tilde{y} = 50,1958 - \underset{(-3,915)}{1,50717}\, x_4 + \underset{(9,088)}{0,608225} \max\{x_3, 2.638063 x_5\}, \qquad (17)$$

для которой $R^2 = 0,546283$. Коэффициенты модели (17) значимы по t-критерию Стьюдента для уровня $\alpha = 0,01$. А если представить её в кусочно-заданной форме [23], то станет ясно, что и знаки коэффициентов не противоречат содержательному смыслу факторов. Коэффициент корреляции между регрессорами x_4 и $\max\{x_3, 2.638063 x_5\}$ равен 0,0417, поэтому незначим. Также незначимы коэффициенты корреляции $r_{x_3 x_4}$ и $r_{x_4 x_5}$ (рис. 3), следовательно, НЛР (17) относится ко вполне интерпретируемым.

Далее строилась НКР (13). Для этого использовались все 50 оставшихся после исключения на первом этапе элементарно преобразованных переменных. Для формирования и исключения неэлементарно преобразованных переменных были заданы параметры CorrY_2 = 0,7 и Razb = 3. В результате было сформировано 7350 неэлементарно преобразованных переменных, которых после исключения осталось 43. Для формирования задачи ЧБЛП были заданы следующие параметры: Preg_2 = 0, Znak_2 = 12, Vklad_2 = 0,05, CorrX_2 = 0,18. Флажок "Вхождение" включен. В результате была построена следующая НКР:

$$\tilde{y} = -247,674 + \underset{(10,78)}{94990,1} \max\{x_4^{-1,5}, 0,000534692 \ln x_2\}, \qquad (18)$$

для которой $R^2 = 0,592322$, что выше, чем у (15) – (17). Коэффициент модели (18) значим по t-критерию Стьюдента для уровня $\alpha = 0,01$, а его знак не противоречат содержательному смыслу факторов. Таким образом, НКР (18) можно считать вполне интерпретируемой.

Представим НКР (18) как кусочно-заданную функцию:

$$\tilde{y} = \begin{cases} -247,674 + 94990,1x_4^{-1,5}, \text{ при } \dfrac{x_4^{-1,5}}{\ln x_2} \geq 0,000534692, \\ -247,674 + 50,79\ln x_2, \text{ при } \dfrac{x_4^{-1,5}}{\ln x_2} < 0,000534692. \end{cases}$$

В таком виде интерпретировать влияние переменных x_2 и x_4 на y затруднительно, поскольку обе они преобразованы с помощью элементарных функций. Но известно, что критерии нелинейности этих преобразованных переменных меньше 0,2, поэтому, как отмечено выше, вместо оценок 50,79 и 94990,1 можно объяснить величины 0,287 и -7,853 соответственно, найденные по формулам $\tilde{\alpha}_{jk} \dfrac{f_k\left(x^j_{\max}\right) - f_k\left(x^j_{\min}\right)}{x^j_{\max} - x^j_{\min}}$. Тогда справедлива следующая интерпретация НКР (18).

1. Длина автомобиля (x_2) влияет на его цену (y) только при условии $\dfrac{x_4^{-1,5}}{\ln x_2} < 0,000534692$, причём, с увеличением x_2 на 10 дюймов цена y возрастает в среднем на 2873 долларов.
2. Высота автомобиля (x_4) влияет на y только при условии $\dfrac{x_4^{-1,5}}{\ln x_2} \geq 0,000534692$, причём, с увеличением x_4 на 1 дюйм цена y убывает в среднем на 7853 долларов.

5. Заключение

В статье представлена разработанная автором программа ВИнтер-2, предназначенная для построения вполне интерпретируемых элементарных и неэлементарных квазилинейных регрессионных моделей. В процессе построения регрессии в ВИнтер-2 можно контролировать количество регрессоров, степень их корреляции с y, степень нелинейности элементарных преобразований, число знаков после запятой в действительных числах, абсолютные вклады переменных в общую детерминацию и величины интеркорреляций. Разработка имеет высокое прикладное значение, поскольку с помощью неё можно решать реальные задачи анализа данных из абсолютно любых предметных областей. Причем, решать их довольно эффективно, поскольку вместо метода “всех регрессий” в ВИнтер-2 формируется задача ЧБЛП, методы решения которой были существенно развиты за последние годы. К тому же, результатом работы ВИнтер-2 является не просто регрессионная модель, по которой можно прогнозировать, но также и интерпретировать влияние каждой входящей в неё переменной на y. Кроме того, ВИнтер-2 позволяет строить уникальные неэлементарные регрессионные модели, позволяющие выявлять новые закономерности функционирования объектов исследования. Например, в [23,24] с помощью ВИнтер-2 уже строились НЛР функционирования железнодорожного транспорта в Иркутской и Тюменской областях. Однако впервые введенные в текущей статье НКР, представляющие собой более сложные зависимости, чем НЛР, а поэтому обладающие большим потенциалом, пока еще ни разу, за исключением небольшого примера в данной работе, не применялись на практике.

В дальнейшем планируется использовать ВИнтер-2 для решения широкого круга реальных прикладных задач, оснастить его возможностью контроля при построении модели факторов вздутия дисперсии [21] и t-критерия Стьюдента [22], а также реализовать в нём возможность построения других вполне интерпретируемых видов регрессий.

Список литературы / References

[1]. Molnar C. Interpretable machine learning. Lulu.com, 2020.

[2]. Doshi-Velez F., Kim B. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608, 2017.

[3]. Montgomery D. C., Peck E. A., Vining G. G. Introduction to linear regression analysis. John Wiley & Sons, 2021.

[4]. Shrestha N. Detecting multicollinearity in regression analysis. American Journal of Applied Mathematics and Statistics, vol. 8, no. 2, 2020, pp. 39-42.

[5]. Базилевский М.П. Построение вполне интерпретируемых линейных регрессионных моделей с помощью метода последовательного повышения абсолютных вкладов переменных в общую детерминацию. Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, ном. 2, 2022, стр. 5-16 / Bazilevskiy M.P. Construction of quite interpretable linear regression models using the method of successive increase the absolute contributions of variables to the general determination. Proceedings of Voronezh State University. Series: Systems Analysis and Information Technologies, no. 2, 2022, pp. 5-16. (in Russian).

[6]. Горбач А.Н., Цейтлин Н.А. Покупательское поведение: анализ спонтанных последовательностей и регрессионных моделей в маркетинговых исследованиях. Киев, Освіта УкраЇны, 2011, 220 с. / Gorbach A.N., Tseytlin N.A. Buying Behavior: Analysis of Spontaneous Sequences and Regression Models in Marketing Research. Kyiv, Education of Ukraine, 2011, 220 p. (in Russian).

[7]. Miller A. Subset selection in regression. CRC Press, 2002.

[8]. Себер Дж. Линейный регрессионный анализ. М., Издательство “Мир”, 1980, 456 с. / Seber Dzh. Linear Regression Analysis. Moscow, Mir Publishing House, 1980, 456 p. (in Russian).

[9]. Фёрстер Э., Рёнц Б. Методы корреляционного и регрессионного анализа. М., Финансы и статистика, 1983, 303 с. / Ferster E., Rents B. Methods of Correlation and Regression Analysis. Moscow, Finance and Statistics, 1983, 303 p. (in Russian).

[10]. Konno H., Yamamoto R. Choosing the best set of variables in regression analysis using integer programming. Journal of Global Optimization, 2009, vol. 44, pp. 273-282. DOI: 10.1007/s10898-008-9323-9.

[11]. Miyashiro R., Takano Y. Mixed integer second-order cone programming formulations for variable selection in linear regression. European Journal of Operational Research, 2015, vol. 247, pp. 721-731. DOI: 10.1016/j.ejor.2015.06.081.

[12]. Miyashiro R., Takano Y. Subset selection by Mallows’ Cp: A mixed integer programming approach. Expert Systems with Applications, 2015, vol. 42, pp. 325-331. DOI: 10.1016/j.eswa.2014.07.056.

[13]. Tamura R., Kobayashi K., Takano Y., Miyashiro R., Nakata K., Matsui T. Mixed integer quadratic optimization formulations for eliminating multicollinearity based on variance inflation factor. Journal of Global Optimization, 2019, vol. 73, pp. 431-446. DOI: 10.1007/s10898-018-0713-3.

[14]. Park Y.W., Klabjan D. Subset selection for multiple linear regression via optimization. Journal of Global Optimization, 2020, vol. 77, pp. 543-574. DOI: 10.1007/s10898-020-00876-1.

[15]. Takano Y., Miyashiro R. Best subset selection via cross-validation criterion. Top, 2020, vol. 28, no. 2, pp. 475-488. DOI: 10.1007/s11750-020-00538-1.

[16]. Bertsimas D., Li M.L. Scalable holistic linear regression. Operations Research Letters, 2020, vol. 48, no. 3, pp. 203-208. DOI: 10.1016/j.orl.2020.02.008.

[17]. Chung S., Park Y.W., Cheong T. A mathematical programming approach for integrated multiple linear regression subset selection and validation. Pattern Recognition, 2020, vol. 108. DOI: 10.1016/j.patcog.2020.107565.

[18]. Bertsimas D., Gurnee W. Learning sparse nonlinear dynamics via mixed-integer optimization. Nonlinear Dynamics, 2023, vol. 111, no. 7, pp. 6585-6604. DOI: 10.1007/s11071-022-08178-9.

[19]. Watanabe A., Tamura R., Takano Y., Miyashiro R. Branch-and-bound algorithm for optimal sparse canonical correlation analysis. Expert Systems with Applications, 2023, vol. 217, pp. 119530. DOI: 10.1016/j.eswa.2023.119530.

[20]. Базилевский М.П. Формализация процесса отбора информативных регрессоров в линейной регрессии в виде задачи частично-булевого линейного программирования с ограничениями на коэффициенты интеркорреляций. Современные наукоемкие технологии, ном. 8, 2023, стр. 10-14 / Bazilevskiy M.P. Formalization the subset selection process in linear regression as a mixed integer 0-1 linear programming problem with constraints on intercorrelation coefficients. Modern High Technologies, no. 8, 2023, pp. 10-14. (in Russian).

[21]. Базилевский М.П. Отбор информативных регрессоров с учётом мультиколлинеарности между ними в регрессионных моделях как задача частично-булевого линейного программирования. Моделирование, оптимизация и информационные технологии, том 6, ном. 2 (21), 2018, стр. 104-118 / Bazilevskiy M.P. Subset selection in regression models with considering multicollinearity as a task of mixed 0-1 integer linear programming. Modeling, Optimization and Information Technology, vol. 6, no. 2 (21), 2018, pp. 104-118. (in Russian).

[22]. Базилевский М.П. Отбор значимых по критерию Стьюдента информативных регрессоров в оцениваемых с помощью МНК регрессионных моделях как задача частично-булевого линейного программирования. Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, ном. 3, 2021, стр. 5-16 / Bazilevskiy M.P. Selection of informative regressors significant by Student's t-test in regression models estimated using OLS as a partial Boolean linear programming problem. Proceedings of Voronezh State University. Series: Systems Analysis and Information Technologies, no. 3, 2021, pp. 5-16. (in Russian).

[23]. Базилевский М.П. Метод построения неэлементарных линейных регрессий на основе аппарата математического программирования. Проблемы управления, ном. 4, 2022, стр. 3-14 / Bazilevskiy M.P. A method for constructing non-elementary linear regressions based on mathematical programming. Control Sciences, no. 4, 2022, pp. 3-14. (in Russian).

[24]. Базилевский М.П. Построение вполне интерпретируемых неэлементарных линейных регрессионных моделей. Вестник Югорского государственного университета, ном. 4 (67), 2022, стр. 105-114 / Bazilevskiy M.P. Construction of quite interpretable non-elementary linear regression models. Yugra State University Bulletin, no. 4 (67), 2022, pp. 105-114. (in Russian).

[25]. Базилевский М.П. Критерии нелинейности квазилинейных регрессионных моделей. Моделирование, оптимизация и информационные технологии, том 6, ном. 4 (23), 2018, стр. 185-195 / Bazilevskiy M.P. Nonlinear criteria of quasi-linear regression models. Modeling, Optimization and Information Technology, vol. 6, no. 4 (23), 2018, pp. 185-195. (in Russian).

Информация об авторах / Information about authors

Михаил Павлович БАЗИЛЕВСКИЙ – кандидат технических наук, доцент, доцент кафедры "Математика" Иркутского государственного университета путей сообщения. Сфера научных интересов: математическое моделирование, анализ данных, оптимизация, эконометрика, машинное обучение, искусственный интеллект.

Mikhail Pavlovich BAZILEVSKIY – Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of Mathematics of the Irkutsk State Transport University. Research interests: mathematical modeling, data analysis, optimization, econometrics, machine learning, artificial intelligence.

DOI: 10.15514/ISPRAS-2023-35(4)-8

Численное моделирование переноса твёрдых частиц в атмосферном городском пограничном слое с использованием лагранжева подхода: физические задачи и параллельная реализация

[1,2] *А.И. Варенцов*, ORCID: 0000-0001-7029-6773 <*aivarentsov98@gmail.com*>
[2] *О.А. Имеев*, ORCID: 0009-0008-5592-0759 <*ochir90@yandex.ru*>
[3] *А.В. Глазунов*, ORCID: 0000-0002-8780-3513 <*and.glas@gmail.com*>
[1,3] *Е.В. Мортиков*, ORCID: 0000-0002-9683-5701 <*evgeny.mortikov@gmail.com*>
[1,2] *В.М. Степаненко*, ORCID: 0000-0003-3033-6712 <*stepanen@srcc.msu.ru*>

[1] *Московский государственный университет имени М.В. Ломоносова*
Научно-исследовательский вычислительный центр
119234, Россия, г. Москва, Ленинские горы, д. 1, стр. 4
[2] *Институт физики атмосферы им. А.М. Обухова Российской академии наук*
119017, Россия, Пыжевский пер., 3
[3] *Институт вычислительной математики им. Г.И. Марчука Российской академии наук*
119333, г. Москва, ул. Губкина, 8

Аннотация. Работа представляет результаты развития численной модели лагранжева переноса частиц и применения методов параллельных вычислений для увеличения эффективности программной реализации модели. Модель реализована в виде программного комплекса, позволяющего проводить расчёты переноса и осаждения аэрозольных частиц с учётом свойств частиц и входных данных, описывающих атмосферные условия и геометрию подстилающей поверхности. Описываются динамическое ядро, физические параметризации, численная реализация и алгоритм работы модели. Изначально модель использовалась для вычислительно несложных задач. В данной работе на фоне необходимости применения модели в вычислительно сложных задачах проводится оптимизация последовательной программной реализации модели, а также создание программных реализаций модели с использованием технологий параллельных вычислений OpenMP, MPI, CUDA. Результаты тестирования различных реализаций модели на вычислительной системе с процессором Intel Xeon E5-2697 v3 2.60GHz и графическим процессором Nvidia P100 показывают, что оптимизация наиболее вычислительно сложных блоков в последовательной версии модели позволяет сократить время выполнения на 27%, в то же время использование технологий параллельных вычислений позволяет добиться ускорения на несколько порядков. Применение OpenMP в динамическом блоке модели привело к ускорению работы блока до 4 раз, применение MPI – до 8 раз, применение CUDA – до 16 раз при прочих равных условиях. Предложены рекомендации по выбору технологии параллельного вычисления в зависимости от свойств вычислительной системы.

Ключевые слова: модель переноса частиц; лагранжев подход; параллельные вычисления; оптимизация численной модели.

Для цитирования: Варенцов А.И., Имеев О.А., Глазунов А.В., Мортиков Е.В., Степаненко В.М. Численное моделирование переноса твёрдых частиц в атмосферном городском пограничном слое с использованием лагранжева подхода: физические задачи и параллельная реализация. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 145–164. DOI: 10.15514/ISPRAS–2023–35(4)–8.

Благодарности: Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова. Работа выполнена при частичной финансовой поддержке Российского научного фонда, гранты 21-17-00249 (проведение численных экспериментов) и 21-71-30023 (разработка параллельной реализации программного кода), и проекта Федеральной Научно-Технической Программы "Исследование процессов в пограничных слоях атмосферы, океана и вод суши вод и их параметризации в моделях системы Земли" в рамках программы "Совершенствование глобальной модели Земной системы мирового уровня для исследовательских целей и сценарного прогнозирования климатических изменений" (постановка задачи, обзор литературы и оптимизация модели).

Numerical Simulation of Particulate Matter Transport in the Atmospheric Urban Boundary Layer Using Lagrangian Approach: Physical Problems and Parallel Implementation

[1,2] *A.I. Varentsov*, ORCID: 0000-0001-7029-6773 <*aivarentsov98@gmail.com*>
[2] *O.A. Imeev*, ORCID: 0009-0008-5592-0759 <*ochir90@yandex.ru*>
[3] *A.V. Glazunov*, ORCID: 0000-0002-8780-3513 <*and.glas@gmail.com*>
[1,3] *E.V. Mortikov*, ORCID: 0000-0002-9683-5701 <*evgeny.mortikov@gmail.com*>
[1,2] *V.M. Stepanenko*, ORCID: 0000-0003-3033-6712 <*stepanen@srcc.msu.ru*>

[1] *Lomonosov Moscow State University, Research Computing Center*
1, Leninskie Gory, building 4, Moscow, 119234, Russia
[2] *Obukhov Institute of Atmospheric Physics of Russian Academy of Sciences*
3, Pyzhyovskiy pereulok, Moscow, 119017, Russia
[3] *Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences*
8, Gubkin str., Moscow, 119333, Russia

Abstract. This paper presents the results of the development of a numerical model of the Lagrangian particle transport and the application of parallel computation methods to increase the efficiency of the software implementation of the model. The model is a software package allowing calculations of transport and deposition of aerosol particles taking into account the properties of particles and input data describing atmospheric conditions and the underlying surface geometry. The dynamic core, physical parameterizations, numerical implementation, and algorithm of the model are described. Initially, the model has been used for computationally low-intensive problems. In this paper, given the need to use the model in computationally intensive problems, we conduct optimization of the sequential software implementation of the model, as well as creation of software implementations of the model with the use of parallel computing technologies OpenMP, MPI, CUDA. The results of testing of different implementations of the model show that optimization of the most computationally complex blocks in the sequential version of the model can reduce the execution time by 27%, at the same time the use of parallel computing technologies allows to achieve acceleration by several orders of magnitude. The use of OpenMP in dynamic block of the model resulted in acceleration of block up to 4 times, the use of MPI – up to 8 times, the use of CUDA – up to 16 times, all other conditions being equal. Recommendations on the choice of parallel computing technology depending on the properties of the computing system are proposed.

Keywords: particle transport model; Lagrangian approach; parallel computing; numerical model optimization.

For citation: Varentsov A.I., Imeev O.A., Glazunov A.V., Mortikov E.V., Stepanenko V.M. Numerical simulation of particulate matter transport in the atmospheric urban boundary layer using Lagrangian approach: physical problems and parallel implementation. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 4, 2023. pp. 145-164 (in Russian). DOI: 10.15514/ISPRAS-20123-35(4)-8.

Acknowledgements. The work was carried out using the equipment of the Center for Collective Use of Ultra-High-Performance Computing Resources of Lomonosov Moscow State University. The work was supported in part by the Russian Science Foundation, grant no. 21-17-00249 (numerical experiments), grant no. 21-71-

30023 (parallel implementation of the program code), and by the Federal Scientific and Technical Program, project "Investigation of Atmospheric, Ocean and Land Water Boundary Layer Processes and Their Parameterization in Earth System Models" under the program "Improvement of the World-Level Global Earth System Model for Research Purposes and Scenario Prediction of Climate Change" (problem statement, literature review and model optimization).

1. Введение

Представленная работа посвящена развитию численной модели лагранжева переноса частиц в геофизических пограничных слоях, её применению в задачах различной вычислительной сложности, оптимизации программной реализации, а также применению методов параллельных вычислений для сокращения времени работы модели.

Актуальность разработки и развития подобной модели обусловлена необходимостью совершенствования прогноза качества воздуха для здравоохранения и экономики, а также разработки исследовательских инструментов, среди которых ключевую роль играет физико-математическое моделирование.

Существует два основных подхода к моделированию переноса частиц, на основе которых создано большинство современных численных моделей. При лагранжевом подходе рассчитываются координаты и скорость для каждой отдельной частицы, могут быть явно учтены все действующие на частицу силы [1; 2]. При эйлеровом подходе производится расчёт переноса и диффузии концентрации частиц, например, в ячейках расчётной сетки [3-5]. В данной работе для создания модели выбран лагранжев подход из-за более явного учёта сил и большей информативности.

Модели лагранжева переноса частиц имеют крайне широкий спектр применения. С одной стороны, явный учёт сил, действующих на частицу, и возможность вычисления траектории каждой частицы позволяют использовать такие модели для изучения вклада в перенос отдельных сил, явлений и эффектов с отслеживанием всего одной или небольшого числа частиц. С другой – при использовании достаточно большого числа частиц можно оценивать различные статистики на их множестве, например пространственное распределение в форме концентрации частиц в ячейках расчётной сетки. Это позволяет применять модели лагранжева переноса в задачах исследования переноса твёрдых и жидких частиц внутри городской среды, когда одновременно важно учесть влияние сложной структуры воздушных течений на движение частиц и получить обоснованные оценки пространственного распределения частиц внутри и над городской застройкой. Лагранжев подход позволяет решить эту задачу, однако для получения надёжных оценок требуется проведение расчётов с большим числом частиц. В то же время подробное описание сложной структуры воздушного течения внутри городской застройки требует представления данных о течении на расчётной сетке с достаточно большим числом ячеек. В сумме это означает, что подобный эксперимент будет иметь высокую вычислительную стоимость.

Помимо прямых задач, траекторные (лагранжевы) модели позволяют решать в наиболее общем виде обратные задачи, предполагающие расчёт переноса частиц в обратном времени для определения источника частиц; частным случаем такой задачи является идентификация футпринтов измерений [6-7]. Для получения статистически значимых оценок, особенно при сложной геометрии подстилающей поверхности, требуется вычисление обратного движения очень большого числа частиц.

Необходимость проведения экспериментов с высокой вычислительной стоимостью создаёт требования к численной модели – алгоритмы должны быть максимально эффективными и оптимизированными, а вычисления внутри модели имеет смысл распараллелить, так как практически любой современный компьютер имеет более одного вычислительного элемента и позволяет проводить параллельные вычисления. Некоторые современные модели лагранжева переноса имеют последовательную программную реализацию [7], однако модели широкого профиля применения, например HYSPLIT, обычно используют параллельные

вычисления [8]. Таким образом, целью данной статьи является развитие вычислительно эффективной модели лагранжева переноса частиц для параллельных архитектур.

2. Модель лагранжева переноса частиц

В данной работе описывается и развивается численная модель переноса взвешенных в воздухе частиц, в которой для описания частиц, их перемещения и взаимодействия со средой используется лагранжев подход. В данной главе приводится описание основных уравнений модели, алгоритма её работы и верификации рассчитанных при помощи модели результатов.

2.1 Физико-математическая модель

При использовании лагранжева подхода движение каждой отдельной частицы описывается уравнением, неизвестными переменными в котором являются скорость и координата частицы [1]:

$$\mathrm{d}\boldsymbol{x_p} = \boldsymbol{u_p}\mathrm{d}t, \tag{1}$$

где $\boldsymbol{u_p} = (u_{p1}, u_{p2}, u_{p3})$ – скорость частицы, $\boldsymbol{x_p} = (x_{p1}, x_{p2}, x_{p3})$ – её координата, t – время. Данное уравнение определяет смещение частицы.

В разработанной в рамках данной работы модели лагранжева переноса частиц рассчитываются изменения и скорости частицы, и её позиции, таким образом движение каждой отдельной частицы описывается системой из двух уравнений: уравнения (1) для позиции частицы и уравнения для её скорости. Уравнение для скорости основано на втором законе Ньютона и имеет следующий вид:

$$\frac{\mathrm{d}\boldsymbol{u_p}}{\mathrm{d}t} = \frac{\boldsymbol{g}(\rho_p - \rho)}{\rho_p} + F_D\left(\boldsymbol{u} - \boldsymbol{u_p}\right), \tag{2}$$

где $\boldsymbol{g}$ – ускорение силы тяжести, ρ_p – плотность частицы (её материала), ρ – плотность окружающего воздуха, $\boldsymbol{u} = (u_1, u_2, u_3)$ – скорость потока воздуха, F_D – коэффициент силы сопротивления среды. Система уравнений (1) – (2) дополняется начальными условиями: для координат они определяются координатами источника частиц, а вектор начальной скорости задаётся равным нулю.

Здесь не учитываются столкновения частиц, т.к. они при не очень больших концентрациях редки. Также в правых частях уравнений (1) – (2) могут быть учтены дополнительные силы (термодиффузионная, броуновская и др.), но предполагается, что данная модель будет использована для условий плотной воздушной среды в нижних слоях атмосферы и частиц, размер которых много больше длины свободного пробега и размера молекул. В таком случае плотность частиц много больше плотности воздуха, так что указанные в уравнении внешние силы становятся преобладающими.

Первое слагаемое в правой части (2) соответствует силе плавучести – результирующей сил тяжести и Архимеда, действующих на частицу. Второе слагаемое характеризует силу сопротивления среды и отвечает за увлечение частицы потоком. Такие параметры системы уравнений движения, как ускорение силы тяжести, плотность окружающего воздуха и плотность частицы, могут быть заданы явно в виде констант или импортированы из входных данных. Каждая частица может иметь собственную плотность, отличную от других. Скорость воздушного потока и коэффициент силы сопротивления рассчитываются при помощи реализованных в модели параметризаций, в том числе с использованием импортированных входных данных.

Параметризации, как упрощённые параметрические описания различных процессов, используются в разработанной модели для описания силы сопротивления среды, влияния турбулентности, распада частиц с ограниченным временем жизни, взаимодействия частиц с твёрдыми поверхностями, влияния высоких концентраций частиц на стратификацию

атмосферы. В большинстве случаев для одного и того же процесса или явления реализовано несколько параметризаций, из которых можно выбрать более подходящую к конкретной задаче.

Коэффициент силы сопротивления среды F_D в модели может быть задан одним из двух способов. Первый способ более универсален, учитывает сопротивление среды при движении сферических частиц в любом направлении и имеет следующий вид [9]:

$$F_D = \frac{3\mu C_D Re}{4\rho_p d_p^2}, \quad (3)$$

где μ – динамическая вязкость воздуха, Re – число Рейнольдса для частицы, d_p – диаметр частицы, C_D – безразмерный коэффициент сопротивления среды, рассчитываемый по эмпирическим формулам для сферических частиц [10]:

$$C_D = a_1 + \frac{a_2}{Re} + \frac{a_3}{Re^2}, \quad (4)$$

где значения коэффициентов a_1, a_2, a_3 зависят от диапазона значений числа Рейнольдса для частицы (8 диапазонов в интервале значений Re от 0 до 50000).

Число Рейнольдса для частицы рассчитывается следующим образом:

$$Re \equiv \frac{\rho d_p |\boldsymbol{u_p} - \boldsymbol{u}|}{\mu} \quad (5)$$

Второй способ описания силы сопротивления среды F_D имеет более простой вид и является следствием закона Стокса для крупных падающих частиц [11-12]. Это предельный случай формулы (3) при малых значениях Re и размера частиц. Он был введён в модель для согласованности формулировки задачи с известными эйлеровыми аналитическими решениями в условиях, когда рассматривается движение снежных частиц в статистически однородном по горизонтали потоке [13]. При данном подходе F_D определяется следующей формулой:

$$F_D = \frac{18\mu}{\rho_p d_p^2} \quad (6)$$

В экспериментах, описанных далее в данной работе, использовалась формула (6) для соответствия условиям движения частиц в модуле лагранжева переноса вихреразрешающей модели ИВМ РАН.

Скорость воздушного потока может сильно меняться во времени и пространстве, в связи с чем она обычно вносит основной вклад в изменение скорости частиц и определяет характер их распространения. Источником данных о скорости потока могут быть результаты измерений или расчётов гидродинамических моделей, аналитические уравнения. Однако такие данные – это аппроксимация реального потока, они либо являются результатом осреднения, либо имеют определённый шаг расчётной сетки – процессы масштабом меньше это шага явно не воспроизводятся. Аэрозоли имеют размер значительно меньше шага сетки входных данных и могут продолжительное время находиться внутри одной ячейки, попадая под влияние подсеточных вихрей, поэтому важен учёт влияния турбулентности на движение частиц. В данной модели влияние турбулентности учитывается путём представления скорости потока в виде следующей суммы:

$$\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}', \quad (7)$$

где $\boldsymbol{u}$ – полная скорость потока, $\overline{\boldsymbol{u}}$ – осреднённая (например, по Рейнольдсу, пространственному фильтру или по времени) скорость потока, $\boldsymbol{u}'$ – пульсационная (подсеточная) составляющая. Средняя по времени компонента считывается из входных данных о средней скорости потока или задаётся аналитически, вторая рассчитывается отдельно при помощи турбулентных параметризаций.

В качестве турбулентных параметризаций реализовано две стохастических модели: одна 0-го порядка и одна 1-го порядка.

Представителем класса параметризаций 0-го порядка является модель случайных смещений

[5, 14]. Важным достоинством её использования является наличие эквивалентных в терминах концентрации эйлеровых подходов [15-16]. Это позволяет провести верификацию траекторной модели на доступных аналитических или численных решениях эйлеровой модели. Пульсация скорости рассчитывается следующим образом:

$$u_i' = \frac{\partial K_s}{\partial x_i} + \frac{\sqrt{2K_s}\xi_i}{dt}\ ,\ \sigma_{\xi_i} = \sqrt{dt}\ , \tag{8}$$

где ξ_i i=1, …, 3, – независимые дельта-коррелированные по времени гауссовы случайные величины, σ_{ξ_i} – среднеквадратическое отклонение, dt – приращение времени, x_i – i-ая эйлерова пространственная координата, K_s – коэффициент турбулентной диффузии.

Модель случайных смещений является стохастической моделью нулевого порядка – пульсационная компонента генерируется на каждом шаге по времени и не зависит от своих предыдущих значений, то есть турбулентное движение описывается последовательностью случайных смещений. В таком случае автокорреляция между смещениями отсутствует, хотя подсеточные вихри могут быть достаточно крупными, чтобы влиять на одну и ту же частицу в течение нескольких последовательных шагов по времени. Этот эффект учитывается более сложными стохастическими моделями первого порядка, ярким примером которых является модель Ланжевена. В ней турбулентное движение – это последовательность случайных затухающих приращений скорости, такой подход обеспечивает наличие автокорреляции смещений, а сила этой автокорреляции зависит от коэффициентов затухания.

На основе уравнения Ланжевена была реализована турбулентная параметризация, относящаяся к стохастическим моделям первого порядка. В итоге было использовано следующее уравнение [17]:

$$\mathrm{d}u_i' = -\frac{1}{2} b^2 \frac{u_i'}{\sigma_{u_i}^2} dt + b\xi_i\ , \tag{9}$$

где $b^2 = C_0\varepsilon$, $\sigma_{u_i}^2$ – дисперсия скорости потока, C_0 – постоянная Колмогорова, ε – скорость диссипации турбулентной кинетической энергии.

В данной параметризации, в отличие от модели случайных смещений, алгебраическим соотношением определяется не сама пульсационная компонента скорости, а её приращение, что позволяет считать параметризацию стохастической моделью первого порядка.

В модель включена возможность учёта распада частиц – это актуально для биоаэрозолей, радиоактивных частиц и некоторых химически активных соединений (в форме аэрозолей), то есть для частиц, имеющих ограниченное время жизни. В реализованном алгоритме время жизни частицы представлено в виде периода полураспада, вероятность распада частицы в течение шага по времени Δt определяется по следующей формуле:

$$P_{decay}(\Delta t) = \frac{N_t - N_{t+\Delta t}}{N_t} = 1 - 2^{-\frac{\Delta t}{\tau}}\ , \tag{10}$$

где N_t – число частиц в момент времени t, N_0 – начальное число частиц, τ – период полураспада частиц.

Для проведения расчётов в условиях городской застройки или идеализированной геометрии, отличной от плоской поверхности, в разработанной модели заданы два вида взаимодействия частиц с твёрдыми поверхностями: при контакте частицы с такой поверхностью происходит или её прилипание, или упругий отскок (с сохранением продольной компоненты импульса и заменой нормальной компоненты на противоположное значение) – выбор зависит от настроек для конкретной поверхности. Реализовано два вида задания твёрдых поверхностей. Первый вид позволяет задавать поверхности в виде выпуклых четырёхугольников произвольных размера и ориентации. Такой подход обеспечивает возможность задания геометрически сложных поверхностей, но требует относительно больших вычислительных затрат на проверку столкновений частиц с этими поверхностями. Второй вид задания твёрдых препятствий описывает не поверхности, а твёрдые ячейки прямоугольной расчётной

сетки, занятые зданиями или другими непроницаемыми объектами. Такой подход активно используется в гидродинамических моделях и позволяет гораздо быстрее производить расчёты в случаях простой ортогональной геометрии, в том числе аппроксимирующей реалистичную городскую застройку [18].

2.2 Численная реализация

Для получения траектории каждой частицы необходимо решить уравнения (2) и (3). Это возможно путём прямого интегрирования и получения полуаналитических решений [9]:

$$\boldsymbol{u}_p^{n+1} = \boldsymbol{u}^n + e^{-\frac{\Delta t}{\tau_D}}\left(\boldsymbol{u}_p^n - \boldsymbol{u}^n\right) - \boldsymbol{a}\tau_D\left(e^{-\Delta t/\tau_D} - 1\right), \tag{11}$$

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t(\boldsymbol{u}^n + \boldsymbol{a}\tau_D) + \tau_D\left(1 - e^{-\Delta t/\tau_D}\right)\left(\boldsymbol{u}_p^n - \boldsymbol{u}^n - \boldsymbol{a}\tau_D\right), \tag{12}$$

$$\boldsymbol{a} = \frac{\boldsymbol{g}(\rho_p - \rho)}{\rho_p}, \tag{13}$$

$$\tau_D = \frac{1}{F_D}, \tag{14}$$

где индекс n обозначает значение переменной в положении частицы в момент времени t_n, индекс $n+1$ – в положении частицы в момент t_{n+1}, $\boldsymbol{a}$ – в общем случае суммарное ускорение всех сил, кроме силы сопротивления среды (в данном случае – только силы плавучести), τ_D – характерное время силы сопротивления среды.

Полуаналитическая схема (11) – (14) становится точной при равенстве скоростей частицы и потока, когда пропадает сила трения. Но это не соблюдается при большом шаге по времени или нарушении гидродинамического равновесия частицы с воздушным потоком, что не позволяет применять данную схему в условиях сложных течений и тяжелых инертных аэрозолей, так как частицы будут постоянно выходить из состояния равновесия со средой.

Существуют конечно-разностные схемы, более применимые для решения этой проблемы. В ранних версиях модели использовался явный метод Рунге-Кутта 4-го порядка, однако на текущий момент он заменён на более вычислительно эффективную схему Кранка-Николсон для скорости и координаты частицы. Уравнения для используемой конечно-разностной схемы имеют следующий вид:

$$\boldsymbol{u}_p^{n+1} = \boldsymbol{u}_p^n + \Delta t\left(\boldsymbol{a} + \frac{1}{\tau_D}\left(\frac{\boldsymbol{u}^n + \boldsymbol{u}^{n+1}}{2} - \frac{\boldsymbol{u}_p^n + \boldsymbol{u}_p^{n+1}}{2}\right)\right), \tag{15}$$

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \frac{1}{2}\Delta t\left(\boldsymbol{u}_p^n + \boldsymbol{u}_p^{n+1}\right), \tag{16}$$

где $\boldsymbol{a}$ предполагается константой, так как предполагается использование модели при условиях, когда плотность воздуха неизменна во времени и много меньше плотности частиц.

2.3 Алгоритм работы модели

В разработанной модели лагранжев подход применяется к частицам, при этом характеристики воздушной среды задаются эйлеровым методом, т.е. как трёхмерные поля на дискретной сетке. Эти поля (компоненты скорости ветра и характеристики турбулентности) и их изменение со временем задаются входными данными, которые могут поступать от различных гидродинамических моделей или быть заданы аналитически. На данный момент реализовано чтение внешних входных данных из моделей ENVI-met [19] и LES-модели НИВЦ МГУ и ИВМ РАН [20-21], планируется чтение данных в формате NetCDF.

Основной алгоритм модели реализует последовательность итераций с заданным шагом по времени, ограниченную периодом моделирования. Схематично алгоритм работы модели представлен на рис. 1 и будет более подробно разобран далее. Реализация модели осуществлена на языке программирования C++, дополнительно присутствует набор скриптов

для чтения и визуализации выходных данных модели, написанный на языке программирования MATLAB.

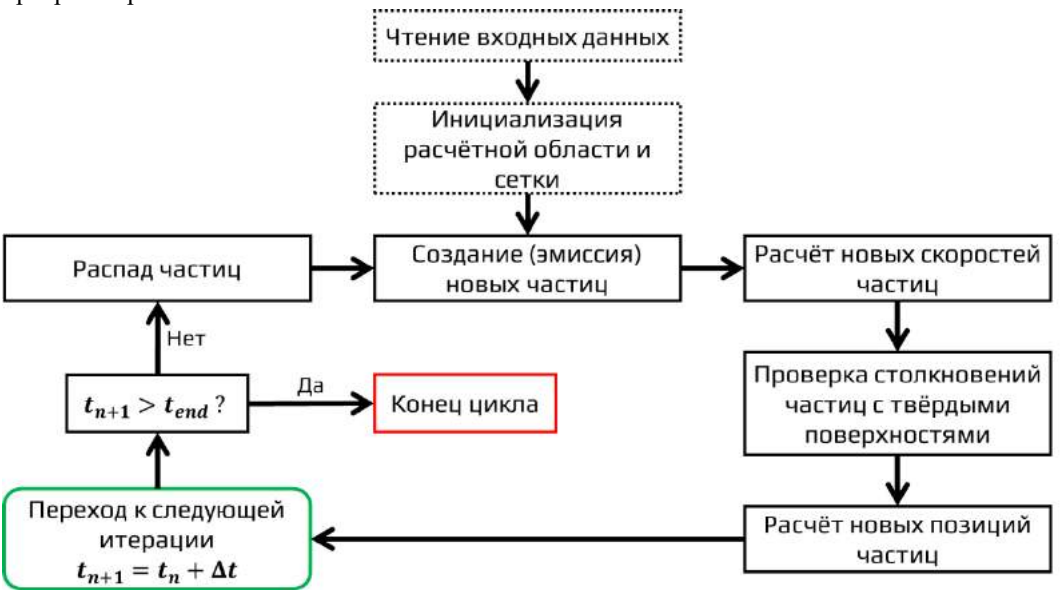

Рис. 1. Схема рабочего алгоритма модели
Fig. 1. Scheme of the model algorithm

Ключевыми частями программной реализации модели являются структуры данных, отвечающие за хранение и обработку информации о частицах, о параметрах среды (метеовеличинах) и о расчётной сетке.

Данные о частицах хранятся в объектах класса *particle* – каждый объект данного класса соответствует одной конкретной частице. Внутри класса хранятся параметры, которые могут быть индивидуальны для каждой частицы: векторы её положения в пространстве, полной скорости и пульсационной компоненты скорости, скалярные значения диаметра и плотности частицы. Также в данном классе хранятся вспомогательные параметры, необходимые для отслеживания состояния частицы или расчёта статистик: логические переменные, отвечающие за состояние частицы – активности, её осаждения, вылета за границы области, распада; скалярные переменные, отвечающие за вероятность распада частицы и момент времени её эмиссии. При необходимости в определённых задачах могут быть добавлены дополнительные переменные или же отключены некоторые из вышеописанных. Всё множество частиц, запускаемых в рамках конкретного эксперимента, описывается массивом объектов класса *particle*.

Данные о расчётной области и сетке, не считая данных о состоянии среды, хранятся в объекте класса *grid*. Данный класс включает, во-первых, множество переменных, описывающих область и сетку: координаты границ области, количество и размеры ячеек сетки, массив объектов класса *node*, отвечающих за хранение информации о состоянии среды в каждом конкретном узле (центре ячейки), и некоторые вспомогательные параметры. Во-вторых, в данном классе присутствует множество функций, описывающих взаимодействие частиц и расчётной сетки: определение ячейки, в которой находится конкретная частица, интерполяция и выдача метеоданных в точке нахождения частицы, расчёт различных статистических параметров, привязанных к расчётной сетке (например, концентрация частиц в ячейках сетки).

Самым простым из трёх ключевых классов является *node* – каждый объект данного класса соответствует одной конкретной ячейке расчётной сетки и хранит в себе информацию о

пространственном положении ячейки и значении характеристик среды в данной ячейке: скорости ветра, свойствах турбулентности и др. Список характеристик среды, включённых в данный класс, может меняться в зависимости от задачи и входных данных.

Рассмотрим подробнее рабочий алгоритм модели (рис. 1). Основной подготовительный этап работы модели при запуске любого численного эксперимента – это инициализация расчётной области и сетки. На этом этапе создаётся объект класса *grid* и заполняются метеоданными ячейки сетки. Если необходимо прочитать метеоданные или характеристики самой сетки из внешнего файла, например из выходного файла гидродинамической модели, предварительно производится чтение входных данных из внешнего файла. Однако все параметры могут быть заданы и аналитически внутри кода модели, такой способ может применяться в тестовых целях или для проведения относительно простых экспериментов. После создания расчётной сетки и сохранения в её узлы исходных метеоданных могут быть рассчитаны и сохранены их производные по пространству, если это требуется. В завершение данного этапа происходит инициализация пока что пустого массива частиц (объектов класса *particle*) и вспомогательных переменных.

Далее начинает работу итерационный цикл с заданным шагом по времени. На каждой итерации, то есть на каждом шаге по времени, выполняется ряд действий над объектами класса *particle*. Первым действием является проверка распада частиц, если имеются частицы с ограниченным временем жизни. Частицы, не прошедшие проверку, то есть распавшиеся, деактивируются – далее их динамика не рассчитывается. Деактивированные частицы удаляются из массива активных частиц во избежание перерасхода памяти и проведения лишних проверок. На первом шаге по времени эта операция пропускается, так как частицы ещё не были созданы. Далее следует этап эмиссии – создания новых частиц. Если на данном шаге по времени с учётом частоты и других настроек эмиссии должны быть созданы частицы, то происходит эмиссия: создаётся определённое количество частиц, которым в соответствии с настройками эксперимента присваиваются свойства (диаметр, плотность, положение, начальная скорость и др.). В отличие от двух предыдущих, этап расчёта новых скоростей частиц обязательно происходит на каждом шаге по времени. Для каждой частицы рассчитывается новая скорость и потенциальное новое положение частицы, но они не принимаются сразу. Проводится проверка на столкновение частицы с твёрдыми поверхностями при их наличии. Если с учётом новой скорости частица столкнётся с такой поверхностью, то происходит обработка столкновения частицы и либо расчёт новых скорости и положения (в случае отскока), либо производится деактивация частицы (в случае осаждения). Если твёрдых поверхностей в эксперименте нет или частица избежала столкновения, то отдельно рассчитывается новая позиция частицы. Завершает данный этап проверка на вылет частицы за границы расчётной области и изменение параметров частицы в соответствии с граничными условиями в случае вылета. На этом основные действия заканчиваются – происходит переход к следующей итерации либо завершение выполнения модели.

Дополнительно в течение итерации могут быть рассчитаны и сохранены в памяти либо выведены во внешние файлы различные статистики и параметры, например общее количество активных частиц или концентрация частиц в каждой ячейке (мгновенная или средняя за определённый период).

Входные данные, включающие термогидродинамические характеристики воздушной среды в определённые моменты времени и геометрические параметры подстилающей поверхности, необходимы для инициализации и интегрирования уравнений модели. В качестве таких данных принимается, во-первых, описание расчётной сетки для исследуемой области, во-вторых, количество моментов времени с внешними данными и шаги по времени между ними, в-третьих, значения различных метеовеличин в каждом узле сетки. К используемым термогидродинамическим характеристикам атмосферы на данный момент относятся:

компоненты скорости ветра по трём осям координат, турбулентная кинетическая энергия, скорость диссипации турбулентной кинетической энергии, коэффициент турбулентной диффузии.

В модели используется расчётная сетка «A» по классификации Аракавы – значения метеовеличин относятся к центрам ячеек. Расчётная сетка привязывается к декартовой системе координат и точке отсчёта в соответствии со входными данными. Так как размеры ячеек исходной сетки G_1, на которой заданы описанные выше входные эйлеровы поля, вдоль осей координат могут различаться (например, неравномерная телескопическая сетка с увеличением вертикального размера ячеек с высотой), после чтения входных данных создаётся вспомогательная равномерная сетка G_2. Длина её ячейки вдоль каждой оси равна наибольшему общему делителю всех длин ячеек сетки G_1 вдоль данной оси (рис. 2); это предполагает, что шаги сетки вдоль любой оси относятся друг к другу, как целые числа. В итоге каждая ячейка сетки G_1 заполняется целым числом одинаковых ячеек сетки G_2. Это позволяет быстро определять, в какой ячейке сетки G_1 в данный момент находится частица, определив номер ячейки сетки G_2.

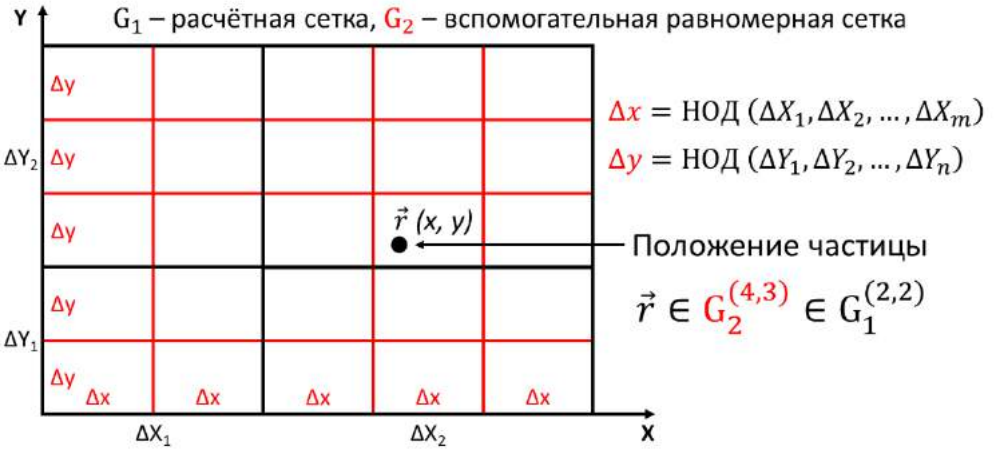

Рис. 2. Основная и вспомогательная сетки хранения данных для эйлеровых полей (НОД – наибольший общий делитель)
Fig. 2. Main and supplementary data storage grids

Значения метеовеличин для точек $\boldsymbol{x_p}$, находящихся между узлами сетки G_1, рассчитываются методом трилинейной интерполяции по 8-ми ближайшим узлам. Если частица находится между крайним узлом и границей области, то применяется билинейная интерполяция между 4-мя ближайшими узлами. Аналогично для граней и углов области применяется интерполяция между 2-мя ближайшими точками или использование значения из единственной ближайшей точки, соответственно. Если для метеовеличины определено граничное условие, оно будет приоритетно учтено при расчёте значения вблизи границы области.

Граничные условия для частиц в модели при отсутствии твёрдой поверхности на границе расчётной области могут иметь вид проницаемой границы, когда частицы считаются улетевшими из области (их расчёт далее не производится), вид периодической границы, когда вылетевшая наружу частица переносится и влетает в область с противоположной стороны, или вид закрытой границы, когда частица не может пересечь границу и упруго отскакивает от неё.

Основным результатом расчётов модели является вывод в отдельные файлы концентраций частиц в ячейках расчётной сетки и траекторий всех частиц. Дополнительно могут быть

выведены такие данные, как концентрация частиц, накопленных на поверхностях, параметры геометрии твёрдых поверхностей, описательная статистика.

2.4 Верификация модели

Оценку точности модели можно провести путём сравнения её результатов с эталонными данными, в роли которых обычно выступают натурно измеренные концентрации аэрозолей в воздухе или аналитические решения для пространственного распределения концентрации в идеализированных условиях.

Для верификации модели ранее было проведено сравнение модельных расчётов с эйлеровыми аналитическими решениями для пассивных частиц – маленьких лёгких частиц (обычно размером не более 2.5 мкм), для которых эффектами инерции и гравитационного оседания можно пренебречь. Были рассмотрены аналитические профили концентраций для двух идеализированных случаев: течения в нейтрально стратифицированном приземном слое и турбулентного течения Куэтта. По итогам сравнения было получено практически полное соответствие лагранжева численного и эйлерова аналитического подходов (рис. 3), что позволяет считать верификацию успешной [22].

3. Оптимизация и распараллеливание модели

3.1 Предпосылки оптимизации: вычислительно ёмкие задачи и недостатки модели

Модель изначально разрабатывалась для решения задач по расчёту переноса относительно небольшого числа частиц (от одной до нескольких десятков тысяч) в идеализированных условиях, в связи с чем эффективность и скорость работы модели не тестировались в задачах с высокой вычислительной стоимостью. Первые предпосылки необходимости оптимизации рабочего алгоритма и программной реализации модели появились при проведении экспериментов в условиях всё ещё идеализированных, но с приближенной к реальной геометрией городской застройке [22-23]. В данных экспериментах в качестве входных данных использовались результаты расчётов RANS-модели ENVI-met [19], количество ячеек сетки с метеоданными имело значение полумиллиона, а число единовременно активных частиц имело порядок тысяч. Однако количество экспериментов было небольшим (не более десяти), а время работы модели исчислялось десятками минут, что было приемлемо.

Полноценная необходимость в оптимизации модели возникла при работе с данными вихреразрешающих LES-моделей, в частности LES-модели ИВМ РАН [5]. В рамках задачи необходимо было сравнить особенности распространения частиц внутри и над городской застройкой при явном разрешении зданий и трёхмерного турбулентного течения в LES-модели (с внутренним модулем лагранжева переноса) и при наличии лишь осреднённых по времени вертикальных профилей метеовеличин данного течения в модели лагранжева переноса. Конфигурации городской среды, граничные условия и внешние воздействия для эйлеровой части модели были идентичны конфигурациям, описанным в работах [24-25]. Вычисления проводились в расчётных областях с размером 512 x 256 x 128 узлов равносторонней расчетной сетки с шагом 0.5 метра. Таким образом, общее количество ячеек составило уже более 16 миллионов, что на два порядка больше, чем в самых вычислительно требовательных экспериментах ранее. Возросло и количество одновременно активных частиц в модели лагранжева переноса – в зависимости от конкретного эксперимента оно достигало нескольких сотен тысяч штук. Следовательно, вычислительная стоимость экспериментов значительно возросла относительно всех ранее проводимых вычислений, что привело и к соответствующему росту времени расчёта – до порядка десятка часов на эксперимент. Такие масштабы времени вычислений уже были неприемлемы и явно показали

необходимость как в оптимизации отдельных элементов и алгоритмов модели, так и в распараллеливании её вычислений. Важной особенностью лагранжевых методов является высокий потенциал для их распараллеливания. Так как в большинстве задач не требуется рассматривать столкновения и другие взаимодействия между отдельными частицами, расчёты переноса и преобразования частиц могут быть распараллелены как по числу частиц, так и с помощью методов декомпозиции вычислительной области.

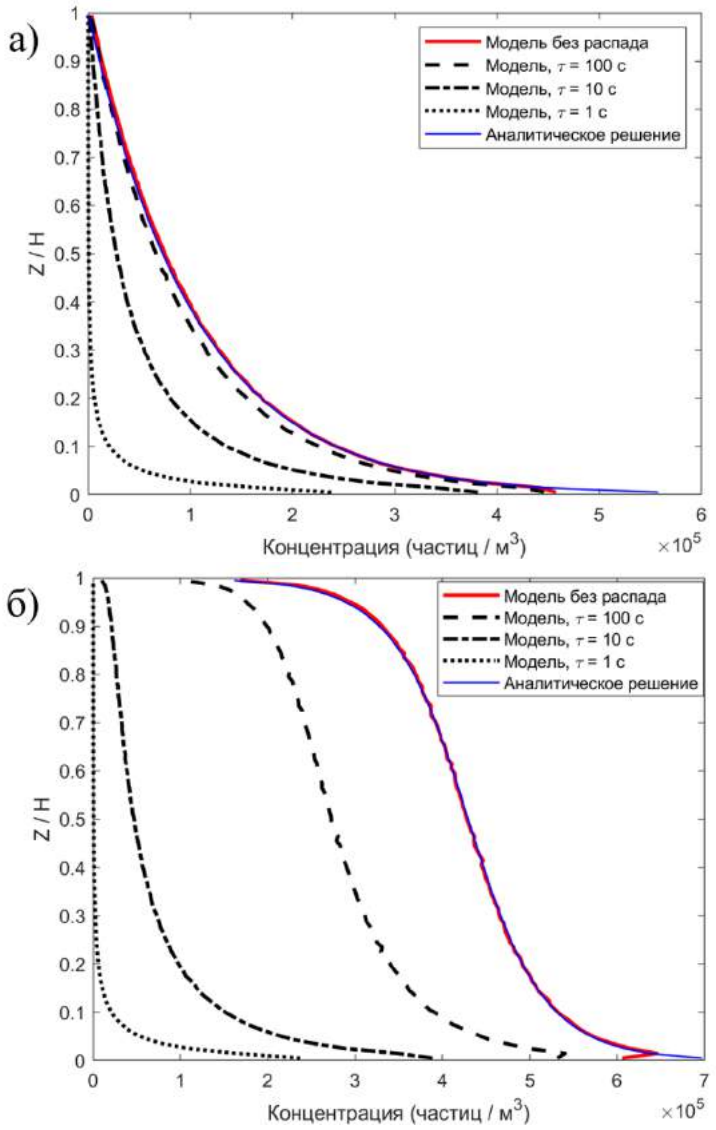

Рис. 3. Рассчитанные моделью профили концентрации частиц для случаев пограничного слоя (а) и течения Куэтта (б) без учёта конечного времени жизни частиц (красные кривые) и с его учётом (чёрные кривые), аналитические профили – синие кривые [22]

Fig. 3. Simulated particle concentration profiles for the case of a boundary layer (a) and Couette flow (b) without taking into account the finite lifetime of particles (red curve) and taking it into account (black curves), analytical profiles are blue curves [22]

3.2 Оптимизация модели и применяемые методы параллельных вычислений

Программную реализацию модели можно разделить на четыре части: этап инициализации, блок расчета динамики, обработку данных и запись результатов в файл. Блок динамики представляет собой цикл по времени, включающий создание новых частиц, обработку значений предыдущего шага, и вычисление новых значений координат и скорости частиц. В данной работе основной акцент сделан на ускорении именно блока расчёта динамики частиц с применением трёх подходов к параллельному программированию: OpenMP и MPI для расчетов на центральных процессорах и технологии CUDA для адаптации алгоритмов для архитектуры графических процессоров. Представленные далее результаты получены на суперкомпьютере Ломоносов-2. Характеристики вычислительной системы, на которой проводились эксперименты, для каждого из трёх случаев (OpenMP, MPI и CUDA) приведены в табл. 1.

Табл. 1. Характеристики вычислительных систем, использованных в экспериментах
Table 1. Characteristics of the computing systems used in the experiments

Архитектура	CPU	GPU	Число процессов
OpenMP	Intel Xeon E5-2697 v3 2.60GHz	-	14
MPI	Intel Xeon E5-2697 v3 2.60GHz	-	14
CUDA	Intel Xeon E5-2697 v3 2.60GHz	Nvidia P100	128 блоков, 64 нити

Единственной оптимизацией, проведенной для последовательной версии программного кода модели, является замена многомерных массивов на одномерные массивы длины, равной произведению всех размерностей. В программе встречается всего два таких массива: массив узлов или ячеек (объектов класса *node*) с информацией о полях метеовеличин и вспомогательный массив, использующийся при расчёте концентраций частиц в ячейках сетки. В обоих случаях изначально массивы имели четыре измерения: пространственные координаты и время. Второй массив влияет на скорость вычисления слабо на масштабе всей программы, так как обращения к нему происходят относительно редко и только при расчёте концентраций, однако его роль возрастает при увеличении интервала осреднения по времени. Зато замена первого массива с четырёхмерного на одномерный оказывает влияние на скорость работы программы на всех этапах алгоритма, в том числе и при отсутствии расчёта статистик. В итоге замена двух указанных многомерных массивов на одномерные позволила сократить время расчёта на 12-27% в зависимости от конфигурации эксперимента. Полученное ускорение достигается за счет более эффективного использования кэш-памяти и векторизации вычислений. Данная оптимизация использовалась и далее при распараллеливании вычислений модели с использованием технологий OpenMP, MPI и CUDA.

Для адаптации программы на OpenMP использовалась директива компилятора *#pragma parallel for*, примененная в функциях для расчёта координат и вектора скорости каждой частицы на следующем шаге по времени. На рис. 4 изображен график зависимости времени выполнения программного кода от модельного времени. Нелинейный характер зависимости вызван эмиссией новых частиц, которая идёт с постоянной скоростью в течение всего модельного времени. OpenMP подход ускоряет работу программы до 4 раз при использовании 14 процессов. Эффективность параллельной программы, η_r, отношение ускорения к количеству задействованных процессов, в данной задаче составляет $\eta_r \approx 2/7$. Основной причиной сравнительно небольшого ускорения является последовательная работа остальных блоков программного кода. Например, расчёты различных статистик в реализации выполняются последовательно – их распараллеливание возможно, однако в данной работе акцент был сделан на ускорении блока расчёта динамики.

На рис. 5 изображен график зависимости времени выполнения программного кода от модельного времени для OpenMP, MPI и CUDA. Использование технологии MPI позволило увеличить эффективность параллельного алгоритма – время работы относительно версии с OpenMP уменьшается примерно в 2 раза на 14 процессах. Эффективность в данном случае составляет $\eta_r \approx 4/7$.

Преимуществом реализации MPI является использование распределенной памяти, выделенной для каждого процесса, что позволяет каждому из них работать с меньшим количеством частиц и повысить эффективность обращений к памяти. Если n – количество процессов, то каждый MPI-процесс работает с массивом частиц в n раз меньшим, по сравнению с размерностью массива в OpenMP подходе. Кроме того, для OpenMP реализации необходимо отдельно обрабатывать блоки кода, которые могут привести к одновременной записи в общий, разделяемый между нитями, участок памяти. Недостатком MPI алгоритма является необходимость постоянного обмена данными между параллельными процессами.

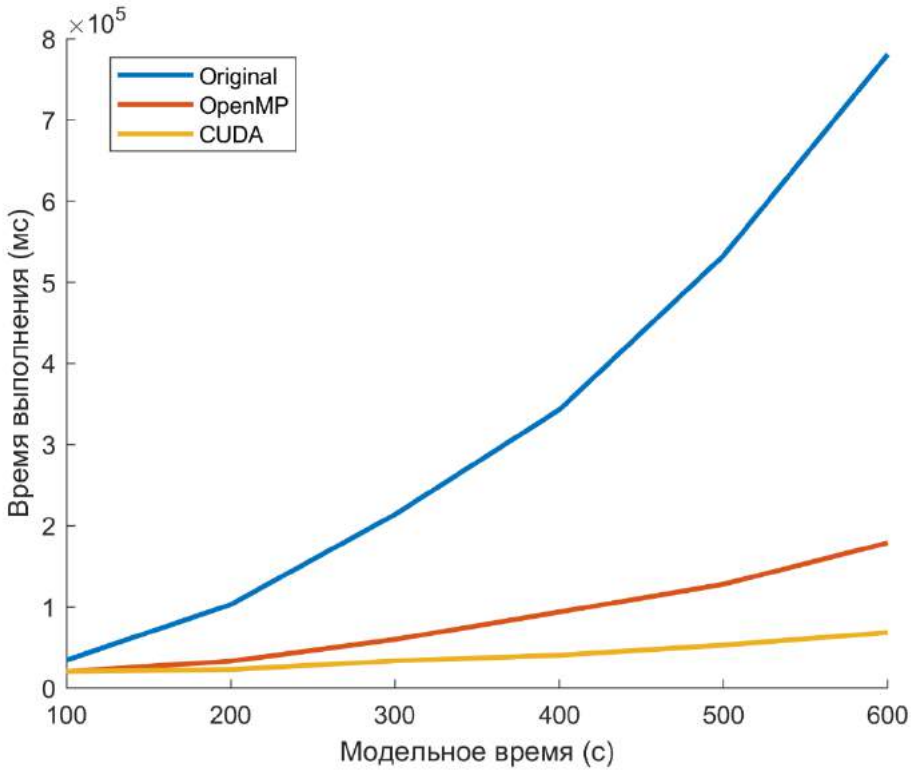

Рис. 4. Сравнение производительности последовательной версии программной реализации и параллельных версий с OpenMP и CUDA
Fig. 4. Performance comparison of the sequential version of the software implementation and parallel versions for OpenMP and CUDA

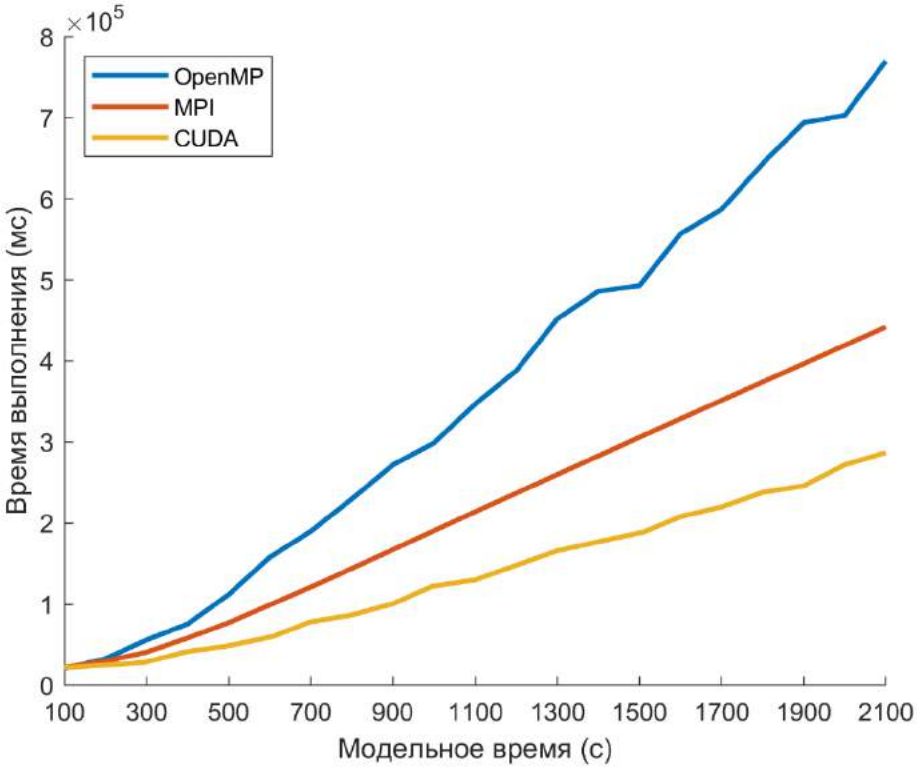

Рис. 5. Сравнение производительности программных реализаций с OpenMP, MPI и CUDA

Fig. 5. Performance comparison of software implementations with OpenMP, MPI and CUDA

Также статистическая обработка данных модели, запись результатов во внешние файлы требуют выделения дополнительной памяти. В настоящей реализации использовался синхронный алгоритм работы процессов, когда на некотором шаге все процессы, кроме первого, пересылают свои данные первому. Первый процесс их обрабатывает и при необходимости записывает результаты в файл.

Для передачи данных используются 3 рабочих массива типов **`int`**, **`double`** и **`bool`**, в которые записывается полное состояние системы. Работу программы можно дополнительно ускорить, если использовать асинхронный алгоритм передачи данных в блоке статистической обработки данных. Алгоритм состоит в том, чтобы на каждой итерации цикла обработку производил не один и тот же процесс, а каждый раз новый. Другими словами, на i-й итерации все процессы, кроме процессора с рангом $i \bmod n$, отдают данные для обработки и продолжают вычисления, где n – количество процессов. В тоже время, процесс $i \bmod n$ принимает и обрабатывает данные. Недостатком подобной модели является требование в n раз больше выделенной памяти для массива частиц. Текущая реализация модели с MPI требует примерно в 2 раза больше памяти по отношению к последовательной версии (первый процесс выделяет память под все частицы, в то время как остальные под 1 / n от всех частиц), вне зависимости от количества процессов.

Работа с графическим процессором (GPU) осложнена необходимостью копирования данных между оперативной памятью и памятью GPU, а также со сложностью адаптации программного кода для подобной архитектуры. В настоящей работе для реализации на GPU была использована технология CUDA (Compute Unified Device Architecture). С помощью

данной технологии, работу всей программы удалось ускорить в 4 раза по сравнению с OpenMP или в 2 раза по сравнению с MPI (см. рис. 5). В тоже время, если анализировать только время, необходимое для работы с вычислением новых скоростей и координат частиц, включая копирование данных на GPU и обратно, ускорение достигает 8 раз по сравнению с OpenMP. Важным изменением при работе с CUDA является генерация значений псевдослучайных величин с помощью встроенной библиотеки curand. Curand позволяет создавать массивы псевдослучайных чисел большой размерности на устройстве с очень высокой скоростью. При этом время такого расчета, как правило, было ограничено лишь накладными расходами на создание структур данных и инициализацию генератора. Данная модификация позволила значительно сократить время создания вектора случайных величин ξ, использующихся при расчёте пульсационных компонент скорости частиц в формулах (8) и (9). На рис. 6 изображен график зависимости времени выполнения для блока расчета новых скоростей и координат частиц от модельного времени для OpenMP и CUDA, включая время копирования на устройство.

4. Заключение

В работе обсуждается развитие, применение и повышение эффективности численной модели лагранжева переноса частиц. Статья описывает физико-математические основы модели, её структуру и алгоритм работы. Важно, что после программной реализации модель была верифицирована и протестирована в условиях различных метеорологических условий и геометрий твёрдых поверхностей. Тем не менее, должное внимание не уделялось оптимизации модели, пока сама модель не использовалась для решения вычислительно требовательных задач. Для решения таких задач было решено провести оптимизацию структуры и алгоритмов работы модели, а также распараллеливание вычислительных процессов для ускорения расчётов.

Как показали результаты данного исследования, оптимизация наиболее вычислительно объёмных блоков модели в её последовательной реализации дало прирост в скорости вычислений, но порядок времени работы остался тем же при любом числе использованных ядер. Значительно больший рост вычислительной эффективности показало применение методов параллельных вычислений. Результаты получены при помощи вычислительной системы с процессором Intel Xeon E5-2697 v3 2.60GHz (использовалось 14 процессов) и графическим процессором Nvidia P100 (128 блоков, 64 нити), количественная оценка ускорения актуальна именно для подобной конфигурации. Использование OpenMP позволило получить ускорение в 4 раза на 14 процессах, что является хорошим результатом, но дальнейшее повышение скорости работы с данной технологией ограничено максимальным числом вычислительных элементов (процессорных ядер) в пределах одного физического процессора или узла кластера с общей памятью и не позволяет без привлечения MPI эффективно использовать доступные возможности суперкомпьютерных технологий.

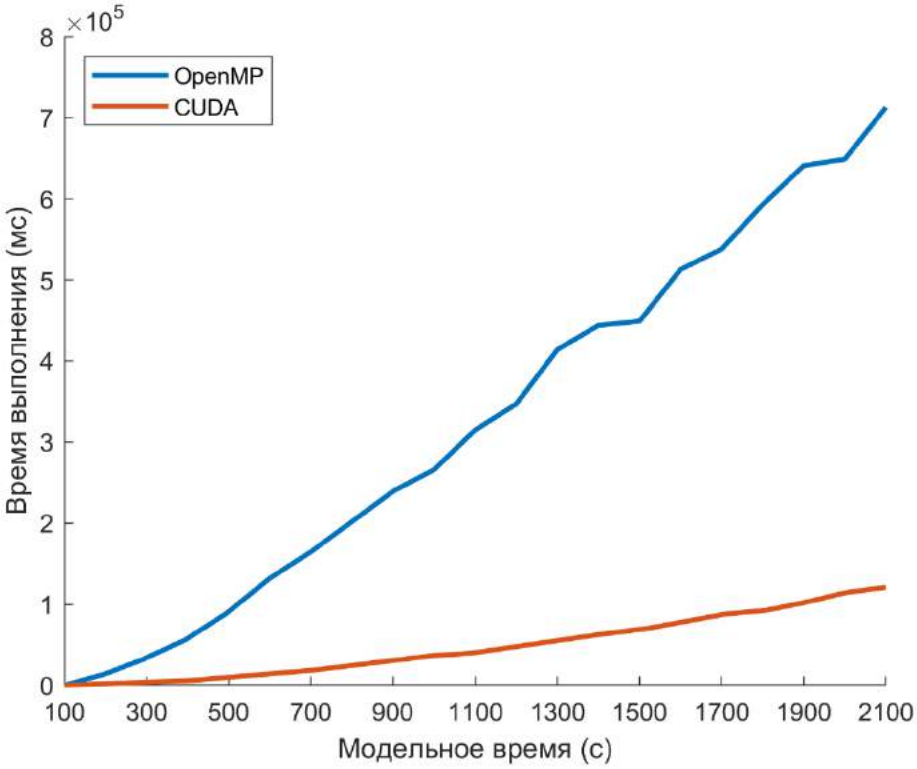

Рис. 6. Сравнение локальной производительности OpenMP и CUDA
Fig. 6. Local performance comparison between OpenMP and CUDA

В то же время данная технология не требует использования дополнительной памяти, поэтому версия модели с OpenMP может быть оптимальна для использования в системах с одним узлом процессоров с общей памятью. Версия модели с использованием технологии MPI позволила получить ускорение примерно в 8 раз относительно последовательной версии модели, что быстрее и версии с OpenMP. Версия с MPI имеет потенциал дальнейшего увеличения производительности модели при увеличении числа процессоров и может использовать потенциал суперкомпьютерных систем, однако требует выделения дополнительных объёмов памяти. Данная версия может быть оптимальна для решения задач с высокой вычислительной стоимостью, однако имеет более высокие требования к используемым вычислительным системам. Версия модели с распараллеливанием на графических процессорах показала наибольший прирост эффективности – до 16 раз относительно последовательной версии программы. Использование технологии CUDA в разработанной модели имеет наибольший потенциал среди рассмотренных технологий, однако минусом данного подхода является то, что графические процессоры, поддерживающие данную технологию, встречаются значительно реже обычных процессоров, то есть версия модели с данной технологией будет иметь строгие требования к техническим характеристикам вычислительной системы.

Список литературы / References

[1]. Thomson D. J., Wilson J. D. History of Lagrangian Stochastic Models for Turbulent Dispersion // Geophysical Monograph Series / Lin J, Brunner D, Gerbig C, Stohl A, Luhar A, Webley P. Washing-ton, D. C. (eds.): American Geophysical Union, 2013. pp. 19–36. doi: 10.1029/2012GM001238.

[2]. Maronga B., Gryschka M., Heinze R., Hoffmann F., Kanani-Sühring F., Keck M., Ketelsen K., Letzel M. O., Sühring M., Raasch S. The Parallelized Large-Eddy Simulation Model (PALM) version 4.0 for atmospheric and oceanic flows: model formulation, re-cent developments, and future perspectives // Geosci. Model Dev. 2015, vol. 8(8), pp. 2515–2551. doi: 10.5194/gmd-8-2515-2015.

[3]. Huttner S. Further development and application of the 3D microclimate simulation ENVI-met // 2012.

[4]. Sofiev M., Vira J., Kouznetsov R., Prank M., Soares J., Genikhovich E. Construction of the SILAM Eulerian atmospheric dispersion model based on the advection algorithm of Michael Galperin // Geosci. Model Dev. 2015, vol. 8(11), pp. 3497–3522. doi: 10.5194/gmd-8-3497-2015.

[5]. Glazunov A., Rannik Ü., Stepanenko V., Lykosov V., Auvinen M., Vesala T., Mammarella I. Large-eddy simulation and stochastic modeling of Lagrangian particles for footprint determination in the stable boundary layer // Geosci. Model Dev. 2016, vol. 9(9), pp. 2925–2949. doi: 10.5194/gmd-9-2925-2016.

[6]. Auvinen M., Järvi L., Hellsten A., Rannik Ü., Vesala T. Numerical framework for the computation of urban flux footprints employing large-eddy simulation and Lagrangian stochastic modeling // Geosci. Model Dev. 2017. Т. 10. № 11. С. 4187–4205. https://doi.org/10.5194/gmd-10-4187-2017.

[7]. Simon H., Heusinger J., Sinsel T., Weber S., Bruse M. Implementation of a Lagrangian Stochastic Particle Trajectory Model (LaStTraM) to Simulate Concentration and Flux Footprints Using the Microclimate Model ENVI-Met // Atmosphere. 2021, vol. 12(8), pp. 977. doi: 10.3390/atmos12080977.

[8]. Stein A. F., Draxler R. R., Rolph G. D., Stunder B. J. B., Cohen M. D., Ngan F. NOAA's HYSPLIT Atmospheric Transport and Dispersion Modeling System // Bulletin of the American Meteorological Society. 2015, vol. 96(12), pp. 2059–2077. doi: 10.1175/BAMS-D-14-00110.1.

[9]. Ansys Fluent. Theory Guide 12.0. [Электронный ресурс]. URL: https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/main_pre.htm (access: 01.05.2023).

[10]. Morsi S. A., Alexander A. J. An investigation of particle trajectories in two-phase flow systems // J. Fluid Mech. 1972, vol. 55(02), 193 p. doi: 10.1017/S0022112072001806.

[11]. Fletcher N. H. The physics of rainclouds. Cambridge University Press // Quarterly Journal of the Royal Meteorological Society. 1962, № 378 (88). С. 559–559. doi: 10.1002/qj.49708837821.

[12]. Budd W. F. The Drifting of Nonuniform Snow Particles. Washington, D. C.: American Geophysical Union, 1966, pp. 59–70. doi: 10.1029/AR009p0059.

[13]. Wamser C., Lykossov V. N. On the friction velocity during blowing snow // Beitr Phys Atmosph. 1995, vol. 68(1), pp. 85–94. https://epic.awi.de/id/eprint/3270/.

[14]. Pope S. B. Turbulent Flows: Cambridge University Press, 2000, Issue 1. doi: 10.1017/CBO9780511840531.

[15]. Durbin P. A. Stochastic differential equations and turbulent dispersion // NASA, 1983. https://ntrs.nasa.gov/citations/19830014275.

[16]. Boughton B. A., Delaurentis J. M., Dunn W. E. A stochastic model of particle dispersion in the atmosphere // Boundary-Layer Meteorol. 1987. Т. 40. № 1–2. С. 147–163. https://doi.org/10.1007/BF00140073.

[17]. Reynolds A. M., Cohen J. E. Stochastic simulation of heavy-particle trajectories in turbulent flows // Physics of Fluids. 2002, vol. 14(1), pp. 342–351. doi: 10.1063/1.1426392.

[18]. Blocken B. LES over RANS in building simulation for outdoor and indoor applications: A foregone conclusion? // Build. Simul. 2018, vol. 11(5), pp. 821–870. doi: 10.1007/s12273-018-0459-3.

[19]. ENVI-met – Decoding urban nature. URL: https://www.envi-met.com/ (access: 01.05.2023).

[20]. Kadantsev E., Mortikov E., Zilitinkevich S. The resistance law for stably stratified atmospheric planetary boundary layers // Q J R Meteorol Soc. 2021, vol. 147(737). pp. 2233–2243. doi: 10.1002/qj.4019.

[21]. Tkachenko E. V., Debolskiy A. V., Mortikov E. V. Intercomparison of Subgrid Scale Models in Large-Eddy Simulation of Sunset Atmospheric Boundary Layer Turbulence: Computational Aspects // Lobachevskii J Math. 2021, vol. 42(7), pp. 1580–1595. doi: 10.1134/S1995080221070234.

[22]. Varentsov A. I., Stepanenko V. M., Mortikov E. V., Konstantinov P.I. Numerical simulation of particle transport in the urban boundary layer with implications for SARS-CoV-2 virion distribution // IOP Conf. Ser.: Earth Environ. Sci. 2020, vol. 611(1), pp. 012017. doi: 10.1088/1755-1315/611/1/012017.

[23]. Varentsov A. I., Stepanenko V. M., Konstantinov P. I. High-resolution simulation of particle transport in the urban atmospheric boundary layer // IOP Conf. Ser.: Earth Environ. Sci. 2019, vol. 386(1), 012045. doi: 10.1088/1755-1315/386/1/012045.

[24]. Glazunov A. V., Debolskiy A. V., Mortikov E. V. Turbulent Length Scale for Multilayer RANS Model of Urban Canopy and Its Evaluation Based on Large-Eddy Simulations // Supercomputing Frontiers and Innovations. 2021, vol. 8(4), pp. 100–116. doi: 10.14529/jsfi210409.

[25]. Glazunov A., Mortikov E., Debolskiy A. Studies of Stable Stratification Effect on Dynamic and Thermal Roughness Lengths of Urban-Type Canopy Using Large-Eddy Simulation // Journal of the Atmospheric Sciences. 2023, vol. 80(1), pp. 31–48. doi: 10.1175/JAS-D-22-0044.1.

Информация об авторах / Information about authors

Александр Иванович ВАРЕНЦОВ – аспирант кафедры метеорологии и климатологии географического факультета МГУ имени М.В. Ломоносова. Сфера научных интересов: численное моделирование переноса аэрозолей в атмосфере, моделирование городского микроклимата и качества воздуха, изменение климата и опасные явления погоды, сезонные прогнозы.

Alexander Ivanovich VARENTSOV - postgraduate student of the Department of Meteorology and Climatology, Faculty of Geography, Lomonosov Moscow State University. Research interests: numerical modeling of aerosol transport in the atmosphere, simulation of urban microclimate and air quality, climate change and weather hazards, seasonal forecasts.

Очир Анатольевич ИМЕЕВ – студент магистратуры кафедры вычислительных технологий и моделирования факультета вычислительной математики и кибернетики МГУ имени М.В. Ломоносова. Сфера научных интересов: оптимизация вычислительных процессов, параллельные вычисления, математическое моделирование атмосферных процессов.

Ochir Anatolievich IMEEV – Master student of the Department of Computing Technologies and Modeling, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University. Research interests: computational processes optimization, parallel computing, mathematical modeling of atmospheric processes.

Андрей Васильевич ГЛАЗУНОВ – доктор физико-математических наук, ведущий специалист лаборатории математического моделирования геофизических пограничных слоёв Научно-исследовательский вычислительного центра МГУ имени М.В. Ломоносова. Сфера научных интересов: математическое моделирование, численные методы, физика геофизических пограничных слоев, турбулентность, гидродинамика, моделирование климата, параллельные вычисления.

Andrey Vasilievich GLAZUNOV – Cand. Sci. (Phys.-Math). Leading Specialist of the Laboratory of Mathematical Modeling of Geophysical Boundary Layers, Research Computing Center, Lomonosov Moscow State University. Research interests: mathematical modeling, numerical methods, physics of geophysical boundary layers, turbulence, hydrodynamics, climate modeling, parallel computing.

Евгений Валерьевич МОРТИКОВ – кандидат физико-математических наук, заведующий лабораторией математического моделирования геофизических пограничных слоёв Научно-исследовательский вычислительного центра МГУ имени М.В. Ломоносова. Сфера научных интересов: математическое моделирование, численные методы, физика геофизических пограничных слоев, турбулентность, гидродинамика, моделирование климата, параллельные вычисления.

Evgeniy Valerievich MORTIKOV – Cand. Sci. (Phys.-Math.). Head of the Laboratory of Mathematical Modeling of Geophysical Boundary Layers, Research Computing Center, Lomonosov Moscow State University. Research interests: mathematical modeling, numerical methods, physics of geophysical boundary layers, turbulence, hydrodynamics, climate modeling, parallel computing.

Виктор Михайлович СТЕПАНЕНКО – доктор физико-математических наук, заместитель директора Научно-исследовательского вычислительного центра МГУ имени М.В. Ломоносова с 2019 года. Сфера научных интересов: математическое моделирование деятельного слоя и экосистем суши, математическое моделирование водоёмов и водотоков

суши, физика геофизических пограничных слоёв и турбулентности, геофизическая гидродинамика, моделирование климата, параллельные вычисления.

Viktor Mihajlovich STEPANENKO – Dr. Sci. (Phys.-Math.). Deputy Director of Research Computing Center, Lomonosov Moscow State University. Research interests: mathematical modeling of the active layer and terrestrial ecosystems, mathematical modeling of terrestrial water bodies and watercourses, physics of geophysical boundary layers and turbulence, geophysical hydrodynamics, climate modeling, parallel computing.

DOI: 10.15514/ISPRAS-2023-35(4)-9

Функциональные особенности падежных показателей в ваховском хантыйском языке (на материале базы современных полевых данных на платформе ЛингвоДок)

[1,2] *В.В. Воробьева*, ORCID: 0000-0002-8729-0375 <*vorobeva@tpu.ru*>
[3] *И.В. Новицкая*, ORCID: 0000-0003-1559-8810 <*irno2012@yandex.ru*>

[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
[2] *Национальный исследовательский Томский политехнический университет,*
634050, Россия, Томск, пр. Ленина, д. 30.
[3] *Национальный исследовательский Томский государственный университет,*
634050, Россия, Томск, пр. Ленина, д. 36.

Аннотация. В настоящем исследовании объектом анализа выступает совокупность падежных морфем имен существительных, выделяемая с учетом маркируемой ими семантики. Спектр значений указанных морфем позволяет объединить их в группу семантических падежей в ваховском хантыйском языке, в противоположность группе синтаксических падежных маркеров. В исследуемом диалекте категория падежа существительного активно обсуждается в связи со спорными моментами относительно используемой терминологии, состава, количества, морфемного статуса и функциональных особенностей падежных маркеров. На основе исследований, проведенных в 2019 году, с помощью функций платформы ЛингвоДок были сопоставлены полевые и уже известные в ханталогии данные, уточнен состав категории падежа ваховского диалекта.

Ключевые слова: хантыйский язык; ваховский диалект; категория падежа; падежные маркеры; полевые данные; корпуса текстов; ЛингвоДок.

Для цитирования: Воробьева В.В., Новицкая И.В. Функциональные особенности падежных показателей в ваховском хантыйском языке (на материале базы современных полевых данных на платформе ЛингвоДок). Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 165–176. DOI: 10.15514/ISPRAS–2023–35(4)–9.

Благодарности: Исследование выполнено при финансовой поддержке РНФ в рамках проекта «Цифровое описание диалектов уральских языков на основании анализа больших данных» (No 20-18-00403).

Functional Characteristics of the Nominal Case Markers in Vakh Khanty (on the Basis of the Recent Field Language Data of the LingvoDoc Platform)

[1,2] *V. V. Vorobeva*, ORCID: 0000-0002-8729-0375 <*vorobeva@tpu.ru*>
[3] *I.V. Novitskaya*, ORCID: 0000-0003-1559-8810 <*irno2012@yandex.ru* >

[1] *Information Systems Department*
Ivannikov Institute for System Programming of the RAS
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[2] *National Research Tomsk Polytechnic University*
30, Lenina Avenue, Tomsk, 634050, Russia
[3] *National Research Tomsk State University*
36, Lenina Avenue, Tomsk, 634050, Russia

Abstract. In this study the object of analysis is a set of case morphemes of nouns, identification of which draws on the semantics they mark. The range of meanings of these morphemes allows us to combine them into a group of semantic cases in the Vakh Khanty language, as opposed to the group of syntactic case markers. In the dialect under study the category of nominal cases is actively discussed in connection with the controversial issues regarding the terminology used, composition, quantity, morphemic status and functional features of case markers. Using the latest field data on this dialect collected in the village of Korliki in 2019, we were able to compare field data and data already known in Khantalogy thus systematizing the case category of this dialect. Field data of more than 10,000 words was processed using the functions of the LingvoDoc platform.

Keywords: Khanty language; Vakh dialect; category of case; case markers; field data; text corpora; LingvoDoc.

For citation: Vorobeva V.V., Novitskaya I.V. Functional characteristics of the nominal case markers in Vakh Khanty (on the basis of the recent field language data of the LingvoDoc platform). *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 4, 2023. pp. 16-176 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-9.

Acknowledgements. The study was funded by the Russian Science Foundation, project no. 20-18-00403 'Digital Description of Uralic Dialects Based on Big Data Analysis'.

1. Введение

Категория падежа хантыйского языка всегда вызывала интерес у исследователей тем, что значительно разнилась по количеству и составу репрезентантов в группах диалектов, объединенных по географическому принципу. Например, в северных диалектах хантыйского падежная система включает три показателя, но их состав различается, а именно, в казымском диалекте – это показатели номинативного, лативного и локативного падежей, в то время как в приуральском диалекте выделяются маркеры номинативного, локативного и транслативного падежей. В утративших в настоящее время свою жизнеспособность южных диалектах выделяется пять-шесть показателей падежей. В восточных диалектах в именную систему склонения включается от семи до одиннадцати падежных репрезентантов. Различающиеся мнения относительно количества падежных показателей в восточных хантыйских диалектах отражают разные представления ученых о статусе некоторых морфем, кодирующих падежную семантику, поэтому один и тот же маркер в грамматических очерках может терминироваться и как падежный показатель, и как частица, и как послелог [1-8].

Такая же вариативность мнений наблюдается и в отношении названий некоторых из выявленных репрезентантов категории падежа. Полифункциональный характер большинства падежных показателей ставит исследователей перед выбором: терминировать полифункциональные падежи, ориентируясь на одну из их функций, понимаемую как приоритетная, или использовать двойное название, отражающее несколько функций какого-либо показателя. Так, в грамматических очерках Н.И. Терёшкина имеются указания на

наличие в ваховском хантыйском таких падежных показателей как направительно-целевой, творительно-объектный, творительно-совместный, отложительно-продольный [1].

Как можно заключить, сформировавшаяся в научной литературе картина компонентного состава категории падежа в восточных диалектах хантыйского языка допускает множественность интерпретаций и, в связи с этим, неоднозначность выводов относительно репрезентативности отдельных падежей. Подобное положение дел значительно осложняет обработку новейшего языкового материала, собранного в полевых условиях, при его оцифровке инструментами платформы ЛингвоДок [9].

Создавая корпус ваховского диалекта на платформе Лингводок, при разметке текстов мы столкнулись с рядом проблем, которые относились к разным уровням языка (аспектуальные и временные показатели, предикативный показатель, сравнительные конструкции, неописанность значения слов и др.). Определенные затруднения были также связаны и с падежными показателями, а именно, со стандартизацией названий падежей, определением их точного количества и функций. При цифровой обработке материала мы опирались на грамматику Н.И. Терёшкина, ученика В. Штейница. Н.И. Терёшкин черпал материал для своей монографии из научных экспедиций, которые он совершал во второй половине XX в. вдоль рек Вах, Васюган, Тромьеган и Салым, а также от своих хантыйских учеников в Институте народов Севера в Ленинграде [1: 5]. Его монография «Очерки диалектов хантыйского языка» (1961 г.) достоверно и достаточно полно описывает ваховский хантыйский, поэтому и послужила для нас опорой при глоссировании корпуса. Однако возникшие при обработке материала сложности решить не удалось, поскольку мы также обнаружили, что некоторые грамматические явления не были описаны в полном объеме или о них нет упоминаний вообще, например, маркер абессивного падежа и некоторые падежные функции, о чем пойдет речь ниже. Мы также привлекали работы зарубежных специалистов по ваховскому хантыйскому Л. Хонти [3] и Я. Гуя [2], но некоторые тайны разгадать не удалось, и задача даже усложнилась, так как ученые по-разному трактуют некоторые грамматические показатели, терминируют их и описывают их функции. Это и побудило нас написать эту статью, в которой мы постарались унифицировать именную падежную систему ваховского хантыйского. Стандартизация падежной системы послужит важной частью при создании парсера по ваховскому хантыйскому, который позволит падежные морфемы глоссировать универсально, стандартизованно, исключая ошибки, которые, учитывая человеческий фактор, неизбежны при ручном глоссировании большого объема материала, и выйти на совершенно другую скорость обработки текстов при создании больших корпусов.

Цель настоящей статьи заключается в уточнении номенклатуры падежных показателей на основе анализа функциональной нагруженности падежных маркеров, выявленных в новейших полевых данных ваховского диалекта хантыйского языка.

2. Пространственные падежи ваховского хантыйского

В настоящей статье мы будем использовать следующие обозначения падежных показателей: маркеры абессивного, аблативного, аллативного, дистрибутивного, комитативного, компаративного, лативного, локативного, номинативного, обликативного и транслативного падежей. Среди приведенных выше падежных показателей два показателя, а именно номинативный и локативный, задействованы для маркирования грамматических (синтаксических) отношений. Остальные показатели, в том числе и локативный в одной из своих функций, функционально ориентированы на передачу семантических отношений. Именно последние и выступят в качестве основного объекта исследования в данной статье.

Анализируемые падежные форманты, в свою очередь, распределяются по трем подгруппам в соответствии с понятийной соотнесенностью их семантики. К первой подгруппе можно отнести падежные показатели, выражающие пространственные отношения, а именно, маркеры аблативного, аллативного, лативного и локативного падежей. Вторая подгруппа

включает маркеры абессивного и комитативного падежей, кодирующих совместность или раздельность обозначаемого действия. Третья подгруппа включает показатели дистрибутивного, объектного и транслативного падежей, используемых для кодирования обстоятельственных или ‘косвенных’ отношений.

Далее проанализируем спектр кодируемой каждым показателем семантики.

Группа маркеров *пространственных падежей,* а именно аблативного, аллативного, лативного и локативного, кодирует семантику направления движения, стативного местоположения, а также нахождения во времени. Большая часть падежных морфем, относящихся к анализируемой группе, являются многофункциональными: маркеры аблатива, латива и локатива. Единственный падежный показатель из этой группы – маркер аллатива – выступает как монофункциональный.

Согласно концепции А.Л. Мальчукова [10], пространственные падежи участвуют в репрезентации базовых семантических ролей, а именно семантических ролей *Место, Путь, Источник* (начальная точка движения), *Цель* (конечная точка движения). Полевой материал ваховского хантыйского не поддерживает четкого разделения пространственных падежей по семантическим ролям. В этом диалекте хантыйского каждая роль может быть выражена одним из пространственных падежей или двумя. Так, для репрезентации роли *Места* используется локативный падеж, для реализации роли *Путь* – аблативный, который также может задействоваться для репрезентации роли *Источника*. Лативный и аллативный падежи ориентированы на репрезентацию роли Цели. Остановимся отдельно на каждом маркере из группы пространственных падежей.

Семантика аблативного маркера *-oɣ/-öɣ* используется для описания движения прочь «от чего-либо/кого-либо», указывая на исходную/начальную точку отсчёта, отметки движения или действия, а также концепции или события, получаемых из объекта. Проиллюстрируем это примерами (1-5):

(1) *lüɣ-ən* *män-t* *kǒr-äm-**öɣ*** *pǒr-ɣas*
3SG-LOC 1SG-ACC нога-POSS.1SG-**ABL** кусать-PST2.SBJ.3SG
‘Она укусила меня в ногу (досл: от ноги).’

(2) *iɣ-nə* *köt-əl-nä-ti* *joɣįm-ɣal-į* *wän’əm-əl-**öɣ***
медведь-LOC рука-POSS.3SG-COM-PRTC ударять-PST3-PASS.3SG лицо-POSS.3SG-**ABL**
‘Медведь его лапой по лицу ударил. (досл: от лица).’

(3) *lin* *paj-**oɣ*** *rįt* *wer-s-əɣən*
3DU осина-**ABL** лодка делать-PST1-SBJ.3DU
‘Они лодку из осины сделали.’

(4) *nüŋ* *paķį* *wer-ɣäs-ən* *tǒnəɣ-**oɣ**?*
2SG кукла делать-PST2-SBJ.2SG береста-**ABL**
‘Ты делал кукол из бересты?’

Устойчивое выражение *semät**öɣ** pittä* ‘рождаться’ (досл: ‘из глаз/на глаза появляться’), в котором слово *semät* ‘глаза’ всегда маркировано аблативным маркером:

(5) *mərəm* *perəɣ* *köl-t* *lullpənə-nə* *sem-ät-**üɣ*** *pit-s-ət:*
только другой слово-PL песня-LOC глаз-PL-**ABL** появляться-PST1-SBJ.3PL
‘Только другие слова в песне появились.’

Также аблативный маркер кодирует пролативное значение движения, выражая значение пути, прохода, обочины «вдоль, через, по», по которым разворачивается действие:

(6) *kiriw* *waɣ-**oɣ*** *mən-wəl*
моторная.лодка Вах-ABL идти-NPST.SBJ.3SG
‘Моторная лодка плывет вдоль реки Вах.’

(7) *lin ļ̈ök-**öɣ** pŏŋl-a mən-s-əɣən*
3DU дорога-**ABL** бок-LAT идти-PST1-SBJ.3DU
‘Они шли по обочине дороги.’

Аллативный маркер *-pa/pä* монофункционален и обычно указывает на конечный пункт движения в ваховском хантыйском. Согласно полевым материалам, аллативный маркер регулярно встречается в образцах речи ваховских ханты, хотя в некоторых случаях он обнаруживает функциональное дублирование показателя лативного падежа. Приведем примеры (8-9):

(8) *lüɣ lawka-j-**pa** jəl-äs päni n’än’ wə-s*
3SG магазин-EP-**ALL** go-PST2.SBJ.3SG и хлеб брать-PST1.SBJ.3SG
‘Он в магазин ушел и хлеб купил.’
(9) *məčəɣ təɣį-l-**pa** mənə-s*
постоянный место-POSS.3SG-**ALL** go-PST1.SBJ.3SG
‘На угодья ушёл.’

Маркер лативного падежа *a/-ä* полифункциональный. Одна из его функций связана с передачей семантики конечного пункта траектории движения, т.е. аллативного значения. Проиллюстрируем эти близкие значения примерами:

(10) *min wor ont-oɣ lar-**a** wįɣəl-s-əmən*
1DU роща нутро-ABL болото-**LAT** выходить-PST1-SBJ.1DU
‘Мы из рощи к болоту вышли.’
(11) *wəlta təɣį-l-**a** mən-il-wəl*
жить-INF место-POSS.3SG-**LAT** идти-MULT-NPST.SBJ.3SG
‘На угодья едет.’

Ещё одна функция – это маркирование лативного значения, заключающегося в обозначении цели, на которую направлено действие.

(12) *–ķəta, əjķa wer-l-əmən, – maɣ löŋkr-äli pŏŋl-a*
ну-ка вместе делать-NPST-SBJ.2DU бобр мышь-DIM бок-LAT
lal’ə-s päni lüɣ-ä pämil-əķə-tə-s
встать-PST1.SBJ.3SG и 3SG-LAT показывать-INCH-TR-PST1.SBJ.3SG
‘«Ну-ка, давай вместе делать», – бобр боком к мышонку встал и ему начал показывать.’

Лативный падеж часто употребляется для маркировки существительных, образующих семантически слитные сочетания с глаголами, например, охотиться, свататься и др. Так, одна из самых частотных в употреблении фраз среди носителей ваховского хантыйского ‘охотиться’ состоит из глагола *kəņčta* ‘искать’ и существительного *wajəɣ* ‘зверь’ в форме лативного падежа. Значение ‘сватать’ передается при помощи глагола *ärəɣtətä* ‘обещать’ и существительного в лативном падеже. Проиллюстрируем употребление данных фраз примерами (14-15):

(13) *käčəŋ ķotl wajɣ-a kəņč-įl-wəl*
каждый день зверь-**LAT** искать-MULT-NPST.SBJ.3SG
‘Он каждый день охотится.’
(14) *änä, män-nə səwəs iki-j-**ä** ärəɣtə-l-əm*
сестра, 1SG-LOC лесной.дух старик-**LAT** обещать-NPST-PASS-1SG
‘Сестра, я тебя за Севсики просватал.’

Маркер лативного падежа *-a/-ä* также может использоваться для кодирования реципиента:

(15) *maɣ mŏ̆ɣi ḳəm köl pükini-ŋ-**ӓ** l'əwəkin-tə-s…*
бобр что-то слово заяц-EP-**LAT** шептать-TR-PST1.SBJ.3SG
'Бобр что-то зайцу шепнул.'

Как следует из описания выше, по сравнению с другими падежными показателями маркер лативного падежа демонстрирует большее разнообразие функциональных возможностей. Однако самым многофункциональным падежным показателем в ваховском хантыйском считается маркер локативного падежа *-nə/-nə̈,* подробно проанализированном в отдельной работе [11]. Приведем примеры (16, см. также пример (5)), в которых показатель локативного падежа функционирует как репрезентант семантической роли Места, указывая на локацию объекта в пространстве:

(16) *köčəɣ satəw-**ən**-əḳi*
нож ножны-**LOC**-PRED
'Нож в ножнах.'

Кратко описывая маркер локативного падежа, нельзя не упомянуть об одной из его функций, которая не была описана в грамматиках Н.И. Терёшкина, Я. Гуя и Л. Хонти. Это одна из грамматических функций падежного показателя, которым маркируется субъект, в конструкции с пассивным маркером. В речи ваховских ханты мы регулярно наблюдали использование локативного маркера на субъекте с формами глаголов с пассивным аффиксом *-i/-į* третьего лица единственного числа в активных конструкциях. В них же отмечается несогласование подлежащего в 1SG, DU, PL; 2 SG, DU, PL или 3DU, PL при глаголе в 3SG. Описание этой грамматической функции локативного падежа в грамматиках отсутствует.

Для иллюстрации приведем примеры:

(18) *män-**nə̈** mon't'į tŏŋəm-tə-s-į*
1SG-**LOC** только.что знать-TR-PST1-PASS.3SG
'Я только что с ним познакомился.'
(19) *kolya-j-**ən** ilə̈n mə̈n-ɣäs-i*
Коля-EP-**LOC** рано покидать-PST2-PASS.3SG
'Коля рано ушел.'

Вторая подгруппа падежных показателей включает маркеры абессивного и комитативного падежей, репрезентирующих семантику совместности или отделенности.

Комитативный маркер *-na/-nä* выражает комитативное значение, т.е. описывает множественного (два или более) участника ситуации, выполняющего одну и ту же семантическую роль.

(20) *pəɣ-ali-nə öɣ-äli-l-**nä-ti** kömən jaŋḳa-l-əɣən*
юноша-DIM-LOC девушка-DIM-POSS.3SG-**COM**-PRTC на.улице играть-NPST-SBJ.3DU
'Мальчик с девочкой играют на улице.'
(21) *vasya marina-**na-tį** mə̈n-ɣäs*
Вася Марина-**COM**-PRTC идти-PST2.SBJ.3SG
'Вася с Мариной ушли.'

Комитативный падеж может передавать орудийное значение, маркируя предмет, который называет орудие и средства действия:

(22) *pükini os nuḳ älmi-l-tə kiriw-**nä** i̯čənwəs rama amə-wəl, jozh-nə päni maɣ-nə ḳaɣərtə-s-tə*

сажать-NPST.SBJ.3SG ёж-LOC и бобр-LOC хватать-PST1-SBJ.3SG:OBJ.SG
*päni käɣi-**nä** ḳat pat'-a täɣ əɣi-lə*
и молоток-**COM** дом край-LAT сюда прибивать-NST.SBJ.3SG:OBJ.SG
'Заяц наверх оконную лампу мотором поднимает, ёж и бобр хватают и молотком к краю дома прибивают.'

(23) *kolya rịt laijm-**na**-tị, suɣəl-**na**-tị, kịr-**na**-tị wer-ɣäs*
Коля облас топор-**COM**-PRTC топор.скребок-**COM**-PRTC топор.боковой-COM-PRTC делать-PST2.SBJ.3SG
'Коля выдолбил облас топором, топором-скребком, боковым топором.'

Также комитативный падеж выражает сопутствующий предмет:

(24) *mara jŏ-ɣäs ul-**na-tị***
Мара приходить-PST2.SBJ.3SG ягода-**COM**-PRTC
'Мара приехала с ягодой.'

В ваховском хантыйском комитативный маркер *-na/-nä* также используется для обозначения средств передвижения, что можно считать четвертой функцией маркера.

(25) *pịrnə mərəm täläɣin mäŋ-ən kätl-min täläɣin os läɣ-nä jəl-il-l-üɣ weli-**nä-ti***
потом лишь зимой 1PL-LOC ловить-CNV зимой ещё 3PL-COM ездить-MULT-NPST-SBJ.3PL олень-**COM**-PRTC
'Потом лишь зимой мы их ловим и ездим на оленях.'

(26) *lüɣ rịt-**na** mən-ɣäs*
3SG boat-**COM** идти-PST2.SBJ.3SG
'Он уехал на лодке.'

Полевые данные свидетельствуют о том, что комитативный маркер чаще других падежных морфем присоединяет частицу *-(ti)/-(ti),* см. пример (20-21, 24-25).

В семантическом отношении комитативному маркеру *-na/-nä* противопоставляется маркер монофункционального абессивного падежа *-ləɣ/-läɣ*.

(27) *mä ämp-**läɣ** ḳoɣa-l-em*
1SG собака-**ABS** идти-NPST-SBJ.1SG
'Я иду без собаки.'

(28) *läɣ püčkän-**läɣ** jəl-il-ɣäs-ət*
3PL gun-**ABS** go-MULT-PST2-SBJ.3PL
'Они пошли без оружия'

В третью подгруппу падежных показателей входят маркеры обликативного, транслативного и дистрибутивного падежей. Перечисленные показатели объединены в данную подгруппу на основе сообщаемой ими разного рода обстоятельственной или «косвенной» семантики, по принципу значений, которые встречаются только в языках с многочисленной падежной парадигмой. Транслативный и дистрибутивный падежи встречаются в других языках финно-угорской группы, например, транслативный представлен в финском и венгерском, а вот дистрибутивный только в венгерском. Обликативный падеж является особенностью только восточных диалектов хантыйского языка.

Маркер обликативного падежа *-(t)ə/-(t)ə* передает семантику, которая похожа на семантику творительного падежа в русском языке, но всё же отличается своей способностью маркировать только неодушевленный объект переходного глагола в субъектном или объектном спряжении, активном или пассивном. При передаче на русский литературный язык мы часто используем винительный падеж, хотя при буквальном переводе подразумевается творительный, например, «Меня водой дай» (см. примеры 29-31):

(29) *män-t* *ə̑čiɣ* *jəŋk-**ə̑*** *məj-itə̑ɣ*
1SG-ACC холодный вода-**OBL** давать-IMP.SBJ.PL.
‘Дай мне холодной воды.’

(30) *pükini-ŋ-äli-nə* *löŋkr-äli* *jəŋk-**ə̑*** *leməɣtə̑-s-i*
заяц-EP-DIM-LOC мышь-DIM вода-**OBL** обливать-PST1-PASS.3SG
‘Зайчонок мышку водой облил.’

(31) *lüɣ* *täləɣ* *köt-**ə̑*** *jȏ̮s*
3SG пустой рука-**OBJ** приходить-PST1.SBJ.3SG
‘Он без добычи вернулся.’

Маркер транслативного падежа *-ɣ/-əɣ/-ə̑ɣ* используется для обозначения изменения формы или состояния. Обычно существительное с транслативным маркером употребляется с глаголами *jäta* «становиться» *и sälimtätä* «превращаться». Это иллюстрируется примером:

(32) *lüɣ* *jol-ta* *ḳu-j-**ɣ*** *jə-ɣäs*
3SG шаманить-INF мужчина-**TRL** становиться-PST2.SBJ.3SG
‘Он стал шаманом.’

Дистрибутивный маркер *-(ə)təl/-(ə̑)tə̑l* выявляется не всеми учёными. В качестве падежного маркера *-(ə)təl/-(ə̑)tə̑l* обозначен только в хантыйской хрестоматии Л. Хонти [3: 60-61]. Падеж имеет необычную семантику ‘каждый’, на русский язык переводится ‘на каждого, на душу населения’ и употребляется он не со всеми словами. Приведем примеры из новейших полевых дынных, которые подтверждают существование данного маркера в падежной парадигме ваховского хантыйского (см. примеры 33-34).

(33) *ej-**ə̑tə̑l*** *mə-j-il-ä*
один-**DISTR** давать-EP-MULT-IMP.SBJ.SG
‘Дай по одной.’

(34) *käčəŋ* *ḳu* *kä* *rɨt-**ətəl*** *wə-ɣäs*
каждый человек два лодка-**DISTR** брать-PST2.SBJ.3SG
‘Каждый взял по две лодки.’

В табл. 1 приведены показатели всех 10 падежей именной системы склонения, используемые для маркирования семантических отношений в современном ваховском хантыйском.

Компаративный элемент *niŋit,* который выделяется Я. Гуя как самостоятельный падежный маркер [2], по результатам анализа полевых данных, свой статус падежного маркера не подтверждает. Он используется для обозначения сравнения одного предмета с другим и функционирует как автономный служебный элемент – послелог, что подтверждается нарушением гармонии гласных элемента *niŋit* с предшествующим словом. Н.И. Терёшкин также определяет элемент *niŋit* как послелог, например (35):

(35) *körľiki* *larjyak* ***niŋit*** *jə̑m-iki*
Корлики Ларьяк от хороший-PRED
‘Корлики лучше, чем Ларьяк.’

В представленной ниже табл. 1 приведены сводные унифицированные данные о падежных маркерах ваховского хантыйского, составленные на базе настоящего исследования с привлечением материалов только последних экспедиций текущего века. В таблице перечислены 10 падежных маркеров в унифицированных терминологических обозначениях. Нами предложена однокомпонентная падежная терминология, где под каждым термином падежа также приведен вариант его обозначения при глоссировании с целью стандартизации глоссирования текстов на ваховском диалекте. Унификация обозначений необходима для универсальной разметки текстов, чтобы при использовании инструментов программы ЛингвоДок, отбиралась вся доступная в корпусе выборка сочетаний с падежными маркерами.

Табл. 1. Свод маркеров семантических падежей ваховского диалекта хантыйского языка по материалам новейших полевых данных
Table 1. A set of semantic cases markers of Vakh Khanty based on the latest field data

№	Падеж	Падежный показатель
Семантические группы		
группа 0 (грамматическая семантика)		
1	Номинативный, локативный NOM, LOC	Ø /nə/nə̈
группа 1 (пространственная семантика)		
2	Аблативный ABL	oɣ/öɣ
3	Аллативный ALL	(a)pa/(ä)pä
4	Лативный LAT	a/ä
5	Локативный LOC	nə/nə̈
группа 2 (семантика совместности или раздельности)		
6	Абессивный ABS	ləɣ/lə̈ɣ
7	Комитативный COM	na/nä
группа 3 («косвенная» семантика)		
8	Дистрибутивный DISTR	-(ə)təl/-(ə̈)tə̈l
9	Обликативный OBL	(t)ə/(t)ə̈
10	Транслативный TRL	ɣ/əɣ/ə̈ɣ (< ḳə/kə̈)

3. Заключение

Полевые данные современного ваховского хантыйского языка свидетельствуют о том, что его именная система склонения включает в себя десять падежных форм, из которых девять ориентированы на кодирование семантических отношений. Анализ функциональных особенностей падежных показателей осуществлен на базе оцифрованного корпуса языковых данных ваховского диалекта, размещенных на платформе ЛингвоДок. Корпус содержит новейшие языковые данные по данному диалекту, собранные в 2019 году в поселке Корлики Нижневартовского района Ханты-Мансийского автономного округа. Проведенное исследование позволило подтвердить устойчивость системы именного склонения ваховского хантыйского языка, уточнить функциональное своеобразие каждого падежного маркера и выявить основания для их преимущественного терминологического обозначения

следующими названиями: 1) аблативный, аллативный, лативный, локативный; 2) абессивный, комитативный; 3) дистрибутивный, обликативный и транслативный. Компаративный элемент *niŋit* статус падежного маркера в ходе анализа не подтвердил. Также было выявлено новое грамматическое значение локативного маркера, которое не описывалось ранее. В настоящем исследовании предложено унифицированное обозначение падежных маркеров для разработки парсеров и разметки корпусов текстов на ваховском хантыйском.

4. Список сокращений

ABL – аблатив
ABS – абессив
ACC – аккузатив
ALL – аллатив
COM – комитатив
DIM – диминутивный
DISTR – дистрибутивный
DU – двойственный
EP – эпентетический
IMP – императив
INF – инфинитив
LAT – лативный
LOC – локативный
MULT – мультипликативный
NPST – непрошедшее
NST – бессуффиксальное
OBJ – объектный
OBL – обликативный
PASS – пассивный
PL – множественный
POSS – посессивный
PRED – предикативный
PRTC – частица
PST1 прошедшее на -s
PST2 – прошедшее на -ɣas/-ɣäs
PST3 – прошедшее на -ɣal/-ɣäl
SBJ – субъектный
SG – единственный
TRL – транслативный

Список литературы / References

[1]. Терёшкин Н. И. Очерки диалектов хантыйского языка (ваховский диалект). Ленинград, Наука, 1961. 204 с. / Tereshkin, N. I. Essays on dialects of the Khanty language (Vakh dialect). Leningrad, Nauka, 1961. 204 p. (in Russia).
[2]. Gulya J. Eastern Ostyak Chrestomathy. Bloomington – The Hague, 1966. 207 p.
[3]. Honti L. Chrestomathia Ostiacica. Budapest, Tankonyvkiado, 1984/1986. 432 p. (in Hungarian).
[4]. Karjalainen K. F., Vértes E. Grammatikalische Aufzeichnungen aus Ostjakischen Mundarten. Helsenki, 1964. 234 p.

[5]. Чепреги М. Сургутский диалект хантыйского языка. Ханты-Мансийск: ООО «Печатный мир г. Ханты-Мансийск», 2017. - 275 с. / Chepregi M. Surgut dialect of the Khanty language. Khanty-Mansiysk, Printed World of Khanty-Mansiysk, 2017. 275 p. (in Russia).
[6]. Filchenko A. Yu. A Grammar of Eastern Khanty. Doctor of Philology Thesis. Houston, Texas, 2007.
[7]. Vorobeva V., Novitskaya I. Inflectional morphology of nouns in Eastern Khanty (Vakh, Vasyugan, Surgut, Salym). Ural-Altaic Studies, vol. 38, 2020, pp. 33-70.
[8]. Мымрина Д. Ф. Категория падежа в хантыйском языке (сопоставительный аспект). Автореферат диссертации на соискание ученой степени к.ф.н. Томск, 2006 / Mymrina D. F. Case category in the Khanty language (comparative aspect). Associate Professor of Philology Thesis, Tomsk, 2006. (in Russia).
[9]. Платформа ЛингвоДок. http://lingvodoc.ispras.ru/corpora_all?language=508%2C44.
[10]. Мальчуков А. Л. Синтаксис эвенского языка: структурные, семантические, коммуникативные аспекты. СПб, Наука, 2008. 425 с. / Malchukov A. L. Syntax of the Even language: structural, semantic, communicative aspects. St. Petersburg, Nauka, 2008. 425 p. (in Russia).
[11]. Воробьева В.В., Новицкая И.В., Федоринова З.В. Синтаксические падежи имени существительного в вахавском хантыйском языке (на материале современных полевых данных платформы LingvoDok). Казанская наука, том 9, 2023, стр. 157-159. ISSN 2078-9955 (Print) ISSN 2078-9963 (Online) / Vorobeva V.V., Novitskaya I.V., Fedorinova Z.V. Syntactic cases of nouns in Vakh Khanty (based on the latest field data of LingvoDok / Kazan Science, vol 9, 2023, pp. 157-159 (in Russia).

Информация об авторах / Information about authors

Виктория Владимировна ВОРОБЬЕВА – кандидат филологических наук, доцент отделения иностранных языков Школы общественных наук Национального Исследовательского Томского политехнического университета с 1999 года. В настоящее время также работает старшим научным сотрудником в лаборатории Лингвистические платформы Института системного программирования РАН. Область научных интересов: сравнительно-типологическое языкознание, уральская языковая семья, корпусная лингвистика, документация и цифровая обработка исчезающих языков.

Victoria Vladimirovna VOROBEVA – Cand. Sci. (Philology), Associate Professor of Department of Foreign Languages, School of Social Sciences of National Research Tomsk Polytechnic University since 1999. 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.

Ирина Владимировна НОВИЦКАЯ – доктор филологических наук, доцент, профессор кафедры английской филологии Национального исследовательского Томского государственного университета с 1997 года. Область научных интересов: морфология и семантика древнегерманских языков, типологическое языкознание, сравнительно-сопоставительные исследования, морфосинтаксис обско-угорских языков, прикладная лингвистика.

Irina Vladimirovna NOVITSKAYA – Dr. Sci. (Philology), professor at the Department of English Philology, at National Research Tomsk State University. Research interests: morphology and semantics of the Old Germanic languages, typological studies, comparative studies, morpho-syntactical properties of Khanty lexical units, applied linguistics.

DOI: 10.15514/ISPRAS-2023-35(4)-10

Автоматическое определение сходства Javadoc-комментариев

[1] *Д.В. Кознов,* ORCID: 0000-0003-2632-3193 <*d.koznov@spbu.ru*>
[2] *Е.Ю. Леденева,* ORCID: 0009-0002-8907-143X <*ekiuled@gmail.com*>
[1] *Д.В. Луцив,* ORCID: 0000-0002-6332-2360 <*d.lutsiv@gmail.com*>
[3] *П.И. Браславский,* ORCID: 0000-0002-6964-458X <*pbras@yandex.ru*>

[1] *Санкт-Петербургский государственный университет,*
Россия, 199034, г. Санкт-Петербург, Университетская наб., д. 7–9
[2] *ООО Яндекс.Технологии,*
Россия, 11902, Москва, 1 ул. Льва Толстого, д. 7
[3] *НИУ «Высшая школа экономики»,*
Россия, 109028, Москва, Покровский бульвар, д. 11

Аннотация. Комментарии в исходном коде являются важной частью документации программного обеспечения. Многие программные проекты страдают от некачественных комментариев, которые часто создаются путем копирования и содержат многочисленные ошибки и неточности. В случае схожих методов, классов и т.п. копирование комментариев с небольшими изменениями оправдано, но и в этом случае разработчики делают ошибки. В этом исследовании мы решаем проблему обнаружения похожих комментариев к исходному коду, что позволяет улучшить комментариев к коду. Применительно к задаче определения сходства JavaDoc-комментариев мы провели оценку традиционных алгоритмов сходства строк и современных методов машинного обучения. В нашем эксперименте мы используем коллекцию комментариев Javadoc из четырех промышленных Java-проектов с открытым исходным кодом. Мы выяснили, что LCS (Longest Common Subsequence) является лучшим алгоритмом для решения нашей задачи, учитывая как качество (точность 94%, полнота 74%), так и производительность.

Ключевые слова: документация программного обеспечения; комментарии Javadoc; метрики схожести.

Для цитирования: Кознов Д.В., Леденева Е.Ю., Луцив Д.В., Браславский П.И. Вычисление схожести комментариев Javadoc. Труды ИСП РАН, том. 35, выпуск 4, 2023. с. 177-186. DOI: 10.15514/ISPRAS-2023-35(4)-10.

Evaluation of Similarity of Javadoc Comments

[1] *D. V. Koznov,* ORCID: 0000-0003-2632-3193 <*d.koznov@spbu.ru*>
[2] *E. Iu. Ledeneva,* ORCID: 0009-0002-8907-143X <*ekiuled@gmail.com*>
[1] *D. V. Luciv,* ORCID: 0000-0002-6332-2360 <*d.lutsiv@gmail.com*>
[3] *P. I. Braslavski,* ORCID: 0000-0002-6964-458X <*pbras@yandex.ru*>

[1] *Saint-Petersburg State Uinversity,*
7/9 Universitetskaya emb., St. Petersburg, 199034, Russia
[2] *Yandex LLC, 7 Lva Tolstogo st., Moscow, 119021, Russia*
[3] *HSE University, 11 Pokrovsky blvd., Moscow, 109028, Russia*

Abstract. Code comments are an essential part of software documentation. Many software projects suffer the problem of low-quality comments that are often produced by copy-paste. In case of similar methods, classes, etc. copy-pasted comments with minor modifications are justified. However, in many cases this approach leads to degraded documentation quality and, subsequently, to problematic maintenance and development of the project. In this study, we address the problem of near-duplicate code comments detection, which can potentially improve software documentation. We have conducted a thorough evaluation of traditional string similarity metrics and modern machine learning methods. In our experiment, we use a collection of Javadoc comments from four industrial open-source Java projects. We have found out that LCS (Longest Common Subsequence) is the best similarity algorithm taking into account both quality (Precision 94%, Recall 74%) and performance.

Keywords: software documentation; Javadoc comments; similarity measure.

For citation: Koznov D.V., Ledeneva E.Iu., Luciv D.V., Braslavski P.I. Calculating similarity of Javadoc comments. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 4, 2023. pp. 177-186. DOI: 10.15514/ISPRAS-2023-35(4)-10.

1. Введение

Комментарии в исходном коде – один из наиболее важных видов документации программного обеспечения. Комментарии играет важную роль при сопровождении и поддержке программного обеспечения, особенно при разработке программных интерфейсов (API), которые создаются для использования другими разработчиками [1].

Существуют специальные инструменты (например, Javadoc [1], Doxygen [2]), которые по комментариям в исходном коде, представленным в специальном формате (а также некоторым другим артефактам – диаграммам в формате GraphViz [3], сигнатурам методов и пр.) генерируют html/pdf документацию. Для Java-приложений стандартом де-факто является технология Javadoc, которая определяет язык разметки и программный инструмент, интегрированный в большинство сред разработки на Java, таких как Eclipse, IntelliJ IDEA и другие.

Обычно в комментариях встречается множество дубликатов, поскольку при комментировании новых классов, методов и прочих программных сущностей разработчики зачастую копируют и вставляют в новые места уже существующие комментарии [2]. Эта практика является общепринятой, поскольку и сами новые сущности также очень часто функционально похожими на существующие. Скопированный комментарий при этом исправляется и дополняется, чтобы отразить специфику новой сущности, в результате чего порождается *неточный дубликат*. Однако скопированные комментарии могут содержать опечатки и ошибки: разработчик может скопировать старый комментарий, слегка изменить его, окончательную доработку отложить на потом, и в конце концов просто забыть о ней.

[1] https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html
[2] https://www.doxygen.nl/
[3] https://graphviz.org/

Также возможны ошибки в изменённых фрагментах комментария, например, когда разработчик забывает заменить в скопированном комментарии часть идентификаторов.

Таким образом, наличие неточных дубликатов считается нормальной практикой, но из-за вышеупомянутых проблем требуются специализированные инструменты для их обнаружения и анализа, чтобы выявить несоответствия между комментариями и исходным кодом, к которому они относятся [2].

Изучению различных дубликатов в комментариях посвящён ряд работ [2]. Тем не менее, исследователи не уделяют должного внимания ни автоматическому обнаружению дубликатов, ни поиску неточных повторов. Из перечисленных работ, лишь [3] предлагает инструментарий для поиска дубликатов, но и там оставлены в стороне неточные дубликаты.

Данная работа восполняет этот пробел, рассматривая неточные дубликаты в Javadoc-комментариях. Мы исследуем различные алгоритмы, которые могут использоваться для оценки попарного сходства комментариев (pairwise similarity), рассматривая как алгоритмы сравнения строк, так и современные алгоритмы машинного обучения. В ходе экспериментов мы проанализировали четыре широко известных Java-проекта с открытым исходным кодом: JSON, JUnit4, Mockito и Slf4J. В результате мы выяснили, что алгоритм LCS (поиск наибольшей общей подпоследовательности) является наилучшим при определении сходства, как с точки зрения качества результата (точность – 94%, полнота – 74%), так и с точки зрения производительности. Мы определили все неточные дубликаты в документации рассмотренных проектов, а также обнаружили некоторые ошибки в комментариях.

2. Background

2.1 Неточные дубликаты в Javadoc-комментариях

Рассмотрим два следующих метода из проекта JUnit4: `assertTrue` и `assertFalse`. Первый активирует исключение (exception), если его аргумент имеет значение «истина», второй делает тоже, если аргумент имеет значение «ложь». Очевидно, что функциональность методов очень похожа, следовательно, их комментарии также должны быть похожи. В самом деле, разница этих комментариев заключается в одном единственном слове – см. пример ниже, в котором различающиеся слова выделены жирным шрифтом и подчеркнуты. Таким образом, эти комментарии являются *неточными дубликатами* (рис.1).

```
/** Asserts that a condition is true.
    If it isn't it throws an AssertionError
    without a message.
    @param condition - condition to be checked */
public static void assertTrue(boolean condition)

/** Asserts that a condition is false.
    If it isn't it throws an AssertionError
    without a message.
    @param condition - condition to be checked */
public static void assertFalse(boolean condition)
```

Рис. 1. Неточные дубликаты комментариев к методам проекта JUnit4
Fig. 1. Near duplicates of JUni4 project methods' comments

2.2 Выбранные для экспериментов алгоритмы

Для определения неточных дубликатов необходима функция сходства (similarity function). Рассмотрим наиболее известные алгоритмы для вычисления функции сходства двух строк:

поиск наибольшей подпоследовательности (Longest Common Subsequence, LCS), косинусное сходство (Cosine Similarity, COS), локально-чувствительное хеширование (Locality-Sensitive Hashing, LSH), а также известные алгоритмы машинного обучения – Word Mover's Distance (WMD), doc2vec (D2V), and Siamese Neural Network (SNN). Краткий обзор этих алгоритмов представлен в табл. 1.

LCS, COS, и LSH являются широко известными алгоритмами сравнения строк и основаны на разных идеях. Мы использовали следующие реализации этих алгоритмов: LCS из библиотеки difflib [4], COS из библиотеки scipy [5], реализацию LSH мы адаптировали из библиотеки datasketch[6].

WMD, D2V и SNN являются ведущими на сегодняшний день алгоритмами машинного обучения, предназначенными для обработки текста. Мы использовали реализацию WDM and D2V из библиотеки gensim[7], реализацию SNN из проекта Keras[8].

Табл. 1. Краткий обзор алгоритмов вычисления сходства строк
Table 1. Brief Overview of Algorithm Selected

Алгоритм	Описание
LCS [6]	Измеряет длину наибольшей общей подпоследовательности двух строк в виде идентичных фрагментов текста, следующих в одном и том же порядке.
COS [7]	Преобразует значения частот вхождения слов в текст (term frequency, TF) в многомерные вектора. Каждая компонента вектора пропорциональна частоте употребления этого слова в тексте и обратно пропорциональна частоте употребления слова в совокупности анализируемых текстов. Сходство этих векторов вычисляется как косинус угла между ними.
LSH [8]	Представляет текст как мультимножество n-грамм (наборов из нескольких последовательно встречающихся слов), к которым применяет алгоритм minhash [9], результирующие битовые вектора разбивает на фрагменты и вычисляет долю совпадающих фрагментов векторов, получившихся из разных комментариев.
WMD [10]	Представляет два документа как множества точек в векторном пространстве (эмбеддинги, embeddings) и вычисляет их сходство как цену перемещения одного множества в другое. Использует алгоритм word2vec [11] для конструирования эмбеддингов для слов документа, основываясь на нейронной сети, обученной на большом корпусе текстов.
D2V [12]	Расширение word2vec, в котором строятся эмбеддинги документов, а не отдельных слов.
SNN [13]	Нейронная сеть с двумя подсетями, имеющими общую архитектуру и веса. Наша реализация использует LSTM (Long Short-Term Memory) компоненты.

[4] https://docs.python.org/3/library/difflib.html
[5] https://scipy.org
[6] http://ekzhu.com/datasketch/
[7] https://radimrehurek.com/gensim/
[8] https://keras.io/

3. Обзор сходных работ

Очень часто комментарии к коду имеют невысокое качество – они могут быть неполными, содержать ошибки, опечатки и противоречия [14]. Более того, если система достаточно велика, комментарии к коду написаны не в едином стиле, поскольку создавались различными разработчиками [16]. Имеется некоторое количество методов и инструментов для определения ошибок в комментариях к коду [5], [17]. Но данная задача всё ещё далека от разрешения и требуются новые подходы.

Имеется ряд исследований, фокусирующихся на дубликатах в программной документации – обзоры могут быть найдены в [16], [22]. В работах [16], [22] рассматриваются неточные дубликаты, но эти работы не имеют дела с JavaDoc-комментариями. Nosál и Porubän [4] рассматривают неточные дубликаты в Javadoc. Однако, они только представляют Javadoc-плагин для реализации повторного использования в комментариях, опуская вопрос поиска таких дубликатов в уже существующих комментариях. Oumaziz и др. [2], а также and Blasi и др. [3] исследовали связь между дубликатами в Javadoc-комментариях и дубликатами в коде, но они не рассматривали неточные дубликаты. Wagner и Fernández [25] писали о алгоритмах выявления повторов документации ПО, но они не рассматривали Javadoc. А между тем, Javadoc является широко используемым на практике форматом, применяясь в большинстве индустриальных Java-проектах.

Таким образом, существующие исследования Javadoc-дубликатов не рассматривают применение современных алгоритмов сопоставления сток и текстов, а также не рассматривают неточные дубликаты.

В своих экспериментах мы использовали известные Java-проекты, представленные в табл. 2. Кроме объёма кода мы также рассматриваем число классов в проектах, а также количество методов, и комментариев. Наконец, мы рассматриваем число пар неточных дубликатов, найденных в результате наших экспериментов.

4. Постановка экспериментов

В работах [16], [22] повторяющиеся фрагменты рассматривались безотносительно к структуре документа, что приводило к нахождению большого количества бессмысленных дубликатов. В работе [2] предлагается средство поиска повторяющихся тэгов в Javadoc, но предложенных подход также приводит к нахождению значительного количества очень «мелкозернистых» дубликатов, и полученную информацию в итоге слишком сложно анализировать.

В данной работе мы решили искать дубликаты в виде целых комментариев, относящиеся к различным сущностям исходного кода – классам, методам, аннотациям и т.д. Это позволило нам работать с заведомо осмысленными фрагментами текста.

Мы привлекли к эксперименту двух Java-разработчиков. Чтобы обеспечить корректность эксперимента, они независимо работали с одними и теми же данными, затем они обсудили результаты своей работы и пришли к общему мнению там, где их решения расходились.

Набор данных, размеченный экспертами, был создан следующим образом. Вначале мы объединили весь исходный код каждого проекта в один текстовый файл и оставили в этом файле лишь комментарии Javadoc и объявления соответствующих им программных сущностей. Затем мы запустили на полученных файлах Clone Miner (инструмент для поиска программных клонов на уровне лексем исходного кода) [4], получили набор групп клонов длинной не менее 5 слов в комментариях, подобно тому, как мы ранее делали это в [22].

Если некоторая группа G содержала два соседних комментария (а такие случаи встречались, поскольку Clone Miner не обращает внимания на структуру анализируемого текста), то мы разделяли каждый её клон на два и получали две группы. В нашем случае этого оказалось достаточно, поскольку более двух соседних комментариев в клоны одной группы не

попадало. Затем мы отсортировали группы по убыванию размера их клонов. Далее мы редактировали группы, заменяя каждую группу G на G' таким образом, что для всех клонов $g \in G$, текст которых являлся частью комментария g', в G' включался полный текст комментария g'. Таким образом, мы получили группы комментариев.

Далее, эксперты вручную анализировали каждую группу комментариев и выбирали пары комментариев, которые являлись неточными дубликатами. Для визуализации повторяющихся фрагментов комментариев одной группы мы использовали Duplicate Finder [24]. Полученный набор *положительных* пар был автоматически дополнен таким же количеством *отрицательных* пар комментариев из различных случайно выбранных групп. Отрицательные пары затем были дополнительно проверены экспертами. Помимо упомянутых инструментов в ходе эксперимента мы также использовали несколько разработанных вспомогательных скриптов на языке Python. В результате был получен размеченный экспертами набор данных, включающий 2600 пар комментариев (см. столбец «Количество пар» в табл. 2). Средняя длина Javadoc-комментария в рассматриваемых проектах равнялась 52 словам.

Табл. 2. Исследованные проекты
Table 2. Evaluated projects

Название проекта	Размер, Кб	Количество классов	Количество методов	Количество комментариев	Количество пар
GSON[9]	480	50	281	426	364
JUnit4[10]	588	103	433	602	744
Mockito[11]	901	110	471	669	786
Slf4J[12]	163	21	158	227	706

Для того, чтобы принять решение касательно того, является ли пара положительной или отрицательной, эксперты оценивали не только сходство текста комментариев, но и то, насколько близкую функциональность они описывали. Встречались ситуации, когда степень текстуального сходства была высока, но по смыслу комментарии были далеки. Такие пары учитывались как отрицательные. Если комментарии пары имели большой по объему сходный текст с одинаковым тегом (например, в них присутствовало одно и то же исключение), но при этом общая функциональность программных сущностей, к которым относились комментарии, значительно различалась, то такие пары учитывались как отрицательные. Наконец, некоторые короткие комментарии включали словосочетание или фразу, относящуюся к одному и тому же объекту (например, «tests to be run by the receiver»), но в целом комментарии описывали совершенно разную функциональность. Такие пары эксперты также считали отрицательными.

Нами было выполнено два эксперимента. Эксперимент 1 был проведён на размеченном экспертами наборе данных с разделением на обучающую и тестовую выборки в пропорции 70/30. Для каждого алгоритма мы нашли пороговое значение схожести, на котором достигалось максимальное значение меры F1. Вдобавок, для лучшей адаптации алгоритмов,

[9] https://github.com/google/gson
[10] https://github.com/junit-team/junit4
[11] https://github.com/mockito/mockito
[12] https://github.com/qos-ch/slf4j

основанных на машинном обучении (WMD, D2V, SNN), мы натренировали их эмбеддинги на большом наборе Javadoc-комментариев (приблизительно 2 миллиона), взятом с Kaggle[13].

Для того, чтобы получить более реалистичные результаты, мы провели эксперимент 2, в котором заместили тестовые данные *полным набором* Javadoc-комментариев из проекта JUnit4. При этом мы исключили пары JUunit, использованные в обучающей выборке. Поскольку в этом эксперименте тестовые данные не были полностью размечены экспертами, последние вручную проверили дополнительные положительные пары, найденные в ходе эксперимента 2. Для упрощения работы эксперты использовали инструмент Duplicate Finder [24], основанный на методе поиска точных повторов [26] и наглядно отображающий одинаковые фрагменты текста в отдельных элементах группы неточных дубликатов (в данном случае – в парах Javadoc-комментариев).

Поскольку наши исходные данные для обоих экспериментов не были сбалансированы в плане количества положительных и отрицательных пар, мы делали выводы о результатах наших экспериментов на основе трёх метрик – точности (P, Precision), полноты (R, Recall) и меры F1.

5. Результаты и дискуссия

Результаты наших экспериментов представлены в табл. 3. Столбцы 2-4 относятся к эксперименту 1, столбцы 5-8 – к эксперименту 2. В рамках эксперимента 1 мера F1 колеблется в пределах от 0,94 до 0,97. Строковые алгоритмы показывают постоянную точность (0,97), тогда как в случае алгоритмов на основе машинного обучения точность меняется от 0,92 (SNN) до 0,98 (WMD).

Табл. 3. Результаты экспериментов
Table 3. Experiment Results

Алгоритмы	Эксперимент 1			Эксперимент 2			
	P	R	F1	P	R	F1	Время, с.
LCS	0,97	0,95	0,96	0,96	0,74	0,84	15
COS	0,97	0,94	0,96	0,88	0,82	0,85	19
LSH	0,97	0,90	0,94	0,93	0,69	0,80	625
D2V	0,94	0,99	0,97	0,64	0,90	0,75	3157
WMD	0,98	0,95	0,96	0,97	0,73	0,84	1949
SNN	0,92	0,97	0,95	0,79	0,86	0,82	82

Полученные в ходе эксперимента 2 результаты ниже результатов, полученных в ходе эксперимента 1. Это объясняется тем, что в последнем случае был использован набор данных, вручную созданный экспертами. Как следствие, обучающие данные состояли из пар комментариев, хорошо разделённых на положительные и отрицательные, сходство комментариев в положительных парах близко к 1, сходство в отрицательных близко к 0. При постановке эксперимента 2 в нашем распоряжении было б*о*льшее количество пар, сходство в которых близк*о* к пороговому значению, отделяющему отрицательные и положительные пары. Их следовало бы использовать в качестве обучающих данных, но из-за большой

[13] https://www.kaggle.com/isofew/code-comment-pairs

трудоёмкости разметки новых данных этого не было сделано. Алгоритмы на основе машинного обучения, особенно D2V, оказались более чувствительны к данной специфике входных данных, нежели строковые алгоритмы.

Эксперимент 2 показывает, что неточных дубликатов среди комментариев очень много – 564 (неточные дубликаты) из 602 (общее количество дубликатов) в случае проекта JUnit4. Можно сделать вывод, большинство проектов в рассматриваемом проекте является неточными дубликатами, и лишь немногие из комментариев уникальны. Это свидетельствует о том, что при создании комментариев в реальных программных проектов копирование и вставка – нормальная практика, хотя для подтверждения этого вывода необходимо провести дополнительные эксперименты.

Время работы всех алгоритмов на данных проекта JUnit4 представлено в таблице 3 (столбец «время»). Хуже всех в этом смысле показали себя D2V и WMD (50 и 30 минут соответственно), при этом JUnit4 не является большим проектом. Таким образом, подобная производительность для практического применения неприемлема. D2V работал так медленно, поскольку по ходу работы он вычисляет свои весовые коэффициенты, в то время как остальные алгоритмы используют коэффициенты, вычисленные при тренировке. Использованный нами вариант алгоритма WMD тратил много времени для вычисления метрики Earth Mover's Distance, и в целом его сложность оценивается как $O(p^3 * \log p)$, где p – это количество уникальных слов в сравниваемых текстовых фрагментах. Время работы LCS, COS и SNN варьировалось от 15 секунд до полутора минут, что вполне приемлемо для использования на практике.

В нашем случае LSH не показал хороших результатов, поскольку Javadoc-комментарии короче топиков (topics) DITA, рассматриваемых в [23], поэтому хеш-сигнатуры работали недостаточно эффективно.

LCS, COS и WMD показали наилучшие результаты в обоих экспериментах. Но напомним, что WMD существенно проигрывает строковым алгоритмам в плане производительности. LCS же напротив, показывает наилучшую производительность, так что в итоге мы считаем его лучшим выбором для применения на практике. К такому же заключению пришли и авторы работы [23], хотя в своём исследовании они рассматривали другой набор алгоритмов.

Наконец, при анализе данных нам удалось найти несколько ошибок в комментариях, возникших вследствие копирования-вставки: в каких-то случаях различия комментариев в некоторых парах были излишними или некорректными, в других случаях различий не было там, где им следовало бы быть.

В качестве дальнейшего направления исследований мы предполагаем разработку подходов, инструментов и сервисов для поиска ошибок, вызванных копированием и вставкой. Также важно улучшить качество алгоритмов вычисления сходства комментариев, дополнив их специфичными для наших задач возможностями и обучая их на данных, лучше сбалансированных в плане количества положительных и отрицательных пар, и имеющих больший объём.

5. Заключение

В работе, применительно к вычислению схожести Javadoc-комментариев, были проанализированы различные алгоритмы вычисления схожести фрагментов текста. Мы установили, что существующие алгоритмы хорошо справляются с этой задачей. Проведённые эксперименты показали, что строковый алгоритм LCS (поиск наибольшей общей подпоследовательности) показал себя наилучшим образом. Также результаты говорят в пользу того, что интерактивные (с участием человека) методы, обладающие специфичными для решаемых задач возможностями, и классические строковые алгоритмы могут лучше показывать себя на практике по сравнению с методами, основанными на машинном обучении. Однако это требует дальнейшего исследования, поскольку основанные на

машинном обучении алгоритмы предположительно имеют значительный потенциал, но требуют и значительных усилий, чтобы его раскрыть. Также представляют интерес визуальные средства анализа экспериментов в духе [27-29], а также средства управления знаниями [30].

Литература

[1]. Spinellis D. Code Documentation // IEEE Softw. – 2010. – Vol. 27, no. 4. – pp. 18–19.
[2]. Oumaziz M. A. et al. Documentation Reuse: Hot or Not? An Empirical Study // Proc. of ICSR 2017. – 2017. – pp. 12–27.
[3]. Blasi A., Gorla A. Replicomment: identifying clones in code comments // Proc. of ICPC 2018, Gothenburg, Sweden. – ACM. – 2018. – pp. 320–323.
[4]. Nosál M., Porubän J. Reusable software documentation with phrase annotations // Central Eur. J. Comput. Sci. – 2014. – Vol. 4, no. 4. – pp. 242–258.
[5]. Corazza A. et al. On the Coherence between Comments and Implementations in Source Code // EUROMICRO-SEAA. – 2015. – pp. 76–83.
[6]. Chin F. Y. L., Poon C. K. Binary Codes Capable of Correcting Deletions, Insertions and Reversals // Afast algorithm for computing longest common subsequences of small alphabet size. – 1991. – Vol. 13(4). – pp. 463–469.
[7]. Manning C. D. et al. Introduction to information retrieval. – 2008. – Vol. 1.
[8]. Gionis A. et al. Similarity Search in High Dimensions via Hashing // Proc. of VLDB 1999, Edinburgh, Scotland, UK. – Morgan Kaufmann. – 1999. – pp. 518–529.
[9]. Broder A. Z. On the resemblance and containment of documents // Proc. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171). – IEEE, 1997. – c. 21-29.
[10]. Kusner M. J. et al. From Word Embeddings To Document Distances // Proc. of ICML 2015, Lille, France. – JMLR.org. – 2015. – Vol. 37. – pp. 957–966.
[11]. Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.
[12]. Le Q.V., Mikolov T. Distributed Representations of Sentences and Documents // ICML 2014. – 2014. – Vol. 32 of JMLR Workshop and Conference Proceedings. – pp. 1188–1196.
[13]. Mueller J., Thyagarajan A. Siamese Recurrent Architectures for Learning Sentence Similarity // Proc. AAAI, 2016. – AAAI Press. – 2016. – pp. 2786–2792.
[14]. Tan S. H. et al. @tComment: Testing Javadoc Comments to Detect Comment-Code Inconsistencies // ICST 2012. – IEEE Computer Society. – 2012. – pp. 260–269.
[15]. Fluri B., Würsch M., Gall H. C. Do Code and Comments Co-Evolve? On the Relation between Source Code and Comment Changes // Proc. of WCRE 2007, Vancouver, BC, Canada. – IEEE Computer Society. – 2007. – pp. 70–79.
[16]. Luciv D., Koznov D., Chernishev G., et al. Detecting Near Duplicates in Software Documentation // Programming and Computer Software. – 2018. – Vol. 44, no. 5. – pp. 335–343.
[17]. Wen F. et al. A large-scale empirical study on code-comment inconsistencies // Proc. of ICPC 2019 / ed. by Guéhéneuc Y. et al. – IEEE / ACM. – 2019. – pp. 53–64.
[18]. Wang D. et al. Deep Code-Comment Understanding and Assessment // IEEE Access. – 2019. – Vol. 7. – pp. 174200– 174209.
[19]. Zhou Y. et al. Analyzing APIs documentation and code to detect directive defects // Proc of the ICSE 2017, Buenos Aires, Argentina. – IEEE / ACM. – 2017. – pp. 27–37.
[20]. Ratol I. K., Robillard M. pp. Detecting fragile comments // Proc. of ASE 2017, Urbana, IL, USA. – IEEE Computer Society. – 2017. – pp. 112–122.
[21]. Otaibi J. A. et al. Machine Learning and Conceptual Reasoning for Inconsistency Detection // IEEE Access. – 2017. – Vol. 5. – pp. 338–346.
[22]. Koznov D. V. et al. Clone Detection in Reuse of Software Technical Documentation // 10th International Andrei Ershov Informatics Conference, PSI 2015. – Springer. – 2015. – Vol. 9609 of LNCS. – pp. 170–185.
[23]. Soto A. J. et al. Similarity-Based Support for Text Reuse in Technical Writing // Proc. of the ACM DocEng 2015, Lausanne, Switzerland / ed. by Vanoirbeek C., Genevès pp. – ACM. – 2015. – pp. 97–106.
[24]. Luciv D., Koznov D., Chernishev G., et al. Duplicate finder toolkit // Proc. of ICSE 2018: Companion Proceeedings. – 2018. – pp. 171–172.

[25]. Wagner S., Fernández D. M. Analyzing Text in Software Projects // The Art and Science of Analyzing Software Data / ed. by Bird Christian et al. – Morgan Kaufmann / Elsevier, 2015. – pp. 39–72.
[26]. Basit H. A. et al. Efficient token based clone detection with flexible tokenization // Proceedings of the ESEC/SIGSOFT FSE, 2007. – 2007. – pp. 513–516.
[27]. Кознов Д.В., Ольхович Л.Б.Визуальные языки проектов // Системное программирование. 2005. Т. 1. с. 148-167.
[28]. Кознов Д.В. Методология и инструментарий предметно-ориентированного моделирования // диссертация на соискание ученой степени доктора технических наук / Санкт-Петербургский государственный университет. Санкт-Петербург, 2016.
[29]. Гаврилова Т., Алсуфьев А., Янсон А.С. Современные нотации бизнес-моделей: визуальный тренд // Форсайт. 2014. Т. 8. № 2. с. 56-70.
[30]. Гаврилова Т.А. Логико-лингвистическое управление как введение в управление знаниями // Новости искусственного интеллекта. 2002. № 6. с. 36-40.

Информация об авторах / Information about authors

Дмитрий Владимирович КОЗНОВ - доктор технических наук, профессор кафедры системного программирования. Научные интересы: программная инженерия, модельно-ориентированная разработка программного обеспечения, программные данные, машинное обучение.

Dmitry Vladimirovich KOZNOV - Doctor of Technical Sciences, Professor of the System Programming Department. Research interests: software engineering, model-driven software development, program data, machine learning.

Дмитрий Вадимович ЛУЦИВ – кандидат физико-математических наук, доцент кафедры системного программирования Санкт-Петербургского государственного университета. Область научных интересов: программная инженерия, анализ данных программных проектов, анализ документации, системное программирование.

Dmitry Vadimovich LUCIV – PhD in computer science, associate professor of System Programming Department at Saint Petersburg State University, Russia. Research interests: software engineering, software data analysis, documentation analysis, systems programming.

Екатерина Юрьевна ЛЕДЕНЕВА – разработчик ООО Яндекс.Технологии, выпускница кафедры системного программирования Санкт-Петербургского государственного университета. Сфера научных интересов: анализ данных программных проектов, анализ технических текстов.

Ekaterina Iurevna LEDENEVA – Software engineer at Yandex LLC, Saint Petersburg State University alumni. Research interests: software data analysis, technical documentation analysis.

Павел Исаакович БРАСЛАВСКИЙ - кандидат технический наук, старший научный сотрудник научно-учебной лаборатории моделей и методов вычислительной прагматики НИУ «Высшая школа экономики». Научные интересы: ресурсы и методы для оценки моделей обработки естественного языка и информационного поиска, вычислительный юмор.

Pavel Isaakovich BRASLAVSKI - PhD in computer science, senior researcher at the Laboratory for Models and Methods of Computational Pragmatics, HSE University. Research interests: resources and methods for evaluation of NLP and IR models, computational humor.

DOI: 10.15514/ISPRAS-2023-35(4)-11

Моделирование русловых процессов в створе канала

[1] *И. И. Потапов, ORCID: 0000-0002-3323-2727 <potapov2i@gmail.com>*
[2] *Д. И. Потапов, ORCID: 0000-0001-6394-228X <potapovdi9@mail.ru>*

[1] *Вычислительный центр Дальневосточного отделения Российской академии наук, г. Хабаровск, Россия*
[2] *Институт горного дела Дальневосточного отделения Российской академии наук, г. Хабаровск, Россия.*

Аннотация. Сформулирована математическая модель эрозии берегового склона песчаного канала, происходящей под действием проходящей паводковой волны. Модель включает в себя уравнение движения квазиустановившегося гидродинамического потока в створе канала. Движение донной и береговой поверхности русла определяется из решения уравнения Экснера, которое замыкается оригинальной аналитической моделью движения влекомых наносов. Модель не содержит в себе феноменологических параметров, учитывает транзитные, гравитационные и напорные механизмы движения донного материала. Движение свободной поверхности гидродинамического потока определяется из интерполяции экспериментальных данных. Модель учитывает изменения средней по створу турбулентной вязкости при изменении створа канала. Исследовано влияние квазиустановившегося гидродинамического потока на потерю массы в створе канала. Введен критерий для определения неравновесности руслового потока. Показано, что для моделирования деформаций створа в данном случае необходимо учитывать ненулевой градиент движения наносов вдоль оси канала. Проведены численные расчеты, демонстрирующие качественное и количественное влияние данных особенностей на процесс определения турбулентной вязкости потока и эрозию берегового склона русла. Сравнение данных по береговым деформациям, полученных в результате численных расчетов, с известными лотковыми экспериментальными данными показало их хорошее согласование.

Ключевые слова: русловые процессы; деформации дна; створ канала; эрозия берегового склона; метод конечных элементов.

Для цитирования: Потапов И. И., Потапов Д. И. Моделирование русловых процессов в створе канала. Труды ИСП РАН, том 35, вып. 4, 2023 г., стр. 187–196. DOI: 10.15514/ISPRAS–2023–35(4)–11.

Modeling of Channel Processes in a Channel Cross Section

[1] *I. I. Potapov, ORCID: 0000-0002-3323-2727 <potapov2i@gmail.com>*
[2] *D. I. Potapov, ORCID: 0000-0001-6394-228X <potapovdi9@mail.ru>*

[1] *Computing Center of the Far Eastern Branch of the Russian Academy of Sciences, Khabarovsk, Russia*
[2] *Institute of Mining, Far Eastern Branch of the Russian Academy of Sciences, Khabarovsk, Russia.*

Abstract. An erosion mathematical model of a sand channel coastal slope, which occurs under the influence of passing flood wave, is formulated. The model includes the equation of motion of a quasi-steady hydrodynamic flow in the channel section. The movement of the bottom and bank surface of the channel is

determined from the solution of the Exner equation, which is closed by an original analytical model of the movement of traction sediments. The model takes into account transit, gravitational and pressure mechanisms of movement of bottom material, and does not contain phenomenological parameters. The movement of the free surface of a hydrodynamic flow is determined from the interpolation of experimental data. The model takes into account changes in the average turbulent viscosity along the alignment when the channel alignment changes. The influence of quasi-steady hydrodynamic flow on mass loss in the channel section was studied. A criterion has been introduced to determine the disequilibrium of the channel flow. It is shown that to model channel deformations in this case, it is necessary to take into account a non-zero gradient of sediment movement along the channel axis. Numerical calculations have been carried out demonstrating the qualitative and quantitative influence of these features on the process of determining the turbulent viscosity of the flow and the erosion of the coastal slope of the channel. Data comparison on coastal deformations obtained as a result of numerical calculations with known flume experimental data showed their good agreement.

Keywords: channel processes, bottom deformations of channel cross section, coastal slope erosion, finite element method.

For citation: Potapov I.I., Potapov D.I. Modeling of channel processes in a channel cross section. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 4, 2023. pp. 187-196 (in Russian). DOI: 10.15514/ISPRAS-2023-35(4)-11.

1. Введение

Изучение процесса эрозии берегового склона песчаного канала под действием протекающей в канале воды является важной прикладной задачей. Первые попытки построения строгих математических моделей для определения формы поперечного сечения русла восходят к работам [1,2], в которых были предложены эвристические модели, позволившие получить качественные оценки изучаемого процесса. Дальнейшее развитие теории было связано с большим количеством экспериментальных работ [3 - 6]. Были предложены математические модели, учитывающие различные механизмы переноса донного материала, путем влечения по дну, взвешивания частиц донного материала в речном потоке и лавинного движения донных частиц при обрушении берегов [4,7]. Однако, применение феноменологических формул транспорта донного материала позволило лишь качественно описывать процессы береговой эрозии. В работе [8] были предложены и развиты [9] аналитические формулы транспорта донного материала, не содержащие в себе феноменологических коэффициентов.

В работах [10,11] предложены математические модели для изучения процессов береговых деформаций русла при постоянном расходе речного потока.

В данной работе предложена математическая модель задачи о развитии процесса эрозии берегового склона с учетом возможного изменения расхода транзитных наносов во времени. Поскольку характерное время установления гидродинамических параметров потока много меньше характерного времени изменения его расхода [9,11], гидродинамический поток описывается в рамках квазистационарного приближения. Однако предлагаемая модель может учитывать потерю массы в створе канала за счет введения поправки на ненулевой градиент транзитного расхода вдоль оси канала.

Турбулентная вязкость потока изменяется во времени из-за изменений геометрии расчетной области, уровня свободной поверхности потока и расхода потока. Для описания изменения донных и береговых отметок русла используется аналитическая модель движения наносов, предложенная в работах [9,11].

Для численного решения задачи предложен алгоритм, основанный на методе конечных элементов. Выполнено сравнение результатов расчетов береговых деформаций с известными экспериментальными данными, [4,6,8] показавшее их хорошее согласование.

2. Математическая постановка задачи гидродинамики

Рассмотрим задачу об эрозии берегового склона песчаного канала, трапециевидного в начальный момент времени, геометрия которого схематично представлена на рис 1. Русло

имеет малый продольный уклон J в направлении движения гидродинамического потока, движущегося в поле силы тяжести. Когда отметки дна Z в направлении оси x изменяются по линейному закону

$$Z(t,x,y)=\zeta(t,y)-xJ, \qquad J=-\frac{\partial Z}{\partial x}=const\ .$$

Ввиду симметрии русла расчет изменения поперечного сечения канала выполнялся только для левой половины сечения. Геометрия расчетной области Ω представлена на рис. 1.

Граница Γ_1 подвижна и представляет собой смоченную береговую и донную поверхности и сухой береговой склон. Γ_2 – свободная поверхность потока, изменяющаяся во времени вследствие изменения геометрии донной поверхности, Γ_3 – граница, совпадающая с центром симметрии канала, изменяющаяся вследствие изменения глубины потока. Определение геометрии подвижных границ Γ_k во времени является целью решения задачи.

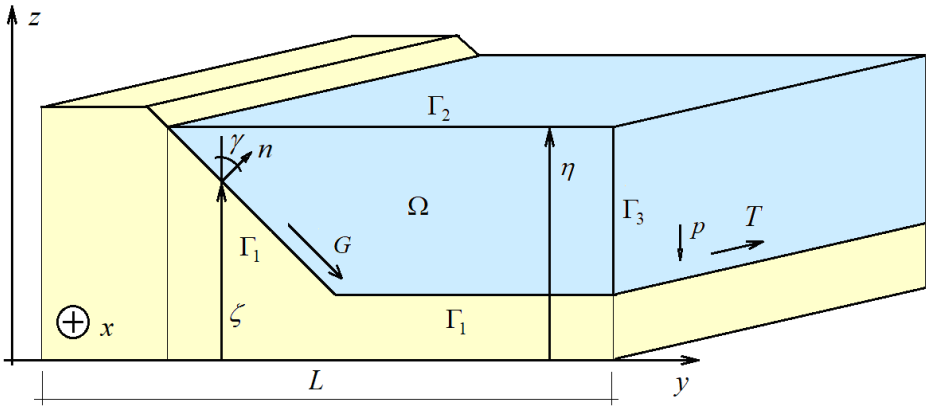

Рис.1. Геометрия расчетной области Ω *и ее границ* Γ_k

Fig.1. Geometry of the computational domain Ω *and its boundaries* Γ_k

Для определения границ Γ_k предложена математическая модель, содержащая в себе:

– уравнение Рейнольдса для гидродинамического потока в створе канала [13]

$$\frac{\partial}{\partial y}\left(\mu_t\frac{\partial u}{\partial y}\right)+\frac{\partial}{\partial z}\left(\mu_t\frac{\partial u}{\partial z}\right)=\rho_w g\,J\ , \tag{1}$$

– уравнение Экснера [14]

$$(1-\varepsilon)\frac{\partial\zeta}{\partial t}+\frac{\partial G_x}{\partial x}+\frac{\partial G_y}{\partial y}=0\ , \tag{2}$$

– уравнения расхода донного материала [9]

$$G_x=A-B\frac{\partial\zeta}{\partial x},\quad G_y=-D\frac{\partial\zeta}{\partial y}, \tag{3}$$

Уравнения (1)–(2) замыкаются граничными условиями

$$u = 0, \quad z \in \Gamma_1, \qquad \frac{\partial u}{\partial z} = 0, \quad y, z \in \Gamma_2, \qquad \frac{\partial u}{\partial y} = 0, \quad z, y \in \Gamma_3, \tag{4}$$

$$\mu_t = \mu_0 , \quad y, z \in \Omega \tag{5}$$

Уравнения (3)–(4) замыкаются начальными

$$\zeta(0, y) = \zeta_0(y), \qquad 0 \le y \le L, \ \ t = 0, \tag{6}$$

и граничными условиями

$$\frac{\partial \zeta(t,\ 0)}{\partial y} = 0, \qquad \frac{\partial \zeta(t,\ L)}{\partial y} = 0, \qquad 0 \le t \le T_p \tag{7}$$

Постановка (1)-(8) замыкается дополнительными зависимостями

$$\int_{\Omega} u(\mu_t) d\Omega - Q_m = 0, \tag{9}$$

$$T_{xy} = \mu_t \frac{\partial u}{\partial y}, \ \ T_{xz} = \mu_t \frac{\partial u}{\partial z}, \quad T = T_{xy} n_y + T_{xz} n_z, \tag{10}$$

Параметры аналитической модели для расходов донного материала (3) определяются следующими зависимостями [9]:

$$A = \frac{G_1 P}{\cos\gamma \left(1 - |\tan\gamma|^2 / (\tan\varphi)^2\right)}, \qquad B = \frac{G_1 P}{\tan\varphi \cos\gamma \left(1 - |\tan\gamma|^2 / (\tan\varphi)^2\right)},$$

$$D = \frac{4}{5}\frac{G_1 P}{\cos\gamma}, \qquad G_1 = \frac{4}{3}\frac{1}{\kappa\sqrt{\rho_w}\,(\rho_s - \rho_w) g\,(\tan\varphi)^2},$$

$$T_c = T_0 \cos\gamma \left(1 - \frac{|\tan\gamma|}{\tan\varphi}\right), \qquad \tan\gamma = \frac{\partial \zeta}{\partial y}, \qquad \cos\gamma = 1/\sqrt{1 + \left(\frac{d\zeta}{dy}\right)^2},$$

$$T_0 = \frac{9}{8}\frac{\kappa^2 (\rho_s - \rho_w) g\, d_{50} \tan\varphi}{c_x}, \qquad P = T \max\left(0, \sqrt{|T|} - \sqrt{T_c}\right),$$

где $u = u(y, z)$ – осредненная скорость речного потока в створе канала Ω, J – уклон речного русла, g – ускорение свободного падения, $\eta = \eta(t)$, $\zeta = \zeta(t, y)$ – отметки свободной и донно-береговой поверхностей потока соответственно, μ_t – турбулентная вязкость потока, T_{0c} – критическое придонное касательное напряжение на ровном дне, ρ_w, ρ_s – плотность воды и песка соответственно, φ – угол внутреннего трения донных частиц, γ – острый угол между нормалью к поверхности дна и вертикальной линией, c_x – лобовое сопротивление частиц, d_{50} – средний диаметр донных частиц, Q_{in} – заданный расход потока, κ – коэффициент Кармана, ζ_0 – отметки дна, μ_t – турбулентная вязкость потока, $s_f = f\,\frac{\rho_s - \rho_w}{\rho_w}$, $f \approx 0.1$ – концентрация наносов в активном слое.

3. Неравновесные русловые процессы в створе канала

В качестве критерия неравновесного руслового процесса в створе канала можно принять интегральную формулу баланса массы

$$\Delta_{\Omega}=\frac{\rho_s\Omega_s-\rho_s\Omega_{\text{e}}}{\rho_s(\Omega_s)}100\%=\frac{\Omega_{\text{e}}-\Omega_s}{\Omega_s}100\% \tag{11}$$

где Ω_e, Ω_s - площади размыва и намыва донного материала в створе канала соответственно. Нулевое значение Δ_{Ω} указывает на протекание руслового процесса в равновесном состоянии, при $\Delta_{\Omega}>0$ происходит заиливание створа, а при $\Delta_{\Omega}<0$ его размыв. Отметим, что критерий (11) зависит от временного периода, на котором он определяется, поэтому для сравнения двух экспериментов с различными периодами наблюдения разумно использовать относительный критерий неравновесного руслового процесса

$$Q_{\Omega}=\frac{\Delta_{\Omega}}{T_p} \tag{12}$$

Данный критерий определяет процентный расход объема в единицу времени, характерный для данного эксперимента.

4. Верификация модели

Для численного решения задач использовался метод и алгоритм, описанный в работе [12]. Для верификации предложенной математической модели (1)-(7) было решено три задачи. Во всех трех задачах экспериментальные результаты, приведенные на рис. 2-4, обозначаются точечными множествами.

В первой задаче рассматривалась деформация донной поверхности затопленного трапециевидного канала. Сравнение вычисленных и экспериментальных отметок донной поверхности приведены на рис.2. Кривой 1 показана начальная геометрия трапециевидного канала [8], кривыми 2-4 эволюция профиля берегового откоса в моменты времени t = 0.3, t = 2.5, t = 5.54 секунд соответственно, при постоянном расходе 112 л/с и уровне свободной поверхности $\eta=0.25$м. Отметим, что экспериментаторам удалось с высокой степенью точности стабилизировать поток, потеря массы в сечении в данном эксперименте не превышает $\Delta_{\Omega}\sim 3.5\%$. При решении данной задачи величина G_x принималась постоянной.

Во второй задаче рассматривалась деформация донной поверхности трапециевидного канала при уровне свободной поверхности потока, равном бровке берегового склона [8]. Расчет второй задачи выполнялся с параметрами экспериментальной работы [4], а полученные результаты сравнивались с экспериментальными данными данной работы. На рис. 3 кривой 1 показана начальная геометрия трапециевидного канала [4], кривой 2 – форма донной поверхности на момент времени t = 600 минут. В отличие от предыдущей задачи в данном примере, русловой процесс рассматривается на длительном периоде времени t = 600 минут, поэтому, не смотря на то, что критерий Δ_{Ω} в данном примере имеет значимую величину достигая 23%, величина $Q_{\Omega}=6.4\circ 10^{-4}$ мала, а следовательно в эксперименте [4] экспериментаторам удалось сформировать установившийся русловой поток с более высокой степенью точности, чем в работе [8].

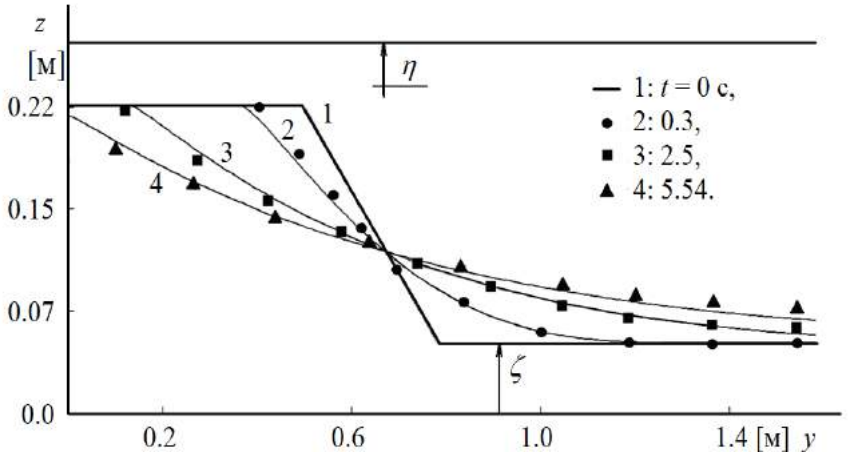

Рис. 2. Изменение геометрии канала во времени, сравнение с результатами работы [8]
Fig. 2. Change in channel geometry over time, comparison with the results of [8]

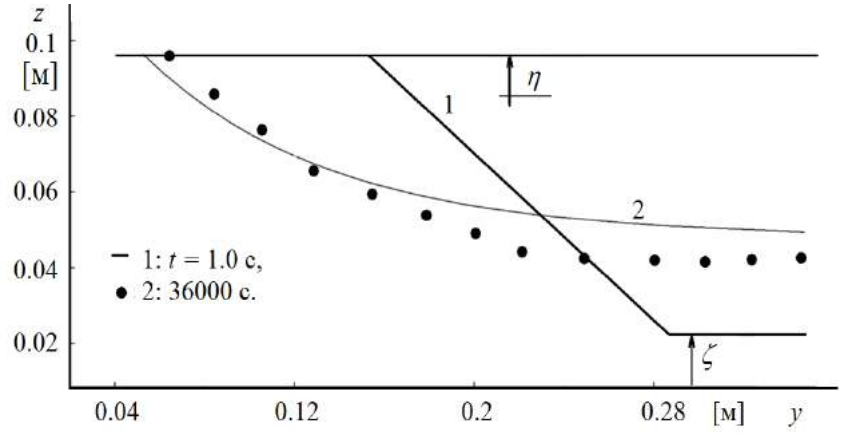

Рис. 3. Изменение геометрии канала во времени, сравнение с результатами работы [4]
Fig. 3. Change in channel geometry over time, comparison with the results of [4]

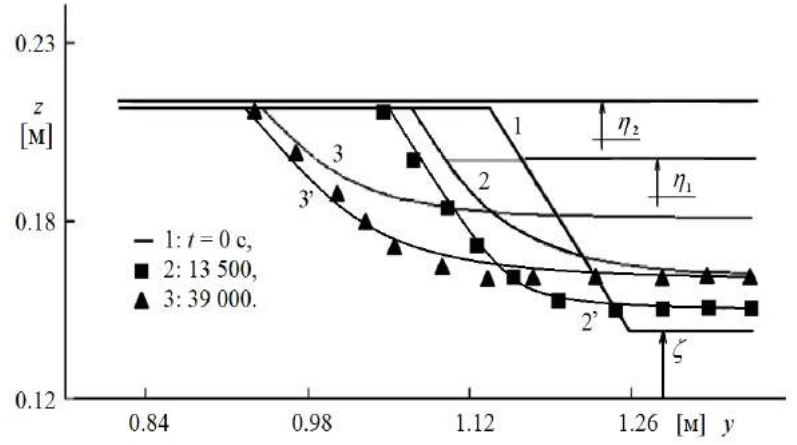

Рис. 4. Изменение геометрии канала во времени, сравнение с результатами работы [6]
Fig. 4. Change in channel geometry over time, comparison with the results of [6]

К сожалению, отсутствие промежуточных отметок донной поверхности не позволяет построить зависимость (13) для внесения поправок на градиент продольных наносов и при решении данной задачи величина G_x принималась постоянной.

В третьей задаче эволюция берегового откоса происходила при переменных расходе и уровне свободной поверхности [6]. Согласно работе [6] глубина потока в начальный момент времени составляет 0.06 м, в момент времени t = 225 минут глубина потока у стенки в течении минуты была изменена, достигнув значения 0.066 м.

Развитие донной поверхности в створе канала на первом этапе t < 225 мин. происходило под воздействием установившегося гидродинамического потока с расходом 11,9 л/с. На втором этапе расход потока достигал 16,3 л/с. Горизонтальной штрихпунктирной линией на рис. 4. обозначена свободная поверхность потока в момент времени t = 650 мин.

Сравнение вычисленных и экспериментальных отметок донной поверхности приведено на рис.4. Кривой 1 на рис. 4 показана начальная геометрия трапециевидного канала. Неразмываемая стенка трапециевидного канала находится справа y = 1.5 м. Кривой 2 показана форма донной поверхности [6] на момент времени t = 225 минут, развившаяся из начальной геометрии дна (кривая 1).

Из анализа кривых 2-3 рис. 4 видно, что сравнение полученных расчетных отметок дна (кривые 2-3) с экспериментальными отметками дна по фронту берегового склона показывает их хорошее согласование, однако уровень средней отметки дна в расчетах существенно выше уровня полученных в экспериментах. Следовательно, несмотря на попытку экспериментаторов в работе [6] получить установившийся процесс береговых деформаций, баланс материала, сползающего с берегового откоса в основное русло канала, не соблюдается как на первом, так и на втором этапе деформации створа канала. Критерий неравновесности руслового процесса на первом этапе достигает Δ_Ω = 700%. То есть отношение объема песка, смытого с берегового откоса (кривая 2) при размыве канала более чем в 7 раз превышает объем песка, перешедшего в ложе канала. В связи с этим определяемая критерием Q_Ω погрешность достигает значения $Q_\Omega = 0.05$%/с, что на два порядка хуже, чем в работах Икеды [4] и соизмеримо с экспериментами Петрова [8]. На втором этапе параметр неравновесности руслового процесса понижается, достигая Δ_Ω = 380% и $Q_\Omega = 0.015$%/с соответственно. Оценка

$$\frac{\partial G_x}{\partial x} = (1-\varepsilon)\frac{\zeta^E - \zeta^C}{T_p} \tag{13}$$

позволяет из экспериментальных данных в центре донной плоской части канала оценить среднее значение для неустановившегося градиента транзитного потока наносов на первом и втором этапе соответственно

$$\frac{\partial G_x}{\partial x} = (1-0.3)\frac{0.15-0.16}{13500} = -0.519\cdot 10^{-6}\ \frac{м}{с}, \tag{14 а}$$

$$\frac{\partial G_x}{\partial x} = (1-0.3)\frac{0.16-0.18}{25500} = -0.549\cdot 10^{-6}\ \frac{м}{с}. \tag{14 б}$$

Учет поправок (14) при расчете донных деформаций в третьей задаче позволяет существенно улучшить прогноз по вычисленным донным деформациям (кривые 2'-3' рис.4), показывая, что численное решение должно учитывать экспериментальные ошибки для правильного описания эволюции донной поверхности, полученной в данных экспериментах.

5. Результаты

В работе выполнен анализ применимости аналитической русловой модели для моделирования развития берегового откоса трапециевидного канала. Полученные результаты показывают, что при верификации математической модели донных деформаций в створе канала с установившимся русловым потоком необходимо учитывать точность, которая достигается экспериментаторами при выполнении требования к установлению потока вдоль оси канала. Если в эксперименте параметр Δ_Ω велик, то необходимо выполнить оценку градиента расхода транзитных наносов вдоль оси канала, а при получении для нее значимых значений использовать данную поправку при расчетах донных деформаций в створе канала.

Выполненные по предложенной модели расчеты показывают, что основным параметром, влияющим на эволюцию береговой линии, является диаметр донных частиц. Для более крупных частиц, использованных в экспериментах Икеды [4], был выполнен подбор больших значений коэффициента лобового сопротивления. Показано, что использованная при решении рассмотренных задач модель гидродинамики позволяет достаточно точно делать прогноз изменений береговой кромки канала.

Список литературы / References

[1]. Glover R. E., Florey Q. L. Stable channel profiles // U. S. Bureau of Reclamation, Washington. 1951.

[2]. Макавеев Н.И. Русло реки и эрозия в ее бассейне. М.: Издательство АН СССР, 1955, 348 р./ [2]. Makaveev N.I. River bed and erosion in its basin. M.: Publishing House of the USSR Academy of Sciences, 1955, 348 p. (in Russian).

[3]. Ikeda S., Parker G., Saway K. Bend theory of river meanders. Part 1. Linear development // J. Fluid Mech. 1981. no. 112. P. 363–377.

[4]. Ikeda S., Parker G., Kimura Y. Stable width and depth of straights gravel rivers with heterogeneous bed materials // J. Water resource research. 1988. Vol. 24, no. 5. P. 713-722.

[5]. Parker G. Self–formed straight rivers with equilibrium banks and mobile bed. Part 1. The sand–silt river // J. Fluid Mech. 1978. part 1, Vol. 89. P. 109–126.

[6]. Pitlick J., Marr J., Pizzuto J. Width adjustment in experimental gravel-bed channels in response to overbank flows // Journal of geophysical research: Earth surface. 2013. Vol. 118. P. 553–570.

[7]. Ikeda S. Stable channel cross–sections of straight sand rivers // J. Water Resources Res., 1991. Vol. 27, no. 9. P. 2429–2438.

[8]. Петров П.Г. Движение сыпучей среды в придонном слое жидкости // ПМТФ, 1991. № 5. С. 72 — 75. / Petrov P.G. Movement of a granular medium in the bottom layer of liquid // PMTF, 1991. No. 5. P. 72 — 75. (in Russian).

[9]. Петров А.Г., Потапов И.И. Избранные разделы русловой динамики // М.: Ленанд. 2019. 244 с. / Petrov A.G., Potapov I.I. Selected sections of channel dynamics // M.: Lenand. 2019. 244 p. (in Russian).

[10]. Бондаренко Б.В., Потапов И.И. Моделирование эволюции поперечного сечения песчаного канала // Вычислительные технологии evolution of the cross section of the sand channel. 2009. Т.14, № 5. С. 1–14. / Bondarenko B.V., Potapov I.I. Modeling the evolution of the cross section of the sand channel // Computational technologies evolution of the cross section of the sand channel. 2009. T.14, no. 5. pp. 1–14. (in Russian).

[11]. Потапов И.И., Бондаренко Б.В. Математическое моделирование эволюции берегового склона в каналах с песчаным руслом // Вычислительные технологии. 2013. Т.18, № 4. С. 25–36./ Potapov I.I., Bondarenko B.V. Mathematical modeling of the evolution of the coastal slope in channels with a sandy bed // Computational technologies. 2013. T.18, no. 4. pp. 25–36 (in Russian).

[12]. Потапов Д. И., Потапов И.И. Развитие берегового откоса в русле трапециевидного канала//Компьютерные исследования и моделирование. 2022, Т. 14, № 3, С. 581–592/ Potapov D.I., Potapov I.I. Development of a coastal slope in the bed of a trapezoidal canal // Computer research and modeling. 2022, T. 14, No. 3, pp. 581–592 (in Russian).

[13]. Гончаров В.Н. Динамика русловых потоков // Л.: Гидрометеоиздат. 1962. с 367.

[14]. Exner, F. M. (1920), Zur physik der du¨nen, Akad. Wiss. Wien Math. Naturwiss. Klasse, 129(2a), 929 – 952.

Информация об авторах / Information about authors

Игорь Иванович ПОТАПОВ – доктор физико-математических наук, профессор, заведующий лаборатории вычислительной механики Вычислительного центра Дальневосточного отделения Российской академии наук с 2009 года. Сфера научных интересов: численные методы, русловые и гидродинамические процессы в равнинных реках.

Igor Ivanovich POTAPOV – Doctor of Physical and Mathematical Sciences, Professor, Head of the Laboratory of Computational Mechanics of the Computing Center of the Far Eastern Branch of the Russian Academy of Sciences since 2009. Area of scientific interests: numerical methods, channel and hydrodynamic processes in lowland rivers.

Дмитрий Игоревич ПОТАПОВ – младший научный сотрудник, аспирант, Института горного дела Дальневосточного отделения Российской академии наук.

Dmitry Igorevich POTAPOV – junior researcher, postgraduate student, Institute of Mining, Far Eastern Branch of the Russian Academy of Sciences.

DOI: 10.15514/ISPRAS-2023-35(4)-12

Символьное вычисление условия резонанса произвольного порядка в системе Гамильтона

[1,2] *Батхин А.Б.,* ORCID:0000-0001-8871-4697<*batkhin@gmail.com*>
[3] Хайдаров З. Х., ORCID: 0000-0002-2580-4887 <*zafarxx@gmail.com*>
[1] *Институт прикладной математики им. М.В. Келдыша РАН,*
125047, Россия, г. Москва, Миусскаяпл.,д.4
[2] *Московский физико-технический институт*
141701, Россия, Московская область, г. Долгопрудный, Институтский переулок, д. 9
[3] *Самаркандский государственный университет им. Ш. Рашидова,*
Узбекистан, г. Самарканд, Университетский бул., д.15

Abstract. Исследование формальной устойчивости положений равновесия многопараметрической системы Гамильтона в случае общего положения традиционно проводится с использованием её нормальной формы при условии отсутствия резонансов небольших порядков. В работе предлагается способ символьного вычисления условия существования резонанса произвольного порядка для системы с тремя степенями свободы. Показано, что это условие для каждого резонансного вектора может быть представлено в виде рациональной алгебраической кривой. Методами компьютерной алгебры получена рациональная параметризация этой кривой для случая общего резонанса. Рассмотрен модельный пример некоторой двупараметрической системы маятникового типа.

Keywords: символьное вычисление, условие резонанса, система Гамильтона.

For citation: Батхин А. Б., Хайдаров З. Х. Символьное вычисление условия резонанса произвольного порядка в системе Гамильтона. Труды ИСП РАН, 2023, том 35 вып. 4, с. 197– 218. 10.15514/ISPRAS-2023-35(4)-12.

Благодарности: Авторы выражают благодарность профессору А.Д. Брюно за указанные замечания и полезное обсуждение работы.

Symbolic Computation of an Arbitrary-Order Resonance Condition in a Hamiltonian System

[1,2] *A.B. Batkhin,* ORCID: 0000-0001-8871-4697 <*batkhin@gmail.com*>
[3] *Z.Kh. Khaydarov,* ORCID: 0000-0002-2580-4887 <*zafarxx@gmail.com* >
[1] *Keldysh Institute of Applied Mathematics, Russian Academy of Sciences,*
4, Miusskaya pl., Moscow, Russia, 125047
[2] *Moscow Institute of Physics and Technology,*
9, Institutskii per., Dolgoprudnyi, Moscow oblast, Russia, 141701
[3] *Sharof Rashidov Samarkand State University,*
15, Universitetskii bul'v., Samarkand, Uzbekistan, 140104

Abstract. The study of formal stability of equilibrium positions of a multiparametric Hamiltonian system in a generic case is traditionally carried out using its normal form under the condition of the absence of resonances of small orders. In this paper we propose a method of symbolic computation of the condition of existence of a resonance of arbitrary order for a system with three degrees of freedom. It is shown that this condition for each resonant vector can be represented as a rational algebraic curve. By methods of computer algebra the rational parametrization of this curve for the case of an arbitrary resonance is obtained. A model example of some two-parameter system of pendulum type is considered.

Keywords: Hamiltonian system, equilibrium state, normal form, formal stability, resonance condition, elimination ideal.

For citation: Batkhin A.B., Khaydarov Z.Kh. Symbolic Computation of an Arbitrary-Order Resonance Condition in a Hamiltonian System. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 4, 2023. pp. 197-218. DOI: 10.15514/ISPRAS-2023-35(4)-12.

Acknowledgements. Authors express their gratitude to Professor A.D. Bruno for comments and useful discussions.

1. Введение

Для многих механических систем наличие резонанса между собственными частотами приводит к появлению сложной динамики, когда энергия колебаний «перекачивается» между теми степенями свободы, чьи соответствующие частоты находятся в резонансе. Однако, наличие нетривиальных решений резонансного уравнения позволяет получить дополнительные формальные первые интегралы и, как следствие, позволяет провести анализ устойчивости положения равновесия или асимптотически проинтегрировать систему уравнений движения, приведённую к нормальной форме. Кроме того, для многопараметрических систем, в которых отсутствуют сильные резонансы (см. определение 3), позволяет при определённых условиях находить в пространстве параметров области формальной устойчивости (устойчивость по Мозеру [1]).

Формальная устойчивость по Мозеру слабее, чем классическая устойчивость по Ляпунову, но гарантирует скорость разбегания траекторий медленнее, чем любая степенная функция с произвольным положительным показателем. С практической точки зрения формальная устойчивость вполне достаточна для большинства технических механических систем.

Существует два типа задач об устойчивости многопараметрических систем.

- Для определённых значений вектора параметров P выяснить устойчивость положения равновесия (ПР) (*частная задача*).

  Найти в пространстве параметров Π все значения P, для которых ПР z = 0 системы устойчиво, т.е. вычислить так называемое множество устойчивости ∑ системы (*общая*

задача).

В статье рассматривается схема решения общей задачи.

Цель работы состоит в описании схемы исследования формальной устойчивости ПР системы Гамильтона с тремя степенями свободы, а также в получении явного представления условий существования резонансов кратности 1 в терминах коэффициентов характеристического многочлена линейной части гамильтоновой системы.

Предварительные результаты работы были опубликованы в препринте авторов [2]. В данной статье авторы предлагают усовершенствованный способ получения полиномиальной параметризации резонансного многообразия, соответствующего трёхчастотному резонансу общего вида. Этот способ может быть обобщён на системы с большим числом степеней свободы.

Структура работы следующая: в разделах 2 и 3 напоминаются результаты о множестве устойчивости линейной системы Гамильтона и гамильтоновой нормальной форме соответственно; в разделе 4 рассматривается общая схема исследования формальной устойчивости положения равновесия системы Гамильтона и формулируется постановка задачи; в разделе 5 доказывается основной результат в виде теоремы 4 и описывается способ символьного вычисления полиномиального представления резонансного многообразия, соответствующего трёхчастотному резонансу общего вида, в явной и неявной формах. Наконец, в разделе 6 приведён пример вычисления всех многообразий, соответствующих сильным резонансам, для двупараметрической системы маятникового типа с тремя степенями свободы.

Обозначения

- Полужирные символы типа $\mathbf{x}, \mathbf{y}, \mathbf{u}, \mathbf{v}$ обозначают столбцы-векторы в n-мерных вещественных $\mathbb{R}^n$ или комплексных $\mathbb{C}^n$ пространствах.
- Полужирные символы типа $\mathbf{p}, \mathbf{q}$ обозначают векторы в n-мерной целочисленной решётке $\mathbb{Z}^n$.
- $|\mathbf{p}| = \sum_{j=1}^{n} |p_j|$ обозначает норму вектора.
- Для $\mathbf{x} = (x_1, \ldots, x_n)^\top$ и $\mathbf{p} = (p_1, \ldots, p_n)^\top$ обозначим через $\mathbf{x}^\mathbf{p} \equiv \prod\limits_{j=1}^{n} x_j^{p_j}$ моном и через $\langle \mathbf{p}, \mathbf{x} \rangle \equiv \sum\limits_{j=1}^{n} p_j x_j$ скалярное произведение пары векторов.

2. Множество устойчивости линейной гамильтоновой системы

Рассмотрим автономную гамильтонову систему с аналитической функцией $H(\mathbf{z}; \mathbf{P})$, ПР которой совпадает с началом координат. Тогда гамильтониан $H(\mathbf{z}; \mathbf{P})$ раскладывается в сходящийся ряд однородных полиномов $H_k(\mathbf{z}; \mathbf{P})$ степени k от своих фазовых переменных $\mathbf{z} = (\mathbf{x}, \mathbf{y})$

$$H(\mathbf{z}; \mathbf{P}) = \sum_{k=2}^{\infty} H_k(\mathbf{z}; \mathbf{P}), \tag{2.1}$$

где $\mathbf{P}$ — вектор параметров.

В случае общего положения ряд (2.1) начинается с квадратичного гамильтониана $H_2(\mathbf{z}; \mathbf{P})$, определяющего локальную динамику вблизи ПР. Поведение фазового потока в первом приближении описывается линейной гамильтоновой системой

$$\dot{\mathbf{z}} = B(\mathbf{P})\mathbf{z}, \quad B(\mathbf{P}) = \frac{1}{2} J \frac{\partial^2 H_2(\mathbf{z}; \mathbf{P})}{\partial \mathbf{z}^2}. \tag{2.2}$$

Напомним здесь основные свойства линейной гамильтоновой системы.

1. Если λ_j есть собственное число матрицы B, то $-\lambda_j$ также является её СЧ [3]. Все СЧ λ_j, $j = 1, \ldots, 2n$, матрицы B могут быть упорядочены таким образом, что $\lambda_{j+n} = -\lambda_j$, $j = 1, \ldots, n$. Обозначим через вектор $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ множество *базисных собственных значений*.
2. Характеристический многочлен $\check{f}(\lambda)$ матрицы B содержит только чётные степени λ, поэтому он является многочленом от $\mu = \lambda^2$. Такой многочлен назван в [4] *полухарактеристическим*

$$f(\mu) = \sum_{k=0}^{n} a_{n-k}(\mathbf{P})\mu^k, \quad a_0 \equiv 1. \tag{2.3}$$

3. Если $\operatorname{Re}\lambda_j \neq 0$ для некоторого j, то ПР неустойчиво.
4. Если все $\operatorname{Re}\lambda_j = 0$, то поведение фазового потока в его окрестности может быть получено только при учёте нелинейных членов.

Определение 1. *Множество устойчивости* Σ линейной системы (2.2) — это множество всех значений параметров $\mathbf{P} \in \Pi$, для которых ПР $\mathbf{z} = 0$ системы (2.2) устойчиво по Ляпунову.

В терминах корней многочлена (2.3) условие устойчивости ПР даётся следующей теоремой.

Теорема 1 ([4])**.** *Положение равновесия* $\mathbf{z} = 0$ *линейной гамильтоновой системы* (2.2) *устойчиво по Ляпунову тогда и только тогда, когда*

1. *все корни* μ_k *полухарактеристического многочлена* (2.3) *вещественны и неположительны;*
2. *все элементарные делители матрицы* B *просты.*

Невыполнение условия 1 приводит к экспоненциальному разбеганию решений, которое не перекрывается нелинейными добавками, привносящими только степенной эффект в поведение решений. Невыполнение условия 2 приводит к степенной неустойчивости, которая может быть перекрыта нелинейными добавками.

Условие вещественности и неположительности корней многочлена $f(\mu)$ определяется следующей теоремой.

Теорема 2. *Для того чтобы все корни многочлена* $f(\mu)$ *степени* n *были вещественны и неположительны, необходимо и достаточно выполнение условий* $a_j(\mathbf{P}) \geqslant 0$, $j = 1, \ldots, n$, $D^{(k)}(f) \geqslant 0$, $k = 0, \ldots, n-2$, *где* $D^{(k)}(f)$ *— k-й субдискриминант многочлена* $f(\mu)$ *[5, 6].*

В гамильтоновом случае согласно п. 4 полный анализ устойчивости возможен только с учётом нелинейных членов. Такое исследование следует выполнять приведя исходный гамильтониан (2.1) к наиболее простому виду, называемому нормальной формой (НФ).

3. Нормальная форма системы Гамильтона

В дальнейшем считаем, что выполнено условие 1 устойчивости теоремы 1. Если не выполнено условие 2 теоремы 1, то устойчивость определяется по НФ общими методами.

Согласно теореме 12 в [7] существует каноническое формальное преобразование в виде степенного ряда, которое приводит исходную систему Гамильтона к её *нормальной форме*

$$\dot{\mathbf{u}} = \frac{\partial h}{\partial \mathbf{v}}, \quad \dot{\mathbf{v}} = -\frac{\partial h}{\partial \mathbf{u}},$$

задаваемой нормализованным гамильтонианом $h(\mathbf{u}, \mathbf{v})$

$$h(\mathbf{u}, \mathbf{v}) = \sum_{j=1}^{n} \sigma_j \lambda_j u_j v_j + \sum h_{\mathbf{pq}} \mathbf{u}^{\mathbf{p}} \mathbf{v}^{\mathbf{q}},\ \sigma_j = \pm 1, \tag{3.1}$$

который содержит только резонансные члены $h_{\mathbf{pq}} \mathbf{u}^{\mathbf{p}} \mathbf{v}^{\mathbf{q}}$, удовлетворяющие условию

$$\langle \mathbf{p} - \mathbf{q}, \boldsymbol{\lambda} \rangle = 0. \tag{3.2}$$

Здесь $0 \leqslant \mathbf{p}, \mathbf{q} \in \mathbb{Z}^n$, $|\mathbf{p}| + |\mathbf{q}| \geqslant 2$ и $h_{\mathbf{pq}}$ — постоянные коэффициенты.
Резонансное уравнение (3.2) имеет два вида решений, которым соответствуют два вида резонансных членов в НФ (3.1):

1. *вековые члены* вида $h_{\mathbf{pp}} \mathbf{u}^{\mathbf{p}} \mathbf{v}^{\mathbf{p}}$, которые всегда присутствуют в гамильтоновой нормальной форме из-за особой структуры матрицы B линейной части системы (2.2); вековые члены являются мономами только чётных степеней от фазовых переменных и входят в соответствующие однородные формы;
2. *строго резонансные члены*, которые соответствуют нетривиальным целочисленным решениям уравнения

$$\langle \mathbf{p}, \boldsymbol{\lambda} \rangle = 0. \tag{3.3}$$

Сама процедура нормализации обычно выполняется в комплексных переменных. Для перехода к комплексным переменным используется линейное преобразование $\Phi : (\mathbf{x}, \mathbf{y}) \to (\boldsymbol{\zeta}, \bar{\boldsymbol{\zeta}})$, которое преобразует исходный гамильтониан к виду

$$h\left(\boldsymbol{\zeta}, \bar{\boldsymbol{\zeta}}\right) = \sum h_{\mathbf{pq}} \boldsymbol{\zeta}^{\mathbf{p}} \bar{\boldsymbol{\zeta}}^{\mathbf{q}}, \tag{3.4}$$

где $\mathbf{p}, \mathbf{q} \in \mathbb{Z}^n$ и $|\mathbf{p}| + |\mathbf{q}| \geqslant 2$. Значение $|\mathbf{p}| + |\mathbf{q}|$ называется *порядком* соответствующего члена разложения.

Определение 2. Функция Гамильтона $h\left(\boldsymbol{\zeta}, \bar{\boldsymbol{\zeta}}\right)$ называется *комплексной нормальной формой* вещественной системы Гамильтона для случая полупростых СЧ, если

1. Его квадратичная часть $h_{\mathbf{2}}$ имеет вид

$$h_{\mathbf{2}} = \sum_{j=1}^{n} \sigma_j \lambda_j \zeta_j \bar{\zeta}_j,\ \sigma_j = \pm 1;$$

2. Разложение (3.4) содержит только члены $h_{\mathbf{pq}} \boldsymbol{\zeta}^{\mathbf{p}} \bar{\boldsymbol{\zeta}}^{\mathbf{q}}$, которые удовлетворяют резонансному уравнению (3.2). Константы σ_j являются инвариантами НФ.

Дальнейшие результаты связаны с существованием резонансов в системе Гамильтона, поэтому напомним их определение.

Определение 3 ([8, Гл.I, § 3])**.** *Кратность резонанса* $\mathfrak{k}$ — это число линейно независимых решений $\mathbf{p} \in \mathbb{Z}^n$ резонансного уравнения $\langle \mathbf{p}, \boldsymbol{\lambda} \rangle = 0$. *Порядок резонанса* равен $\mathfrak{q} = \min |\mathbf{p}|$ по $\mathbf{p} \in \mathbb{Z}^n$, $\mathbf{p} \neq 0, \langle \mathbf{p}, \boldsymbol{\lambda} \rangle = 0$. Если решение резонансного уравнения содержит только два собственных значения, то такой резонанс называется *двухчастотным резонансом*, если более двух — то *многочастотным резонансом*. Резонансы порядков 2, 3 и 4 назовём *сильными*, больших порядков — *слабыми* резонансами.

Пусть $\mathcal{L}$ подпространство, образованное всеми решениями уравнения (3.3). Тогда согласно [8, Гл.I, § 3] величина

$$\langle \boldsymbol{\rho}, \boldsymbol{\omega} \rangle = \text{const}, \quad \boldsymbol{\rho} = (\rho_1, \ldots, \rho_n), \quad \rho_j = \zeta_j \bar{\zeta}_j, \tag{3.5}$$

есть первый интеграл для любого вектора $\boldsymbol{\omega}$ из ортогонального дополнения $\mathcal{L}^{\perp}$ к подпространству $\mathcal{L}$.

Рассмотрим каким образом наличие резонансов и их кратность влияет на количество первых интегралов системы Гамильтона в НФ.

1. Пусть уравнение (3.3) не имеет нетривиальных решений. Тогда кратность резонанса $\mathfrak{k} = 0$ и имеется n независимых первых интегралов (3.5) НФ. Это есть НФ Биркгофа [9]

   $$h\left(\boldsymbol{\zeta}, \bar{\boldsymbol{\zeta}}\right) = \sum_{k=1}^{\infty} h_{2k}\left(\boldsymbol{\zeta}, \bar{\boldsymbol{\zeta}}\right),$$

   состоящая из однородных форм чётных степеней $2k$, зависящих только от переменных вида $\zeta_j \bar{\zeta}_j$, $j = 1, \ldots, n$, при этом каждая из величин $\zeta_j \bar{\zeta}_j$ является формальным первым интегралом. В переменных действие-угол $(\boldsymbol{\varphi}, \boldsymbol{\rho})$ НФ Биркгофа может быть записана в виде

   $$h(\boldsymbol{\rho}) = \sum_{k=1}^{\infty} h_k(\boldsymbol{\rho}).$$

   Поскольку все ρ_j независимы, то в этом случае НФ интегрируема.
2. Пусть резонанс имеет кратность $\mathfrak{k} = 1$. Это всегда выполнено, если резонанс двухчастотный. Для многочастотных резонансов кратность 1 означает, что для любого меньшего набора частот резонансное уравнение имеет только тривиальное решение. Резонанс кратности 1 обеспечивает наличие $n-1$ независимых первых интегралов (3.5), которые вместе с интегралом $h = \text{const}$ дают интегрируемость НФ.
3. Если система имеет две степени свободы, то её НФ всегда интегрируема, поскольку при отсутствии резонансов имеет место НФ Биркгофа п. 1, а при его наличии этот резонанс всегда двухчастотный, т. е. обеспечивает наличие ещё одного независимого интеграла (3.3).
4. Для системы с тремя степенями свободы согласно теореме 3.1 в [8, Гл.I, § 3] достаточное для интегрируемости нормализованной системы Гамильтона число первых интегралов вида (3.3) обеспечивает либо двухчастотный резонанс, либо трёхчастотный резонанс кратности 1.

Ниже потребуется условие, с помощью которого последующие результаты проще формулируются.

Условие A_k^n[10]
Будем говорить, что для нормализованной системы Гамильтона выполняется условие A_k^n, *если резонансное уравнение* (3.3) *не имеет целочисленных решений* $\mathbf{p}$ *с* $|\mathbf{p}| \leqslant k$.

4. Схема исследования формальной устойчивости

Определение 4. ПР $\mathbf{z} = 0$ системы с функцией Гамильтона $H(\mathbf{z})$ *формально устойчиво*, если существует возможно расходящийся степенной ряд $G(\mathbf{z})$, который является формальным положительно определенным первым интегралом $\{G, H\} = 0$.

В [10] было дано схематическое описание метода изучения формальной устойчивости ПР. Этот метод основан на следующих ключевых результатах:

- вычисляется НФ системы Гамильтона в окрестности ПР;
- применяется теорема Брюно [11] о формальной устойчивости;
- используются q-аналоги объектов классической теории исключений.

Пусть имеет место условие A_4^n, т. е. $\langle \mathbf{L}, \boldsymbol{\lambda} \rangle \neq 0$ для $\mathbf{L} \in \mathbb{Z}^n$, $0 < |\mathbf{L}| \leqslant 4$, тогда существует аналитическое каноническое преобразование $(\mathbf{x}, \mathbf{y}) \to (\boldsymbol{\rho}, \boldsymbol{\varphi})$ такое, что новый гамильтониан g имеет вид

$$g(\boldsymbol{\rho}, \boldsymbol{\varphi}) = g_1(\boldsymbol{\rho}) + g_2(\boldsymbol{\rho}) + r(\boldsymbol{\rho}, \boldsymbol{\varphi}),$$

где $g_1(\boldsymbol{\rho}) = \langle \boldsymbol{\lambda}, \boldsymbol{\rho} \rangle$, $g_2(\boldsymbol{\rho}) = \langle C\boldsymbol{\rho}, \boldsymbol{\rho} \rangle$, $C = [c_{ij}]_{i,j=1}^n$, и $r(\boldsymbol{\rho}, \boldsymbol{\varphi})$ является сходящимся степенным рядом переменных $(\boldsymbol{\rho}, \boldsymbol{\varphi})$ степени три или выше в $\boldsymbol{\rho}$.

При отсутствии сильных резонансов условие формальной устойчивости гамильтоновой системы в окрестности ПР определяется следующей теоремой.

Теорема 3 (Брюно [11])**.** *Если условие A_4^n выполнено и для любых ненулевых целых векторов $\mathbf{L}$, которые являются решением уравнения $\langle \mathbf{L}, \boldsymbol{\lambda} \rangle = 0$, квадратичная форма $\langle C\mathbf{L}, \mathbf{L} \rangle \neq 0$ при $\boldsymbol{\lambda} \neq 0$, то ПР $\mathbf{z} = 0$ гамильтоновой системы формально устойчиво.*

Таким образом, для применения теоремы 3 о формальной устойчивости необходимо найти границы областей в пространстве параметров Π, определяемых резонансными многообразиями (см. определение 5 ниже), соответствующие сильным резонансам.

Определение 5. *Резонансным многообразием $\mathcal{R}_n^{\mathrm{P}}$* в пространстве $\mathbb{K}$ коэффициентов $a_1, \ldots, a_n$ полухарактеристического многочлена $f_n(\mu)$ степени n назовём такое алгебраическое многообразие, на котором вектор базовых собственных значений $\boldsymbol{\lambda}$ соответствующего характеристического многочлена $\check{f}(\lambda)$ является нетривиальным решением резонансного уравнения (3.3) для фиксированного целочисленного вектора $\mathbf{p}$. Аналитическое представление многообразия $\mathcal{R}_n^{\mathrm{P}}$ в неявной или параметрической формах далее обозначим R_n^{P}.

Постановка задачи

Для исследования формальной устойчивости ПР гамильтоновой системы необходимо:

- в пространстве параметров Π найти множество устойчивости Σ линейной системы;
- в ней определить области, в которых квадратичная форма $H_2(\mathbf{z}; \mathbf{P})$ не является знакоопределённой;
- в найденных областях выделить их части S_k, в которых отсутствуют сильные резонансы;
- в каждой из таких частей S_k выполнить процедуру нормализации гамильтониана до четвёртого порядка включительно и применить теорему 3.

Для выполнения последнего пункта достаточно выбрать какую-либо точку в каждой из частей S_k в пространстве параметров и воспользоваться каким-либо алгоритмом нормализации функции Гамильтона. Поскольку в каждой внутренней точке части S_k все собственные числа λ_k, $k = 1, \ldots, n$ простые, то легко применим алгоритм инвариантной нормализации [12].

В данной работе рассматривается описание резонансных многообразий порядков 2, 3 и 4 в пространстве коэффициентов $\mathbb{K}$ полухарактеристического многочлена $f(\mu)$ для системы Гамильтона с тремя степенями свободы.

Задача 1 (Основная задача)**.** Для многопараметрической системы Гамильтона с 3 степенями свободы дать описание областей в пространстве параметров системы, в которых отсутствуют резонансы порядков 2, 3 и 4.

Рассмотрим более подробно, при каких условиях реализуются резонансы указанных выше порядков. Для резонанса

- порядка $\mathfrak{q} = 2$: $p = (1, 1, 0)$ — это случай кратных корней, который описывается дискриминантным множеством $R_3^{(1,1,0)} \equiv D(f) = 0$;
- порядок $\mathfrak{q} = 3$: для двухчастотного случая $p = (2, 1, 0)$, описывается q-дискриминантом $R_3^{(2,1,0)} \equiv D_4(f) = 0$;
- порядок $\mathfrak{q} = 4$: для двухчастотного случая $p = (3, 1, 0)$, описывается q-дискриминантом $R_3^{(3,1,0)} \equiv D_9(f) = 0$;
- в трёхчастотном случае: для порядка 3 описывается условием $R_3^{(1,1,1)} = 0$, а для порядка 4 — условием $R_3^{(2,1,1)} = 0$.

Для решения этой задачи следует получить описание границ областей, свободных от сильных резонансов. Эти границы состоят из участков алгебраических многообразий, на которых резонансное уравнение (3.3) имеет нетривиальное решение.

Основную задачу разобьём на несколько вспомогательных задач.

1. Получить аналитическое представление в пространстве коэффициентов $\mathbb{K} = (a_1, a_2, a_3)$ кубического многочлена резонансных многообразий $\mathcal{R}_3^{\mathbf{p}}$ для всех векторов $\mathbf{p}$ порядков 2, 3 и 4.
2. Выяснить взаимное расположение всех найденных выше резонансных многообразий, т. е. определить, каким образом указанные выше резонансные многообразия касаются или пересекаются в пространстве $\mathbb{K}$.

5. Вычисление условия существования резонансов в системе с тремя степенями свободы

Рассмотрим два способа вычисления условий существования резонансных соотношений для заданного резонансного вектора $\mathbf{p}^*$.

Первый способ позволяет получить неявное представление резонансного многообразия $\mathcal{R}_3^{\mathbf{p}}$ в пространстве коэффициентов $\mathbb{K}$, а уже затем исследовать его структуру.

Второй способ позволяет получить сначала параметрическое представление резонансного многообразия $\mathcal{R}_3^{\mathbf{p}}$, а затем с его помощью вычислить неявное представление, выполнить визуализацию многообразия и исследовать структуру особых точек последнего.

Каждый из этих способов предполагает, что сначала найдено резонансное соотношение между корнями μ_j многочлена $f(\mu)$ для заданного вектора $\mathbf{p}^*$, которое вычисляется следующим образом.

1. Для заданного вектора $\mathbf{p}^* = (r, q, 1)$, где $r, q \in \mathbb{Q}$, $r, q \neq 0$, удовлетворяющего резонансному уравнению $\langle \mathbf{p}, \boldsymbol{\lambda} \rangle = 0$, составляется полиномиальный идеал

$$\mathcal{J} = \left\{ \langle \mathbf{p}^*, \boldsymbol{\lambda} \rangle, \lambda_j^2 - \mu_j, j = 1, \ldots, n \right\}$$

2. Вычисляется базис Грёбнера $\mathcal{G}$ этого идеала с исключающим мономиальным порядком переменных $\lambda_j, \mu_j, j = 1, \ldots, n$. Первый многочлен $\mathfrak{g}_1$ из $\mathcal{G}$ является квазиоднородным многочленом по переменным $\mu_j, j = 1, \ldots, n$. Его нули определяют условие существования резонанса для данного вектора $\mathbf{p}^*$.

Для резонансного вектора $\mathbf{p}^* = (r, q, 1)$ это условие имеет вид квадратичной формы от корней μ_k, $k = 1, 2, 3$:

$$R_3^{(r,q,1)}(\mu_j) \equiv q^4\mu_2^2 - 2q^2r^2\mu_1\mu_2 + r^4\mu_1^2 - 2q^2\mu_2\mu_3 - 2r^2\mu_1\mu_3 + \mu_3^2 = 0 \tag{5.1}$$

Отметим, что все описанные выше вычисления выполнялись с использованием систем компьютерной алгебры (СКА) `Maple`. При этом применялись следующие пакеты и процедуры:

- Для вычисления и исследования идеалов $\mathcal{G}, \mathcal{F}$ использовалась процедура `Basis` построения базисов Грёбнера для различных лексикографических порядков, а также некоторые дополнительные процедуры пакета `Groebner`.
- Для определения рода кривой использовалась команда `genus`, а для нахождения параметризации — `parametrization` из пакета `algcurves`.

Апробация методов была выполнена на модельном примере, описанном в п. 6.

5.1 Первый способ вычисления

Данный способ подробно описан в работах авторов [2, 13] поэтому здесь приведём его лишь краткую схему вычислений.

Для получения соответствующего резонансного условия $\mathcal{R}_3^{\mathbf{p}^*}$ в неявной форме в виде нулей полинома с коэффициентами $a_j, j = 1, \ldots, 3$ полинома $f(\mu)$, строится новый базис Грёбнера $\mathcal{F}$ идеала, который содержит полученное условие (5.1) для μ_j и множество элементарных симметрических многочленов, связывающих коэффициенты a_j многочлена $f(\mu)$ и его корни μ_k:

$$\begin{aligned} &\mu_1 + \mu_2 + \mu_3 + a_1 = 0, \\ &\mu_1\mu_2 + \mu_1\mu_3 + \mu_2\mu_3 - a_2 = 0, \\ &\mu_1\mu_2\mu_3 + a_3 = 0. \end{aligned} \tag{5.2}$$

При вычислении базиса $\mathcal{F}$ указывается следующий порядок исключения переменных: в начале μ_j, а затем $a_j, j = 1, \ldots, n$. Таким же образом, нули первого полинома $\mathfrak{f}_1$ вычисленного базиса $\mathcal{F}$, зависящие только от a_j, дают условие существования резонанса.

Многочлен $\mathfrak{f}_1$ получается в виде квазиоднородного многочлен от a_j, т. е. его носитель в пространстве векторных показателей степеней коэффициентов a_j лежит в плоскости с нормальным вектором $\mathbf{n} = (1, 2, 3)$. Это позволяет после определённого степенного преобразования уменьшить число переменных полинома $\mathfrak{f}_1$ на 1 и получить условия существования резонансов в виде набора плоских алгебраических кривых.

Этот метод оказался довольно трудоёмким для резонансов общего вида и приводит к очень громоздким выражениям. Его обобщение на случаи со степенями свободы больше 3 авторами не представляется возможным.

5.2 Второй способ вычисления

Для условия (5.1) строится степенное преобразование (см., например, [**BrunoAzimovProg202**]), определяемое матрицей

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

с соответствующей заменой переменных

$$\mu_1 = s_2 s_3, \quad \mu_2 = s_1 s_3, \quad \mu_3 = s_3. \tag{5.3}$$

Это преобразование приводит квадратичную форму (5.1) после сокращения на s_3^2 к квадратичной функции

$$\tilde{R}_3^{(r,q,1)} \equiv q^4 s_1^2 - 2q^2 r^2 s_1 s_2 + r^4 s_2^2 - 2q^2 s_1 - 2r^2 s_2 + 1 = 0,$$

которая, как известно, допускает рациональную параметризацию. Эта параметризация может быть построена таким образом

$$\begin{aligned} s_1 &= \left(\frac{u}{u+2q}\right)^2, \\ s_2 &= \left(\frac{(q+1)u+2q}{r(u+2q)}\right)^2, \\ s_3 &= r^2 v (u+2q)^2, \end{aligned}$$

что параметризация (5.3) корней получится полиномиальной:

$$\begin{aligned} \mu_3 &= v r^2 (u+2q)^2, \\ \mu_2 &= v r^2 u^2, \\ \mu_1 &= v((q+1)u+2q)^2. \end{aligned} \tag{5.4}$$

Структура параметризации (5.4) такова, что для вещественных значений параметров u, v корни μ_k являются вещественными. Более того, выбирая значения параметра $v \leqslant 0$, получим, что все корни μ_k вещественны и неположительны, что согласно теореме 2 обеспечивает линейную устойчивость. Проверим, что найденное параметрическое представление корней (5.4) обеспечивает выполнение резонансного уравнения (3.3) для вектора $\mathbf{p}^* = (r, q, 1)$. Полагая $v = -w^2$, получаем следующий набор корней

$$\begin{aligned} \lambda_{1,4} &= \pm \mathrm{i} w((q+1)u+2q), \\ \lambda_{2,5} &= \pm \mathrm{i} r w u, \\ \lambda_{3,6} &= \pm \mathrm{i} r w (u+2q). \end{aligned}$$

Тогда непосредственной проверкой получаем

$$r\lambda_1 + q\lambda_5 + \lambda_6 = r\lambda_4 + q\lambda_2 + \lambda_3 = 0.$$

Теперь параметрическое представление корней (5.4) с помощью элементарных симметричных многочленов из формулы (5.2) даёт полиномиальную параметризацию коэффициентов a_k, $k = 1, 2, 3$,

$$\begin{aligned} a_1 =& -v\left[(2r^2+(q+1)^2)u^2 + 4q(r^2+q+1)u + \right. \\ & \left. +4q^2(r^2+1)\right], \\ a_2 =& v^2\left[\left(r^2+2(q+1)^2\right)u^4 + 4q\times\right. \\ & \times\left(r^2+q^2+4q+3\right)u^3 + \\ & +4q^2\left(r^2+q^2+6q+7\right)u^2 + \\ & \left. 16q^3(q+1)u + 16q^4\right], \\ a_3 =& -r^4 v^3 u^2 \left(u(q+1)+2q\right)^2 (u+2q)^2 \end{aligned} \tag{5.5}$$

Рассматривая формулы (5.5) как полиномиальный идеал относительно переменных u, v, a_1, a_2, a_3 и вычисляя элиминационный идеал для него, можно получить неявное представление резонансного многообразия $\mathcal{R}_3^{(r,q,1)}$ в виде многочлена $R_3^{(r,q,1)}$ от коэффициентов a_j полухарактеристического многочлена $f_3(\mu)$. Его выражение здесь не приводится из-за его громоздкости: этот квазиоднородный многочлен состоит из 19 мономов, коэффициенты которого суть многочлены от элементов r, q вектора $\mathbf{p}^*$.

Заметим, что параметризация (5.5) не является глобальной параметризацией многообразия $\mathcal{R}_3^{(r,q,1)}$, а только той его части, на которой все корни μ_k вещественны. Однако, не представляет труда получить параметризацию в полиномиальной форме всего резонансного многообразия $\mathcal{R}_3^{(r,q,1)}$. Таким образом, доказано следующая

Теорема 4. *Резонансное многообразие $\mathcal{R}_3^{(r,q,1)}$, где $r, q \in \mathbb{Q}$, допускает полиномиальную параметризацию.*

Далее приведём вычисления резонансных многообразий, соответствующих сильным резонансам согласно списка предыдущего раздела.

5.3 Вычисление условия двухчастотного резонанса

Рассмотрим случай двухчастотного резонанса: $\mathbf{p}^* = (q, 1, 0)$, где $q \in \mathbb{Q}$. Здесь имеется пара соизмеримых СЧ, а третье СЧ несоизмеримо ни с каким из них. В пространстве параметров такой резонанс описывается в терминах резонансного множества $\mathcal{R}_{q^2}(f_3)$ полухарактеристического многочлена $f(\mu)$ [15], либо в терминах обобщённых дискриминантных множеств $\mathcal{D}_{(q^2,0)}(f_3)$[16].

Вычисления проведём с использованием первого способа из подраздела 5.1.

Для начала составим идеал $\mathcal{J}$, содержащий соотношения зависимости между корнями характеристического и полухарактеристического многочленов вида $\lambda_j^2 - \mu_j, j = 1, 2, 3$, а также резонансное соотношение $q\lambda_1 + \lambda_2$. Первый многочлен исключающего базиса Грёбнера $\mathcal{G}$ этого идеала, с последовательным порядком исключения переменных λ_j, μ_j, $j = 1, 2, 3$, есть многочлен

$$\mathfrak{g}_1 \stackrel{\text{def}}{=} q^2\mu_1 - \mu_2. \tag{5.6}$$

Равенство нулю данного многочлена даёт условие на корни полухарактеристического полинома. Чтобы получить условие на коэффициенты, составляем новый идеал $\mathcal{I}$, включающий в себя полученное условие (5.6) и их связь с коэффициентами a_j многочлена (2.3) через элементарные симметрические многочлены (5.2).

Для идеала $\mathcal{I}$ вычисляем исключающий базис Грёбнера $\mathcal{F}$ с соответствующим порядком исключения переменных μ_j, a_j, $j = 1, 2, 3$. Равенство нулю первого его многочлена

$$\begin{aligned}\mathfrak{f}_1 \stackrel{\text{def}}{=} &(q^4 + q^2 + 1)^3 a_3^2 + q^4(q^2 + 1)^2 a_2^3 + \\ &+ q^4(q^2 + 1)^2 a_1^3 a_3 - q^6 a_1^2 a_2^2 - \\ &- q^2(q^4 + q^2 + 1)(q^4 + 4q^2 + 1) a_1 a_2 a_3,\end{aligned} \tag{5.7}$$

зависящего только от a_j, является условием существования двухчастотного резонанса в общем виде для некоторого рационального значения $q \in \mathbb{Q}$.

Проверим полученный результат, сравнивая с полученными ранее условиями для случая, когда $q = 1, 2, 3$. Если в полученных в [16] формулах обобщённых дискриминантов для f_3 положить $\omega = 0$, а $q = q^2$, то получим совпадающие с точностью до знака выражения для соответствующих резонансных многообразий:

- при $q = 1$ условие принимает вид

$$R_3^{(1,1,0)} \stackrel{\text{def}}{=} 4a_1^3a_3 - a_1^2a_2^2 - 18a_1a_2a_3 + 4a_2^3 + 27a_3^2 = 0, \tag{5.8}$$

- при $q = 2$ оно имеет вид

$$R_3^{(2,1,0)} \stackrel{\text{def}}{=} 400a_1^3a_3 - 64a_1^2a_2^2 - 2772a_1a_2a_3 + 400a_2^3 + 9261a_3^2 = 0, \tag{5.9}$$

- а при $q = 3$ оно выглядит

$$R_3^{(3,1,0)} \stackrel{\text{def}}{=} 8100a_1^3a_3 - 729a_1^2a_2^2 - 96642a_1a_2a_3 + 8100a_2^3 + 753571a_3^2 = 0. \tag{5.10}$$

Для выражения (5.7) соответственно со вторым способом из подраздела 5.2 производим степенное преобразование с аналогичной заменой переменных вида (5.3) и использую выражения (5.2), получаем параметрические представления коэффициентов

$$\begin{aligned} a_1 &= -\,s_3\left(\left(q^2+1\right)s_2+1\right), \\ a_2 &= \;\; s_2s_3^2\left(q^2\left(s_2+1\right)+1\right), \\ a_3 &= -\,s_2^2s_3^3q^2. \end{aligned} \tag{5.11}$$

Зная параметризацию коэффициентов можно найти параметрическое представление многообразия (5.7) для произвольного значения q, а так же для каждого из случаев при $q = 1, 2, 3$. Ранее, в работе авторов [13] было использовано степенное преобразование

$$a_1 = 3\nu_3, \quad a_2 = 3\nu_1\nu_3^2, \quad a_3 = \nu_2\nu_3^3, \tag{5.12}$$

которое позволило каждое из резонансных многообразий, представленных в виде нулей квазиоднородных многочленов $R_3^{\mathbf{P}^*}$ от трёх переменных a_1, a_2, a_3 привести к алгебраическим кривым $\tilde{R}_3^{\mathbf{P}^*}$ от двух переменных ν_1, ν_2. Для этих кривых в указанной выше работе были вычислены соответствующие рациональные параметризации. Здесь аналогичная параметризация кривой $\tilde{R}_3^{(q,1,1)}$ легко получается с помощью обращения степенного преобразования (5.12) и подстановки (5.11):

$$\left\{\nu_1 = \frac{3s_2\left(q^2s_2+q^2+1\right)}{\left(q^2s_2+s_2+1\right)^2}, \quad \nu_2 = \frac{27s_2^2q^2}{\left(q^2s_2+s_2+1\right)^3}\right\}.$$

Таким образом, для многообразий (5.8), (5.9) и (5.10) получаются рациональные параметризации соответствующих алгебраических кривых в следующем виде:

- Для $R_3^{(1,1,0)}$:

$$\left\{\nu_1 = \frac{3s_2\left(s_2+2\right)}{\left(2s_2+1\right)^2}, \quad \nu_2 = \frac{27s_2^2}{\left(2s_2+1\right)^3}\right\};$$

- для $R_3^{(2,1,0)}$:

$$\left\{\nu_1 = \frac{3s_2\left(4s_2+5\right)}{\left(5s_2+1\right)^2}, \quad \nu_2 = \frac{108s_2^2}{\left(5s_2+1\right)^3}\right\};$$

- для $R_3^{(3,1,0)}$:

$$\left\{ \nu_1 = \frac{3s_2\,(9s_2+10)}{(10s_2+1)^2}, \quad \nu_2 = \frac{243s_2^2}{(10s_2+1)^3} \right\}.$$

Данные параметризации позволяют изобразить указанные многообразия на плоскости переменных (ν_1, ν_2).

5.4 Вычисление условия трёхчастотного резонанса

В случае трёхчастотного резонанса его кратность может быть равна 1 или 2. Если кратность $\mathfrak{k} = 2$, то это означает, что имеет место попарная соизмеримость между базисными частотами λ_j характеристического многочлена. Такая ситуация исследуется с помощью условий существования двухчастотного резонанса пункта 5.3. Далее рассматриваем только случай кратности 1.

Например, в случае базисных частот $\lambda_1 = -2$, $\lambda_2 = 2+\sqrt{2}$, $\lambda_3 = 2-\sqrt{2}$ собственные значения попарно несоизмеримы, но их взвешенная сумма с резонансным вектором $\mathbf{p}^* = (2,1,1)$ равна нулю. Это означает, что существует трёхчастотный резонанс кратности 1 порядка 4. Поэтому мы рассмотрим случай, когда взвешенная алгебраическая сумма всех трёх базисных собственных значений $\lambda_j (j = 1,2,3)$ равна нулю для некоторого вектора $\mathbf{p}^* \in \mathbb{Z}^3$, т. е. $\langle \mathbf{p}^*, \boldsymbol{\lambda} \rangle = 0$.

Для резонансного вектора $\mathbf{p}^*$ общего вида $\mathbf{p}^* = (r,q,1)$ параметризация соответствующего резонансного многообразия $\mathcal{R}_3^{\mathbf{p}^*}$ была получена выше в п. 5.2. Подставляя $r = q = 1$, получим параметризацию многообразия $\mathcal{R}_3^{(1,1,1)}$ в виде

$$\begin{aligned} a_1 =& -2v\left(3(u+1)^2+1\right), \\ a_2 =& v^2\left(3(u+1)^2+1\right)^2, \\ a_3 =& -4u^2v^3(u+1)^2(u+2)^2. \end{aligned}$$

Здесь нетрудно видеть, что соответствующее неявное представление многообразия имеет вид:

$$R_3^{(1,1,1)} \stackrel{\text{def}}{=} a_1^2 - 4a_2 = 0. \tag{5.13}$$

Непосредственные вычисления показывают, что при $r = q = 1$ многочлен $R_3^{(r,q,1)}$, неявно описанный в подпункте 5.2, есть шестая степень многочлена (5.13).

Подставляя $r = 2$, $q = 1$, получим параметризацию многообразия $\mathcal{R}_3^{(2,1,1)}$ в виде

$$\begin{aligned} a_1 =& -4v\left(3(u+1)^2+2\right), \\ a_2 =& 16v^2\left(3(u+1)^4+1\right), \\ a_3 =& -64u^2v^3(u+1)^2(u+2)^2. \end{aligned}$$

Здесь исключая переменные u, v получаем соответствующее неявное представление многообразия:

$$\begin{aligned} R_3^{(2,1,1)} \stackrel{\text{def}}{=} & 16a_1^6 - 264a_1^4a_2 + 36a_1^3a_3 + 1425a_1^2a_2^2 - \\ & - 630a_1a_2a_3 - 2500a_2^3 + 9261a_3^2 = 0. \end{aligned} \tag{5.14}$$

Непосредственные вычисления показывают, что при $r = 2$, $q = 1$ многочлен $R_3^{(r,q,1)}$ есть полный квадрат многочлена (5.14).

Здесь левые части формул (5.13) и (5.14) суть квазиоднородные многочлены, допускающие с помощью подстановки (5.12) приведение к многочленам от двух переменных ν_1, ν_2. Эти многочлены определяют алгебраические кривые, рациональные параметризации которых имеют следующий вид.

Для $\mathcal{R}_3^{(1,1,1)}$:

$$\left\{ \nu_1 = \frac{3}{4}, \quad \nu_2 = \frac{27u^2\left(u^2-1\right)}{2\left(3u^2+1\right)^3} \right\}. \tag{5.15}$$

Для $\mathcal{R}_3^{(2,1,1)}$:

$$\left\{ \nu_1 = \frac{3\left(768u^4+1\right)}{4\left(24u^2+1\right)^2},\ \nu_2 = \frac{54u^2\left(16u^2-1\right)}{\left(24u^2+1\right)^3} \right\}. \tag{5.16}$$

Как уже было отмечено выше, формулы (5.15) и (5.16) дают параметризацию не всей кривой, а только той её части, на которой многочлен f_3 имеет только вещественные корни.

Пользуясь параметрическим представлением образов резонансных многообразий, можно построить плоские алгебраические кривые на координатной плоскости (ν_1, ν_2).

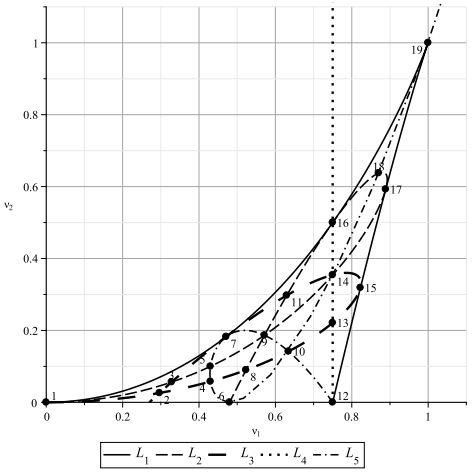

Рис. 1: Резонансные многообразия в переменных ν_1, ν_2.

Для полного представления о взаимном расположении многообразий, соответствующих сильным резонансам, дадим описание рис. 1. Кривые обозначим символами $L_k, k = 1, ..., 5$. Их особые точки, а также точки их взаимного пересечения обозначим символами P_j, где нумерация индексов j точек выбрана в соответствии с увеличением расстояния от начала координат.

Кривая L_1 играет особую роль, она задаёт границу области устойчивости ПР по линейному приближению. Эта кривая является образом дискриминантного множества $\mathcal{D}(f_3)$, которое делит пространство коэффициентов $\mathbb{K}$ многочлена третьей степени на две части. В одной части все корни многочлена вещественные, а в другой части имеется пара комплексно сопряжённых корней и один вещественный корень. Согласно теореме 2, криволинейный треугольник

$P_1P_{19}P_{12}$ является границей области Σ. Остальные резонансные кривые полностью или частично располагаются внутри этой области. Резонансные кривые L_2 и L_3, соответствующие двухчастотному резонансу, полностью располагаются в ней, касаясь кривой L_1. Это связано с тем, что если есть две частоты участвующие в резонансе, то они как корни характеристического уравнения должны быть одной природы – либо одновременно вещественные, либо одновременно комплексные. Но пара комплексно сопряжённых корней не может находиться в резонансе, а третий корень всегда должен быть вещественным. Отметим, что ранее кривые L_1, L_2 и L_3 были изображены в [15], но их параметризации были получены другим способом.

Ещё два резонансных многообразия, образами которых являются кривые L_4 и L_5, соответствуют трёхчастотным резонансам. В них, в отличие от двух частотных резонансов, могут участвовать корни различной природы. Это говорит о том, что трёхчастотные резонансные многообразия будут находиться и в области вещественности корней, и в области, где существуют комплексные корни. Во всех изображённых точках на кривых кратность резонанса ₹ становится равной 2.

Ранее, в [16] было показано, что резонансное многообразие $\mathcal{R}_3^{(q,1,0)}$, соответствующее двухчастотному резонансу, имеет одномерное многообразие особых точек самопересечения. Используя параметризацию (5.5) покажем, что все резонансные многообразия $\mathcal{R}_3^{(r,1,1)}$ при $q \neq 1$ имеют однопараметрическое множество особых точек самопересечения. Для этого подставим в указанную параметризацию значение $q = 1$ и, повторяя выкладки п. 5.3, получим общую параметризацию соответствующей кривой для произвольного значения r:

$$\nu_1 = \frac{3\left(r^6(r^2+8)u^4 - 2r^2(r^2-4)u^2 + 1\right)}{4\left(r^2(r^2+2)u^2+1\right)^2}, \quad \nu_2 = \frac{27r^2u^2\left(r^4u^2-1\right)^2}{2\left(r^2(r^2+2)u^2+1\right)^3}.$$

Эта параметризация после исключения параметра u даёт неявное представление алгебраической кривой в виде

$$\begin{aligned}\tilde{R}_3^{(r,1,1)} = &-108(r^2+1)^4\nu_1^3 + 81(r^4+10r^2+1)\times \\ &\times (r^2+1)^2\nu_1^2 - 18(r^4-9)(r^4-1)(r^2-1)\nu_1\nu_2 + \\ &+ (r^2-1)^3(r^2+3)^3\nu_2^2 - 486r^2(r^4+4r^2+1)\nu_1 + \\ &+ 54(r^4-3r^2-2)(r^2-1)^2\nu_2 + 729r^4 = 0.\end{aligned} \tag{5.17}$$

Особые точки алгебраической кривой – это точки, в которых обнуляются само уравнение кривой и обе её первые частные производные. Полученную полиномиальную систему решаем относительно переменных ν_1, ν_2, рассматривая r в качестве параметра:

$$\nu_1^* = \frac{6r^4+3r^2+3}{(r^2+3)\left(r^2+1\right)^2}, \quad \nu_2^* = \frac{27\left(r^2-1\right)^2}{(r^2+1)\left(r^2+3\right)^3}.$$

Остаётся показать, что найденная особая точка с координатами (ν_1^*, ν_2^*) есть точка самопересечения. Для этого достаточно разложить многочлен (5.17) в этой точке до квадратичных членов включительно и убедиться, что полученная квадратичная форма раскладывается на два линейных множителя.

6. Пример

Рассмотрим три математических маятника одинаковой длины l, точки подвеса которых расположены на равных расстояниях d на горизонтальной прямой, а массы маятников выбраны

равными $\mathbf{m} = (1, \alpha, 1)$ соответственно. Пусть маятники соединены между собой невесомыми линейно упругими пружинами жёсткости k длиной d в недеформированном состоянии. Точки прикрепления пружин расположены на расстоянии $b \leqslant d$ от точек подвеса маятников (см. рис. 2).

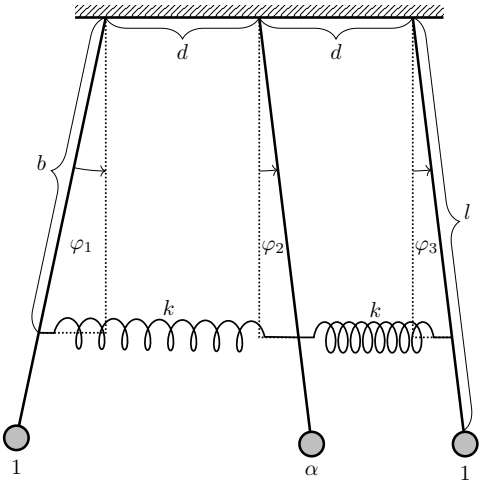

Рис. 2: Три плоских математических маятника, соединённые невесомыми пружинами.

Выбрав в качестве обобщённых координат $\boldsymbol{\varphi}$ углы φ_i, $i = 1, 2, 3$, отклонения маятников от вертикали, запишем функцию Лагранжа этой системы

$$L(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) = \frac{1}{2} \langle T\dot{\boldsymbol{\varphi}}, \dot{\boldsymbol{\varphi}} \rangle + \Pi. \tag{6.1}$$

Матрица T диагональна: $T = \operatorname{diag}(l^2, \alpha l^2, l^2)$. Потенциальная энергия Π есть сумма потенциальной энергии упругой деформации пружин Π_1 и энергии Π_2 математических маятников в однородном поле силы тяжести:

$$\Pi_1 = \frac{k}{2} \sum_{j=1}^{2} \left\{ \sqrt{\left(\Delta_j^{(1)}\right)^2 + \left(\Delta_j^{(2)}\right)^2} - d \right\}^2,$$

$$\Pi_2 = -gl \sum_{j=1}^{3} \cos\varphi_j - gl(\alpha - 1)\cos\varphi_2,$$

где $\Delta_j^{(1)} = b\,(\cos\varphi_{j+1} - \cos\varphi_j)$ и $\Delta_j^{(2)} = b(\sin\varphi_{j+1} - \sin\varphi_j) + d$ суть проекции величин деформаций пружин между j-м и $j+1$-м маятниками на оси абсцисс и ординат соответственно. Следуя [17], введём безразмерные переменные $\tau = \sqrt{g/l}\,t, \quad \beta = b^2k/(gl)$.

Перейдём к гамильтоновой форме уравнений движения, но при этом будем использовать новые канонические переменные, определяемые нормальными модами колебаний. Для этого раскладываем функцию Лагранжа (6.1) в ряд Тейлора в окрестности положения равновесия $\boldsymbol{\varphi} = 0$, получим линейную систему уравнений Лагранжа с СЧ

$$\boldsymbol{\lambda} = \left(1, \sqrt{1+\beta}, \sqrt{\beta\alpha + 2\beta + 1}\right).$$

Этим частотам соответствуют амплитудные векторы

$$\mathbf{u}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{u}_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{u}_3 = \begin{pmatrix} 1 \\ -2/\alpha \\ 1 \end{pmatrix},$$

которые задают матрицу перехода $U = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ к новым переменным $\mathbf{Q}$: $\boldsymbol{\varphi} = U\mathbf{Q}$.
Переход к канонически сопряжённым импульсам $\mathbf{P}$ осуществляется с помощью матрицы $\mathbf{p} = (U^*)^{(-1)}$:

$$\mathbf{p} = (A^*)^{(-1)}\,\mathbf{P}, \text{ где } \mathbf{p} = gl\,\langle \mathbf{m}, \dot{\boldsymbol{\varphi}} \rangle\,.$$

Здесь звёздочка означает транспонирование.
Тогда в новых переменных разложение функции Гамильтона вблизи начала координат имеет вид

$$H(\mathbf{Q}, \mathbf{P}) = H_2(\mathbf{Q}, \mathbf{P}) + H_4(\mathbf{Q}, \mathbf{P}) + \cdots\,, \tag{6.2}$$

где первые два члена разложения следующие:

$$\begin{aligned} H_2 =& \frac{2P_1^2}{2+\alpha} + P_2^2 + \frac{\alpha P_3^2}{2+\alpha} + \left(1+\frac{\alpha}{2}\right) Q_1^2 + \\ &+ (\beta+1)\,Q_2^2 + \frac{(2+\alpha)\,(\beta\alpha+\alpha+2\beta)\,Q_3^2}{\alpha^2}, \\ H_4 =& -\frac{(2+\alpha)\,Q_1^4}{24} - \frac{(2\beta+1)\,Q_1^2Q_2^2}{2} - \\ &- \frac{(2+\alpha)\,(2\beta\alpha+\alpha+4\beta)\,Q_1^2Q_3^2}{2\alpha^2} - \\ &- \frac{(3\beta\alpha+\alpha+2\beta)\,Q_1Q_2^2Q_3}{\alpha} - \\ &- \frac{\left(\alpha^2-4\right)(3\beta\alpha+\alpha+6\beta)\,Q_1Q_3^3}{3\alpha^3} - \\ &- \frac{(4\beta+1)\,Q_2^4}{12} - \frac{(4\beta\alpha+\alpha+4\beta)\,Q_2^2Q_3^2}{2\alpha} - \\ &- \frac{(2+\alpha)\left(\alpha^2-2\alpha+4\right)(4\beta\alpha+\alpha+8\beta)\,Q_3^4}{12\alpha^4}. \end{aligned} \tag{6.3}$$

Все дальнейшие вычисления выполнялись в СКА `Maple` согласно схемы раздела 5.
Поскольку в (6.2) $H_3(\mathbf{Q}, \mathbf{P}) \equiv 0$, то резонансы порядка $\mathfrak{q} = 3$ проявят себя при приведении к НФ только в формах степени 6 и выше. Следовательно, в данном случае можно ограничиться исследованием резонансов порядка $\mathfrak{q} = 4$, а именно:

- в случае двухчастотного резонанса исследуем многообразие $\mathcal{R}_3^{(3,1,0)}$;
- в случае трёхчастотного резонанса исследуем многообразие $\mathcal{R}_3^{(2,1,1)}$.

Обозначим через $A(H_2)$ матрицу квадратичной формы (6.3), тогда полухарактеристический

многочлен матрицы $JA(H_2)$, соответствующей линейной системы Гамильтона, есть

$$f(\mu) = \mu^3 + \frac{(2\beta\alpha + 3\alpha + 2\beta)\,\mu^2}{\alpha} + \\ + \frac{\left(\beta^2\alpha + 4\beta\alpha + 2\beta^2 + 3\alpha + 4\beta\right)\mu}{\alpha} + \\ + \frac{(\beta\alpha + \alpha + 2\beta)\,(\beta + 1)}{\alpha}. \tag{6.4}$$

Таким образом, пространство параметров $\Pi \equiv (\alpha, \beta)$ задачи двумерно и значения параметров должны быть неотрицательны: $\{\alpha, \beta \geqslant 0\}$. Квадратичная форма (6.3) в указанной области пространства параметров является знакоопределённой, следовательно, нижнее равновесное положение маятников является устойчивым по Ляпунову. Поэтому далее будет приведено описание резонансных множеств 4-го порядка.

Подставляем коэффициенты a_1, a_2, a_3 многочлена (6.4) в уравнение (5.10) резонансного многообразия $\mathcal{R}_3(3, 1, 0)$, раскладываем его на множители и отбираем среди них только те, нули которых располагаются в первом квадранте плоскости (α, β). Аналогично поступаем и с уравнением (5.14) резонансного многообразия $\mathcal{R}_3(2, 1, 1)$. В результате получается пять резонансных кривых: три соответствуют двухчастотному резонансу, а два — трёхчастотному:

$$\mathcal{R}_a^{(3,1,0)} \stackrel{\text{def}}{=} \beta = \frac{8\alpha}{2 + \alpha}, \tag{6.5}$$

$$\mathcal{R}_b^{(3,1,0)} \stackrel{\text{def}}{=} \beta = 8, \tag{6.6}$$

$$\mathcal{R}_c^{(3,1,0)} \stackrel{\text{def}}{=} \beta = \frac{4\alpha}{1 - 4\alpha}, \tag{6.7}$$

$$\mathcal{R}_a^{(2,1,1,)} \stackrel{\text{def}}{=} \beta = 4\alpha^2 + 4\alpha, \tag{6.8}$$

$$\mathcal{R}_b^{(2,1,1)} \stackrel{\text{def}}{=} \beta = \frac{8\alpha\,(2 - \alpha)}{(3\alpha - 2)^2}. \tag{6.9}$$

Эти кривые показаны на рис. 3, на котором, двухчастотные резонансные кривые изображены сплошными и штриховыми линиями, а трёхчастотные — штрих-пунктирными.

Кривые обозначим символами $L_k^i, k = 3, 5, i = a, b, c$, где нижний индекс соответствует индексу кривых L_k, описанных в подпункте 5.4. Их точки взаимного пересечения обозначим символами P_j, где нумерация индексов j точек выбрана в соответствии с увеличением расстояния от начала координат. Дадим краткое описание структуры СЧ на соответствующих резонансных кривых.

1. На кривой L_3^a, которая изображена длинной штриховой линией, с уравнением (6.5) СЧ следующие: $\lambda_{1,4} = \pm\mathrm{i}$, $\lambda_{2,5} = \pm\mathrm{i}\sqrt{(9\alpha + 2)/(\alpha + 2)}$, $\lambda_{3,6} = \pm 3\mathrm{i}$. Таким образом, для всех значений параметра α имеет место двухчастотный резонанс, кроме значения $\alpha = 6/5$, соответствующего точке P_1 (см. ниже).
2. На прямой L_3^b показанной в виде сплошной линии с уравнением (6.6) СЧ следующие: $\lambda_{1,4} = \pm\mathrm{i}$, $\lambda_{2,5} = \pm 3\mathrm{i}$, $\lambda_{3,6} = \pm\mathrm{i}\sqrt{(9\alpha + 16)/\alpha}$. Эта прямая пересекается с другими кривыми в трёх точках: P_2, P_3, P_4.
3. На кривой L_3^c, которая изображена короткой штриховой линией, с уравнением (6.7) СЧ следующие:

$$\lambda_{1,4} = \pm\mathrm{i}, \lambda_{2,5} = \pm\frac{\mathrm{i}}{\sqrt{1 - 4\alpha}}, \lambda_{3,6} = \pm\frac{3\mathrm{i}}{\sqrt{1 - 4\alpha}}.$$

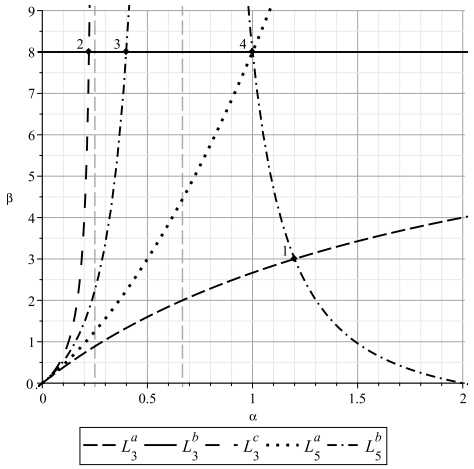

Рис. 3: Резонансные многообразия примера в переменных α, β.

4. На кривой L_5^a изображённой в виде пунктирной линии с уравнением (6.8) СЧ следующие:

$$\lambda_{1,4} = \pm\mathrm{i}, \lambda_{2,5} = \pm\mathrm{i}(2\alpha+1), \lambda_{3,6} = \pm\mathrm{i}(2\alpha+3).$$

На этой кривой для всех значений α имеет место трёхчастотный резонанс, кроме значения $\alpha = 1$, соответствующего точке P_4.

5. На кривой L_5^b изображённой в виде штрих-пунктирной линии с уравнением (6.9) СЧ следующие:

при $0 \leqslant \alpha < 2/3$

$$\lambda_{1,4} = \pm\mathrm{i}, \quad \lambda_{2,5} = \pm\mathrm{i}\frac{\alpha+2}{2-3\alpha}, \qquad \lambda_{3,6} = \pm\mathrm{i}\frac{6-\alpha}{2-3\alpha};$$

при $\alpha > 2/3$

$$\lambda_{1,4} = \pm\mathrm{i}, \quad \lambda_{2,5} = \pm\mathrm{i}\frac{\alpha+2}{3\alpha-2}, \qquad \lambda_{3,6} = \pm\mathrm{i}\frac{6-\alpha}{3\alpha-2}.$$

На этой кривой для всех значений α имеет место трёхчастотный резонанс, кроме точек P_2, P_3,P_4.

Кривая L_3^a пересекается только с одной ветвью кривой L_5^b в точке $P_1 = (6/5, 3)$. Подставив в выражения (6.4) значения переменных α, β, являющихся координатами точки P_1, получим

$$f_3 = (\mu+1)(\mu+4)(\mu+9).$$

Корнями этого полухарактеристического многочлена являются значения $\mu_1 = -1$, $\mu_2 = -4$, $\mu_3 = -9$. А в корнях характеристического уравнения они запишутся в виде

$$\lambda_{1,4} = \pm\mathrm{i}, \quad \lambda_{2,5} = \pm 2\mathrm{i}, \quad \lambda_{3,6} = \pm 3\mathrm{i}.$$

Зная значения корней, проверим выполнение резонансных соотношений. Так как точка P_1 принадлежит двум резонансным многообразиям, соответствующим двухчастотному и трёхчастотному резонансу, проверим эти условия для каждого из них:

1. для $\mathcal{R}_a^{(3,1,0)}$ имеются 2 пары соизмеримых корней с отношением $3:1$, т. е. $3\lambda_1 + \lambda_6 = 3\lambda_4 + \lambda_3 = 0$.

2. для $R_b^{b(2,1,1)}$ также должно выполняться соотношение 2 : 1 : 1, которое выглядит как: $\lambda_1 + 2\lambda_5 + \lambda_3 = \lambda_4 + 2\lambda_2 + \lambda_6 = 0$.

Кривая L_3^b пересекается с кривой L_3^c в точке P_2 = (2/9, 8), с кривой L_5^b в точке P_3 = (2/5, 8) и P_4 = (1, 8). В точке P_4 она также пересекается с кривой L_5^a . Подставив в выражения (6.4) значения переменных α, β, являющихся координатами этих точек, получим соответствующие выражения для f_3:

$P_2 : f_3 = (\mu + 1)(\mu + 9)(\mu + 81)$;
$P_3 : f_3 = (\mu + 1)(\mu + 9)(\mu + 49)$;
$P_4 : f_3 = (\mu + 1)(\mu + 9)(\mu + 25)$.

Аналогично, как и в вышеуказанном случае, здесь можно убедиться в выполнении резонансных соотношений.

7. Заключение

Для системы Гамильтона с тремя степенями свободы предложен способ символьного вычисления резонансного многообразия, соответствующего резонансу общего вида, а также доказано, что это многообразие допускает полиномиальную параметризацию. Предлагается схема исследования областей формальной устойчивости положения равновесия системы Гамильтона с тремя степенями свободы.

Список литературы / References

[1]. Moser J.K. New aspects in the theory of stability of hamiltonian systems. В Comm. PureAppl. Math. Том 11, № 1, страницы 81—114, 1958.
[2]. Батхин А.Б., Хайдаров З.Х. Сильные резонансы в нелинейной системе Гамильтона. ПрепринтыИПМ им. М.В. Келдыша, (59): 1—28, 2022. DOI: https://doi.org/10.20948/prepr-2022-59.
[3]. Зигель К., Мозер Ю. Лекции по небесной механике. НИЦ «Регулярная и хаотическая динамика»: 384, 2001.
[4]. Батхин А.Б., Брюно А.Д., Варин В.П. Множества устойчивости многопараметрических гамильтоновых систем. Прикладная математика и механика, том 76, №1,страницы 80—133, 2012.
[5]. Калинина Е. А., Утешев А.Ю. Теория исключения: Учеб. пособие. НИИ химии СПбГУ, СПб, 2002.
[6]. Basu S., Rollack R., Roy M.F. Algorithms in real algebraic geometry. Algorithms andComputations in Mathematics 10, 2006.
[7]. Брюно А.Д. Аналитическая форма дифференциальных уравнений (II). Встраницы 199—239, 1972.
[8]. Брюно А.Д. Ограниченная задача трех тел: Плоские периодические орбиты. Наука,1990.
[9]. Биркгоф Дж.Д. Динамические системы. Изд. дом «Удмуртский университет», Ижевск, 1999.
[10]. Bruno A.D. Survey of eight modern methods of Hamiltonian mechanics. В том 10, № 4, 2021. DOI: https://doi.org/10.3390/axioms10040293.
[11]. Брюно А. Д. О формальной устойчивости систем Гамильтона. Мат. заметки, том 1, № 3, страницы 325—330, 1967.
[12]. Журавлев В.Ф., Петров А.Г., Шундерюк М.М. Избранные задачи гамильтоновой механики. ЛЕНАНД, 2015.
[13]. Батхин А.Б., Хайдаров З.Х. Вычисление условия сильного резонанса в системе Гамильтона. В ЖВМиМФ, том 63, № 5, страницы 697—714, 2023. DOI: 10.31857/S0044466923050071.
[14]. Брюно А.Д., Азимов А.А. Вычисление унимодулярных матриц степенных преобразований. Программирование, том 49, № 1, страницы 38—47, 2023. DOI: 10.31857/S013234742301003X.

[15]. Батхин А. Б. Резонансное множество многочлена и проблема формальной устойчивости. Вестник ВолГУ. Серия 1. Математика. Физика, том 35, №4, страницы 5—23, 2016. DOI: https://doi.org/10.15688/jvolsu1.2016.4.1.

[16]. Батхин А.Б. Параметризация множества, определяемого обобщенным дискриминантом многочлена. Программирование, страницы 5—17, 2018.

[17]. Маркеев А.П. О движении связанных маятников. Нелинейная динамика, том 9, № 1, страницы 27—38, 2013.

Информация об авторах / Information about authors

Александр Борисович БАТХИН — доктор физико-математических наук, доцент, старший научный сотрудник Института прикладной математики им. М.В. Келдыша РАН. Сфера научных интересов: гамильтонова и небесная механика, разрешение особенностей алгебраических и дифференциальных уравнений, полиномиальная компьютерная алгебра, численно-аналитические методы решения обыкновенных дифференциальных уравнений.

Alexander Borisovich BATKHIN — Doctor of Physics and Mathematics, Associated Professor, senior researcher of the Keldysh Institute of Applied Mathematics of RAS. Research interests: Hamiltonian and Celestial Mechanics, resolution of singularities of algebraic and differential equations, polynomial computer algebra, numerical and analytical solutions to ordinary differential equations.

Зафар Хайдар угли ХАЙДАРОВ – ассистент преподаватель кафедры Алгебры и геометрии Математического факультета Самаркандского государственного университета имени Ш.Рашидова. Сфера научных интересов: степенная геометрия, базисы Грёбнера, алгебраические структуры, системы Гамильтона.

Zafar Khaydar ugli KHAYDAROV — assistant teacher of the Department of Algebra and Geometry, Faculty of Mathematics, Samarkand State University named after Sh.Rashidov. Research interests: power geometry, Gröbner basis, algebraic structures, Hamiltonian systems.