ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 32 Выпуск 4

ISSN Online 2220-6426
Volume 32 Issue 4

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2020

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:

Google Scholar, ULRICHSWEB Global Serials Directory, Russian Science Citation Index, OCLC WorldCat, BASE, OpenDOAR, CYBERLENINKA, DOAJ Directory of Open Access Journals, eLIBRARY.RU, ROAR Registry of Open Access Repositories, Crossref, Math-Net.Ru

УДК004.45

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

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: proceedings@ispras.ru
Web: https://ispranproceedings.elpub.ru/

Содержание

Table of Contents

DOI: 10.15514/ISPRAS–2020–32(4)–1

Использование Big Data в международном бизнесе

К.А. Алексеев, ORCID: 0000–0002–8350–2158 <alxkonstantin@gmail.com>
EPAM Systems,
Польша, 40–101, Катовице, Хожовска, 148

Аннотация. Термин Big Data («Большие данные») вызывает много споров у специалистов, многие из которых полагают, что он означает только объемы накопленных данных, но не стоит забывать и о технической стороне: рассматриваемое направление включает в себя технологии вычисления, хранения, а также сервисы услуг. Big Data – термин, который обозначает технологии обработки неструктурированных и структурированных данных большого объема для получения понятных и полезных человеку итогов. В бизнесе Big Data используют для поддержки принятия решений руководителем (к примеру, на основании анализа финансовых показателей из учетной системы) или маркетологом (к примеру, на основании анализа предпочтений клиентов из социальных сетей). Сами по себе алгоритмы Больших данных появились при внедрении первых мэйнфреймов (высокопроизводительных серверов), которые обладают необходимыми ресурсами для оперативной обработки данных и пригодны для компьютерных вычислений и для последующего анализа данных. Поскольку число встраиваемых компьютеров увеличивается благодаря уменьшению цен на процессоры и повсеместному распространению Интернета, также увеличиваются и объемы передаваемых данных с последующей их обработкой (зачастую в режиме реального времени). Поэтому можно предположить, что в ближайшие годы будет повышаться значимость облачных вычислений и Интернета вещей. Следует отметить, что технология обработки Big Data сводится к трем основным направлениям, которые решают три типа задач, а именно, (1) перевод и хранение поступаемой информации в гигабайтах, терабайтах, петабайтах и т.д. данных для их обработки, хранения и применения на практике; (2) структурирование разрозненного контента: фотографий, текстов, аудио, видео и всех других видов данных; (3) анализ Больших данных и внедрение разных методов обработки неструктурированных данных, создание разных аналитических отчетов. В сущности, применение Больших данных подразумевает все направления работы с большими объемами самых разрозненных данных, постоянно обновляемых и разбросанных по различным источникам. Цель достаточно проста – наибольшая эффективность работы, внедрение новых продуктов и повышение конкурентоспособности. В данной статье рассматриваются особенности решения проблем использования Big Data в международном бизнесе.

Ключевые слова: большие данные; международный бизнес; эволюционирование; облачные технологии; мейнфрэйм; стартап

Для цитирования: Алексеев К.А. Использование Big Data в международном бизнесе. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 7–20. DOI: 10.15514/ISPRAS–2020–32(4)–1

Using Big Data in International Business

K.A. Alekseev, ORCID: 0000–0002–8350–2158 <alxkonstantin@gmail.com>
EPAM Systems,
Chorzowska 148, 40–101 Katowice, Poland

Abstract. The term BigData causes a lot of controversy among specialists, many of whom note that it only means the volumes of accumulated data, but do not forget about the technical side, the considered direction includes technologies for computing, storage, and also service services. Big Data is a term that denotes technologies for processing large unstructured and structured data to obtain results that are understandable and useful to humans. In business, Big Data is used to support the adoption of transformations by a manager (for example, based on an analysis of financial indicators from an accounting system) or a marketer (for example, based on an analysis of customer preferences from social networks). By themselves, Big Data algorithms appeared with the introduction of the first mainframes (high–performance servers), which have the necessary resources for operational data processing and are suitable for computer calculations and subsequent data analysis. As the number of embedded computers rises due to falling processor prices and the ubiquity of the Internet, so too is the amount of data transferred and then processed (often in real time). Therefore, we can assume that the importance of cloud computing and the Internet of Things will increase in the coming years. It should be noted that Big Data processing technology boils down to three main areas that solve three types of tasks: (1) translation and storage of incoming information in gigabytes, terabytes, petabyte, etc. for their processing, storage and application in practice; (2) structuring of disparate content, namely: photos, texts, audio, video and all other types of data; (3) analysis of Big Data and the implementation of different methods of processing unstructured data, the creation of various analytical reports. In essence, the application of Big Data implies all areas of work with large volumes of the most disparate data, constantly updated and scattered across various sources. The goal is quite simple – the most efficient work, the introduction of new products and increased competitiveness. In this article, we will consider the features of solving the problems of using Big Data in international business.

Keywords: big data; international business; evolution; cloud technology; mainframe; startup

For citation: Alekseev K.A. Using Big Data in International Business. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 7–20 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–1

1. Введение

Термин Big Data был впервые введен в 2008 году на страницах журнала Nature в статье главного редактора К. Линча. Этот номер издания посвящался взрывному росту глобальных объемов данных и их значимости в научной среде. К. Линч предложил для новой парадигмы название «Большие данные», которое он выбрал по аналогии с такими метафорами, как «большая руда», «большая нефть» и т.п., которые отражают не столько объемы чего–либо, сколько переход количества в качество [1].

В настоящее время мировое сообщество снова заговорило о Big Data. Причины данного явления заключаются в увеличении объемов информации и отсутствии в ней какой–то структуры. Исследователей волнуют вопросы качественной интерпретации информации, содержащейся в данных, разработки средств для работы с ними и эволюционирование технологий хранения.

Эффективные преобразования при работе с Big Data для разнообразных направлений деятельности осуществляются благодаря большому числу существующих на сегодняшний день комбинаций аппаратного и программного обеспечения.

Важное достоинство Больших данных заключается в возможности применения новых инструментов вместе с теми, которые уже используются в данной области. Это имеет в особенности важное значение в случае кросс–дисциплинарных проектов. Примером может служить поддержка потребителей и мультиканальные продажи.

Для работы с Большими данными важна определенная последовательность действий, приведенная ниже:

- производится сбор данных;
- после этого происходит структурирование информации; для этого используются дашборды (Dashboards) – средства структурирования;
- на следующей стадии создаются контексты и инсайты, на основе которых формируются предложения для принятия преобразований.

Вследствие высоких расходов на сбор данных, основная задача заключается в определении цели использования полученных данных [2].

Приведем пример. Рекламные агентства могут использовать агрегированные у телекоммуникационных фирм информационные данные о местоположении. Данный подход должен обеспечить таргетированную рекламу. Эту же информацию применяют и в других областях, которые непосредственно связаны с продажей и оказанием товаров и услуг [3]. Таким образом, полученные сведения могут оказаться ключевыми в принятии преобразования об открытии магазина в конкретной местности.

Целью настоящей работы является исследование особенностей использования Big Data в международном бизнесе. В статье рассматриваются основные понятия и функции Big Data, методы использования их в международном бизнесе; анализируется зарубежный опыт применения Big Data в международном бизнесе: на его основе этого анализа делаются выводы о проблемах и путях совершенствования механизма применения Big Data.

2. Общие аспекты использования Big Data в международном бизнесе

Если рассмотреть особенности использования outdoor–щитов в Лондоне, не приходится сомневаться, что в настоящее время такой опыт является возможным только тогда, когда возле каждого щита располагается специальный измерительные прибор. В то же время мобильные операторы всегда знают основные данные о своих абонентах: их семейное положение, расположение и т.д. [4]. Вполне возможно, что в скором будущем реклама на любом щите будет подстраиваться под каждого конкретного человека.

Еще одна потенциальная область применения Больших данных заключается в сборе информации о числе посетителей разных мероприятий. Рассмотрим еще один пример. Организаторы футбольных матчей не могут знать точное количество заранее пришедших на матч. Однако они получили бы такие данные, если бы воспользовались сведениями от операторов мобильной связи: где находятся потенциальные посетители за определенный период времени — неделю, месяц, день — до матча. Следовательно, у организаторов была бы возможность планирования локации мероприятия в зависимости от предпочтений целевой аудитории [5].

Большие данные также дают значительные преимущества для банковского сектора, который может использовать обработанные данные для того, чтобы выявить недобросовестных держателей карт. Приведем следующий пример. При заявлении картодержателя о краже или утере карты у банка есть возможность отслеживания местоположения карты, по которой был произведен расчет, и мобильного телефона картодержателя, чтобы удостовериться в правдивости данных. У представителя банковской организации есть возможность увидеть, что мобильный телефон и платежная карта держателя находятся в одной зоне. Следовательно, картой пользуется ее владелец [6].

Благодаря таким преимуществам использование информации предоставляет фирмам множество новых возможностей, а предложения Больших данных продолжают свое эволюционирование [7].

Главная сложность внедрения Больших данных заключается в сложности расчета кейса. Этот процесс осложняет наличие большого числа неизвестных переменных.

Сложно делать какие–то прогнозы на будущее, в то время как информация о прошлом не всегда находится в зоне доступа. В такой ситуации самое главное — планирование своих изначальных действий.

Определение конкретного вопроса, в преобразовании которого будет применяться технология обработки Больших данных, поможет определиться с концепцией и задаст вектор будущих действий. Сделав акцент на сборе сведений именно по отмеченному вопросу, также стоит использовать все доступные инструменты и методы для получения более ясных сведений. Более того, этот подход в значительной степени упростит процесс принятия преобразования в будущем.

Возможность того, что проект Больших данных будет реализован командой без определенных опыта и навыков — достаточно невелика. Знания, которые следует использовать в таком сложном исследовании, обычно приобретают посредством упорного труда, следовательно, предыдущий опыт является важным в рассматриваемой области. Сложно переоценить воздействие культуры использования информации, которая была получена посредством таких исследований. Они предоставляют разные возможности, в том числе и злоупотребления полученными сведениями. Чтобы использовать сведения во благо, необходимо придерживаться простых правил корректной обработки данных.

Инсайты являются основной ценностью технологий. Рынок все еще испытывает значительную нехватку специалистов, которые имеют понимание законов ведения бизнеса, важнось информации и сферы ее применения. Нельзя не учитывать то, что анализ данных является ключевым способом эволюционирования бизнеса и достижения поставленных целей, необходимо стремиться к выработке конкретных моделей восприятия и поведения [8]. В данном случае Big Data могут принести пользу и сыграть позитивную роль в преобразовании вопросов ведения дел.

3. Анализ зарубежного опыта использования Big Data в международном бизнесе

Поистине неисчерпаемые и беспрецедентные возможности открываются для субъектов предпринимательства в плане принятия более выверенных и обоснованных преобразований, повышения и оптимизации эффективности деятельности, а также создания новых типов сервисов и продуктов, опираясь на учитывающий все аспекты, комплексный анализ проблем и ситуаций, планирование их тенденций и динамики, установление причинно-следственных связей с не принимавшимися ранее в расчет факторами и выявление воздействия обстоятельств, считавшимися незначимыми [9].

Как уже отмечалось выше, схожее видение и подходы к обращению с Big Data появились совсем недавно. Но соответствующие преобразования и технологии активно продвигаются фирмами–разработчикам и ПО, для которых они формируют новый емкий сегмент рынка. Например, IBM уже инвестировала в разработки в данном направлении 12 млрд. долл., открыв по всему миру 6 центров анализа данных с общей численностью занятых 4 тыс. человек. Что же касается бизнеса, который является адресатом итогом таких НИОКР, то, по сведениям SAS, 26% фирм уже пользуются Big Data на системной основе и отмечают в связи с этим повышение эффективности в течение прошедших 3 лет, в то время как 41% – ожидает повышения в ближайшие 3 года; в соответствии же с IBM, 28% субъектов запустили пилотные проекты по оценке метаданных, а 47% – имеют намерение внедрения соответствующих технологий [10].

Перспективность и важность Больших данных осознается и государством как центральным институтом экономической координации. Стратегии в отношении метаданных, или Big Data, уже приняты и реализуются в США, Республике Корея, Сингапуре и Великобритании, которые являются более крупными участниками мирового информационно–коммуникационного рынка [11].

К примеру, Республика Корея первая в мире в октябре 2011 г. выдвинула стратегию в сфере Больших данных в рамках курса на формирование электронного правительства, задействуя их для обеспечения прозрачности государства, укрепления и эволюционирования конкурентоспособности экономики [12]. При этом сама стратегия представляет собой самую

системную по сравнению с программами остальных государств, охватывая весь комплекс аспектов по внедрению Больших данных в практику бизнеса.

Основные инструменты достижения поставленных задач в отношении продвижения Больших данных предусматривают собой следующие меры содействия:

- применению Больших данных в сферах телерадиовещания, инфокоммуникационных технологиях, здравоохранения, образования, транспорта;
- подготовке кадров;
- последующему совершенствованию платформ и технологий Больших данных;
- организации центра поддержки пилотных проектов и НИОКР в сфере Больших данных;
- формированию благоприятной среды для распространения технологий Больших данных;
- защите личных данных и снижения злоупотреблений ей;
- структурированию нормативно–правовых основ для ведения бизнеса на основании Больших данных [13].

Вместе с тем, также осуществляются НИОКР в сфере разработки базовых и вспомогательных технологий анализа данных с использованием суперкомпьютеров. Схожие задачи были поставлены и США, которые приступили к реализации инициативы в сфере Больших данных в марте 2012 г. Программа концентрирует свои усилия на разработке необходимых при хранении, сборе, распределении, управлении и анализе информации ноу–хау, активизации использования основных технологий обработки данных для ускорения инновационных процессов в науке (здравоохранении, биотехнологии, фундаментальных исследованиях) и инженерии (добыча полезных ископаемых, энергетика), а также в подготовке кадров по рассматриваемому профилю. При этом для хозяйствующих субъектов и населения публикуется 227 тыс. массивов статистических данных по сферам ИКТ, транспорта, общественной безопасности, фармацевтики, занятости, правонарушений [14].

В настоящее время технологии Big Data зачастую внедряются в фирмах США, но уже сейчас и другие государства стали проявлять интерес. В 2014 году, по сведениям IDC, на государства Ближнего Востока, Европы, Азии (за исключением Японии) и Африки пришлось 45% рынка услуг, ПО и оборудования в области Больших данных.

В соответствии с опросом CIO, фирмы государств Азиатско–Тихоокеанского региона быстрыми темпами осваивают новые преобразования в сфере анализа Big Data, облачных технологий и безопасного хранения. Латинская Америка находится на втором месте по числу инвестиций в эволюционирование технологий Big Data, опережая государства США и Европы [15].

Рассмотрим прогнозы эволюционирования рынка Big Data отдельных государств.

Объемы данных Китая составляет 909 эксабайт, что равняется 10% общих объемов информации в мире, к 2020 году объемы информации должны достигнуть 8060 эксабайт, повысится и доля информационных данных в общемировой статистике, через 5 лет она будет равняться 18%. Потенциальный рост Больших данных КНР имеет одну из наиболее быстрорастущих динамик.

Бразилия по результатам 2016 года накопила информации на 212 эксабайт, что составляет 3% от общемировых объемов. К 2020 году объемы информации должны вырасти до 1600 эксабайт, что составит 4% общемировой информации.

По сведениям EMC, объем накопленных данных Индии по результатам 2016 года составляет 326 эксабайт, что составляет 5% от общих объемов информации. К 2020 году объем информации возрастет до 2800 эксабайт, что составит 6% общемировой информации.

Объем накопленных Big Data Японии по результатам 2016 года составляет 495 эксабайт, что составляет 8% от общих объемов информации. К 2020 году объем информации возрастет до 2200 эксабайт, но снизится доля рынка Японии и составит 5% от общих объемов

общемировой информации. Следовательно, объем рынка Японии снизится на более, чем 30%.

По сведениям EMC, объем накопленных данных в Германии по результатам 2016 года составляет 230 эксабайт, что составляет 4% от общего объема общемировой информации. К 2020 году объем информации увеличится до 1100 эксабайт и составит 2%.

На рынке Германии большую долю выручки, по прогнозам Experton Group, будет генерировать сегмент сервисных услуг, доля которых в 2015 году составялет 54%, а в 2019 году повысится до 59%, доли программного оборудования и обеспечения, напротив, снизятся (см. рис. 1).

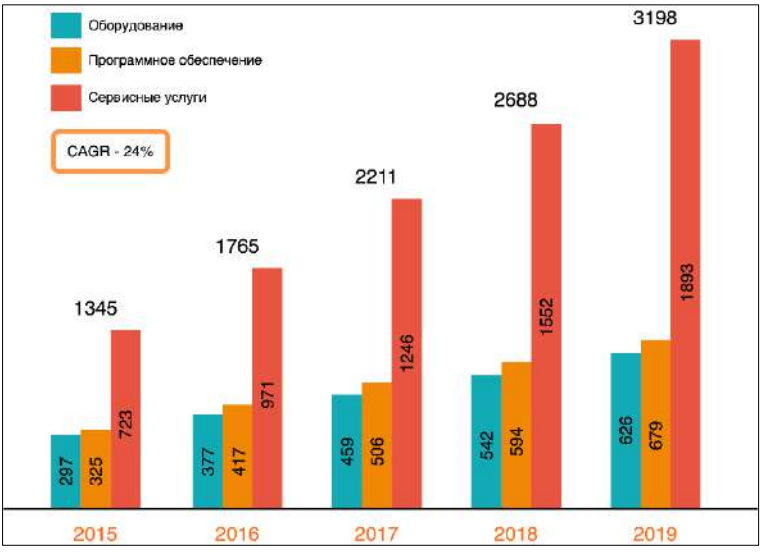


Рис. 1. Объем рынка Больших данных Германии (млн. евро), источник: Experton Group
Fig. 1. The size of the German Big Data market (€ million), source:

В целом, объем рынка возрастет с 1, 345 млрд евро в 2015 году до 3,198 млрд евро в 2019 году, средний темп роста составит 24%.

Следовательно, на основании аналитики EMC и CIO, можно заключить, что развивающиеся государства в ближайшие годы станут рынками активного эволюционирования технологий Big Data.

По сведениям IDG Enterprise, в 2015 расходы фирм на сферу Big Data составили в среднем 7,4 млн долл. США на фирму, крупные фирмы потратили около 13,8 млн долл. США, средние и малые – 1,6 млн долл. США.

Более всего инвестировано в такие сферы, как визуализация и анализ данных и их сбор.

В соответствии с текущими тенденциями и спросом на рынке, инвестиции в 2015 году использованы на улучшение качества данных, совершенствование прогнозирования и планирования, а также на повышение скорости обработки данных.

Фирмами финансового сектора, по сведениям Bain Company's Insights Analysis, были произведены инвестиции, так, в 2015 году потрачено 6,4 млрд долл. США на технологии Больших данных, средний темп роста инвестиций составит 22% до 2020 года. Интернет–фирмы потратили 2,8 млрд долл. США, средний темп роста повышения расходов на Big Data составит 26%.

При проведении опроса Economist Intelligence Unit survey выявлены приоритетные пути эволюционирования Больших данных, распределение ответов респондентов представлено на рис. 2.

Рис. 2. Какие направления эволюционирования Больших данных являются приоритетными для компаний? Источник: Economist Intelligence Unit

Fig, 2. What areas of evolution of Big Data are priority for companies? Source: Economist Intelligence Unit

По прогнозам IDC тенденции эволюционирования рынка Больших данных выглядят так.

- В следующие 5 лет расходы на облачные преобразования в области технологий Big Data будут возрастать в 3 раза быстрее, чем расходы на локальные преобразования. Станут востребованными гибридные платформы для хранения данных.
- Увеличение приложений с использованием прогнозной и сложной аналитики, включая машинное обучение, ускорится, предложение таких приложений будет возрастать на 65% быстрее, чем приложения, которые не используют прогнозную аналитику.
- Медиа аналитика утроится и станет весомым драйвером роста рынка технологий Big Data.
- Ускорятся тенденции внедрения преобразований для анализа постоянного потока информационных данных, которые применяются для интернета вещей.

К 2018 году 50% пользователей будут взаимодействовать с сервисами, которые основываются на когнитивном вычислении.

Эксперты IDC выделили 3 драйвера рынка Big Data:

- массовые поглощения клиентской базы фирм, которые предлагают мобильные приложения и иные платформы;
- эволюционирование облачной инфраструктуры;
- изменения в законодательстве о конфиденциальности данных.

Кроме этого также стоит выделить:

- увеличенный интерес на обработку медиа–материалов, которые ранее относились к неструктурированной информации;
- увеличение популярности обучающих курсов в области Больших данных;
- инвестиции в визуализацию данных и активное storytelling аналитиками данных;
- постоянные инвестиции веб–гигантами в Большие данные, например, Amazon, Google, Facebook и др.

Среди ограничителей рынка Больших данных можно выделить:

- все еще высокую стоимость внедрения технологий Больших данных;
- необходимость обеспечение защиты сведений и их конфиденциальности;
- нехватку квалифицированных кадров;
- недоверие фирм к таким технологиям;

- недостаточный объем накопленных сведений;
- поддержка базы данных требует постоянного финансирования, что создает дополнительный барьер на внедрение Больших данных;
- трудности интеграции с существующими системами;
- ограниченное количество поставщиков данных.

В соответствии с опросом Accenture, вопросы безопасности данных сейчас являются основным барьером на пути внедрения технологий Big Data, более 51% опрошенных подтвердили, что беспокоятся за обеспечение защиты сведений и их конфиденциальности. 47% фирм сообщили, о невозможности внедрения Больших данных в связи с ограниченным бюджетом, 41% фирм в качестве проблемы отметили недостаток квалифицированных кадров (см. рис. 3).

Рис. 3. Основные проблемы при внедрении проектов Big Data, источник: Accenture
Fig. 3. The main problems in the implementation of Big Data projects, source: Accenture

Компания Wikibon спрогнозировала, что объем рынка Больших данных возрастет до 38,4 млрд долл. США и повысится по сравнению с предыдущим годом на 36%. В ближайшие годы можно будет наблюдать снижение темпов роста до 10% по результатам 2017 года. С учетом таких прогнозов, объем рынка в 2020 году может составить 68,7 млрд долл. США (см. рис. 4).

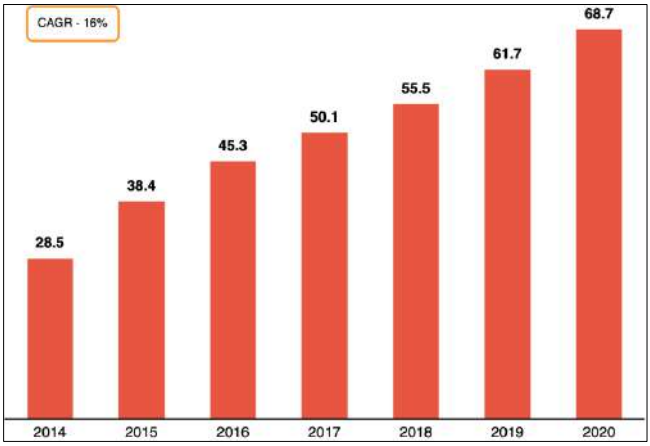


Рис. 4. Объем рынка Больших данных 2014–2020 гг. (млрд. долл. США), источник: Wikibon, IPOboard
Fig.4. The size of the Big Data market in 2014–2020 (USD billion), source: Wikibon, IPOboard

Распределение общемирового рынка Big Data по бизнес–категориям будет выглядеть так (см. рис. 5).

Как видно из представленной гистограммы, большую часть рынка Больших данных будут занимать технологии из сферы улучшения клиентского сервиса. Точечный маркетинг будет на втором месте по приоритетности у фирм вплоть до 2019 года; в 2020 году, по прогнозам Heavy Reading, он уступит место преобразованиям по улучшению операционной эффективности.

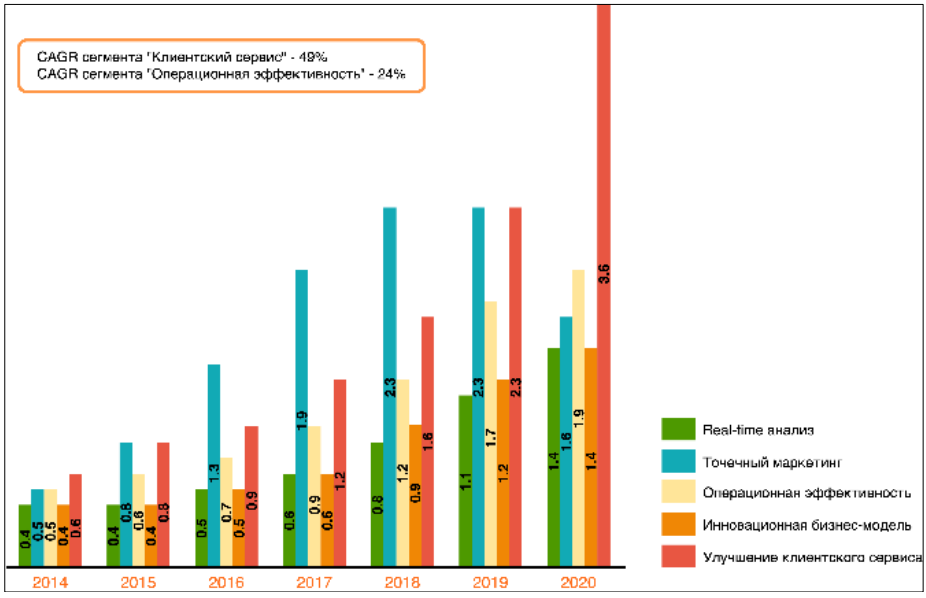


Рис. 5. Большие данные по бизнес–категориям (млрд. долл. США), источник: Heavy Reading
Fig.5. Big data by business category (USD billion), source: Heavy Reading

Наиболее высокий темп роста будет также у сегмента «улучшение клиентского сервиса», прирост составит 49% ежегодно.

Прогноз рынка по подтипам Больших данных будет выглядеть следующим образом (см. рис. 6).

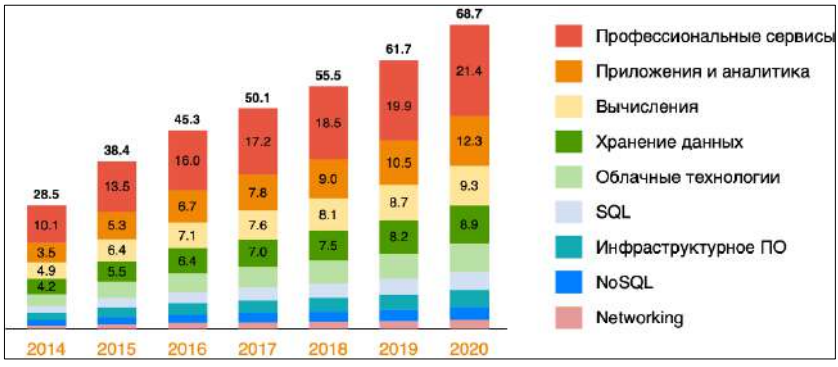


Рис. 6. Объем рынка Больших данных по потипам (млрд. долл. США), источник: Wikibon, IPOboard
Fig. 6. Big Data Market Size by Subtype (USD billion), source: Wikibon, IPOboard

Доминирующую долю рынка, как видно из гистограммы, занимают профессиональные сервисы, наиболее высокий темп рост будет у приложений с аналитикой, их доля возрастет с сегодняшних 12% до 18% в 2020 году и объем данного сегмента будет равняться 12,3 млрд долл. США, доля вычислительного оборудования, напротив, снизится с 20% до 14% и составит около 9,3 млрд долл. США в 2020 году, предложение облачных технологий будет постепенно повышаться и в 2020 году достигнет 6,3 млрд долл. США, доля рынка преобразований для хранения данных, напротив, снизится с 15% в 2014 году до 13% в 2020 году и в денежном выражении будет равняться 8,9 млрд долл. США.

В соответствии с прогнозом Bain & Company's Insights Analysis, распределение рынка Больших данных по отраслям в 2020 году будет выглядеть так.

- финансовая отрасль будет осуществлять расходы на Большие данные в размере 6,4 млрд долл. США со средним темпом роста 22% в год;
- Интернет–фирмы потратят 2,8 млрд долл. США и средний темп увеличения расходов составит 26% за следующие 5 лет;
- расходы государственного сектора будут соразмерными расходам интернет–фирм, но темп роста будет ниже – 22%;
- сектор телекоммуникаций будет возрастать со средним темпом роста 40% и достигнет 1,2 млрд долл. США в 2020 году.
- энергетические фирмы будут инвестировать в эти технологии сравнительно небольшую сумму — 800 млн долл. США, но темп роста будет одним из наиболее высоких – 54% каждый год.

Следовательно, большую долю рынка Больших данных в 2020 году займут фирмы финансовой отрасли, а наиболее быстрорастущим сектором будет являться энергетика.

Следуя прогнозам специалистов, общий объем рынка в ближайшие годы будет увеличиваться. Рост рынка будет обеспечиваться за счет внедрения технологий Big Data в развивающихся государствах, как видно из представленного ниже графика (см. рис. 7).

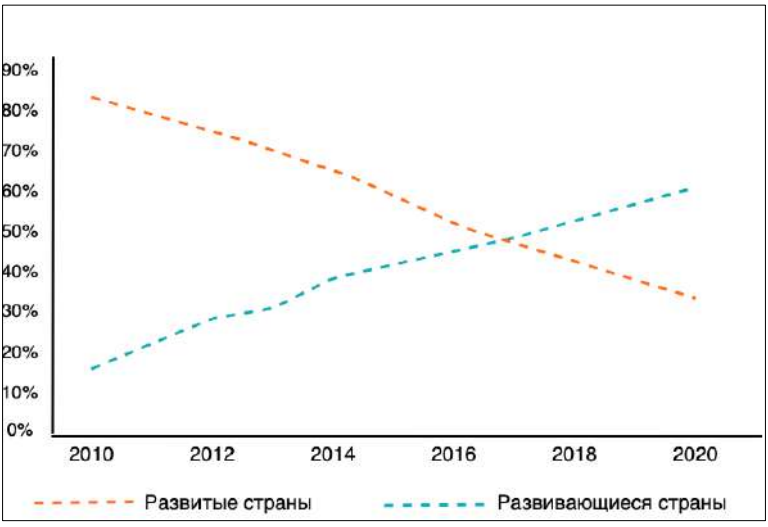


Рис. 7. Доля развитых и развивающихся государств в общем объеме Big Data, %, источник EMC
Fig. 7. The share of developed and developing countries in the total volume of Big Data,%, source: EMC

Прогнозируемый объем рынка будет зависеть от того, как развивающиеся государства воспримут технологии Big Data, будет ли они также популярны как в развитых государствах. В 2014 году развивающиеся государства занимали 40% от объема накопленных данных. По прогнозу EMC, сегодняшняя структура рынка, с преобладанием развитых государств, изменится уже в 2017 году. В соответствии с аналитикой EMC, в 2020 году доля развивающихся государств будет более 60%.

По мнению EMC и Cisco, развивающиеся государства будут активно работать с Большими данными, это будет во многом связано с доступностью технологий и накоплением достаточного объема данных до уровня Больших данных. На карте мира, которая представлена на рис. 8, изображен прогноз повышения темпов роста и объема Big Data по регионам.

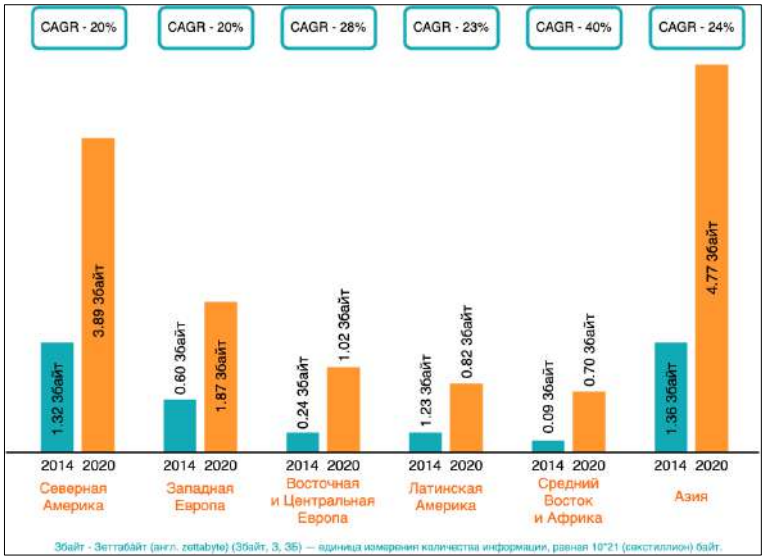


Рис. 8. Темпы роста рынка больших данных по регионам, источник: Cisco, IPOboard
Fig. 8. Growth rates of the big data market by region, source: Cisco, IPOboard

4. Особенности российского рынка Больших данных

Некоторые из нижеперечисленных ситуаций были более удачными в сборе информации, другие — в аналитике Больших данных и способах применения данных, которые были получены в процессе исследования.

Компания «Тинькофф Кредитные Системы» использовала платформу EMC2 Greenplum для массивно-параллельных вычислений. В связи с непрерывным повышением потока пользователей карт в банке появилась необходимость выполнения обработки данных быстрее. Было принято преобразование о применении Больших данных и деятельности с неструктурированными данными, а также корпоративными данными, которые были получены из различных источников. При этом на сайте ФНС России внедряется аналитический слой федерального хранилища информации. Впоследствии на его основе запланировано организовать пространство, которое предоставляет доступ к сведениям налоговой системы для последующей обработки и получения статистики.

Отдельно необходимо рассмотреть российский стартап Synqera, который занимается анализом Больших данных online и разработал платформу Simplate. Ее суть состоит в том, что производится обработка большого массива данных, анализируются данные о потребителях, их возрасте, покупках, душевном состоянии и настроении. Сеть магазинов косметики установила на кассах датчики, которые способны распознавать эмоции покупателей. После определения настроения, анализируются данные о времени покупки, покупателе. После этого потребителю целенаправленно поступают данные об акциях и скидках. Данное преобразование повысило лояльность потребителя и смогла увеличить доход продавца.

Стоит также отметить случай применения технологий Больших данных в фирме Dunkin` Donuts, которая, по аналогии с рассмотренным выше примером, использовала проведение анализа online для повышения своей прибыли. Таким образом, в торговых точках дисплеи отображали специальные предложения, содержимое которых изменялось каждую минуту. Основанием замен в тексте были как время суток, так и товар в наличии. Из кассовых чеков фирма получала данные, какие позиции пользовались максимальным спросом. Этот метод позволил повысить оборот и доход складских запасов.

Как видно, обработка Больших данных оказывает положительное влияние на преобразование бизнес-задач. Важным фактором здесь является выбор стратегии и использование новых разработок в сфере Больших данных.

5. Заключение

В настоящее время Big Data представляют собой один из ключевых драйверов эволюционирования информационно-коммуникационных технологий в условиях высокотехнологического производства. Это направление является относительно новым для российского бизнеса, но получило обширное распространение в западных государствах.

Непрерывно возрастающие возможности обработки больших объемов данных на сегодняшний день кардинально изменяют бизнес-среду и бизнес-процессы. Использование глобальных технологий Больших данных, по мнению автора, может иметь ключевое значение в современном инновационном эволюционировании постиндустриальной экономики. Технологии Больших данных являются совершенно новым трендом эволюционирования, что подтверждается представителями мирового сообщества.

Еще недавно специализированное обучение в сфере Больших данных в мире велось только в трех вузах США (университетах Миссури, Беркли и De-Paul). Однако именно в США к 2018 г. прогнозировалась нехватка 140-190 тыс. специалистов в области Big Data и 1,5 млн. специалистов и менеджеров по извлечению экономической ценности из информационных данных.

Рассмотренные сведения говорят о том, что использование Больших данных связано с большими объемами работ. Успешность же их задается на восприятии бизнесом и государством метаданных как экономической ценности и источника роста, осознании полезности и готовности нести связанные с этим значительные расходы, не ожидая незамедлительной окупаемости вложенных финансовых средств.

Список литературы / References

[1] Lynch C. Big data: how do your data grow? Nature, vol. 455, № 7209, 2008, pp. 28-29.

[2] Артемов С. Big Data: новые возможности для растущего бизнеса. URL: https://www.itweek.ru/upload/iblock/d05/jet-big-data.pdf, 12.07.2020 / Artemov S. Big Data: New Opportunities for a Growing Business (in Russian).

[3] Головина Т.А., Авдеева И.Л., Парахина Л.В. Использование цифровых и мобильных инноваций для развития предприятий регионального интернет-рынка. Вопросы современной экономики, no. 3, 2014 г. / Golovina T.A., Avdeeva I.L., Parakhina L.V. Use of digital and mobile innovations for development of the enterprises regional the market of Internet. Contemporary economic issues, no. 3, 2014 (in Russian).

[4] Корытникова Н.В. Online Big Data как источник аналитической информации в online-исследованиях. Социологические исследования, no. 8, 2015 г., стр. 14-24 / Korytnikova N.V. Online Big Data as a source of analytic information in online research. Sotsiologicheskie issledovaniya [Sociological Studies], no. 8, 2015, pp. 14-24 (in Russian).

[5] Измалкова С.А., Головина Т.А. Использование глобальных технологий «Big Data» в управлении экономическими системами. Известия Тульского государственного университета. Экономические и юридические науки, вып. 4-1, 2015 г., стр. 151-158 / Izmalkova S.A., Golovina T.A. The use of global technologies «Big Data» in the management of economic systems. Bulletin of the Tula State University. Economic and legal sciences, issue 4-1, 2015, pp. 151-158 (in Russian).

[6] Марков Н.Г., Сонькин Д.М., Фадеев А.С., Шемяков А.О., Газизов Т.Т. Интеллектуальные навигационно-телекоммуникационные системы управления подвижными объектами с применением технологии облачных вычислений. Горячая Линия – Телеком, 2014 г., 158 стр. / Markov N.G., Sonkin D.M., Fadeev A.S., Shemyakov A.O., Gazizov T.T. Intelligent navigation and telecommunication systems for controlling mobile objects using cloud computing technology. Hot line – Telecom, 2014, 158 p. (in Russian).

[7] Mayer-Schoenberger V., Cukier K. Big Data: A Revolution That Will Transform How We Live, Work, and Think. Eamon Dolan/Mariner Books, 2014, 272 p.

[8] Савельев А.И. Проблемы применения законодательства о персональных данных в эпоху «Больших данных» (Big Data). Право. Журнал Высшей школы экономики, no. 1, 2015 г., стр. 43–66 / Savelyev A.I. The Issues of Implementing Legislation on Personal Data in the Era of Big Data. Pravo. Zhurnal Vysshey shkoly ekonomiki, no.1, 2015, pp. 43–66 (in Russian).

[9] Толстова Ю. Н. Социология и компьютерные технологии. Социологические исследования, no. 8, 2015 г., стр. 3-13 / Tolstova Yu.N. Sociology and computer technologies. Sotsiologicheskie issledovaniya [Sociological Studies], no. 8, 2015, pp. 3-13 (in Russian).

[10] Соколянский В.В., Пашков Б.С. Технологии Big Data и их инсталляции в экономические исследования. Вопросы экономических наук, no. 4, 2015 г., стр. 169-171 / Sokolyansky V.V., Pashkov B.S. Big Data technologies and their installations in economic research. Issues of economic sciences, no. 4, 2015, pp. 169-171 (in Russian).

[11] Черняк Л. Большие Данные – новая теория и практика. Открытые системы. СУБД, вып. 10, 2011 / Chernyak L. Big Data: New Theory and Practice. Open systems. DBMS, issue 10, 2011 (in Russian).

[12] Japek L., Crater F., Berg M., et al. AAPOR Report: Big Data. American Association of Opinion Researchers. URL: https://www.aapor.org/Education-Resources/Reports/Big-Data.aspx, 12.07.2020

[13] Namiot D., Sneps-Sneppe M. On M2M Software Platforms. International Journal of Open Information Technologies, vol. 2, no. 8, 2014, pp. 29-33.

[14] Namiot D., Sneps-Sneppe M. On IoT Programming. International Journal of Open Information Technologies, vol. 2, no. 10, 2014, pp. 25-28.

[15] Cugola G., Margara A. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys (CSUR), vol. 44, issue 3, 2012, article no. 15.

Информация об авторах / Information about authors

Константин Анатольевич АЛЕКСЕЕВ имеет степень магистра и является разработчиком программного обеспечения с более чем десятилетним опытом. Его научные интересы включают большие данные, разработку в сфере финансовых услуг, мобильную разработку, IoT.

Konstantin Anatolievich ALEKSEEV has a Master's degree and he is a Software Engineer with over ten years of experience. His research interests include Big Data, Financial Services Engineering, Mobile Development, IoT.

DOI: 10.15514/ISPRAS–2020–32(4)–2

Методика выявления центров компетенций авиационной науки на основе публикационной и патентной активности

[1] *В.Г. Беленков, ORCID: 0000-0003-3892-2601 <vbelenkov@ipiran.ru>*
[1] *В.И. Будзко, ORCID: 0000-0002-8235-0404 <vbudzko@ipiran.ru>*
[1] *Д.А. Девяткин, ORCID: 0000-0002-0811-725X <devyatkin@isa.ru>*
[2] *А.В. Кан, ORCID: 0000-0001-9410-406X <avkan@nrczh.ru>*
[2] *И.С. Михайлин, ORCID: 0000-0002-0173-3754 <mikhaylinis@nrczh.ru>*
[1] *И.В. Соченков, ORCID: 0000-0003-3113-3765 <sochenkov@isa.ru>*
[1] *И.А. Тихомиров, ORCID: 0000-0003-0698-7689 <tih@isa.ru>*
[2] *В.С. Шапкин, ORCID: 0000-0003-0812-8319 <shapkinvs@nrczh.ru>*

[1] *Федеральный исследовательский центр «Информатика и управление» РАН, 119333, РФ, Москва, ул. Вавилова, 44, корп. 2*
[2] *Национальный исследовательский центр «Институт им. Н.Е. Жуковского», 125167, РФ, Москва, ул. Викторенко, 7, корп. 12*

Аннотация. Устойчивое развитие и повышение конкурентоспособности являются главными задачами управления научной организацией. Анализ компетенций обеспечивает детальное понимание имеющихся ресурсов при формировании стратегии развития, а их оценка – знание сильных сторон и рисков при ее реализации. Развитие компетенций неизбежно приводит к организационному закреплению ресурсов в подразделениях научной организации – центрах компетенций. Целью исследования является разработка методической базы для выявления и оценки центров компетенции в области авиастроения. Предложенная методика предполагает использование полнотекстовых средств поиска и анализа научно-технических документов для идентификации направлений исследований, технологий и аффилированных центров. Для получения категориальных оценок уровня развития центров компетенций предложен оригинальный подход, включающий аппроксимацию массы научно-технических документов с помощью s-кривых и их анализ с применением теории нечетких множеств. В статье предложена методика выявления и оценки центров научных компетенций авиационной науки и представлены результаты её апробации на примере 143 таких центров в области авиастроения в России. Разработанная методика позволяет использовать полнотекстовые поисково-аналитические инструменты для анализа компетенций, что обеспечивает сформирование детальной оценки имеющихся ресурсов при планировании развития научной организации в области авиастроения. В дальнейшем предполагается автоматизировать предложенную методику путем интеграции соответствующих модулей в состав разрабатываемой экспертной информационной системы по поиску, анализу и учету знаний в авиастроении.

Ключевые слова: цифровой контент; семантический поиск; искусственный интеллект; нечеткие множества; летательный аппарат; безопасность воздушной деятельности; комплексные научно-технологические проекты; авиастроение; стратегическое планирование.

Для цитирования: Беленков В.Г., Будзко В.И., Девяткин Д.А., Кан А.В., Михайлин И.С., Соченков И.В., Тихомиров И.А., Шапкин В.С. Методика выявления центров компетенций авиационной науки на основе публикационной и патентной активности. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 21-40. DOI: 10.15514/ISPRAS–2020–32(4)–2

Благодарности. Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов № 18-29-03215 и № 17-29-07016.

Methodology for identifying centers of excellence in aviation science based on publication and patent activity

[1] *V.G. Belenkov, ORCID: 0000-0003-3892-2601 <vbelenkov@ipiran.ru>*
[1] *V.I. Budzko, ORCID: 0000-0002-8235-0404 <vbudzko@ipiran.ru>*
[1] *D.A. Devyatkin, ORCID: 0000-0002-0811-725X <devyatkin@isa.ru>*
[2] *A.V. Kan, ORCID: 0000-0001-9410-406X <avkan@nrczh.ru>*
[2] *I.S. Mikhailin, ORCID: 0000-0002-0173-3754 <mikhaylinis@nrczh.ru>*
[1] *I.V. Sochenkov, ORCID: 0000-0003-3113-3765 <sochenkov@isa.ru>*
[1] *I.A. Tikhomirov, ORCID: 0000-0003-0698-7689 <tih@isa.ru>*
[2] *V.S. Shapkin, ORCID: 0000-0003-0812-8319 <shapkinvs@nrczh.ru>*

[1] *Federal Research Center «Computer Science and Control» of RAS,*
44, Vavilova str., bldg. 2, Moscow, 119333, Russia
[2] *National Research Center «Zhukovsky Institute»*
7, Viktorenko str., bldg. 12, Moscow, 125167, Russia

Abstract. Sustainable development and increasing competitiveness are the main tasks of managing a scientific organization. The analysis of competencies provides a detailed understanding of the available resources when forming a development strategy, and their assessment - knowledge of the strengths and risks in its implementation. The development of competencies inevitably leads to the organizational consolidation of resources in the departments of a scientific organization - centers of competence. The aim of the study is to develop a methodological framework for identifying and assessing centers of competence in the field of aircraft construction. The proposed technique involves the use of full-text search and analysis tools for scientific and technical documents to identify research areas, technologies and affiliated centers. To obtain categorical assessments of the level of development of centers of competence, an original approach is proposed, including the approximation of the mass of scientific and technical documents using s-curves and their analysis using the theory of fuzzy sets. The article proposes a methodology for identifying and evaluating centers of scientific competence in aviation science and presents the results of its testing on the example of 143 such centers in the field of aircraft construction in Russia. The developed methodology allows the use of full-text search and analytical tools for the analysis of competencies, which ensures the formation of a detailed assessment of the available resources when planning the development of a scientific organization in the field of aircraft construction. In the future, it is planned to automate the proposed methodology by integrating the relevant modules into the developed expert information system for the search, analysis and accounting of knowledge in aircraft construction.

Keywords: digital content; semantic search; Artificial Intelligence; fuzzy sets; aircraft; safety of air activities; complex scientific and technological projects; aircraft construction; strategic planning

For citation: Belenkov V.G., Budzko V.I., Devyatkin D.A., Kan A.V., Mikhailin I.S., Sochenkov I.V., Tikhomirov I.A., Shapkin V.S. Methodology for identifying centers of excellence in aviation science based on publication and patent activity**.** Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 21-40 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–2

Acknowledgments. The study was carried out with the financial support of the Russian Foundation for Basic Research within the framework of research projects No. 18-29-03215 and No. 17-29-07016**.**

1. Введение

В настоящее время мониторинг результативности деятельности научных организаций, выполняющих научно-исследовательские, опытно-конструкторские и технологические работы, проводится в соответствии с Постановлением Правительства Российской Федерации от 8 апреля 2009 г. № 312 «Об оценке и о мониторинге результативности деятельности

научных организаций, выполняющих научно-исследовательские, опытно-конструкторские и технологические работы гражданского назначения» (далее – Постановление Правительства).

Вместе с тем, в соответствии с Проектом распоряжения Правительства Российской Федерации «Об утверждении Стратегии развития авиационной промышленности Российской Федерации на период до 2030 года» (далее – Стратегия) предполагается создание центров специализации и компетенций. При этом предполагается создание условий для эффективного использования предприятиями авиационной промышленности имеющихся производственных мощностей, в том числе, проведение инвентаризации и оценки компетенций и ресурсов, обладающих высоким потенциалом использования в различных видах экономической деятельности, по предприятиям авиационной промышленности, что позволит реализовать мероприятия по реструктуризации предприятий, включая экономически эффективную дозагрузку производственных мощностей за счет выпуска продукции общемашиностроительного применения.

Процесс создания таких условий предполагает проведение работ по выявлению и оценки центров компетенций авиационной науки (ЦКАН), регламентации процесса подготовки планов развития центров и их мониторинга, а также формирование необходимых форм сбора первичной информации и разработку форм отчетности.

Обеспечение устойчивого развития и конкурентоспособности является центральной задачей управления научной организацией. Анализ компетенций дает детальное понимание имеющихся ресурсов при формировании стратегии развития научной организации, а их оценка – знание сильных сторон и рисков при ее реализации.

В общем понимании компетенция – это совокупность ресурсов и знаний, которые обеспечивают получение необходимых результатов. Для выполнения научных исследований ключевыми ресурсами являются специалисты, обладающие знаниями и опытом в том или ином направлении науки, и инфраструктура, в том числе информационная. В связи с этим центр компетенций может быть определен как устойчивый коллектив специалистов, обладающий доступом к научной и информационной инфраструктуре, и способный получать конкурентоспособные научные результаты в определенном направлении науки. Очевидно, что одни и те же специалисты могут относиться к нескольким центрам компетенций, а специалисты из них могут работать в различных организациях сразу по нескольким направлениям. Вместе с тем, развитие компетенций необходимо приводит к закреплению ресурсов в подразделениях научной организации, поэтому конкретный центр компетенций рассматривается, в первую очередь, как структурное подразделение (или их совокупность) научной организации.

В связи с этим разработка методической базы, регламентирующей основные процессы, связанные с центрами компетенций, является актуальной и важной задачей. Анализ компетенций может быть практически важен при выполнении ряда условий.

Первым условием является возможность «измерения» компетенций, т.е. их оценки, выраженной в числовой форме в виде системы показателей. Выбор системы показателей не является универсальным и требует обоснования. Показатели должны обеспечивать мониторинг текущего состояния центров компетенций и позволять отслеживать динамику их развития.

Вторым условием является использование однозначно интерпретируемых показателей. При этом необходимо закрепление правил определения показателей в методических документах. При выполнении этого условия возможен сопоставительный анализ различных центров компетенций.

Третьим условием является унификация состава показателей в выбранной отрасли науки для разных центров компетенций, что дает основу для сопоставления и ранжирования центров компетенций между собой. Выполнение этого условия обеспечивает возможность сопоставительного анализа.

И, наконец, четвертым условием является формализация процессов, связанных с анализом центров компетенций. Это оказывается важным из-за высокой ресурсоемкости процесса анализа компетенций.

Целью настоящего исследования является разработка методической базы для выявления и оценки центров компетенции в области авиастроения. Предлагаемая методика предполагает использование полнотекстовых средств поиска и анализа научно-технических документов для идентификации направлений исследований, технологий и аффилированных центров. В качестве таких средств может выступать экспертная информационная система по поиску, анализу и учету знаний в авиастроении (ЭИС). Макет такой системы, основанный на программном обеспечении TextAplliance [1], был создан в ходе работ 2017-2019 гг. ЭИС – это поисково-аналитическая система, которая позволяет реализовать представляемое методическое обеспечение на массиве данных из открытых источников.

Для получения категориальных оценок уровня развития центров компетенций рассмотрим оригинальный подход, включающий аппроксимацию массы научно-технических документов с помощью s-кривых и их анализ с применением теории нечетких множеств. В заключительной части статьи также представлены результаты апробации методики, в ходе которой было выявлено и проанализировано 143 ЦКАН в России. Предлагаемый в статье аппарат и методическое обеспечение являются до некоторой степени универсальными, т.е. они могут применяться с незначительными изменениями для оценки центров компетенций по инженерным, техническим и естественным наукам в прикладных и фундаментальных сферах.

2. Подходы к выявлению перспективных технологий и центров компетенции

Анализ больших коллекций научно-технических документов, связанных с развитием перспективных направлений науки и техники, а также больших данных, порождаемых на различных этапах жизненного цикла продуктов, позволяет вырабатывать обоснованные решения при управлении научными и промышленными организациями [2]. Задача выявления центров компетенций не является исключением. Для ее решения необходимо использовать методы, средства и инструменты автоматизированного анализа коллекций документов.

Основным подходом, применяющимся для выявления и оценки компетенций и ресурсов, обладающих высоким потенциалом применения в науке и технике, является использование цитатных и патентных баз данных, таких как Scopus, Web of Science, TotalPatent One, Derwent Innovation др. [3]. В России среди подобных решений преобладают зарубежные базы и инструменты. Они характеризуются достаточно высоким качеством данных для англоязычных источников, но для многих направлений науки и техники, в частности для авиастроения, в них отражается лишь незначительная часть российских публикаций, что затрудняет использование этих баз данных для выявления ЦКАН. Кроме того, глобальные технологические и научные тренды не всегда соотносятся с актуальными проблемами и стратегиями развития науки и техники в РФ, что ограничивает применимость этих инструментов.

При использовании подобных инструментов также возникает задача идентификации анализируемой предметной области. Для ее решения обычно используют системы классификации: ГРНТИ, УДК, МКПО, МПК, РИНЦ, Web of Science, Scopus и др. Однако, они учитывают далеко не все существующие направления науки и техники, либо покрывается сразу несколько направлений, не позволяя проводить детальный анализ. Кроме того, эти классификаторы не связаны между собой, что затрудняет интеграцию данных из различных источников, например, патентов и научных публикаций. Вместе с тем, в рамках одного направления могут существовать сразу несколько центров компетенций, занимающихся развитием технологий на различных уровнях готовности (УГТ) [4]. Шкала УГТ содержит 9 уровней, из которых первые два соответствуют формированию научного

задела в виде публикаций и отчетов, уровни 3-6 охватывают период создания технического задела в виде патентов, а последующие три относятся к разработке опытных и серийных образцов. Очевидно, что прямое сопоставление этих центров друг с другом не имеет смысла, поэтому необходимо выявлять специализацию каждого центра и научно-технические документы, характеризующие этот центр компетенций на определенном уровне УГТ, для чего необходимо сопоставление классификаторов.

Перспективным решением этой проблемы является интеграция баз разнородных научно-технических документов и внедрение средств полнотекстового поиска и анализа, а также разработка методик их использования. Такой подход был реализован авторами в ЭИС. В основе подхода лежат идеи, предложенные авторами в ряде исследований, которые мы рассмотрим далее.

Разработке методического обеспечения выявления перспективных направлений науки и техники посвящена работа [5], описывающая концепции прорывных технологий и формальные критерии, позволяющие выявлять такие технологии. Прорывные технологии – это технологии, которые позволяют создавать изделия с новыми сочетаниями свойств, обеспечивающих их качественное отличие от известных аналогов. Развитие таких технологий может быть непривлекательно для крупных организаций, так как их модели получения прибыли предполагают ограниченный объем инвестиций в создание принципиально новых технологий. Однако, такие технологии, могут быть привлекательными для новых игроков. Вместе с тем, развитие таких технологий зачастую приводит к появлению новых и схлопыванию традиционных рынков. В статье [5] предлагаются подходы к выявлению и оценке прорывных технологий, а также стратегии реагирования на их появление, которые могут быть полезны для управления исследованиями и разработками.

В статье [6] представлена циклическая модель для оценки эволюции направлений науки и техники. Рассматриваются подходы к количественному и качественному изучению эволюции технологий. Один из основных таких подходов предложен в работе Р. Фостера [7]. Он состоит в аппроксимации кумулятивной массы научно-технических документов с помощью s-кривых кривых (логистическая кривая, кривая Гомпертца и др.) [8], являющихся решениями уравнения [9]. В работе [10, 11] представлен подход, позволяющий согласовать методы оценки исследований с целями научно-технологического развития государства. В частности, показано, что оценка социальных эффектов, связанных с развитием научных направлений, оказывает важное влияние на расстановку приоритетов при организации исследований, в то же время, приводя к повышению социальной значимости полученных результатов.

В работе [12] предлагается методология управления технологическим развитием авиастроения в условиях сильной неопределенности, присущей периоду смены технологических укладов. Согласно этой методологии, на верхнем уровне управления должны задаваться оптимальные функции отдельных областей науки и техники. В каждой области определяются оптимальные для выполнения этих функций концепции, то есть наборы взаимосвязанных методов или технологий. Предложена иерархическая организации процессов планирования и принятия решений, моделирования, управления компетенциями и знаниями. В работе [13] представлена методология управления прикладными исследованиями и разработками, в которой предусматривается мониторинг готовности технологий к внедрению путем измерения их уровня готовности по формализованным шкалам. Методология основана на принципах управления с учетом специфики научно-исследовательских работ как вида деятельности. Она предполагает повышение уровня обоснованности принимаемых управленческих решений за счет формализованного целеполагания развития технологий. Согласно методологии, приоритет отдается различным типам проектов и механизмам финансирования, в зависимости от УГТ технологий, развиваемых в этих проектах. Такой подход обеспечивает диверсификацию исследований при разработке заданного типа изделий – поскольку одна задача может быть решена

несколькими способами. Это снижает риск отсутствия необходимого задела при разработке изделий. Помимо этого, диверсифицируются и области применения задела, созданного в ходе выполнения проектов. Даже если изделия того или иного типа не будут востребованы рынком в данный момент, разработанные технологии могут найти применение для решения других задач.

Основная же часть работ посвящена исследованию перспективности отдельных технологий, используемых в авиастроении, без привязки к ЦКАН. Так в [14] с применением методов форсайта, выявляются исследования и технологии, оказывающие наибольшее влияние на развитие производства новых конструкционных материалов. Проведена оценка современного состояния отечественных исследований в области новых материалов для выявления «белых пятен», а также зоны паритета и лидерства. При этом необходимо отметить отсутствие исследований, посвященных методикам выявления и оценки научных и технологических центров в области авиастроения, хотя присутствуют программные статьи, описывающие деятельность отдельных центров. Например, в [15], рассматривается работа центра SFB880, созданного в Техническом университете Брауншвейга. Основным направлением деятельности центра является создание прорывных технологий в области снижения шума и повышения взлетно-посадочных характеристик летательных аппаратов (ЛА). Авторы этой статьи отмечают, что подобные задачи требуют разработки новых технологий производства шумопоглощающих поверхностей, а также исследования их аэроакустических и аэродинамических характеристик.

Выявлению и оценке центров компетенции в других областях науки и техники посвящены работы [16, 17]. В этих работах для оценки используются наукометрические индикаторы (индекс Хирша, публикационная и патентная активность), а также связанность и тематическое дублирование работ, выполняемых центрами.

Опираясь на вышеизложенный опыт в сфере оценки перспективности исследований, отметим, что предложенная в настоящей статье методика выявления и оценки ЦКАН предусматривает использование полнотекстовых средств поиска и анализа научно-технических документов для идентификации направлений исследований, технологий и аффилированных центров, а также s-кривых и наукометрических показателей для оценки состояния исследований в выявленных ЦКАН. Это позволяет преодолеть ограничения существующих методов и систем за счёт применения технологий анализа больших массивов полнотекстовых данных: для более детального анализа научных тем и направлений, для оценки траекторий их развития, а также для выявления коллективов, обладающих научными заделами в этих темах.

3. Методическое обеспечение выявления ЦКАН и оценки уровня их развития на основе показателей, характеризующих их публикационную и патентную активность с использованием экспертной информационной системы

Разработанное методическое обеспечение применения экспертных информационных систем для анализа ЦКАН представляет собой набор связанных методик решения следующих задач.

- Выявление и анализ деятельности (состояния) ЦКАН.
- Оценка уровня развития выявляемых ЦКАН в соответствии с выявляемыми перспективными технологическими направлениями в авиастроении и в смежных областях.
- Формирование перечня ЦКАН, сгруппированных по перспективным технологическим направлениям в авиастроении, и оценки уровня их развития на основе показателей, характеризующих их публикационную и патентную активность
- Ранжирование ЦКАН.

Рассмотрим эти методики более подробно.

3.1 Методика выявления и анализа деятельности (состояния) ЦКАН

В методике под *состоянием* понимается совокупность следующих характеристик ЦКАН:

- наименование технологического направления;
- наименование тематики работы;
- состав ключевых слов;
- состав ключевых исполнителей (авторов) и др.

Под *оценкой состояния* ЦКАН понимаются сведения о положении дел со ЦКАН, выявленные в результате анализа материалов, собранных по источникам информации.

Методика применения ЭИС для выявления и анализа деятельности (состояния) ЦКАН включает три шага:

- формирование коллекций документов, относящихся к рассматриваемым организациям;
- расширение коллекций документов (опционально);
- получение инфографической информации на основе обработки коллекций документов для ЦКАН.

Далее приведен набор пошаговых инструкций.

Шаг 1. Формирование коллекций документов, относящихся к рассматриваемым ЦКАН.

Формирование коллекций начинается со сбора научно-технических документов российских организаций в области авиастроения и смежных областей. Для сбора коллекции документов могут быть использованы различные сервисы и средства полнотекстового поиска научно-технической информации.

В качестве запросов используются различные варианты сокращенного и полного наименования организаций. Для более точного наполнения коллекций на этом шаге рекомендуется использовать язык запросов. Результаты поиска документов каждой организации необходимо сохранить в отдельной заранее созданной коллекции.

После сбора коллекции необходимо выполнить ее тематическую кластеризацию, например, с помощью сервиса кластеризации: реализованного в ЭИС (рис. 1). Перед запуском кластеризации необходимо эмпирически подобрать ее порог, после чего сохранить каждый полученный тематический кластер, как отдельную коллекцию.

Рис. 1. Кластеризация пользовательской коллекции
Fig. 1. Clustering custom collection

Каждый кластер является отдельной тематикой деятельности организации. Далее полученные тематики агрегируются в направления деятельности организации с

привлечением эксперта. Аналогично, с привлечением эксперта формируются названия полученных тематик и технологических направлений (в методике ЦКАН – это группа сотрудников организации, работающих в рамках отдельного технологического направления). Пример полученной в итоге структуры пользовательских коллекций для одной из организаций приведен на рис.2.

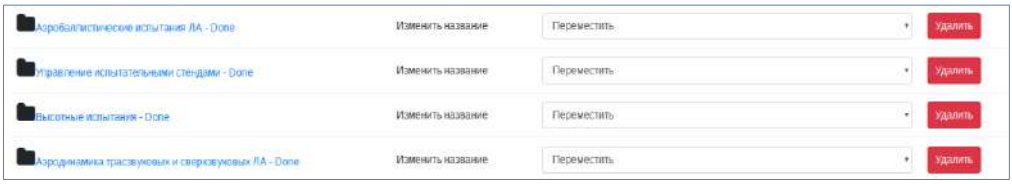


Рис. 2. Пользовательские коллекции, созданные для сохранения документов по различным технологическим направлениям одной из анализируемых организаций
Fig. 2. Custom collections created to save documents in various technological areas of one of the analyzed organizations

Шаг 2 – расширение коллекций документов (опционально).

Предполагается, что часть найденных документов может быть аффилирована с организациями, не входящими в исходный список для анализа (0). Поэтому построенные коллекции могут быть расширены путем включения релевантных документов этих организаций.

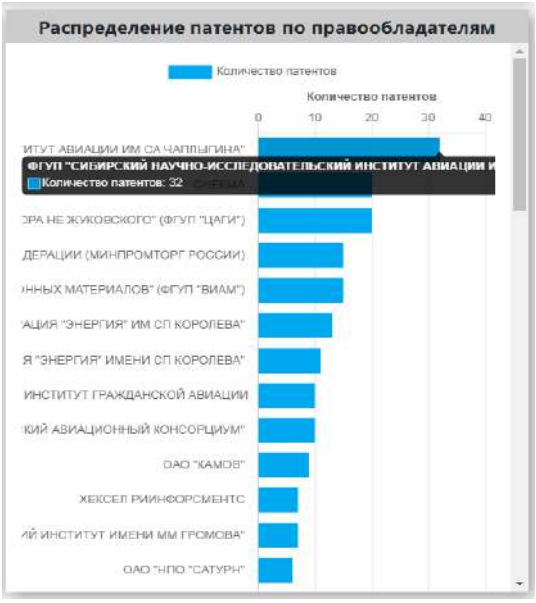


Рис. 3. Распределение документов пользовательской коллекции по организациям
Fig. 2. Distribution custom collection documents by organizations

Шаг 3 –. получение инфографической информации на основе обработки коллекций документов для ЦКАН.

Из сформированных коллекций необходимо извлечь информацию о ключевых исполнителях в ЦКАН. В качестве ключевых исполнителей ЦКАН необходимо выбрать исследователей с наибольшим количеством публикаций внутри коллекции, соответствующей этому ЦКАН. Для получения этой информации необходимо воспользоваться функцией агрегатного анализа. Запуск этой функции производится внутри каждой коллекции, соответствующей направлению деятельности организации. (рис.4).

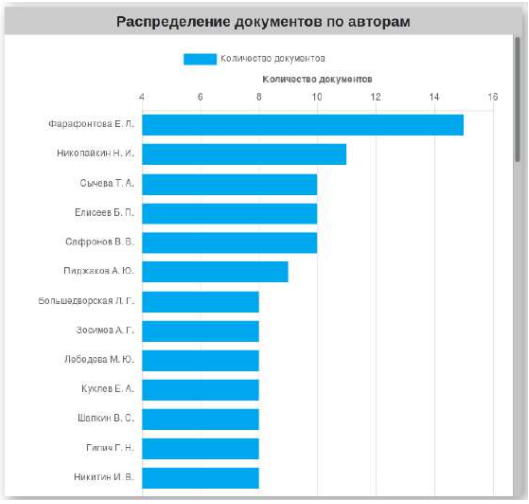


Рис. 4. Распределение документов пользовательской коллекции по авторам
Fig. 4. Distribution custom collection documents by authors

В итоге, для каждого ЦКАН заполняется карточка, содержащая сведения о состоянии ЦКАН, это: его номер, наименование технологического направления, наименование тематики работы, список ключевых слов, список ключевых исполнителей (авторов).

3.2 Методика формирования перечня ЦКАН, сгруппированных по перспективным технологическим направлениям в авиастроении, и оценки уровня их развития на основе показателей, характеризующих их публикационную и патентную активность

Согласно рассматриваемой методике под *уровнем развития* ЦКАН по некоторому технологическому направлению понимается уровень и динамика прироста НТЗ, сформированного в центре по этим направлениям. Оценка уровня развития ЦКАН производится в сравнении с другими ЦКАН в рамках выявленных перспективных авиационных технологических направлений. Все ЦКАН делятся на *три основных категории*: с уровнем развития ниже среднего, со средним и с высоким уровнем развития, при этом:

- к ЦКАН со средним уровнем развития относятся те, чьи показатели динамики патентной или публикационной активности (по годам) за последние 5 лет и уровня заделов за последние 10 лет (по отношению к иным ЦКАН) являются средними или высокими;
- к ЦКАН с высоким уровнем развития относятся те, чьи показатели динамики патентной или публикационной активности (по годам) за последние 5 лет и уровня заделов за последние 10 лет (по отношению к иным ЦКАН) являются высокими, либо показатель динамики патентной или публикационной активности за последние 5 лет является высоким, а показатель уровня заделов за последние 10 лет средним;
- к ЦКАН с уровнем развития ниже среднего относятся те, чьи показатели не позволяют отнести их к ЦКАН со средним или высоким уровнем развития.

3.3 Определение динамики патентной или публикационной активности ЦКАН

Определение динамики патентной или публикационной активности осуществляется для каждого ЦКАН по годам (за ряд лет). Для определения динамики как высокой, средней и ниже среднего (невысокий) для ЦКАН введем лингвистическую переменную (ЛП) «Динамика ЦКАН», которая может принимать значения из множества $X = \{X_{выс}, X_{низ}, X_{сред}\}$, где $X_{выс} = 'высокий'$, $X_{низ} = 'невысокий'$, $X_{сред} = 'средний'$ и построим для нее функции

принадлежности $\mu_{выс.}$, $\mu_{сред}$, $\mu_{низк}$. Для этого выполним следующую последовательность действий.

Рис. 5. Накопление кумулятивной массы патентов
Fig. 5. Accumulation of the cumulative mass of patents

1. Получить данные о кумулятивной массе НТД, созданных в анализируемом ЦКАН за заданный период (рис. 5). Для получения этих данных могут использоваться различные сервисы и средства полнотекстового поиска научно-технической информации (см. подраздел 3.1 – методику выявления и анализа деятельности (состояния) ЦКАН).
2. Аппроксимировать полученные данные с помощью S-образной кривой и определить ее параметры. В качестве S-образной кривой может использоваться логистическая функция $f: \mathbb{R} \to \mathbb{R}^+$, $f(x) = \frac{C}{b+e^{-ax}}$., где $a, b, C \epsilon \mathbb{R}^+$, x – год за который производится оценка динамики (рис. 6).

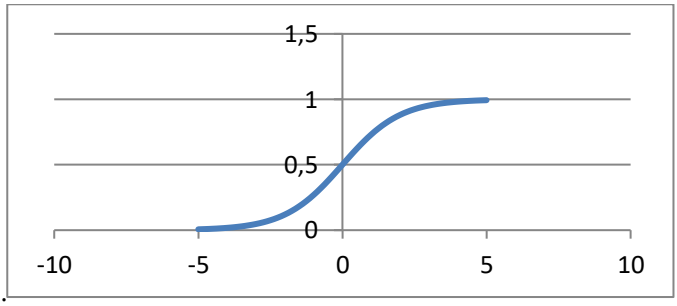


Рис. 6. Пример графика функции, аппроксимирующей кумулятивную массу научно-технических документов
Fig. 6. Example graph of the function that approximates the cumulative mass of scientific and technical documents

Наибольшей скорости прироста массы научно-технических документов (НТД) соответствует максимум первой производной этой функции:

$$f'(x) = \frac{Cae^{-ax}}{(b+e^{-ax})^2}.$$

Динамика (скорость) прироста НТД, как было отмечено выше, характеризуются значением $f'(x)$. Эта производная всюду положительна и ограничена сверху. В соответствии с условием $\mu \leq 1$ она должна быть нормирована на максимум. Значение аргумента x_{max}, соответствующее этому максимуму определяется выражением:

$$f''(x) = -\frac{Ca^2\left(1-\frac{2e^{-ax}}{b+e^{-ax}}\right)e^{-ax}}{(b+e^{-ax})^2},$$
$$f''(x_{max}) = 0, x_{max} = -\frac{ln\,(b)}{a}.$$

3. Построить функции принадлежности для введенных ЛП.

$$\mu_{\text{выс}}, \ \mu_{\text{сред}}, \ \mu_{\text{низк}}: \mathbb{R} \to [0,1].$$

Для этого в соответствии с подходом, изложенным в [18], определим $\mu_{\text{выс}}$, характеризующую высокие показатели динамики прироста НТД, как

$$\mu_{\text{выс}}(x) = \frac{f\prime(x)}{f'_{max}},$$

$$f'_{max} = \frac{Ca}{4b}.$$

Так как значение «невысокий» семантически является отрицанием значения «высокий», функция $\mu_{\text{низк}}$ принадлежности множества значений аргумента функции $f(x)$, относящихся к невысоким показателям динамики прироста НТД, будет определяться как

$$\mu_{\text{низк}}(x) = 1 - \mu_{\text{выс}}(x).$$

Средней динамике соответствует ситуация, в которой динамика является одновременно, в равной степени, высокой и низкой. С точки зрения нечеткой логики, этой ситуации соответствует равенство значений функций принадлежности к высокой и низкой динамике, поэтому средней динамике прироста НТД будет соответствовать функция $\mu_{\text{сред}}$ имеющая следующий вид:

$$\mu_{\text{сред}}(x) = 1 - |\mu_{\text{выс}}(x) - \mu_{\text{низк}}(x)|.$$

Пример графиков функций принадлежности $\mu_{\text{выс}}$, $\mu_{\text{сред}}$ и $\mu_{\text{низ}}$ приведен на рис.7.

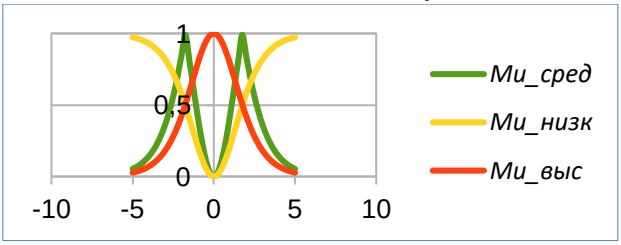


Рис. 7. Пример графиков функций принадлежности $\boldsymbol{\mu}_{\text{выс}}$, $\boldsymbol{\mu}_{\text{сред}}$ *и* $\boldsymbol{\mu}_{\text{низ}}$
Fig. .7 Example graphs of membership functions $\boldsymbol{\mu}_{hi}$, $\boldsymbol{\mu}_{mid}$ *and* $\boldsymbol{\mu}_{low}$

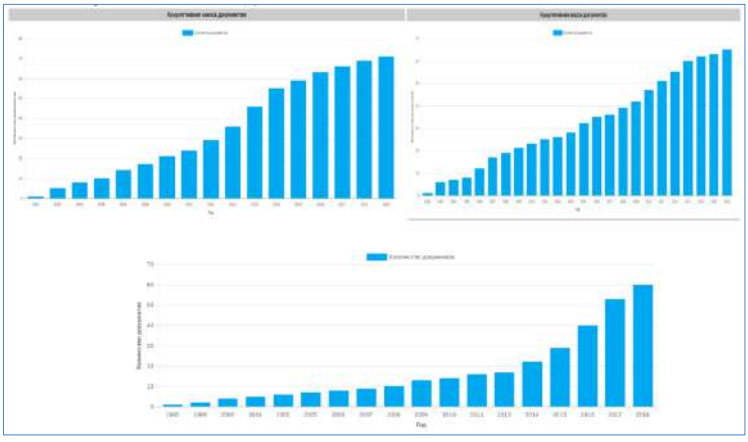


Рис. 8. Кумулятивная динамика массы публикаций и патентов ЦКАН: низкая, средняя и высокая динамика публикационной или патентной активности
Fig. 8. Cumulative mass dynamics ASCC publications and patents: low, medium and high dynamics of publication or patent activity

4. Для оценки динамики НТД за год в отдельном ЦКАН (точка – x) выбрать функцию $\mu_{\text{выс}}$., $\mu_{\text{сред}}$, $\mu_{\text{низк}}$ с максимальным значением и соответствующее ей значение ЛП. При равных

значениях предпочтение отдается оптимистической оценке. Критерием отнесения динамики НТД за год к высокой, средней и ниже среднего является приведенное выше правило определения той из функций принадлежности, значение которой в этот год максимально (рис.8).

В качестве оценки по конкретному ЦКАН динамики НТД за 5 лет выбирается та оценка, которая получена для максимального количества лет из числа рассматриваемых, при наличии нескольких равноправных оценок – та из них которая была наиболее характерна для последних лет (сумма лет по которой больше).

Определение уровня патентной или публикационной активности осуществляется для каждого ЦКАН по годам (за фиксированный интервал дат). Для определения по данным за год уровня патентной или публикационной активности ЦКАН (по отношению к иным ЦКАН) как высокого среднего и ниже среднего для группировки ЦКАН, по которым осуществляется анализ, введем лингвистическую переменную «Уровень заделов ЦКАН», принимающую значения из множества Z.

$$Z = \{Z_{\text{выс}}, Z_{\text{низ}}, Z_{\text{сред}}\}, \text{ где } Z_{выс} = \text{'высокий'}, Z_{низ} = \text{' низкий'}, Z_{сред} = \text{'средний'}$$

и построим для нее функции принадлежности $\mu_{\text{выс}}$., $\mu_{\text{сред}}$, $\mu_{\text{низк}}$. Для этого выполним следующую последовательность действий.

1. Получить данные о распределении выявленных ЦКАН определенного направления деятельности (НД) по количеству НТД (рис. 9).

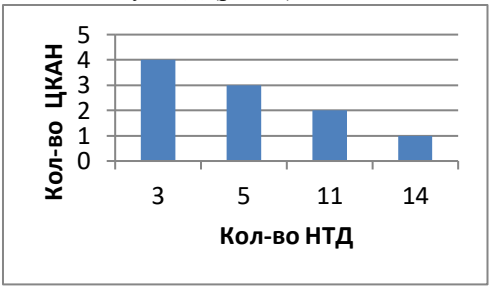


Рис. 9. Пример гистограммы распределения НТД по ЦКАН
Fig .9. Example histogram distribution of NTD ASCC

2. Построить кривую Парето, аппроксимирующую полученные данные и определить ее параметры $(x_{0,} k)$. Кривая Парето задается функцией

$$f_з(x) = \frac{k x_0^k}{x^{k+1}},$$
$$x \geq x_{0,} k > 0.$$

Выбор этой функции обусловлен эмпирически выявленной закономерностью в распределениях количества НТД анализируемых ЦКАН. Эта функция всюду положительна, ее максимум соответствует ЦКАН с наименьшим числом НТД в рамках анализируемого НД. Аргумент этой функции x, соответствует группе ЦКАН с определенным уровнем заделов (ЦКАН, относящихся к одному столбцу гистограммы). Обозначим x_{max} аргумент этой функции, соответствующий ЦКАН с наибольшим количеством НТД.

Далее, по аналогии с построением оценок динамикой НТД по конкретному ЦКАН выполним следующие действия.

Определим функцию принадлежности к множеству значений аргумента функции $f_з(x)$, относящихся к низким показателям заделов ЦКАН. Для этого нормируем эту функцию таким образом, чтобы ее область значений лежала в интервале *[0,..,1]*:

$$\mu_{\text{низк}}(x) = \frac{f_з(x) - f_з(x_{\max})}{f_з(x_0) - f_з(x_{\max})} = \left(\frac{x_0^k}{x^{k+1}} - \frac{x_0^k}{x_{\max}^{k+1}}\right) \frac{x_0 x_{\max}^{k+1}}{x_{\max}^{k+1} - x_0^{k+1}}.$$

Определим функцию принадлежности множества значений аргумента функции $f_з(x)$, относящихся к высоким показателям заделов ЦКАН, как

$$\boldsymbol{\mu}_{\text{выс}}(\boldsymbol{x}) = \mathbf{1} - \boldsymbol{\mu}_{\text{низк}}(\boldsymbol{x}).$$

При этом средним показателям заделов ЦКАН соответствует функция $\mu_{\text{сред}}$ имеющая следующий вид:

$$\mu_{\text{сред}} = 1 - |\mu_{\text{выс}}(x) - \mu_{\text{низк}}(x)|.$$

Для оценки уровня заделов по конкретному (точка – x) ЦКАН (по отношению к иным ЦКАН) по данным за год выбирается функция $\mu_{\text{выс}}$., $\mu_{\text{сред}}$, $\mu_{\text{низк}}$ с максимальным значением и соответствующее ей значение ЛП. При равных значениях предпочтение отдается оптимистической оценке. То есть, критерием отнесения уровня заделов по конкретному (точка – x) ЦКАН (по отношению к иным ЦКАН) по данным за год к высокому, среднему и низкому является приведенное выше правило определения той из функций принадлежности, значение которой в этот год максимально. В качестве оценки по конкретному ЦКАН уровня заделов за ряд лет выбирается та оценка, которая получена для максимального количества лет из числа рассматриваемых, при наличии нескольких равноправных оценок – та из них, которая была наиболее характерна для последних лет (сумма лет по которой больше).

Для отнесения конкретных ЦКАН к ЦКАН с уровнем развития ниже среднего, со средним и с высоким уровнем развития выполним следующие действия.

1. Введем ЛП «Уровень развития ЦКАН». Для этого зададим множество $L = \{L_{\text{выс}}, L_{\text{ср}}, L_{\text{низк}}\}$, где $L_{\text{выс}}$='высокий уровень развития', $L_{\text{ср}}$='средний уровень развития', $L_{\text{низк}}$ ='уровень развития ниже среднего'.
2. Для определения значений ЛП воспользуемся полученными выше результатами и формулами теории нечетких множеств:

$$\boldsymbol{L}_{\text{выс}} = (\boldsymbol{X}_{\text{выс}} \cap (\boldsymbol{Z}_{\text{выс}} \cup \boldsymbol{Z}_{\text{сред}})),$$

$$\boldsymbol{L}_{\text{ср}} = (\boldsymbol{Z}_{\text{сред}} \cup \mathbf{Z}_{\text{выс}}) \cap \mathbf{X}_{\text{сред}},$$

$$L_{\text{низк}} = \overline{(L_{\text{выс}} \cup L_{\text{ср}})},$$

где:

$$\mu_{Z \cap X}(x_з, x_д) = min(\mu_Z(x_з), \mu_X(x_д)),$$
$$\mu_{Z \cup X}(x_з, x_д) = max(\mu_Z(x_з), \mu_X(x_д)) –$$

функции, участвующие в выполнении операций объединения и пересечения нечетких множеств, $x_з$ – кол-во НТД ЦКАН, $x_д$ – год, в который оценивается динамика ЦКАН.

Примечания.

1. Для качественной оценки уровня развития ЦКАН необходимо обращать внимание не только на низкие или высокие показатели одного из указанных критериев, но также и на род деятельности ЦКАН. Так, ЦКАН, ведущему фундаментальную исследовательскую деятельность, допустимо иметь невысокий показатель динамики патентной активности, при условии высоких показателей динамики публикационной активности.
2. В ходе оценки приведенных критериев с использованием средств и сервисов поиска и анализа научно-технической информации необходимо учесть, что анализ динамики должен производиться отдельно для патентов и публикаций, то есть для каждого ЦКАН должны быть предварительно сформированы отдельные коллекции по видам научно-технических документов.

3.4 Методика оценки уровня развития выявляемых ЦКАН в соответствии с выявляемыми перспективными технологическими направлениями в авиастроении и в смежных областях

Согласно этой методике для оценки уровня развития выявляемых ЦКАН в соответствии с выявляемыми перспективными технологическими направлениями в авиастроении и в смежных областях, необходимо выполнить следующую последовательность действий:

- в соответствии с методикой произвести выявление и оценку состояния ЦКАН;
- выявить перспективные технологические направления российских исследовательских организаций в авиастроении и смежных областях;
- для ЦКАН, работающих в перспективных областях, в соответствии с методикой сформировать перечень ЦКАН, сгруппированных по перспективным технологическим направлениям в авиастроении, и оценить уровень их развития на основе показателей, характеризующих публикационную и патентную активность.

3.5 Методика ранжирования ЦКАН

Методика ранжирования ЦКАН, относящихся к одному направлению деятельности (к перспективному технологическому направлению в авиастроении и в смежных областях), а также построения ранжированного перечня выявленных российских ЦКАН в соответствии с разработанными подходами и критериями включает четыре шага, это:

- формирование перечня ЦКАН, сгруппированных по технологическим направлениям;
- расчет ранга ЦКАН, относящихся к одному направлению деятельности, на основе основных критериев;
- уточнение информации для определения ранга на основе дополнительных критериев;
- сортировка ЦКАН в соответствии с рангами.

Для ранжирования выявленных ЦКАН необходимо проделать следующую последовательность действий:

Шаг 1 (Формирование перечня ЦКАН, сгруппированных по технологическим направлениям). Результатом выполненных на этом шаге действий является список карточек ЦКАН. Каждая карточка содержит информацию о тематике работ ЦКАН, технологическом направлении, ключевых исполнителях, публикационных и патентных заделах, о публикационной и патентной динамике.

Шаг 2 (Расчет ранга ЦКАН, относящихся к одному направлению деятельности, на основе основных критериев). На этом шаге на основе информации, полученной на предыдущем шаге, формируется рейтинг ЦКАН с использованием следующих основных критериев.

- динамика публикационной активности сотрудников ЦКАН за последние 5 лет (имеет второй приоритет применительно к ЦКАН, деятельность которых носит исследовательскую направленность);
- динамика патентной активности сотрудников ЦКАН за последние 5 лет (имеет второй приоритет применительно к ЦКАН, деятельность которых носит производственную направленность);
- задел (общее количество публикаций или патентов) сотрудников ЦКАН за последние 10 лет (имеет первый приоритет).

Информация, необходимая для вычисления этих критериев уже содержится в карточках ЦКАН после выполнения шага 1.

Шаг 3 (Уточнение информации для определения ранга на основе дополнительных критериев). На этом шаге при необходимости (необходимость определяется экспертно) результаты формирования информации для определения рейтинга ЦКАН уточняются с использованием следующих дополнительных критериев, использующих информацию,

получаемую из внешних источников, это: наукометрические показатели наиболее активно публикующихся авторов: индекс Хирша, индекс Хирша без самоцитирований, импакт фактор журналов, в которых опубликованы работы (рис.10).

Данную процедуру имеет смысл проводить при сравнении ЦКАН, занимающихся фундаментальными исследованиями. Таковым учреждение будет являться, при наличии высокой положительной динамики публикационной активности (научные публикации). ЦКАН, имеющие высокую динамику патентной активности, следует отнести к организациям, занимающимся прикладными исследованиями.

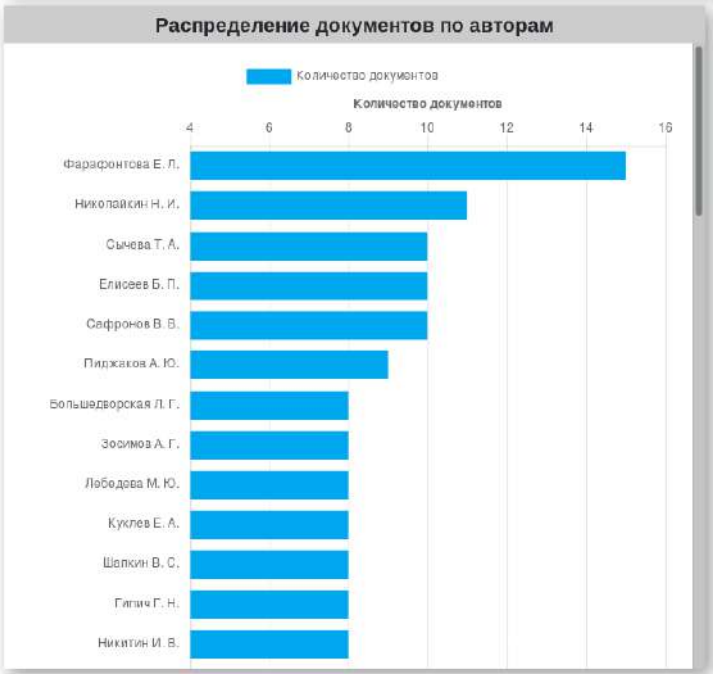


Рис. 10. График распределения документов по авторам
Fig. 10. Graph of distribution of documents by author

Для получения сведений, необходимых для уточнения наукометрических показателей по каждому отобранному автору, используются следующие источники:

- РИНЦ (Российский индекс научного цитирования) – eLibrary.ru;
- реферативная база данных Scopus – scopus.com;
- реферативная база данных Web of Science – webofknowledge.com.

Итоговый рейтинг ЦКАН определяется экспертно на основе перечисленных выше критериев.

Шаг 4 (Сортировка ЦКАН в соответствии с рангами). На этом шаге на основе данных, представленных в ранжированном списке ЦКАН, ЦКАН разбиваются на три группы. К первой группе относятся 10% ЦКАН с наиболее высокими показателями. Ко второй группе относятся 30% ЦКАН с наиболее высокими показателями. К третьей группе относятся 30% ЦКАН с наиболее низкими показателями. Интервалы значений рейтинга определены для условий высокой неопределенности в соответствии с подходом [19].

4. Апробация методик на модельных данных

Для апробации методик использовался макет ЭИС, в который были загружены информационные базы научно-технических и патентных документов в количестве ~25,5 млн. документов общим объемом ~2,35 ТБ. В ходе апробации поиск проводился по следующим коллекциям: «Российские журналы», «Российские научные издания по авиастроению»,

«Авторефераты диссертаций», «ФИПС Полезные модели», «ФИПС Промышленные образцы» и «ФИПС Изобретения».

Основные результаты ранжирования ЦКАН, полученные на основе приведенной выше методики, представлены в табл.1, содержащей сведения о центрах с наивысшим рангом по каждому направлению. Представленные результаты носят модельный характер, так как были получены исключительно путем анализа открытых источников информации, не содержащих, в виду специфики области, описания многих перспективных технологий, методов и образцов техники.

Табл. 1. ЦКАН с наивысшим рангом по направлениям научно-технической деятельности
Table 1. Aviation Science Centers of Excellence with the highest rank in areas of scientific and technical activities

ЦКАН	**Патенты**	**Публикации**	**Динамика прироста патентов**	**Динамика публикаций**
Аэродинамика гиперзвуковых ЛА				
ЦАГИ	3	62	низкая	средняя
Малоразмерные ГТД				
СГАУ	0	33	-	средняя
Аэродинамика трансзвуковых и сверхзвуковых ЛА				
ЦАГИ	35	41	средняя	средняя
Вихревая безопасность				
ЦАГИ	0	6	-	средняя
Лопатки\диски ГТД				
ВИАМ	19	49	средняя	высокая - рост в последние годы (2013-2017)
Гиперзвуковые аэродинамические трубы				
ЦАГИ	5	14	средняя	средняя
Авиабортовые системы электроснабжения				
УГАТУ	0	19	-	средняя
Навигационные системы				
МГТУ ГА	0	44	-	средняя, небольшой рост в 2018
Гиперзвуковые ПВРД				
ЦИАМ	17	5	Средняя	низкая - спад в последние годы
Ресурсные испытания ЛА				
ВИАМ	11	50	низкая	высокая - значительный рост с 2014 года
Имитация целей				
ГОСНИИАС	6	0	высокая	-
Рельсовые стенды				
ГКНИПАС	4	0	низкая	-

ЦКАН	Патенты	Публикации	Динамика прироста патентов	Динамика публикаций
Интегрированные платформы авионики				
Центр комплексирования ОАК	0	5	-	средняя
Компрессоры и турбины ГТД				
УМПО	218	12	высокая - резкий рост с 2014	низкая
Прочностные испытания ЛА				
ЦАГИ	12	12	средняя	средняя
Снижение шума ЛА				
ЦИАМ	20	6	средняя	средняя
Управление ГТД				
МАИ	0	23	-	высокая - рост с 2015 года
Управление испытательными стендами				
ЦАГИ	2	10	средняя небольшой рост в последние годы	низкая
Обработка изображений				
МГТУ им Баумана	1	7	Низкая	средняя
Производство наноматериалов для авиастроения				
МГТУ им Баумана	1	7	Низкая	средняя
Высотные испытания				
ЦИАМ	13	2	средняя	низкая
Датчики				
КАИ	15	12	низкая - последний патент в 2015	средняя

5. Заключение

В ходе апробации представленной методики было выявлено 25 технологических направлений российских исследовательских организаций в авиастроении и смежных областях, из них 7 перспективных, а также 20 глобальных технологических трендов, оказывающих влияние на развитие авиастроения в России и мире. В рамках этих технологических направлений выявлено, оценено и ранжировано 143 ЦКАН. Корректность полученных результатов была оценена экспертами, которые подтвердили применимость предложенной методики для выявления и анализа ЦКАН.

Разработанная методика позволяет использовать поисково-аналитические инструменты, такие как ЭИС, для анализа компетенций, что позволит сформировать детальную оценку имеющихся ресурсов при планировании развития научной организации в области авиастроения. В дальнейшем предполагается автоматизировать шаги предложенной методики путем интеграции соответствующих модулей в состав разрабатываемой ЭИС по поиску, анализу и учету знаний в авиастроении.

Предложенное методическое и информационное обеспечение для выявления ЦКАН может быть адаптировано и для других предметных областей с целью выявления и ранжирования центров компетенций по тематикам и направлениям, принятым в этих областях.

Список литературы / References

[1] Ананьева М. И., Девяткин Д.А., Зубарев Д.В., Осипов Г.С., Смирнов И.В., Соченков И.В., Тихомиров И.А., Швец А.В., Шелманов А.О. TextAppliance: поиск и анализ больших массивов текстов. Труды Пятнадцатой национальной конференции по искусственному интеллекту с международным участием, 2016 г., стр. 220-228 / Ananyeva M.I., Devyatkin D.A., Zubarev D.V., Osipov G.S., Smirnov I.V., Sochenkov I.V., Tikhomirov I.A., Shvets A.V., Shelmanov A.O. TextAppliance: Search and Analysis of Large Arrays of Text. In Proc. of the Fifteenth National Conference on Artificial Intelligence with International Participation, 2016, pp. 220-228 (in Russian).

[2] S. Ren, Yingfeng Zhang, Y. Liu, T. Sakao, D. Huisingh, C. Almeida. A comprehensive review of big data analytics throughout product lifecycle to support sustainable smart manufacturing: A framework, challenges and future research directions. Journal of Cleaner Production, vol. 210, 2019, pp. 1343-1365.

[3] Liu G. F., Sun H. P., Song X. P. Visualizing and mapping the research on patents in information science and management science. Malaysian Journal of Library & Information Science, vol. 19, № 1, 2014, pp. 87-103.

[4] ГОСТ Р 57194.1-2016. Трансфер технологий. Общие положения / GOST R 57194.1-2016. Technology transfer. General Provisions (in Russian),

[5] Christensen C.M., McDonald R., Altman E.J., Palmer J.E. Disruptive innovation: An intellectual history and directions for future research. Journal of Management Studies, vol. 55, № 7, 2018, pp. 1043-1078.

[6] Zhang G., Morris E., Allaire D., and McAdams D.A. Research Opportunities and Challenges in Engineering System Evolution. Journal of Mechanical Design, vol. 142, № 8, 2020.

[7] Foster R.N. Working the S-curve: assessing technological threats. Research Management, vol. 29, № 4, 1986, pp. 17-20.

[8] Mansfield E. Technical Change and the Rate of Imitation. Econometrica, vol. 29, № 4, 1961, pp. 741-766.

[9] Richards F.J. A Flexible Growth Function for Empirical Use. Journal of Experimental Botany, vol. 10, № 2, 1959, pp. 290–300.

[10] Arocena R., Göransson B., Sutz J. Towards making research evaluation more compatible with developmental goals. Science and Public Policy, vol. 46, № 2, 2019, pp. 210-218.

[11] Andersen B. The hunt for S-shaped growth paths in technological innovation: a patent study. Journal of evolutionary economics, vol. 9, № 4, 1999, pp. 487-526.

[12] Дутов А.В., Клочков В.В., Рождественская С.М. Стратегическое управление технологическим развитием и научно-техническими знаниями (на примере авиастроения). Друкеровский вестник, № 1, 2019 г., стр. 177-191 / Dutov A.V., Klochkov V.V., Rozhdestvenskaya S.M. Strategic management of technological development and scientific and technical knowledge (on the example of aircraft construction). Drukerovskiy Vestnik, № 1, 2019, pp. 177-191 (in Russian).

[13] Рождественская С.М., Клочков В.В. Методический инструментарий формирования программ технологического развития и перечня критических технологий в авиастроении. Россия: тенденции и перспективы развития, № 12-2, 2017 г., стр. 469-503 / Rozhdestvenskaya S.M., Klochkov V.V. Methodical toolkit of formation of technological development programs and the list of critical technologies in aircraft. Russia: Trends and Prospects, № 12-2, 2017, pp. 469-503 (in Russian).

[14] Vishnevskiy K., Yaroslavtsev A. Russian S&T Foresight 2030: case of nanotechnologies and new materials. Foresight, vol. 19, No. 2, 2017, pp. 198-217.

[15] Radespiel R., Heinze W., Bertsch L. High-lift research for future transport aircraft. Deutsche Luft- und Raumfahrtkongress (DLRK), 2017.

[16] Земцов С. П. Опыт выявления и оценки потенциала инновационных кластеров (на примере отрасли «Рациональное природопользование»). Региональные исследования, том 40, № 2, 2013, стр. 12-19 / Zemtsov S. P. Experience of identifying and assessing the potential of innovative clusters (on the example of the "Rational use of natural resources" industry). Regional research, vol. 40, № 2, 2013, pp. 12-19 (in Russian).

[17] Devyatkin D.A., Suvorov R.E., Tikhomirov I.A. A Method for the Identification of Competence Centers Based on the Example of the Artificial Intelligence Domain. Scientific and Technical Information Processing, vol. 44, № 4, 2017, pp. 253-260.

[18] Zadeh L.A. The concept of a linguistic variable and its application to approximate reasoning – I. Information sciences, vol. 8, № 3, 1975, pp. 199-249.

[19] Беленков В. Г. Вопросы методического обеспечения построения перспективного КСА (VI). Наукоемкие технологии, том 11, № 1, 2010 г., стр. 038-072 / Belenkov V.G. Issues of methodological support for the construction of a promising CSA (VI). Science-intensive technologies, vol. 11, № 1, 2010, pp. 038-072 (in Russian).

Информация об авторах / Information about authors

Виктор Геннадьевич БЕЛЕНКОВ – кандидат технических наук, ведущий научный сотрудник Института проблем информатики Федерального исследовательского центра «Информатика и управление» РАН. Имеет большой опыт работы по научно техническому сопровождению и обеспечению создания автоматизированных систем.

Viktor Gennadievich BELENKOV – PhD, Leading Researcher at the Institute of Informatics Problems RAS of the Federal Research Center "Informatics and Management" RAS. Has extensive experience in scientific and technical support and ensuring the creation of automatic systems.

Владимир Игоревич БУДЗКО – доктор технических наук, заместитель директора по научной работе, Институт проблем информатики Федерального исследовательского центра «Информатика и управление» РАН. Сфера его научных интересов лежит в области создания крупномасштабных информационно-телекоммуникационных систем (ИТС).

Vladimir Igorevich BUDZKO – Doctor of Technical Sciences, Deputy Director for Science, Institute of Informatics Problems of the Federal Research Center "Information and Management" RAS. The sphere of his scientific interests lies in the creation of large-scale information and telecommunication systems (ITS).

Дмитрий Алексеевич ДЕВЯТКИН – научный сотрудник. Его научные интересы включают интеллектуальный анализ текстов, построение когнитивных моделей, семантический анализ.

Dmitry Alekseevich DEVYATKIN – lead scientist. His research interests include text mining, cognitive model building, semantic analysis.

Анна Владимировна КАН – кандидат технических наук, начальник аналитического отдела департамента координации и сопровождения программ. Основные научные интересы в области системного анализа, имитационного моделирования и искусственного интеллекта.

Anna Vladimirovna KAN – Ph.D., head of the analytical department of the department of coordination and support programs. Her main research interests are in the field of systems analysis, simulation and artificial intelligence.

Иван Сергеевич МИХАЙЛИН – заместитель генерального директора. Основные научные интересы в области системного анализа, имитационного моделирования и искусственного интеллекта.

Ivan Sergeevich MIKHAILIN – Deputy General Director. His main research interests are in the field of systems analysis, simulation and artificial intelligence.

Илья Владимирович СОЧЕНКОВ – кандидат физико-математических наук, начальник лаборатории. Его научные интересы включают интеллектуальный анализ текстов, построение когнитивных моделей, семантический анализ.

Ilya Vladimirovich SOCHENKOV – Candidate of Physics and Mathematics Sciences, Head of Laboratory. His research interests include text mining, cognitive model building, semantic analysis.

Илья Александрович ТИХОМИРОВ – кандидат технических наук. Его научные интересы включают интеллектуальный анализ текстов, построение когнитивных моделей, семантический анализ, наукометрию, анализ научных цитирований; автоматизированную оценку научных работ.

Ilya Aleksandrovich TIKHOMIROV – Candidate of Technical Sciences. His research interests include text mining, building cognitive models, semantic analysis, scientometrics, scientific citations analysis; automated evaluation of scientific papers.

Василий Сергеевич ШАПКИН –доктор технических наук, профессор, первый заместитель генерального директора. Известный ученый в области поддержания летной годности воздушных судов.

Vasily Sergeevich SHAPKIN – Doctor of Technical Sciences, Professor, First Deputy General Director. A well-known scientist in the field of maintaining the airworthiness of aircraft.

DOI: 10.15514/ISPRAS–2020–32(4)–3

Общие подходы к проектированию подсистемы доступа высокопроизводительных вычислительных систем

С.Ю. Мокшин, ORCID: 0000-0002-7454-6597 <sumo@rambler.ru>
Всероссийский НИИ технической физики имени академика Е.И. Забабахина,
456770, Россия, г. Снежинск, Челябинская область, ул. Васильева, 13

Аннотация. Создание высокопроизводительной вычислительной системы, предназначенной для решения задач численного моделирования различных физических процессов, является сложной и трудоемкой задачей. В статье рассматриваются основные подходы по проектированию подсистемы доступа такой высокопроизводительной вычислительной системы, позволяющие систематизировать и упростить процесс ее разработки. Проводится анализ основных факторов, оказывающих влияние на структуру и состав подсистемы доступа. Приводится пример методики расчета размерности подсистемы доступа.

Ключевые слова: высокопроизводительная вычислительная система; кластер; подсистема доступа; высокопроизводительные вычисления; моделирование.

Для цитирования: Мокшин С.Ю. Общие подходы к проектированию подсистемы доступа высокопроизводительных вычислительных систем. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 41–52. DOI: 10.15514/ISPRAS–2020–32(4)–3

General approaches to the design of the access subsystem of high-performance computing systems

S.Yu. Mokshin, ORCID: 0000-0002-7454-6597 <sumo@rambler.ru>
E. I. Zababakhin All-Russian Scientific Research Institute of Technical Physics,
13, Vasilieva street, Chelyabinsk region, Snezhinsk, 456770, Russia

Abstract. Mathematical modeling allows to create a digital model of a product, so-called digital twin. Such digital twins are quite complex mathematical models, the description of which as well as the calculation of their parameters and characteristics is a very serious computational problem that can be solved only on high-performance computing systems. However, building of a high-performance supercomputing system designed for solving problems of computer simulation of various physical processes is a difficult time-consuming task. The paper discusses basic approaches to design a login nodes subsystem for such high-performance supercomputing system allowing to systematize and simplify the process of its development. An analysis of main factors affecting a structure and a composition of login nodes subsystem is carried out. These factors include a number of potential users of a high-performance computing system; availability of categories of users of the system and their percentage; characteristics of computing tasks solved on the system; hardware platform chosen for system building. An example of a methodology for calculating of the login nodes subsystem size is given.

Keywords: high-performance computing system; cluster; computer simulation; login nodes subsystem; HPC modeling; HPC.

For citation: Mokshin S.Yu. General approaches to the design of the access subsystem of high-performance computing systems. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 41–52 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–3

1.Введение

Любое научное или производственное предприятие в своей деятельности рано или поздно сталкивается с тем, что создание опытных образцов результатов своей деятельности, тестовых или конечных продуктов – это достаточно дорогой и затратный по времени процесс. Причем неважно, какую именно область деятельности затрагивает производственный или научный процесс – авиастроение ли это, машиностроение или атомная энергетика, медицина или фармакология и т.д. И тогда на помощь ученым и производственникам приходит математическое моделирование, которое позволяет создать цифровую модель продукта, так называемый цифровой двойник. Естественно, такие цифровые двойники представляют собой достаточно сложные математические модели, описание которых, а также расчет их параметров и характеристик, представляет собой очень серьезную вычислительную задачу, решаемую на высокопроизводительных вычислительных системах (ВВС). В данной статье мы сделаем попытку раскрыть основные подходы к проектированию одной из функциональных подсистем ВВС – подсистемы доступа, учитывая опыт создания ВВС для открытых исследований в Российском федеральном ядерном центре – Всероссийском научно-исследовательском институте технической физики имени академика Е.И. Забабахина (РФЯЦ-ВНИИТФ).

Любая высокопроизводительная вычислительная система, предназначенная для решения задач численного моделирования, состоит из множества функциональных подсистем, среди которых, как правило, присутствует подсистема доступа ВВС. На протяжении последних 20-25 лет рабочее место инженера или научного сотрудника на любом предприятии представляет собой обычный персональный компьютер (ПЭВМ) с установленной на него операционной системой Microsoft Windows какого-либо поколения. По данным Statcounter GlobalStats [1] в апреле 2020 года 76.52% персональных компьютеров использовали операционные системы (ОС) от компании Microsoft и только 1.61% – операционные системы семейства Linux. В тоже время практически на всех ВВС в качестве операционных систем используются различные сборки ОС Linux [2].

Отсюда и возникает потребность в создании своего рода промежуточного слоя между ВВС и ПЭВМ пользователя ВВС – подсистемы доступа (ПСД), реализующей функции полноценно однородного с ВВС рабочего места, оснащенном средствами разработки, компиляции программ, а также всем необходимым системным и прикладным программным обеспечением, которое обеспечивает функциональное использование ВВС. Фактически ПСД в составе ВВС выполняет функции некоего «мощной» ПЭВМ, на которой ведется разработка и отладка программ, выполняется расчет начальных данных (РНД), проводятся процедуры постобработки и визуализации расчетных данных.

Термин «подсистема доступа» возник в конце 1990-ых годов при разработке ВВС в ядерных центрах РФ. В зарубежных публикациях по соответствующей тематике обычно речь идет об отдельных вычислительных узлах, выполняющих функции доступа к ВВС, и не объединяемых в отдельную подсистему (так называемые login nodes).

В данной статье мы рассмотрим ключевые аспекты, учитываемые при создании подсистемы доступа ВВС на примере исследования, которое было проведено для ВВС «Зубр», созданной в РФЯЦ-ВНИИТФ. [3].

2. Выбор ключевых характеристик ПСД ВВС

Очевидно, что структура и состав ПСД ВВС будут зависеть от нескольких факторов:

- количества потенциальных пользователей ВВС;
- наличия категорий пользователей ВВС и их процентного соотношения;
- характеристик вычислительных задач, решаемых на ВВС;
- аппаратной платформы, выбранной для ВВС.

Рассмотрим подробнее каждый из этих факторов и их влияние на структуру и состав ПСД ВВС.

Естественно, если в организации имеется некоторое количество потенциальных пользователей ВВС, то возникает первое желание создать ПСД, используя количество узлов (серверов) ПСД равное количеству пользователей ВВС. Такой подход даже в случае исключительно хорошего финансового состояния организации будет, безусловно, экономически неэффективен и никоим образом не окупит себя, так как стоимость серверного оборудования всегда намного выше стоимости обычных персональных компьютеров. Поэтому количество узлов ПСД конечно будет пропорционально зависеть от количества пользователей ВВС, но с какими-то поправочными коэффициентами, определяющими эту зависимость. Формулу этой зависимости мы постараемся определить позднее.

Режим работы с ПСД ВВС, как правило, удаленный и подразумевает, что пользователь может обращаться к узлу ПСД по протоколам SSH [3], X [4], RFB (VNC) [5], FTP [6], используя различные программные инструменты, такие как PuTTY [7], OpenText Exceed [8], Xming [9], X2Go [10], TurboVNC [11], TightVNC [12] и многие другие. В данной статье мы не будем останавливаться на некоторых других протоколах, используемых для удаленного доступа, которые разработаны для использования с коммерческим ПО наподобие VMware [13] (протоколы Blast, PCoIP, Microsoft RDP). Подобные продукты обладают рядом очень полезных характеристик, но их исходные коды являются закрытыми, а стоимость самого ПО достаточно велика.

Подсистема доступа ВВС должна быть подключена к некоторой локальной вычислительной сети (ЛВС) предприятия. Параметры такого подключения тоже определяются исходя из количества потенциальных пользователей ВВС, но, естественно, учитывают финансовые возможности предприятия. Как правило, для подключения ПСД ВВС к ЛВС предприятия используются сети на базе Ethernet со скоростями от 1 Гбит/с до 10 Гбит/с. Стоимость других решений пока еще слишком велика для массового использования и может быть сравнима со стоимостью самой ПСД ВВС. Имеющийся у автора статьи многолетний опыт использования ПСД ВВС показывает, что на данный момент один пользователь ВВС в среднем утилизирует канал примерно в 50 Мбит/с, причем за 10 лет использования ВВС [14] ширина этого канала выросла примерно в 5 раз. Данный рост обуславливается спецификой использования удаленного графического стола, ростом графических возможностей программных оболочек ОС, программ предварительной и постобработки результатов расчетов, количеством расчетов и объемом насчитанных данных. Поэтому это значение в 50 Мбит/с можно принять в данное время в качестве минимальной базовой пропускной способности канала сети доступа на одного пользователя ВВС – E_{min}, которое попробуем использовать в дальнейшем при расчетах размерности ПСД.

Опыт использования ВВС показывает, что пользователь ВВС – это такое достаточно общее понятие, объединяющее в себе и высококвалифицированного разработчика прикладных программ, программиста, математика, и потребителя результатов расчетов – физика, инженера, экономиста и др. Конечно, потребности в составе ПО, в вычислительных ресурсах, а также интенсивность их работы с ВВС у всех этих категорий пользователей абсолютно разные. Кто-то из пользователей ВВС умеет и может заниматься созданием математических моделей исследуемых объектов, генерацией расчетных сеток для трехмерных объектов и их декомпозицией, распараллеливанием алгоритмов физико-математического моделирования различных процессов, написанием программ и их адаптацией под оборудование ВВС и т.п.

(назовем такую категорию пользователей категорией А). А кто-то использует готовое прикладное программное обеспечение для своих расчетов, не вдаваясь в тонкости создания программ и написания алгоритмов (категория Б). При проектировании ПСД важно знать, хотя бы приблизительно процентное соотношение между категориями А и Б, либо их абсолютные значения, которые мы обозначим как n_A и $n_Б$ соответственно.

Как правило, потребности в вычислительных ресурсах, как-то количество ядер CPU – N_{cpu}, количество оперативной памяти, объем необходимых файловых ресурсов для категории А в 2-3 раза больше, чем для категории Б. Такое соотношение обусловлено прежде всего необходимостью компиляции (часто параллельной), отладки программ (как правило тоже параллельной), визуализации насчитанных данных, наличии процедур расчета начальных данных. Напоминаем, что речь идет именно о подсистеме доступа, на которой, как правило, не производятся сами расчеты; на вычислительном поле ВВС это соотношение потребностей в вычислительных ресурсах может быть совсем иным.

Важно отметить еще один момент: как правило, ВВС – это многопользовательские системы. И часто, помимо указанных категорий пользователей А и Б, в них формируются и устойчиво существуют так называемые тематические группы пользователей. Объединение пользователей в такие группы подразумевает решение ими общих для конкретной тематической группы вычислительных задач и, следовательно, общей потребности в определенном количестве вычислительных ресурсов и составе прикладного и системного ПО. В том числе, введение таких тематических групп в ВВС проводится с целью разграничения доступа к вычислительным и файловым ресурсам ВВС.

Безусловно, на структуру ПСД ВВС влияют и характеристики вычислительных задач, предполагаемых к решению на ВВС. Не секрет, что большинство потребителей ВВС старается использовать для решения своих конкретных математических, конструкторских или иных задач готовое прикладное программное обеспечение, в которое заложены существующие и известные десятилетиями математические алгоритмы. Это вполне понятно и оправдано, так как придумывать самому алгоритмы и писать программы достаточно сложно, требует наличия высокой квалификации у разработчика; фактически это малоэффективно и дорого.

Исходя из этого, при проектировании ПСД ВВС и ВВС, требуется провести анализ необходимого к использованию на ВВС прикладного и системного ПО, определить все зависимости такого ПО от сторонних оптимизирующих библиотек (BLAS [15], SCALAPACK [16], PLASMA [17], MKL [18] и др.), его возможности в плане использования тех или иных средств распараллеливания (pthreads [19], OpenMP [20], MPI [21], Linda [22] и т.д.), возможности в плане использования на специализированных ускорителях вычислений, например программируемых логических матрицах, либо графических ускорителях и т.п.

Кроме того, важно учитывать характеристики ввода-вывода прикладного ПО, касающиеся его работы с системой хранения ВВС (формат выводимых данных, возможность параллельного чтения и записи данных, количество операций ввода-вывода в секунду, объемы формируемых файлов, частота обращений к ним, наличие контрольных операций с файлами и т.п.).

Только на основе такого достаточно глубокого и многофакторного анализа, делается вывод о ключевых параметрах ВВС – архитектуре процессоров ВВС, необходимости в ускорителях, архитектуре высокопроизводительной коммуникационной среды (Ethernet, Infiniband, Omni-Path и т.д.), количестве и типе необходимой оперативной памяти, объемах и типах системы хранения ВВС, операционной системе ВВС и составе системного и прикладного ПО. И уже после этого, на основе разработанной общей архитектуры ВВС, выбранной для ВВС аппаратной платформы, формируется вывод и об общей архитектуре ПСД ВВС. В рамках данной статьи мы не будем описывать процедуру выбора аппаратной платформы для ВВС,

отметим лишь что она, как правило, строится из расчета требуемой реальной или пиковой производительности ВВС.

Учитывая все вышеперечисленные факторы, можно сделать вывод, что ПСД ВВС должна полноценно реализовывать все необходимые для её пользователя функции – разработку приложений, их компиляцию, отладку, постобработку результатов расчетов, но с обязательным учетом всех заложенных в общую архитектуру ВВС особенностей – её аппаратной конфигурации, типа и версии операционной системы, коммуникационного ПО. Такая жесткая взаимозависимость ПСД от ВВС особенно справедлива для ВВС, где присутствует пользователь-разработчик прикладного ПО (категория А) и в целом обусловлена тем, что любое прикладное ПО, во-первых, имеет определенные программные зависимости, и просто может не работать, например, на ОС, отличной от типа и версии ОС, на которой было собрано это ПО (даже в независимости от вида сборки ПО – динамической или статической). Те же проблемы с запуском и корректной работой прикладного ПО наблюдаются так же при отсутствии другого необходимого системного окружения, различных математических и оптимизирующих библиотек, компиляторов, библиотек для задания формата данных и т.д.

Во-вторых, для того, чтобы максимально эффективно использовать вычислительные ресурсы ВВС, прикладное ПО должно быть оптимизировано под архитектуру ВВС (процессоров, ускорителей, коммуникационной среды и т.д.) По опыту автора публикации, оптимизация прикладного ПО – процесс трудоемкий и многоэтапный, и подразумевает прямой и интерактивный режим работы с оборудованием ВВС. На первом этапе производится исследование архитектуры процессора или ускорителя, изучения свойств и характеристик оперативной памяти и памяти устройств. На этом этапе пользователь ВВС должен иметь возможность взаимодействия с оборудованием ВВС минуя промежуточный программный слой в виде набора ПО системы управления ВВС или коммуникационных библиотек, которые всегда присутствуют на вычислительном поле ВВС. Именно на этом этапе функцию рабочего места должна выполнять ПСД в ВВС, реализуя функции ПЭВМ с прямым доступом к оборудованию.

На втором этапе обычно исследуются механизмы распараллеливания ПО, прикладное ПО переписывается при необходимости и с учетом тех возможностей, которые дают коммуникационная среда и система управления ВВС. На третьем этапе, когда программа каким-то образом уже оптимизирована, её дальнейшая отладка и адаптация непосредственно происходит на вычислительном поле ВВС.

Таким образом, можно сформулировать ключевое правило построения ПСД ВВС – архитектура процессоров и выбранные типы ускорителей, а также состав системного и прикладного ПО должны быть, как минимум, аналогичными используемым в ВВС в целом. Многолетний опыт вычислительных расчетов в РФЯЦ-ВНИИТФ показывает, что остальные характеристики ПСД достаточно независимы от ВВС и определяются ранее названными факторами (количества пользователей, наличия их категорий, характеристик прикладного ПО для ПСД).

3. Общая структура ПСД ВВС

Поскольку подсистема доступа является неотъемлемой частью ВВС, для понимания того, каким образом должна выглядеть общая структура ПСД, надо рассмотреть структуру типовой ВВС (рисунок 1).

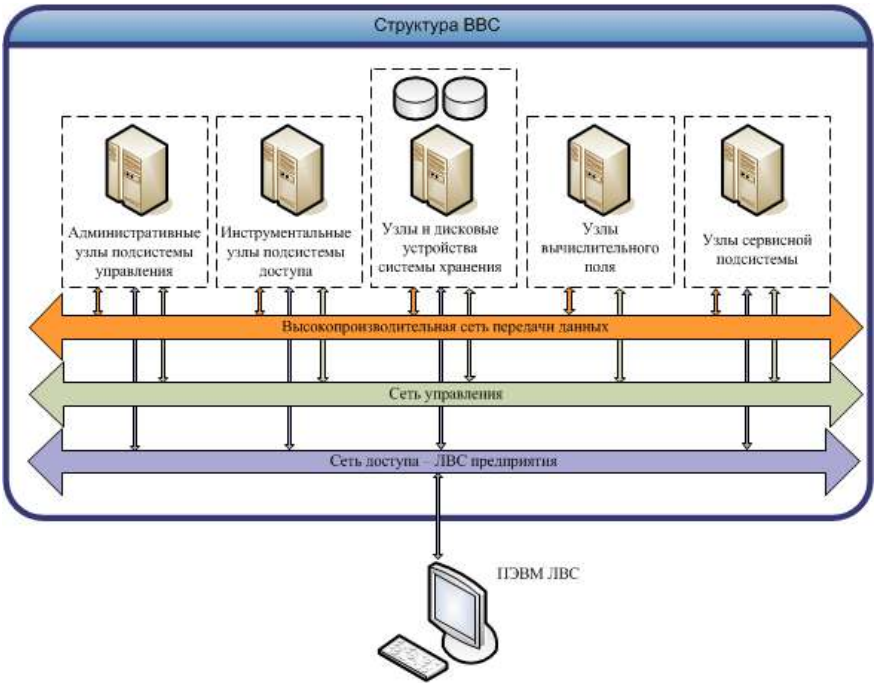


Рис. 1. Общая структура ВВС
Fig.1. High-performance supercomputer structure

С учетом сложившейся общемировой практики построения ВВС, структура ВВС может выглядеть так, как представлено на рис. 1. Она состоит из подсистемы доступа, системы хранения, подсистемы управления, сервисной подсистемы и вычислительного поля, объединенных высокопроизводительной коммуникационной средой (высокопроизводительной сетью передачи данных) и сетью управления. Кроме того, обычно подсистема управления, сервисная подсистема и подсистема доступа объединены с ЛВС предприятия (организации) через сеть доступа.

Предлагаемая к рассмотрению общая структура ПСД ВВС представлена на рис. 2. ПСД ВВС состоит из некоторого количества инструментальных узлов (ИУ) F1…FN, объединенных сетью управления (СУ), высокопроизводительной коммуникационной средой (ВКС) и сетью доступа, которая в свою очередь должна быть сопряжена с ЛВС предприятия (организации). Термин «узел», в данном контексте используется для единицы вычислительного оборудования и отличается от термина «сервер», так как физически современный сервер может состоять из нескольких вычислительных модулей – узлов, объединенных, в одном корпусе с одним, например, блоком питания. Таких модулей, в серверах на данный момент времени, например, может быть от двух до восьми в корпусах размерности 1U и 2U соответственно (Twin server, Twin^2 server, Twin^3 server).

В зависимости от требуемого количества инструментальных серверов, потребность в которых мы попробуем рассчитать чуть позже, ПСД может быть построена с использованием механизмов бездисковой или дисковой загрузки ОС. Бездисковая загрузка используется для удобства администрирования ПСД при большом количестве инструментальных узлов (как правило, от 10 и более), и позволяет хранить образ системы на одном системном узле (СУ) ПСД, загружая его на инструментальные узлы с помощью протоколов и инструментов PXE, TFTP, DHCP, NFS.

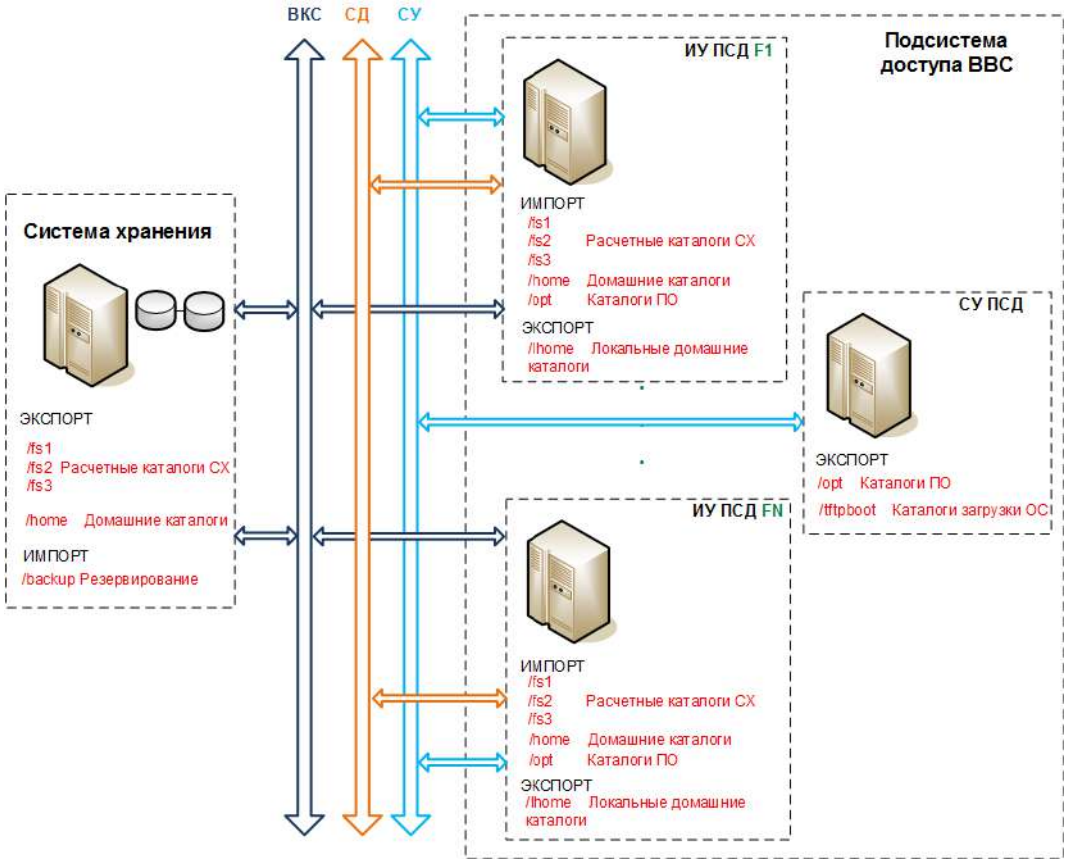


Рис. 2. Общая структура ПСД ВВС
Fig.2. High performance supercomputer access subsystem structure

На инструментальных узлах создаются каталоги/точки монтирования сетевых файловых систем – каталоги расчетных данных и домашние каталоги. Домашние каталоги реализуют функцию личного хранилища данных, а также точки входа на ВВС, с помощью которой можно задавать системное окружение, глобальные настройки для конкретного пользователя на всем пространстве ВВС. Как правило, домашний каталог содержит настройки окружения в конфигурационных файлах командной оболочки (например, для широко используемой Bourne-Again Shell (BASH) это файлы .bashrc и .bash_profile), настройки рабочего стола, файловых менеджеров и редакторов, компиляторов и отладчиков, исходные тексты программ в случае их разработки на ПСД. Существует два подхода к организации домашних каталогов:

- их можно размещать на локальных дисках каждого инструментального узла (например, используя массивы RAID 1, RAID 10, RAID 5, RAID 6),
- их можно размещать на файловой системе системы хранения ВВС (СХ) и экспортировать по протоколам сетевой файловой системы.

Первый подход можно использовать для небольших ВВС, так как необходимо помнить, что точка входа пользователя на любой узел ВВС должна быть одинаковой на всем пространстве ВВС (для единообразия настроек окружения). В этом случае обычно на каждом инструментальном узле запускается NFS сервер, обеспечивающий экспорт домашних каталогов на вычислительное поле. В случае если ВВС содержит в своем составе множество узлов (клиентов) вычислительного поля (как правило, более 200), такая схема становится

неудобной и неспособной обеспечить требуемую функциональность из-за слишком большого количества клиентских запросов. Собственно, эти ограничения накладываются самим протоколом NFS и не зависят от версии этого протокола.

Гораздо более удобной является схема размещения домашних каталогов на параллельной файловой системе, способной обеспечивать множественные клиентские запросы (например, Lustre). Кроме того, домашние каталоги могут быть построены не на обычных серверных RAID системах (с использованием дисковых серверов), а на дисковых подсистемах профессионального уровня, что обеспечивает их сохранность и доступность, а также упрощает функции масштабирования и резервирования с использованием достаточно больших объемов дисковых подсистем.

Аналогичный подход используется и при организации расчетных каталогов; для больших ВВС целесообразно использовать несколько файловых систем в качестве каталогов расчетов, это облегчает их обслуживание, снижает загрузку на конкретную файловую систему при интенсивном ее использовании расчетными программами. В этом случае на инструментальных узлах ПСД создается несколько точек монтирования для сетевых файловых систем.

Использование сетевых файловых систем для домашних каталогов, несомненно, имеет огромные плюсы в плане унификации, администрирования, обеспечения сохранности данных и т.д. Но есть и некоторые минусы. Таковым, например, является неудобства, возникающие у разработчиков программ при проведении процедуры параллельной компиляции программ. Такая потребность возникает часто для больших программных продуктов, где количество файлов исходных текстов может достигать нескольких тысяч и более. В этом случае скорость работы с файловой системой, на которой одновременно несколько пользователей может совершать подобную процедуру, становится неприемлемой.

Кроме того, при параллельной компиляции сетевая файловая система должна обеспечивать функцию временной блокировки файла (flock); не каждая файловая система может обеспечить такой режим без существенного замедления работы. В этом случае, помимо обычных домашних каталогов, обеспечивающих функцию точки входа, автором статьи была предложена идея использовать специализированные локальные домашние каталоги, обеспечивающие максимальную скорость работы с локальными файловыми системами.

Современные технологии позволяют реализовывать локальные RAID массивы из быстрых SSD дисков (различного типа и форм-фактора: SATA, M2, U2, Intel Optane и т.д.), обеспечивающие скорости чтения до 5000 Мбит/с и записи до 3500 Мбит/с, объемы до 10 ТБ и время наработки на отказ до 2 млн. часов. Конечно, в этом случае должно быть предусмотрено резервирование всей ценной информации, хранимой в локальных домашних каталогах. Такой локальный домашний каталог доступен только в пределах каждого инструментального узла ПСД, экспорт данных с его файловой системы на другие узлы ВВС обычно не предусматривается.

Для унификации общедоступного системного и прикладного программного обеспечения целесообразно на системном узле ПСД создавать каталог, который будет содержать весь набор неизменяемого прикладного и системного ПО: расчетных программ, библиотек оптимизации, компиляторов, отладчиков и т.д. Данный каталог обычно экспортируется по протоколу NFS на все инструментальные узлы ПСД, его содержимое синхронизируется встроенными в ОС инструментами типа Rsync на системные узлы вычислительного поля, экспортирующие в свою очередь по протоколу NFS эти данные на вычислительные узлы (пример – каталог /opt на рис. 2).

Таким образом, для создания ПСД в качестве вычислительного оборудования должно выбираться оборудование, учитывающее особенности организации технологии работы

пользователей на ПСД. Как минимум, требования к вычислительному оборудованию могут быть такими:

- серверная платформа должна позволять использовать процессоры однородной архитектуры с процессорами, используемыми на вычислительном поле;
- серверная платформа должна иметь не менее двух портов Ethernet для подключения к сети управления и сети доступа;
- серверная платформа должна позволять устанавливать адаптеры высокопроизводительных сетей (например, Infiniband), либо иметь такие интегрированные порты;
- центральный процессор, выбираемый для использования на инструментальном узле ПСД, должен иметь достаточно высокую тактовую частоту ядра (более высокую или сравнимую с частотой ядра процессора обычного ПЭВМ в ЛВС);
- количество памяти на инструментальном узле должно выбираться из соотношения:

$$M = M_0 * (n_{\text{А}} + kn_{\text{Б}}),$$

где

$n_{\text{А}}$ – количество пользователей категории А в одной тематической группе;

$n_{\text{Б}}$ – количество пользователей категории Б в одной тематической группе;

k – условный коэффициент сложности работ, определяемый необходимостью решения на ПСД некоторого количества N дополнительных задач (РНД, задач визуализации и т.д., $k = 1..N$);

M_0 – величина, соотносимая или равная размеру памяти на используемых в ЛВС предприятия (организации) ПЭВМ.

В зависимости от наличия в ВВС пользователей-разработчиков, их потребностей, требования к оборудованию, используемому в качестве инструментальных узлов ПСД, могут быть расширены до следующих:

- серверная платформа должна иметь возможность установки полноразмерного графического ускорителя или видеокарты (либо нескольких единиц этого оборудования) для работы с мощными графическими приложениями;
- серверная платформа должна иметь возможность установки ускорителей той же архитектуры, которые используются (если используются) на вычислительном поле ВВС для возможности разработки и интерактивной отладки прикладных программ, адаптированных под архитектуру ускорителей;
- серверная платформа должна иметь возможность установки RAID контроллера, поддерживающего RAID 1, RAID 10, RAID 5, RAID 6 для организации локального файлового пространства, в том числе, если требуется и на SSD накопителях разного типа и форм-фактора.

Из всего вышеуказанного набора требований можно сделать вывод, что серверная платформа, выбираемая для использования в качестве инструментальных узлов ПСД, в некоторых случаях будет иметь достаточно высокую стоимость, поэтому очень важно на этапе проектирования примерно определить количество инструментальных узлов, удовлетворяющее требованиям заказчика ВВС.

4. Примерный расчет размерности ПСД ВВС

При проектировании ВВС в стоимость проекта закладывается стоимость всех создаваемых подсистем ВВС, поэтому естественно важно каким-то образом определить количество единиц оборудования, необходимых для создания ПСД ВВС. Расчет размерности (количества инструментальных узлов) ПСД должен строиться на базе уже выбранной для ВВС аппаратной платформы (модели процессора, модели серверов, типа коммуникационной

среды). Исходя из всего вышеизложенного, можно предложить несколько простейших зависимостей для расчета размерности ПСД ВВС.

Во-первых, учитывая наличие нескольких категорий пользователей, примерное количество инструментальных узлов может быть рассчитано по формуле:

$$F_1 \approx \sum_{i=1}^{G} \frac{(n_{\text{А}}+kn_{\text{Б}})_i}{N_{fcpu}} N_{cpu},$$

где

F_1 – минимальное количество инструментальных узлов в составе ПСД;

G – общее количество тематических групп пользователей;

$n_{\text{А}}$ – количество пользователей категории А в одной тематической группе;

$n_{\text{Б}}$ – количество пользователей категории Б в одной тематической группе;

k – условный коэффициент сложности работ, определяемый необходимостью решения на ПСД некоторого количества N дополнительных задач (РНД, задач визуализации и т.д., $k = 1..N$);

N_{cpu} – минимально достаточное количество вычислительных ядер на инструментальном узле для сеанса одного пользователя ПСД;

N_{fcpu} – максимально доступное количество вычислительных ядер на инструментальном узле ПСД.

Значение N_{fcpu} рассчитывается как произведение физически возможного количества процессоров в вычислительном узле (сервере) и количества ядер одного процессора. Значение N_{cpu} выбирается эмпирическим путем. Например, его можно определить равным количеству ядер процессора на ПЭВМ в составе сети предприятия. Фактически значение N_{cpu} определяет комфортные условия работы пользователя в сеансе на ПСД.

Во-вторых, мы должны выбрать такое количество инструментальных узлов в составе ПСД, которое бы позволяло обеспечить минимальную базовую пропускную способность канала сети доступа для одного пользователя ВВС:

$$F_2 \approx \sum_{i=1}^{G} \frac{(n_A+kn_{\text{Б}})_i}{E} E_{min},$$

где

F_2– минимальное количество инструментальных узлов в составе ПСД;

G – общее количество тематических групп пользователей;

$n_{\text{А}}$ – количество пользователей категории А в одной тематической группе;

$n_{\text{Б}}$ – количество пользователей категории Б в одной тематической группе;

k – условный коэффициент сложности работ, определяемый необходимостью решения на ПСД некоторого количества N дополнительных задач (РНД, задач визуализации и т.д., $k = 1..N$);

E_{min} – минимальная базовая пропускная способность канала сети доступа для одного пользователя ВВС;

E – пропускная способность канала сети доступа одного инструментального узла ПСД (физическая скорость канала).

Таким образом, количество необходимых для ПСД узлов можно определить из этих двух приблизительных значений, выбрав максимальное:

$$F \approx \max\{F_1, F_2\}.$$

Безусловно, такая методика расчета является приблизительной и служит для определения стоимости оборудования ПСД при создании проекта ВВС.

5. Заключение

Рассмотренные в данной статье подходы по построению подсистемы доступа высокопроизводительных вычислительных систем были использованы автором публикации

при проектировании нескольких ВВС в РФЯЦ-ВНИИТФ, в том числе по ряду контрактных работ по созданию ВВС для научных организаций РФ. Расчет размерности ПСД по приведенным в статье формулам позволил РФЯЦ-ВНИИТФ упростить проектирование ВВС для нужд своих заказчиков. Автор выражает надежду, что изложенные в статье подходы могут быть полезны и другим специалистам, занимающимся созданием ВВС.

Список литературы / References

[1]. GlobalStats. Available at: https://gs.statcounter.com/os-market-share/desktop/worldwide, accessed 01.06.2020.
[2]. Top500 Supercomputers. Available at: https://www.top500.org/, accessed 01.06.2020.
[3]. Глазырин А.И., Мокшин С.Ю. Суперкомпьютер «Зубр» средней производительности. Труды Всероссийской конференции «Информационные технологии в оборонно-промышленном комплексе, 2016 г. / Glazyrin A.I., Mokshin S.Yu. Supercomputer «Zubr» of average performance. In Proc. of the All-Russian conference "Information technologies in the military-industrial complex, 2016 (in Russian).
[4]. The Secure Shell (SSH) Protocol Architecture. Available at: https://tools.ietf.org/html/rfc4251, accessed 01.06.2020.
[5]. XWindow System Protocol. Available at: ftp://ftp.x.org/pub/X11R7.0/doc/PDF/proto.pdf, accessed 01.06.2020.
[6]. The Remote Framebuffer Protocol. Available at: http://www.realvnc.com/docs/rfbproto.pdf, accessed 01.06.2020.
[7]. File Transfer Protocol. Available at: https://tools.ietf.org/html/rfc959, accessed 01.06.2020.
[8]. PuTTY. Available at: https://www.putty.org/, accessed 01.06.2020.
[9]. OpenText Exceed. Available at: https://www.opentext.com/products-and-solutions/products/specialty-technologies/connectivity/exceed, accessed 01.06.2020.
[10]. Xming X Server. Available at: http://www.straightrunning.com/XmingNotes/, accessed 01.06.2020.
[11]. X2Go. Available at: https://wiki.x2go.org/doku.php, accessed 19.05.2020.
[12]. TurboVNC. Available at: https://www.turbovnc.org/, accessed 01.06.2020.
[13]. TightVNC. Available at: https://www.tightvnc.com/, accessed 01.06.2020.
[14]. VMware Product. Available at: https://www.vmware.com/, accessed 24.05.2020.
[15]. Basic Linear Algebra Subprograms. Available at: https://www.netlib.org/blas/, accessed 24.05.2020.
[16]. Scalable Linear Algebra Package. Available at: https://www.netlib.org/scalapack/, accessed 01.06.2020.
[17]. Parallel Linear Algebra Software for Multicore Architectures. Available at: https://bitbucket.org/icl/plasma/src/default/, accessed 01.06.2020.
[18]. Intel® Math Kernel Library. Available at: https://software.intel.com/content/www/us/en/develop/tools/math-kernel-library.html, accessed 01.06.2020.
[19]. The Open Group Base Specifications Issue 6, pthread.h. Available at: http://www.opengroup.org/onlinepubs/007904975/basedefs/pthread.h.html, accessed 01.06.2020.
[20]. The OpenMP API specification for parallel programming. Available at: https://www.openmp.org/, accessed 01.06.2020.
[21]. MPI: The Message Passing Interface. Available at: http://parallel.ru/tech/tech_dev/mpi.html, accessed 01.06.2020.
[22]. Tutorial on Parallel Programming with Linda. Available at: http://www.cse.chalmers.se/edu/course/TDA384_LP1/files/lectures/semantic-linda-biglecture.pdf, accessed 01.06.2020.

Информация об авторах / Information about authors

Сергей Юрьевич МОКШИН – начальник отдела. Сфера научных интересов: проектирование вычислительных систем, разработка функциональных подсистем для высокопроизводительных вычислительных систем, разработка операционных систем, методы и средства защиты информации.

Sergey Yureivich MOKSHIN – Head of the Department. Research interests: design of supercomputer systems, development of functional subsystems for high performance supercomputing systems, operating systems development, methods and means for protecting information.

DOI: 10.15514/ISPRAS–2020–32(4)–4

Модель производительности графического конвейера для однопроходной схемы рендеринга динамических трехмерных сцен

В.И. Гонахчян, ORCID: 0000-0002-4348-8443 <pusheax@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В работе рассматривается однопроходная схема рендеринга динамических трехмерных сцен с использованием современных видеокарт (GPU) и графических интерфейсов. В рамках этой схемы используются следующие методы и техники: отсечение объектов с использованием методов пространственной декомпозиции и индексирования, аппаратные проверки видимости, фрагментация и кэширование командных буферов. Для выполнения этих методов требуются значительные вычислительные ресурсы, а объем работы на этапах графического конвейера зависит от их результатов. Поэтому важно сбалансированное использование ресурсов при конвейерной обработке и передаче графических данных. Предлагается модель производительности графического конвейера применительно к задачам рендеринга динамических трехмерных сцен, позволяющая оценивать требуемые ресурсы в зависимости от применяемых базовых методов и характеристик отображаемой сцены. В отличие от существующих методов и моделей, предлагаемая модель позволяет рассчитать затраты на составление буферов команд с использованием различных техник записи, затраты на отправку, выполнение, получение результатов аппаратных проверок видимости. Выводятся формулы для расчета временных затрат в зависимости от количества проверок видимости. Предлагается метод оценки количества аппаратных проверок видимости для эффективного выполнения рендеринга динамических сцен. Проводятся вычислительные эксперименты, показывающие релевантность предложенной модели и эффективность разработанного метода при отображении больших динамических сцен.

Ключевые слова: рендеринг; составление командных буферов; удаление невидимых поверхностей; аппаратные проверки видимости; модель производительности графического конвейера

Для цитирования: Гонахчян В.И. Модель производительности графического конвейера для однопроходной схемы рендеринга динамических трехмерных сцен. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 53–72. DOI: 10.15514/ISPRAS–2020–32(4)–4.

Performance model of graphics pipeline for single-pass dynamic 3d scene rendering scheme

V.I. Gonakhchyan, ORCID: 0000-0002-4348-8443 <pusheax@ispras.ru>
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. The paper considers a single-pass scheme for rendering dynamic 3D scenes using modern GPUs and graphics interfaces. The following methods and techniques are used with this scheme: clipping of objects using spatial decomposition and indexing methods, hardware occlusion queries, fragmentation and caching of command buffers. These methods require significant computational resources to execute, and the amount of work in the stages of the graphics pipeline depends on their results. Therefore, a balanced use of resources when transferring graphics data and executing commands is important. A performance model of the graphics pipeline

is proposed, which makes it possible to estimate the required resources depending on the applied base methods and characteristics of the displayed scene. Unlike existing methods and models, the proposed model allows to calculate the costs of composing command buffers using various recording techniques, the costs of sending, executing, and receiving the results of hardware occlusion queries. Formulas are derived to calculate frame rendering time depending on the number of occlusion queries. A method is proposed to estimate the number of occlusion queries for efficient rendering of dynamic scenes. Computational experiments are carried out to show the relevance of the proposed model and the effectiveness of the developed method when displaying large dynamic scenes. Section 1 provides an overview of related work as well as general purposes of given paper and its structure. Section 2 describes the proposed performance model and method used to calculate the number of occlusion queries for efficient rendering of dynamic scenes. Section 3 presents the performance analysis, which contains derived and measured rendering time when rendering scenes and employing frustum culling, occlusion queries. Section 4 summarizes the main conclusions.

Keywords: 3d rendering; command buffer recording; occlusion culling; hardware occlusion queries; performance model of graphics pipeline.

For citation: Gonakhchian V.I. Performance model of graphics pipeline for single-pass dynamic 3d scene rendering scheme. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 53–72 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–4

1. Введение

Рендеринг трехмерных сцен является одной из ключевых задач компьютерной графики и широко применяется в таких предметных областях, как научная визуализация, автоматизация проектирования, инженерии и производства (CAD/CAM/CAE), компьютерные игры и анимация, виртуальная и дополненная реальность. В связи с перманентным ростом сложности сцен повышаются требования и к эффективности программных и аппаратных средств рендеринга. Несмотря на обширные исследования в этой области и большое количество опубликованных работ, задачам рендеринга больших динамических сцен уделяется относительно мало внимания. Вместе с тем, класс приложений, оперирующих с подобными сценами, чрезвычайно широк, что определяет актуальность темы исследований. Особую важность тема приобретает в связи со стремительным развитием технологий информационного моделирования зданий и сооружений (BIM), предусматривающих, в частности, интерактивные графические средства динамического моделирования процессов возведения сложных строительных объектов и реализации масштабных инфраструктурных программ.

В работе рассматриваются задачи отображения динамических трехмерных сцен с использованием современных видеокарт (GPU) и графических интерфейсов к ним (OpenGL, DirectX, Vulkan). Видеокарты обеспечивают высокоэффективную поточно-параллельную обработку графических элементов (треугольников, пикселей) для достижения требуемой частоты генерации изображений. Графические интерфейсы позволяют задать описание трехмерной сцены и передать его на видеокарту, установить программы для геометрических преобразований, проецирования треугольников на экранную плоскость, расчета функций освещения, а также сгенерировать итоговые изображения.

Однако при отображении сложных сцен, содержащих большое количество графических элементов и предполагающих использование сложных моделей материалов и освещения, видеокарты часто не справляются с большим объемом вычислений. Подобные ограничения делают невозможными разработку и применение интерактивных графических приложений, предполагающих генерацию и вывод изображений с частотой, соответствующей скорости отклика на пользовательские события.

Одним из традиционных подходов к повышению производительности рендеринга является уменьшение числа растеризуемых графических элементов за счет предварительного анализа и удаления невидимых поверхностей. Распространенные варианты методов удаления невидимых поверхностей описаны в работах [1-5]. В методах с использованием

потенциально видимых множеств объектов, порталов, расширенных проекций (Potentially visible sets, Portals and mirrors, Extended projections) выполняется предварительный анализ видимости элементов из разных пространственных ячеек. При изменении положения камеры и смене занимаемой ячейки уточняется множество видимых элементов. Данные методы эффективны для статических сцен, в которых статус видимости объектов не меняется в зависимости от происходящих в сцене событий и может быть вычислен заранее.

Метод иерархического буфера глубины (Hierarchical z-buffer) [6], метод исключения на основе бинарного пространственного разбиения (BSP tree culling) [7] отбирают объекты сцены, невидимые из заданного положения камеры, с помощью специальных пространственных структур данных. Проверки видимости, выполняясь на центральном процессоре (CPU), позволяют составить буфер команд только для видимых объектов и, тем самым, уменьшить нагрузку на видеокарту. Ограниченные вычислительные ресурсы CPU являются наиболее узким местом для подобных реализаций, обычно проявляющимся при отображении больших сцен.

Методы, описанные в работах [8-10], выполняют проверки видимости с учетом имеющихся в сцене больших объектов-преград (large occluders). На этапе предобработки подобные преграды, если присутствуют в сцене, выявляются. На этапе отображения определяется статус перекрытий объектов, для чего они сортируются в порядке удаленности от камеры. Методы имеют отмеченные ограничения, а анализ видимости объектов с учетом множественных преград (occluder fusion) может требовать значительных вычислительных ресурсов.

Одним из общих подходов к ускорению методов удаления невидимых элементов является использование структур пространственной декомпозиции сцены, таких как окто-деревья и k-d-деревья [11-12]. Для этих же целей используются иерархии ограничивающих объемов BVH (Bounding Volume Hierarchies), основанные на произвольно ориентированных и ориентированных вдоль координатных осей параллелепипедах OBB (Oriented Bounding Box), AABB (Axis Aligned Bounding Box), а также на многогранниках с фиксированным количеством ориентаций граней k-DOP. Структуры декомпозиции сцены и объектов, выполняющие функции пространственных индексов, позволяют существенно сократить объем вычислений за счет быстрого вынесения отрицательного вердикта относительно видимости объектов, что достигается за счет недорогих, иерархически организованных проверок видимости ограничивающих их пространственных ячеек или объемов.

Важным достоинством методов пространственной декомпозиции и индексирования является применимость к динамическим сценам, хотя в этом случае необходимо эффективно обновлять соответствующие пространственные структуры при наступлении соответствующих изменений в сцене. Возможные способы решения этой проблемы предложены в работах [13-14]. Метод временного ограничивающего объема (Temporal Bounding Volume) вычисляет результирующий параллелепипед, ограничивающий объект с учетом всех его положений на траектории движения, и подготавливает необходимые структуры для эффективного рендеринга сцены с удалением невидимых элементов. Метод успешно применяется для сцен с детерминированным характером динамики, однако имеет ограничения для сцен, в которых события происходят случайно или моделируются специальными условиями.

В современных видеокартах реализован функционал для выполнения аппаратных проверок видимости (hardware occlusion queries) [15]. В ряде случаев он позволяет повысить эффективность отображения сцен, однако регулярное использование аппаратных проверок может приводить к деградации производительности, принимая во внимание дополнительные расходы на подготовку и осуществление самих проверок. Поэтому необходимыми становятся оценки эффективности применения аппаратных проверок с учетом вычислительных затрат и изменений в сцене. В частности, такие оценки могут применяться для отложенных проверок

видимости для элементов структур пространственной декомпозиции, описанных в работах [16-17].

Важными факторами, влияющими на эффективность процесса рендеринга динамических сцен, являются способы составления и отправки командных буферов [19]. Как правило, описание объектов сцены формируется в виде буферов команд рендеринга, которые записываются в основную память один раз и отправляются на видеокарту по шине. Однако при отображении динамических сцен и использовании проверок видимости требуется повторная запись, которая может занимать продолжительное время. При незначительных локальных изменениях на каждом временном шаге моделирования динамической сцены можно сократить затраты на составление и отправку буферов за счет их фрагментации и кэширования в основной памяти. В этом случае требуется обновление только тех буферов, объекты которых подверглись изменениям на текущем временном шаге.

Таким образом, для эффективного рендеринга больших динамических сцен с использованием современных видеокарт перспективным представляется применение следующих методов и техник:

- удаление невидимых объектов с использованием методов пространственной декомпозиции и индексирования;
- аппаратные проверки видимости с учетом временной когерентности статуса видимости;
- фрагментация и кэширование командных буферов с учетом локальных изменений в сцене.

Поскольку данные методы выполняются одновременно в вычислительной системе, важно сбалансированное использование ее ресурсов при конвейерной обработке и передаче графических данных. В частности, должна быть сбалансирована загрузка CPU и GPU процессоров. Вместе с тем, вариативность в количестве и сложности индивидуальных объектов сцены, характере и интенсивности динамики делает подобную балансировку крайне сложной или даже невозможной при фиксировании конкретных методов и техник. Обеспечить высокую производительность вычислительной системы на широком классе задач рендеринга больших динамических сцен представляется возможным в результате адаптивного управления графическим конвейером, предусматривающего выбор и настройку альтернативных базовых методов и техник с учетом доступных ресурсов системы и особенностей отображаемой сцены на конкретном интервале модельного времени в конкретной пространственной области. Требуется разработать модель производительности графического конвейера для оценки затрачиваемых ресурсов при выполнении рендеринга динамических сцен с использованием описанных методов.

Рассмотрим существующие работы, применяющие оценки вычислительных затрат в процессе рендеринга. В работе [20] предложен адаптивный метод рендеринга с заданной частотой кадров. Отображение каждого объекта выполняется с подходящим уровнем детализации (level of detail) для поддержания целевой частоты кадров. Как правило, уровень детализации выбирается на основе расстояния до объекта и размера объекта, но это не всегда дает равномерную частоту кадров. Частота кадров может сильно меняться, когда появляются новые объекты или изменяются их уровни детализации. Для устранения этого недостатка можно адаптивно изменять пороговые размеры для вывода уровней детализации. Эта техника хорошо справляется со сценами, в которых видимость объектов слабо меняется при переходе к следующему кадру. В некоторых сценах количество объектов значительно меняется при движении камеры. Для таких сцен требуется выполнить предварительную оценку затрат на отображение. Исходя из этой оценки, можно выбрать подходящие уровни детализации объектов. Для каждого объекта осуществляется выбор уровня детализации и алгоритма затенения (равномерное затенение, затенение по Гуро, использование текстурных карт). Задача сводится к поиску значений, которые осуществимы с целевой частотой кадров и соответствуют максимальному качеству изображения. Рассматривается упрощенная модель

графического конвейера, которая включает в себя обработку вершин, полигонов и фрагментов. Предполагается, что отправка команд рендеринга происходит мгновенно, на GPU процессоре выполняется только рассматриваемое приложение. Время работы конвейера определяется временем самого медленного этапа. Для оценки времени обработки графических элементов предлагается использовать линейную комбинацию количества полигонов и вершин. Получаем формулу:

$$Cost(O, L, R) = max\begin{pmatrix} C_1 Poly(O, L) + C_2 Vert(O, L) \\ C_3 Pix(O) \end{pmatrix},$$

где O — объект,

L — уровень детализации объекта,

$Poly(O, L)$ — количество полигонов объекта O,

$Vert(O, L)$ — количество вершин объекта O,

C_1, C_2, C_3 — константы рендеринга, которые предлагается определять экспериментальным путем.

В работе [21] предложена модель производительности для определения времени рендеринга трехмерной сцены. Рассматривается общая формула для вычисления времени рендеринга:

$$t = RT(SG, RA, HW, ST),$$

где SG — обход объектов сцены (scene graph),

RA — процедура по отображению объекта (rendering action),

HW — аппаратное обеспечение (hardware),

ST — текущее состояние операционной системы и аппаратного обеспечения (state).

Для упрощения этой формулы вводится обозначение для массива объектов $X = (x_1, \dots, x_n)$. Считается, что каждый объект описывается своими атрибутами и геометрическим представлением $x_i = (g_i, a_i)$. Предполагается, что процедура по отображению объекта определяется его атрибутами. Тогда формула выглядит следующим образом: $t = RT(X, HW, ST)$. Для каждого кадра выполняется обход объектов сцены X, установка состояния GPU процессора согласно атрибутам a_i, отправка геометрии g_i. Драйвер отправляет составленные команды в командный буфер (очередь FIFO), видеокарта читает отправленные команды и выполняет этапы конвейера GPU для преобразования, растеризации, вычисления цвета и вывода на экран треугольников объектов. Графические элементы состоят из вершин и индексов, геометрическое представление хранится либо в памяти AGP, либо в видеопамяти. Предлагаются следующие формулы для оценки временных затрат CPU и GPU процессоров:

$$RT = ET_{system} + \max(ET^{CPU}, ET^{GPU}),$$
$$ET^{CPU} = ET_{nr}^{CPU} + ET_{r}^{CPU} + ET_{mm}^{CPU} + ET_{idle}^{CPU},$$
$$ET^{GPU} = ET_{fs}^{GPU} + ET_{r}^{GPU} + ET_{mm}^{GPU} + ET_{idle}^{GPU},$$

где nr — затраты, не относящиеся к рендерингу,

r — затраты на рендеринг,

fs — затраты на установку состояния,

mm — затраты на обращения к памяти,

$idle$ — время простоя.

В наилучшем сценарии, CPU и GPU процессоры работают параллельно без простоев (с загруженностью 100%). Зачастую этот сценарий является недостижимым. Задержка при выполнении обхода сцены на центральном процессоре вызывает простой в ожидании команд. Долгая обработка элементов на конвейере GPU вызывает простой центрального процессора. Для достижения наилучшей производительности необходимо сократить время простоя. Используется следующий метод оценки времени рендеринга. Пусть задано количество

вершин, треугольников, пикселей. Тогда время рендеринга вычисляется как максимум времени преобразования вершин, растеризации треугольников, вычисления цвета пикселей. Для получения консервативной оценки предлагается суммировать времена работы разных этапов конвейера.

2. Модель производительности графического конвейера

В данном разделе предлагается модель производительности графического конвейера, которая расширяет существующие модели, описанные в работах [20-21]. Новизна заключается в том, что учитываются следующие аспекты рендеринга:

- затраты на запись и отправку командных буферов, в том числе с использованием рассматриваемых техник;
- затраты на отправку, выполнение, получение результатов аппаратных проверок видимости.

Выводятся формулы для расчета времени рендеринга использованием альтернативных базовых методов и техник:

- удаление невидимых объектов с использованием методов пространственной декомпозиции и индексирования;
- отложенные аппаратные проверки видимости;
- фрагментация и кэширование командных буферов с учетом локальных изменений в сцене.

Полученные формулы позволяют своевременно выбирать наиболее эффективные способы реализации рендеринга в рамках однопроходной схемы в процессе отображения динамических сцен.

2.1 Исследуемая модель графического конвейера

Рассмотрим модель графического конвейера, которая описывает рендеринг за один проход по объектам и графическим элементам. На рис. 1 изображены основные этапы рендеринга, которые включают обработку объектов на CPU и GPU процессорах.

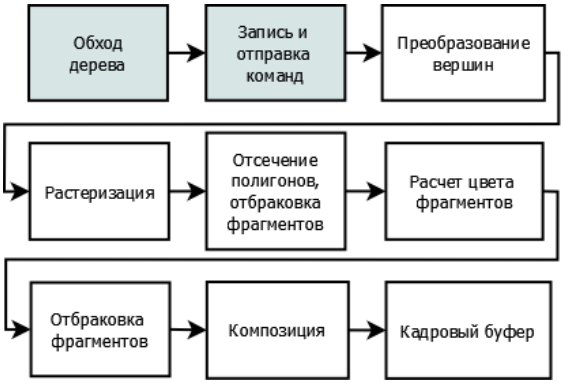


Рис. 1. Основные этапы рендеринга объектов сцены. Первые два этапа выполняются на центральном процессоре. Остальные этапы выполняются на GPU процессоре
Fig. 1. Main stages of 3D rendering. First two stages execute on CPU. Other stages execute on GPU

На вход подается сцена, включающая множество объектов, которые представлены в виде треугольных сеток с заданным положением и материалом. Применяется структура пространственного разбиения для сокращения затрат при выполнении отсечений (frustum culling) и аппаратных проверок видимости (hardware occlusion queries). На первом этапе

выполняется отбор узлов структуры, которые попадают в область видимости камеры. Затем на центральном процессоре проводится запись и отправка команд рендеринга. Команда рендеринга содержит информацию о конфигурации конвейера GPU процессора и смещение в памяти, по которому хранятся вершины (индексы вершин) объекта. Команды посылаются по шине на GPU процессор.

2.2 Исследуемая схема рендеринга

В данной работе рассматривается прямая однопроходная схема рендеринга динамических трехмерных сцен. Это означает, что записывается одна команда рендеринга для каждого объекта сцены, командный буфер составляется на центральном процессоре, осуществляется один проход по графическим элементам для генерации кадрового буфера. Рассматриваемые способы реализации процессов рендеринга заключаются в выполнении следующих шагов:

1. Изменение видимости, положений, материалов объектов сцены при изменении анимационного времени.
2. Получение результатов аппаратных проверок видимости.
3. Ожидание готовности командного буфера.
4. Обход структуры пространственного разбиения и выполнение отсечений.
5. Запись команд рендеринга для отдельных объектов сцены.
6. Запись ссылок на вспомогательные буфера в главный командный буфер.
7. Запись буфера для выполнения аппаратных проверок видимости.
8. Отправка главного командного буфера на выполнение.

Шаги 1, 3, 8 выполняются для всех способов. Запись команд рендеринга может проводиться напрямую в главный командный буфер (шаг 5) или может выполняться запись командных буферов для групп объектов сцены (шаг 6). Для ускорения рендеринга может использоваться структура пространственного разбиения H (шаг 4), могут выполняться аппаратные проверки видимости (шаги 2, 7).

2.3 Описание динамической сцены

Пусть кортеж S содержит описание динамической сцены:

$$S =< Mesh, MeshInst, Mat, Tex, Idx, Vert, TC, Norm, Vp, t_{anim}, TrnTl, VisTl, MatTl, batchSize, visFrags, d_{anim} >, \quad (1)$$

где $Mesh$ — множество треугольных сеток,

$MeshInst$ — множество объектов сцены, которые имеют заданное положение, материал, видимость,

Mat — множество материалов,

Tex — множество текстур,

Idx — множество индексов, которое задается в случае использования проиндексированных множеств вершин, нормалей, текстурных координат,

$Vert$ — множество вершин,

TC — множество текстурных координат,

$Norm$ — множество нормалей,

Vp — текущее положение камеры (viewpoint), которое задается с помощью матрицы размерности 4×4,

t_{anim} — время анимации, для которого определяются характеристики объектов сцены,
$TrnTl$ — множество временных событий, которые задают положение объектов сцены,
$VisTl$ — множество временных событий, которые задают видимость объектов сцены,
$MatTl$ — множество временных событий, которые задают материалы объектов сцены.

Также в кортеж S добавлены переменные, которые зависят от положения камеры Vp и анимационного времени t_{anim}:
$batchSize$ — среднее количество вершин в видимых объектах сцены во время t_{anim},
$visFrags$ — суммарное количество фрагментов видимых объектов сцены во время t_{anim},
d_{anim} — средняя доля объектов сцены, свойства которых изменились за последние несколько кадров.

Кортеж S задает описание всех объектов сцены (видимых и невидимых). Для сцен с объектами, прошедшими проверки видимости, в момент времени t_{anim} при положении камеры Vp будем использовать обозначение S_{vis}, если не оговорено иначе. Под проверками видимости подразумевается метод удаления объектов, не попадающих в область видимости камеры, и метод выполнения аппаратных проверок видимости ограничивающих параллелепипедов AABB узлов дерева относительно буфера глубины (hardware occlusion query).

2.4 Оценка потребляемой памяти

Данные для выполнения рендеринга записываются в память. При использовании разных способов рендеринга расходуется различный объем памяти. Предварительная запись командных буферов, использование дополнительных уровней детализации объектов (levels of details) повышают объем расходуемой памяти. Если памяти не хватает, то, как правило, выполняется завершение программы. Чтобы убедиться, что памяти достаточно для выполнения рендеринга, выведем формулы для вычисления требуемого объема памяти. Объем памяти, который занимает полигональное представление сцены, выражается с помощью формулы:

$$M_{geom}(S, HW) = |\mathrm{Vert}|\mathrm{M}(vert_1, HW) + |\mathrm{Idx}|\mathrm{M}(idx_1, HW) + |\mathrm{TC}|\mathrm{M}(tc_1, HW) + |\mathrm{Norm}|\mathrm{M}(norm_1, HW), \quad (2)$$

где HW — конфигурация вычислительной системы (CPU и GPU),
$|Vert|$, $|Idx|$, $|TC|$, $|Norm|$ — количества вершин, индексов, текстурных координат, нормалей,
$M(a, HW)$ — объем памяти, который занимает элемент a.

Объем памяти, который занимают текстуры сцены, выражается с помощью формулы:

$$M_{tex}(\mathrm{S}, \mathrm{HW}) = \sum_{i=1}^{|Tex|} W(tex_i)H(tex_i)BPP(tex_i, HW), \quad (3)$$

где $W(tex)$ — ширина текстуры tex,
$H(tex)$ — высота текстуры tex,
$BPP(tex)$ — объем памяти, который занимает один пиксель текстуры tex.

Объем памяти, который занимают буфера рендеринга с матрицами и материалами, выражается с помощью формулы:

$$M_{uniforms}(\mathrm{S}, \mathrm{HW}) = \sum_{i=1}^{|MeshInst|} M_{ubo}(Mat(inst_i), HW), \quad (4)$$

где $M_{ubo}(mat, HW)$ — объем памяти, который занимает буфер, хранящий значения переменных для шейдера, соответствующего данному материалу,

$Mat(inst)$ — материал объекта сцены $inst$.

В процессе рендеринга записываются буфера с командами рендеринга. Каждая команда содержит информацию для отображения объекта сцены с заданным положением и материалом. Объем памяти, который занимает буфер, содержащий команды для рендеринга объектов сцены, выражается с помощью формулы:

$$M_{cmds}(\mathrm{S},\mathrm{HW}) = \sum_{i=1}^{|MeshInst|} M_{cmd}(Mat(inst_i), HW), \tag{5}$$

где $M_{cmd}(mat, HW)$ — объем памяти, который занимает команда рендеринга объекта сцены с материалом mat.

Пусть множество объектов сцены S разбито на подмножества $\mathrm{G} = \{g_i : \{inst_{i_1}, \dots, inst_{i_n}\}\}$. Будем считать, что для каждого подмножества g_i записывается отдельный командный буфер (secondary command buffer). Минимальный объем памяти, который занимает командный буфер, составляет $M_{\mathrm{sec\_buf}}(HW)$. Команды рендеринга подмножества g_i могут занимать больший объем памяти. В этом случае производится расширение командного буфера. Объем памяти, который занимают буфера, содержащие команды рендеринга для подмножеств g_i, выражается с помощью формулы:

$$M_{grouped\_cmds}(\mathrm{G},\mathrm{HW}) = \sum_{i=1}^{|G|} \max\left(M_{\mathrm{sec\_buf}}(HW), M_{cmds}(g_i,\mathrm{HW})\right). \tag{6}$$

Для выполнения рендеринга необходимо задать конфигурацию графического конвейера, которая используется при отображении объекта сцены с заданным материалом. Зачастую используется одна конфигурация конвейера на материал. Конфигурации всех конвейеров, как правило, составляются заранее и хранятся в памяти для сокращения затрат в процессе рендеринга. Объем памяти, который занимают конфигурации графических конвейеров программного интерфейса, выражается с помощью формулы:

$$M_{rendering}(\mathrm{S},\mathrm{HW}) = \sum_{i=1}^{|Mat|} \mathrm{M}_{pipeline}(m_i, HW), \tag{7}$$

где $\mathrm{M}_{pipeline}(m_i, HW)$ — объем памяти, который требуется для хранения описания конфигурации конвейера для вывода объектов сцены с материалом m_i.

Объем памяти, который требуется для хранения описаний объектов сцены, выражается с помощью формулы:

$$M_{instances}(\mathrm{S},\mathrm{HW}) = |MeshInst|M(inst_1, HW) + |Mat|M(mat_1, HW), \tag{8}$$

где $|MeshInst|$, $|Mat|$ — количества объектов сцены, материалов,

$M(inst, HW)$ — объем памяти, который занимает объект сцены при использовании вычислительной системы HW,

$M(mat, HW)$ — объем памяти, который занимает материал при использовании вычислительной системы HW.

При выполнении рендеринга может использоваться структура пространственного разбиения для кластеризации объектов сцены. Объем памяти, который требуется для хранения пространственного индекса, выражается с помощью формулы:

$$M_{hierarchy}(\mathrm{H},\mathrm{HW}) = |\mathrm{H}|M(n_1, HW), \tag{9}$$

где H — множество узлов структуры пространственного разбиения (дерева),

$M(n, HW)$ — объем памяти, который занимает узел структуры пространственного разбиения при использовании вычислительной системы HW.

Зачастую геометрия сцены, текстуры и буфера с матрицами и описанием материалов ($M_{geom}(S, HW)$, $M_{tex}(S, HW)$, $M_{uniforms}(S, HW)$) хранятся в видеопамяти. Остальные данные ($M_{cmds}(S, HW)$, $M_{grouped\_cmds}(G, HW)$, $M_{rendering}(S, HW)$, $M_{instances}(S, HW)$, $M_{hierarchy}(H, HW)$) хранятся в основной памяти (RAM). Когда недостаточно видеопамяти, можно выполнить запись в основную память. Таким образом, на этапе подготовки к рендерингу определяется доступная память и проводится запись данных либо в видеопамять, либо в основную память.

2.5 Оценка времени рендеринга

Графический конвейер выполняет работу для выявления треугольников, которые не попадают в область видимости камеры. Для каждой вершины треугольников выполняется преобразование для перевода в плоскость изображения. Только после растеризации выполняется отсечение треугольников, которые не попали в область видимости камеры. Таким образом, расходуются лишние вычислительные ресурсы на отправку команд и преобразование вершин треугольников, которые не выводятся на экран. Подобная ситуация возникает и при отображении сложных сцен с большим количеством перекрывающихся поверхностей. Для определения видимости фрагментов используется метод z-буфера. Однако до применения метода z-буфера необходимо выполнить преобразование вершин, растеризацию, расчет цвета фрагментов. Таким образом, часть ресурсов тратится на обработку фрагментов, которые отсекаются методом z-буфера и в конечном итоге не попадают в кадровый буфер. Эта проблема особенно проявляется в сценах с большим количеством перекрывающихся поверхностей. Объем вычислений можно сократить с помощью методов удаления невидимых поверхностей, но они не всегда эффективны. Актуальной является разработка модели производительности рендеринга, в рамках которой можно оценить время выполнения различных этапов рендеринга. Это позволит определить сценарии, в которых методы удаления невидимых поверхностей являются эффективными.

Рассмотрим формулы для оценки времени выполнения приведенных способов реализации процессов рендеринга. Время выполнения рендеринга на центральном процессоре выражается с помощью формулы:

$$\begin{gathered} T_{CPU}\left(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW\right) = \\ T_{update}(S, HW) + [h = 1]T_{update}(H, HW) + \\ [h = 0]T_{fc}(S, HW) + [h = 1](T_{traverse}(H, HW) + T_{fc}(H, HW)) + \\ [g = 0]T_{buf}(S_{vis}, HW) + [g = 1](T_{buf}\left(H_{upd}, HW\right) + T_{\text{sec\_buf}}(H_{vis}, HW)) + \\ [q = 1](T_{buf}\left(H_q, HW\right) + T_{recv\_query}\left(H_q, HW\right)), \end{gathered} \quad (10)$$

где S_{vis} — кортеж с информацией об объектах сцены, которые прошли проверки видимости и находятся в области видимости камеры,

H_{vis} — множество узлов структуры пространственного разбиения в области видимости камеры,

H_{upd} — множество узлов структуры пространственного разбиения, в которых произошли изменения после движения объектов сцены,

H_q — множество узлов, для которых выполняются аппаратные проверки видимости,

q — параметр, который определяет, используются ли проверки видимости,

$[q = 1]$ — выражение, которое равняется 1, если $q = 1$ (скобка Айверсона),

h — параметр, который определяет, используется ли структура пространственного разбиения H,

g — параметр, который определяет, используется ли составление и отправка вспомогательных буферов (secondary command buffers) для узлов структуры пространственного разбиения H,

$T_{update}(S, HW)$ — время изменения свойств объектов сцены с помощью структур $TrnTl$, $VisTl$, $MatTl$ при изменении анимационного времени,

$T_{update}(H, HW)$ — время обновления пространственного индекса при изменении анимационного времени,

$T_{fc}(H, HW)$ — время выполнения отсечений узлов, не попадающих в область видимости камеры,

$T_{traverse}(H, HW)$ — время обхода структуры пространственного разбиения H,

$T_{buf}(S, HW)$ — суммарное время составления и отправки командных буферов со всеми объектами сцены S,

$T_{\text{sec\_}buf}(H, HW)$ — суммарное время отправки вспомогательных командных буферов для узлов структуры пространственного разбиения H,

$T_{recv\_query}(H, HW)$ — время получения всех результатов проверок видимости.

Время выполнения рендеринга на GPU процессоре выражается с помощью формулы:

$$T_{GPU}(S, H_q, q, HW) = \max\left(T_{state}(S, HW), T_{vert}(S, HW), T_{frag}(S, Vp, HW)\right) + [q = 1]\max(T_{state}(H_q, HW), T_{vert}(H_q, HW), T_{frag}(H_q, Vp, HW), T_{exec\_query}(H_q, HW)), \quad (11)$$

где $T_{state}(S, HW)$ — суммарное время установки состояний конвейера для всех материалов, используемых при отображении сцены S,

$T_{vert}(S, HW)$ — суммарное время преобразования вершин объектов сцены S,

$T_{frag}(S, Vp, HW)$ — суммарное время обработки фрагментов объектов сцены S,

$T_{exec\_query}(H, HW)$ — время выполнения проверок видимости узлов структуры пространственного разбиения H.

Итоговое время рендеринга сцены S при использовании структуры пространственного разбиения H выражается с помощью формулы:

$$T(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW) = \begin{cases} T_{CPU}(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW) + T_{GPU}(S_{vis}, H_q, q, HW), \text{один гл. ком. буфер} \\ \max\left(T_{CPU}(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW), T_{GPU}(S_{vis}, H_q, q, HW)\right), \text{иначе.} \end{cases} \quad (12)$$

В формуле (12) выполняется расчет времени рендеринга для двух сценариев. В первом сценарии используется один главный командный буфер (при этом может использоваться много вспомогательных буферов). Запись команд возможна только тогда, когда чтение буфера завершено на GPU процессоре. Это означает, что необходимо выполнять синхронизацию доступа к буферу, при этом работа на CPU и GPU будет выполняться последовательно. Во втором сценарии используется несколько командных буферов и выполняется параллельная работа над разными буферами (рис. 2).

Для вычисления затрат при использовании различных способов реализации процессов рендеринга в приведенные формулы добавлены параметры h, g, q. Если $h = 0, g = 0, q = 0$, получаем формулу для расчета времени рендеринга без использования структуры пространственного разбиения, без записи команд рендеринга для узлов дерева (фрагментации), без проверок видимости узлов дерева. Если $h = 1, g = 0, q = 0$, получаем формулу для расчета времени рендеринга с использованием структуры пространственного разбиения для выполнения отсечений, без записи команд рендеринга для узлов дерева, без проверок видимости узлов дерева. Если $h = 1, g = 1, q = 0$, получаем формулу для расчета времени рендеринга с использованием структуры пространственного разбиения для

выполнения отсечений, с составлением команд рендеринга для узлов дерева, без проверок видимости узлов дерева. Если $h = 1, g = 1, q = 1$, получаем формулу для расчета времени рендеринга с использованием структуры пространственного разбиения для выполнения отсечений, с составлением команд рендеринга для узлов дерева, с проверками видимости узлов дерева.

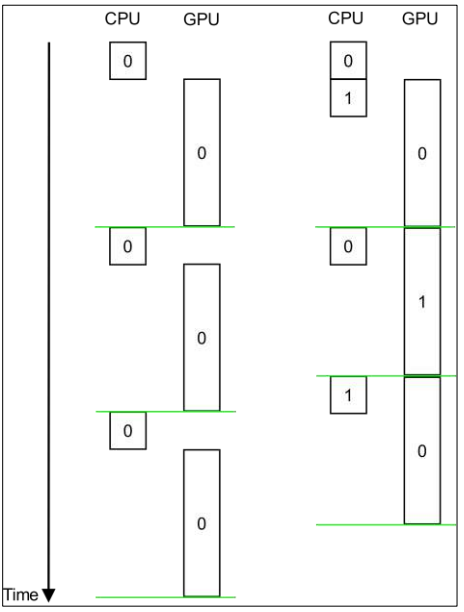


Рис. 2. Синхронизация доступа к главным командным буферам. Номерами помечены командные буфера. На CPU производится запись, на GPU — выполнение. Использование одного главного буфера (слева). Использование двух главных буферов для параллелизации записи и выполнения буфера команд (справа). Зеленая линия показывает момент, когда доступен для записи следующий командный буфер

Fig. 2. Synchronizing access to primary command buffers. Numbers are used to mark different command buffers. CPU performs writing, GPU — execution. Employing one primary command buffer (left). Employing two command buffers to parallelize writing and execution of commands (right). Green line shows the moment of time when next buffer is ready for writing

2.6 Время выполнения этапов графического конвейера

Оценим время выполнения различных этапов рендеринга. Время изменений свойств объектов сцены с помощью структур $TrnTl$, $VisTl$, $MatTl$ зависит от количества изменений. Время изменений вычисляется по формуле:

$$T_{update}(S, HW) = d_{anim}|MeshInst|t_{update}(HW), \quad (13)$$

где $t_{update}(HW)$ — среднее время записи свойств (матрицы преобразования, материала, видимости) для одного объекта сцены. Оно также включает время обращения к памяти UBO, в которой хранятся значения переменных шейдерных программ.

Время обхода структуры пространственного разбиения вычисляется по формуле:

$$T_{traverse}(H, HW) = |H|t_{traverse}(HW), \quad (14)$$

где $t_{traverse}(HW)$ — время обхода одного узла на данной вычислительной системе.

Время выполнения отсечений узлов структуры пространственного разбиения вычисляется по формуле:

$$T_{fc}(H, HW) = |H|t_{fc}(HW), \quad (15)$$

где $t_{fc}(HW)$ — время отсечения одного узла на данной вычислительной системе.

Время записи командного буфера для всех объектов сцены вычисляется по формуле:

$$T_{buf}(MeshInst, HW) = |MeshInst| t_{buf}(HW), \quad (16)$$

где $t_{buf}(HW)$ — время записи и отправки одной команды на данной вычислительной системе.

Время отправки вспомогательных командных буферов вычисляется по формуле:

$$T_{\text{sec\_buf}}(H, HW) = |H| t_{\text{sec\_buf}}(numInstPerNode, HW), \quad (17)$$

$t_{\text{sec\_buf}}(numInstPerNode, HW)$ — время отправки одного вспомогательного командного буфера с количеством команд $numInstPerNode$.

Времена выполнения этапов конвейера GPU вычисляются по формулам:

$$T_{state}(S, HW) = |Mat| t_{state}(HW), \quad (18)$$

$$T_{vert}(S, HW) = \sum_{i=1}^{|MeshInst|} NumVerts(inst_i) t_{vert}(Mat(inst_i), batchSize, |MeshInst|, HW), \quad (19)$$

$$T_{frag}(S, Vp, HW) = \sum_{i=1}^{|MeshInst|} NumFrags(inst_i, Vp) t_{frag}(Mat(inst_i), visFrags, HW), \quad (20)$$

$$T_{exec\_query}(H, HW) = |H| t_{exec\_query}(HW), \quad (21)$$

$$T_{recv\_query}(H, HW) = |H| t_{recv\_query}(HW), \quad (22)$$

где $T_{state}(HW)$ — время установки состояния графического конвейера на данной вычислительной системе,

$t_{vert}(mat, batchSize, numObjects, HW)$ — время работы вершинного шейдера данного материала с данным количеством вершин в объекте $batchSize$ и количеством объектов в сцене $numObjects$ (важность этих параметров объясняется ниже),

$NumVerts(inst)$ — количество вершин в объекте $inst$,

$t_{frag}(mat, visFrags, HW)$ — время работы фрагментного шейдера данного материала при обработке одного фрагмента (важность параметра $visFrags$ объясняется ниже),

$NumFrags(inst, Vp)$ — количество фрагментов объекта, которые видимы при данном положении камеры,

$t_{exec\_query}(HW)$ — время выполнения проверки видимости на этой вычислительной системе,

$t_{recv\_query}(HW)$ — время получения результата проверки видимости на данной вычислительной системе.

2.7 Вычисление покрываемого объектом количества фрагментов

Количество фрагментов, которые объект (узел дерева) занимает на экране, можно определить путем проекции его ограничивающего параллелепипеда AABB на экран. Доля видимых фрагментов рассчитывается по формуле:

$$NumFrags = w\,h \left(\frac{1}{\sqrt{w/h}\; 2\, tg\left(fov/2\right)} \right)^2 \frac{bb_x bb_y}{d^2}, \quad (23)$$

где w — ширина экрана,

h — высота экрана,

fov — вертикальный угол обзора,

bb_x — ширина видимой стороны ограничивающего параллелепипеда AABB объекта сцены,

bb_y — высота видимой стороны ограничивающего параллелепипеда AABB объекта сцены,

d — расстояние от камеры до ближайшей грани ограничивающего параллелепипеда AABB объекта сцены.

Вместо произведения $bb_x bb_y$ можно использовать среднюю площадь грани ${A_{bb}}/{6}$.

2.8 Генерация тестовых наборов для вычисления параметров модели

Оценим скорость работы вычислительной системы HW при выполнении этапов рендеринга. Современные видеокарты содержат большое количество потоковых процессоров, которые выполняют параллельную обработку данных. Итоговое время вычислений зависит от объема работы (количества вершин, треугольников и объектов в сцене).

Производится генерация трехмерных сцен с различными параметрами:

- количество вершин, приходящихся на объект ($batchSize$),
- количество объектов ($numObjects$),
- материал.

Параметр $batchSize$ имеет значение для производительности, потому что при выполнении рендеринга производится установка значений переменных для каждого объекта (uniform variables). Частое изменение этих переменных ухудшает производительность рендеринга. Произведение параметров $batchSize * numObjects$ определяет объем работы (суммарное количество вершин).

Пусть характерный размер объекта равняется a. Тогда длина, ширина и высота генерируемой сцены вычисляются, как $\sqrt[3]{numObjects}\ a$. Объекты равномерно заполняют весь объем сцены. Используется один материал для всех объектов. Предлагается выполнять генерацию таких сцен при увеличении значений параметров $batchSize, numObjects$ с некоторым шагом до тех пор, пока выполняются условия:

- Геометрия сцены помещается в доступную видеопамять.
- Командный буфер помещается в доступную основную память.

Для оценки времени выполнения этапов рендеринга нужно взять тестовую сцену с близким значением параметров $batchSize, numObjects$.

Для оценки $t_{frag}(mat, visFrags, HW)$ достаточно произвести вывод пары треугольников на весь экран и измерить время работы конвейера GPU. Обозначим количество обработанных фрагментов при выполнении рендеринга сцены, как $visFrags$. Тогда время обработки фрагмента равняется времени работы конвейера, поделенному на значение $visFrags$. Также производится оценка времени обработки фрагмента в случаях, когда пара треугольников занимает только часть экрана: $3/4$, $1/2$, $1/4$, $1/8$, $1/16$. В качестве значения $t_{frag}(mat, visFrags, HW)$ для некоторой входной сцены берется время обработки фрагмента при выполнении рендеринга тестовой сцены с наиболее близким значением $visFrags$.

Для вычисления $t_{state}(HW)$ выполняется рендеринг одного треугольника, что вызывает задержку, связанную со сменой состояния конвейера. При этом количество обрабатываемых фрагментов треугольника должно быть незначительным ($NumFrags(inst, Vp) \leq 10$). Воспользуемся формулой (23) и получим расстояние от камеры, на котором выполняется это условие:

$$d \geq \frac{h}{2\ tg\left({fov}/{2}\right)} \sqrt{\frac{bb_x bb_y}{10}}. \tag{24}$$

2.9 Метод оценки количества аппаратных проверок видимости для эффективного выполнения рендеринга

Аппаратные проверки видимости позволяют выполнить отбраковку большого количества невидимых треугольников, чтобы сократить время работы графического конвейера. Однако на выполнение аппаратных проверок видимости также расходуются вычислительные ресурсы. Необходимо оценивать эффективность аппаратных проверок видимости в процессе отображения динамической сцены.

В отличие от существующих методов предложенный в этом разделе метод предназначен для динамических сцен. В течение некоторого времени накапливаются данные о видимости объектов. Затем производится вычисление и адаптивное обновление количества проверок видимости в процессе отображения. Использование аналитических формул позволяет сократить затраты и получить релевантные оценки для текущего состояния сцены.

Выделяются следующие стадии метода:

1. Применение ограниченного количества проверок видимости $N_{queries}$. Постепенное накопление данных о видимости объектов.
2. Вывод количества проверок видимости $N_{queries}^{efficient}$ для эффективного рендеринга.
3. Применение нового количества проверок видимости $N_{queries}^{efficient}$ до тех пор, пока это выгодно и количество изменившихся объектов незначительно.

Во время первой стадии производится обход дерева в ширину для приоритетного отбора узлов с наибольшим количеством объектов. При этом пропускаются узлы, для которых не выполняется критерий отбора:

$$t_{\text{sec\_buf}}(HW) + \max\left(t_{state}(HW), NumVerts(node)\hat{t}_{vert}, NumFrags(node, Vp)\hat{t}_{frag}\right) \geq 10\left(t_{buf}(HW) + t_{exec\_query}(HW) + t_{recv\_query}(HW)\right),$$

где $\hat{t}_{vert}$ — среднее время преобразования вершины для всех материалов сцены,

$\hat{t}_{frag}$ — среднее время расчета цвета фрагмента для всех материалов сцены.

Отметим, что в случае использования нескольких главных командных буферов работа на CPU и GPU выполняется параллельно, и в этой формуле нужно использовать максимум времени работы на CPU и GPU.

Проверяется видимость отобранных узлов. Собираются данные, задающие зависимость количества невидимых элементов (узлов дерева, объектов, вершин, фрагментов) в сцене от количества проверок видимости. Исходя из этого, выводится количество проверок видимости для эффективного рендеринга.

Определим количество проверок видимости с помощью предложенной модели производительности графического конвейера. Сначала рассмотрим сценарий, когда этапы рендеринга на CPU и GPU выполняются последовательно. Самым долгим этапом конвейера GPU является преобразование вершин или расчет цвета фрагментов. Тогда время рассматриваемых этапов определяется по формуле:

$$T(x) = c(x)t_c + s(x)t_s + e(x)t_e,$$

где x — количество проверок видимости узлов структуры пространственного разбиения,

$c(x)$ — количество записываемых команд рендеринга,

$s(x)$ — количество отправляемых командных буферов,

$e(x)$ — количество графических элементов на лимитирующем этапе конвейера (вершин или фрагментов),

t_c — время записи одной команды рендеринга,

t_s — время отправки вспомогательного командного буфера,

t_e — время выполнения лимитирующего этапа конвейера (преобразования вершины, расчета цвета фрагмента).

Накопленные данные о видимости узлов дерева позволяют определить функции $c(x)$, $s(x)$, $e(x)$. Количество невидимых узлов, объектов, вершин (фрагментов) монотонно растет с увеличением количества проверок видимости. Зачастую рост происходит быстрее при малых x, а затем останавливается при больших x. Приближенное описание таких данных задается с помощью логарифмической зависимости $c_1 + c_2 \ln(1 + x)$, где c_1 и c_2 — коэффициенты, которые вычисляются методом наименьших квадратов [22].

Накопив достаточное количество данных о видимости узлов дерева, определим коэффициенты регрессии и подставим в функции:

$$c(x) = \alpha_n(o_0 - o_1 - o_2 \ln(1 + x)) + x,$$
$$s(x) = (n_0 + 1 - s_1 - s_2 \ln(1 + x)),$$
$$e(x) = (e_0 + px - e_1 - e_2 \ln(1 + x)),$$

где α_n — средняя доля перезаписываемых командных буферов узлов дерева,

o_0 — текущее количество объектов в сцене,

n_0 — количество непустых узлов окто-дерева,

e_0 — количество вершин (фрагментов) при выполнении рендеринга сцены,

p — количество вершин (фрагментов), приходящихся на один узел окто-дерева.

Определение минимума $T(x)$ вместе с ограничениями на количество проверок видимости $0 \le x \le n_0$ является задачей нелинейного программирования. Функция $T(x)$ является непрерывной и дифференцируемой на множестве ограничений, поэтому минимум является либо стационарной точкой, либо лежит на границе множества ограничений [23]. Для нахождения стационарной точки решим уравнение $T'(x) = 0$ и получим:

$$x_1^* = \frac{\alpha_n o_2 t_c + s_2 t_s + e_2 t_e}{p t_e + t_c} - 1.$$

Сравнив $T(0)$, $T(n_0)$, $T(x_1^*)$, найдем минимальное время выполнения рендеринга сцены и соответствующее количество проверок видимости x.

Аналогичные вычисления можно провести и для сценария с параллельной работой CPU и GPU. Тогда время рендеринга определяется по формуле: $T(x) = \max(T_{CPU}(x), T_{GPU}(x))$. Нужно определить минимум двух функций:

$$T_{CPU}(x) = c(x)t_c + s(x)t_s,$$
$$T_{GPU}(x) = e(x)t_e.$$

Уравнение $T'_{CPU}(x) = 0$ имеет решение: $x_2^* = \frac{\alpha_n o_2 t_c + s_2 t_s}{t_c} - 1$. Сравнив $T_{CPU}(0)$, $T_{CPU}(n_0)$, $T_{CPU}(x_2^*)$, найдем минимальное время выполнения рендеринга на CPU и соответствующее значение x_2. Уравнение $T'_{GPU}(x) = 0$ имеет решение: $x_3^* = \frac{e_2}{p} - 1$. Сравнив $T_{GPU}(0)$, $T_{GPU}(n_0)$, $T_{GPU}(x_3^*)$, найдем минимальное время выполнения рендеринга на GPU и соответствующее значение x_3. После этого сравним $T(0)$, $T(n_0)$, $T(x_2)$, $T(x_3)$ и возьмем количество проверок видимости x, которое соответствует минимуму $T(x)$.

3. Вычислительные эксперименты

Проведем вычислительные эксперименты для подтверждения эффективности разработанной модели производительности графического конвейера и предложенного метода. Характеристики тестовой вычислительной системы: Intel Core i7-7700 3.6GHz, 16GB RAM, NVIDIA Geforce GTX 1070.

Для тестирования производительности методов используются сцены (рис. 3), характеристики которых приведены в табл. 1.

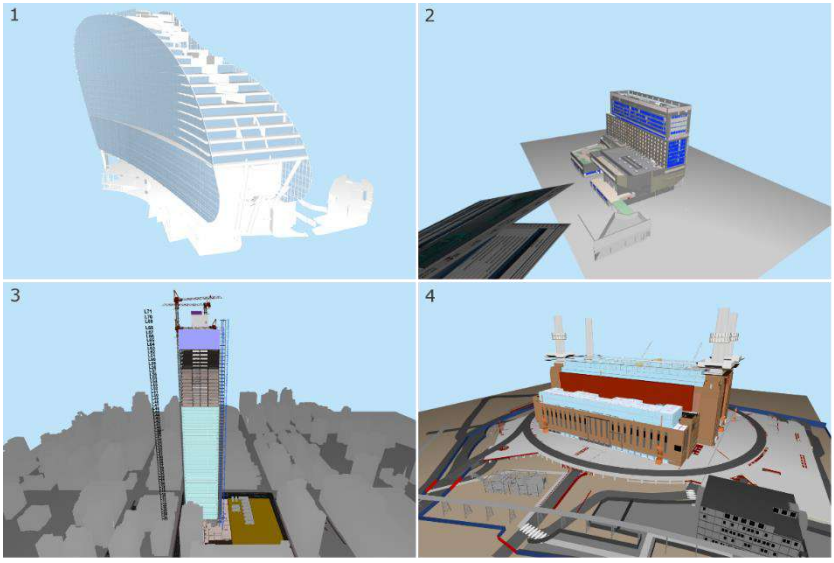


Рис. 3. Тестовые сцены 1–4
Fig. 3. Test scenes 1–4

Табл. 1. Характеристики тестовых сцен
Table 1. Properties of test scenes

Сцена	**Количество вершин**	**Количество треугольников**	**Количество объектов**
1	15037746	5012582	50521
2	32483139	10827713	71961
3	34609623	11536541	109991
4	94388454	31462818	270431

Проведем тестирование с использованием способа рендеринга с записью команд без кэширования. Проводилось отсечение объектов, не попадающих в область видимости камеры, а затем выполнялась запись командного буфера. Время работы рассчитывалось по формуле $T(S, S_{vis}, \emptyset, \emptyset, \emptyset, \emptyset, h = 0, g = 0, q = 0, HW)$. В табл. 2 приведены результаты тестирования с использованием тестовых сцен 1–4 при задании конечного времени анимации.

Табл. 2. Сравнение измеренного и рассчитанного времени рендеринга при использовании способа рендеринга с записью команд без кэширования
Table 2. Comparison of measured and calculated frame rendering time when employing rendering process without prerecorded commands

Сцена	**Измеренное время рендеринга, мс.**	**Рассчитанное время рендеринга, мс.**	**Отклонение (%)**
1	15.16	12.92	15
2	20.75	20.28	2
3	19.81	26.76	35
4	69.87	65.75	6

Проведем тестирование с использованием способа рендеринга с записью команд без кэширования и аппаратными проверками видимости каждого объекта. Проводилось получение результатов проверок видимости, отсечение объектов, не попадающих в область

видимости камеры, а затем выполнялась запись командного буфера и отправка новых проверок видимости для каждого объекта. Время работы рассчитывалось по формуле $T(S, S_{vis}, \emptyset, \emptyset, \emptyset, S_{vis}, h = 0, g = 0, q = 1, HW)$. В табл. 3 приведены результаты тестирования с использованием тестовых сцен 1–4 при задании конечного времени анимации.

Табл. 3. Сравнение измеренного и рассчитанного времени рендеринга при использовании способа рендеринга с записью команд без кэширования и аппаратными проверками видимости каждого объекта

Table 3. Comparison of measured and calculated frame rendering time when employing rendering process with occlusion queries per object and without prerecorded commands

Сцена	**Измеренное время рендеринга, мс.**	**Рассчитанное время рендеринга, мс.**
1	19.53	19.31
2	24.09	24.61
3	41.01	45.58
4	83.14	92.75

Таким образом, результаты тестирования предложенной модели производительности рендеринга показали, что ее можно применять для получения достаточно точных оценок времени рендеринга при использовании разнообразных сцен и способов рендеринга.

Для проверки эффективности метода, предложенного в разделе 2.9, проводится тестирование с использованием динамических сцен 1–4. В процессе анимации сцен происходит добавление новых объектов, удаление временных объектов, изменение цвета объектов. Тестирование заключается в отображении сцены с фиксированным положением камеры, проигрывании анимации и измерении времени рендеринга с применением следующих способов реализации рендеринга:

- Frustum Culling. Выполняется обход окто-дерева (single reference octree) с отсечением узлов, не попадающих в камеру. Проводится запись команд в главный командный буфер без кэширования.
- Рендеринг с применением предложенного метода по расчету количества проверок видимости для узлов окто-дерева.

В табл. 4 приведены результаты тестирования. Таким образом, расчет и адаптивное изменение количества проверок видимости узлов окто-дерева в процессе рендеринга позволяет получить прирост производительности до 2.3 раз по сравнению с распространенным методом Frustum Culling.

Табл. 4. Среднее время рендеринга при отображении динамических сцен

Table 4. Average frame rendering time when displaying dynamic scenes

Сцена	**Frustum Culling, мс.**	**Рендеринг с применением предложенного метода, мс.**
1	17.88	7.51
2	43.58	26.79
3	56.89	31.33
4	105.44	52.29

4. Заключение

Предложена и исследована математическая модель производительности графического конвейера для однопроходной схемы рендеринга динамических трехмерных сцен. Модель позволяет оценивать требуемые ресурсы (время обработки и передачи графических данных, объем основной и графической памяти) в зависимости от применяемых базовых методов и

характеристик отображаемой сцены. Для получения релевантных оценок на оборудовании пользователя параметры модели калибруются с использованием тестовых наборов. Предложен метод оценки количества аппаратных проверок видимости для эффективного выполнения рендеринга динамических сцен. Проведены вычислительные эксперименты, которые показали релевантность предложенной модели и эффективность разработанного метода при отображении больших динамических сцен.

Список литературы / References

[1]. Cohen-Or D., Chrysanthou Y. L., Silva C. T., Durand F. A survey of visibility for walkthrough applications. IEEE Transactions on Visualization and Computer Graphics, vol. 9, no. 3, 2003, pp. 412–431.

[2]. Airey J. Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations. PhD diss., University of North Carolina, Chappel Hill, 1991.

[3]. Teller S. J. Visibility computations in densely occluded polyhedral environments. PhD diss., University of California at Berkeley, 1992.

[4]. Luebke D., Georges C. Portals and mirrors: Simple, fast evaluation of potentially visible sets. In Proc. of the 1995 Symposium on Interactive 3D Graphics, 1995, pp. 105-106.

[5]. Durand F., Drettakis G., Thollot J., Puech C. Conservative visibility preprocessing using extended projections. In Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 2000, pp. 239–248

[6]. Greene N., Kass M., Miller G. Hierarchical Z-buffer visibility. In Proc. of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 1993, pp. 231–238.

[7]. Bittner J., Havran V., Slavik P. Hierarchical visibility culling with occlusion trees. In Proc. of the Computer Graphics International, 1998, pp. 207-219.

[8]. Coorg S., Teller S. Temporally coherent conservative visibility. Computational Geometry, vol. 12, no. 1-2, 1999, pp. 105–124.

[9]. Coorg S., Teller S. Real-time occlusion culling for models with large occluders. In Proc. of the 1997 Symposium on Interactive 3D Graphics,1997, pp. 83–90.

[10]. Hudson T., Manocha D., Cohen J., Lin M., Hoff K., Zhang H. Accelerated occlusion culling using shadow frustra. In Proc. of the 13th Annual ACM Symposium on Computational Geometry, 1997, pp 1–10.

[11]. Glassner A. S. Space subdivision for fast ray tracing. IEEE Computer Graphics and applications, vol. 4, no. 10, 1984, pp. 15–24.

[12]. Pharr M., Jakob W., Humphreys G. Physically based rendering: From theory to implementation. 3rd ed., Morgan Kaufmann, 2016, 1266 p.

[13]. Sudarsky O., Gotsman C. Dynamic scene occlusion culling. IEEE Transactions on Visualization and Computer Graphics, vol. 5, no. 1, 1999, pp. 13–29.

[14]. Morozov S., Semenov V., Tarlapan O., Zolotov V. Indexing of Hierarchically Organized Spatial-Temporal Data Using Dynamic Regular Octrees. Lecture Notes in Computer Science, vol 10742, 2018, pp. 276–290.

[15]. NV_occlusion_query. URL: https://www.khronos.org/registry/OpenGL/extensions/NV/NV_occlusion_query.txt, accessed 26.08.2020.

[16]. Bittner J., Wimmer M., Piringer H., Purgathofer W. Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful. Computer Graphics Forum, vol. 23, issue 3, 2004, pp. 615-624.

[17]. Mattausch O., Bittner J., Wimmer M. CHC++: Coherent Hierarchical Culling Revisited. Computer Graphics Forum. vol. 27, issue 2, 2008, pp. 221–230.

[18]. Guthe M., Balázs Á., Klein R. Near Optimal Hierarchical Culling: Performance Driven Use of Hardware Occlusion Queries. In Proc. of the 17th Eurographics Symposium on Rendering, 2006, pp. 207–214.

[19]. Gonakhchyan V.I. Efficient command buffer recording for accelerated rendering of large 3d scenes. In Proc. of the 12th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, 2018, pp. 397–402.

[20]. Funkhouser T.A., Séquin C.H. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In Proc. of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 1993, pp. 247–254.

[21]. Wimmer M., Wonka P. Rendering time estimation for real-time rendering. In Proc. of the 14th Eurographics Workshop on Rendering, 2003, pp. 118–129.

[22]. Weisstein, Eric W. Least Squares Fitting – Logarithmic. From MathWorld – A Wolfram Web Resource. URL: https://mathworld.wolfram.com/LeastSquaresFittingLogarithmic.html. accessed 26.08.2020.

[23]. Черняк А.А., Черняк Ж.А., Метельский Ю.М., Богданович С.А. Методы оптимизации: теория и алгоритмы. 2 изд., Юрайт, 2017, 357 стр. / Chernyak A.A., Chernyak Zh.A., Metelsky Yu.M., Bogdanovich S.A. Optimization methods: theory and algorithms. 2nd ed., Urait, 2017, 357 p. (in Russian).

Информация об авторах / Information about authors

Вячеслав Игоревич ГОНАХЧЯН – младший научный сотрудник отдела системной интеграции и прикладных программных комплексов. Сфера научных интересов: компьютерная графика, вычислительная геометрия.

Viacheslav Igorevich GONAKHCHIAN – junior research fellow of the department of System integration and multi-disciplinary collaborative environments. Research interests: computer graphics, computational geometry.

DOI: 10.15514/ISPRAS–2020–32(4)–5

Эффективные методы и алгоритмы синтеза видео 360 градусов на основе кубической проекции виртуального окружения

П.Ю. Тимохин, ORCID: 0000-0002-0718-1436 <webpismo@yahoo.de>
М.В. Михайлюк, ORCID: 0000-0002-7793-080X <mix@niisi.ras.ru>
Е.М. Вожегов, ORCID: 0000-0003-2676-1206 <vozhegovem@icloud.com>

ФНЦ Научно-исследовательский институт системных исследований РАН,
117218, Россия, г. Москва, Нахимовский просп., д. 36, к. 1

Аннотация. В статье рассматривается задача создания и воспроизведения панорамного видео с обзором 360 градусов, обеспечивающего погружение исследователя в виртуальную среду вне родительской системы виртуального окружения (СВО). Для решения этой задачи предлагается расширение метода проекции в кубическую карту, при котором разрешение карты определяется с учетом угла обзора камеры зрителя и разрешения экрана (Adequate Cubemap Projection, ACMP). В работе исследовано влияние ориентации камеры зрителя внутри куба на отношение «пиксел карты/пиксел экрана», определяющее качество визуализации панорамы, и предложен метод вычисления разрешения кубической карты для качественной визуализации панорамы при всех возможных ориентациях камеры. В работе рассмотрены эффективные метод и алгоритм создания ACMP-видео на GPU с помощью технологии рендеринга в текстуру, которые позволяют синтезировать панорамы с постоянной ориентацией или с привязкой к направлению взгляда наблюдателя. Также в исследовании предложены эффективные методы и алгоритмы воспроизведения ACMP-видео, основанные на визуализации видимых граней куба и адаптивной буферизации кадров. Полученные методы и алгоритмы реализованы в программном комплексе синтеза ACMP-видео (C++, OpenGL, FFmpeg), который включает в себя модуль захвата кадров (встраиваемый в СВО) и плеер. Разработанное решение было протестировано в системе «Виртуальная Земля» по обучению наблюдению объектов земной поверхности с орбиты Международной космической станции (МКС). С помощью модуля захвата было создано ACMP-видео полета вдоль участка подспутниковой трассы МКС. При воспроизведении данного видео обучаемый летит по орбите над виртуальной 3D поверхностью Земли и может исследовать ее, поворачивая камеру. Апробация комплекса подтвердила адекватность разработанных методов и алгоритмов поставленной задаче. Полученные научные и практические результаты позволяют расширить возможности и сферу применения СВО, систем научной визуализации, видеотренажеров и виртуальных лабораторий, эффективно обмениваться опытом между исследователями и др.

Ключевые слова: видео 360 градусов; панорамное видео; кубическая проекция; визуализация; виртуальное окружение.

Для цитирования: Тимохин П.Ю., Михайлюк М.В., Вожегов Е.М. Эффективные методы и алгоритмы синтеза видео 360 градусов на основе кубической проекции виртуального окружения. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 73–88. DOI: 10.15514/ISPRAS–2020–32(4)–5

Благодарности: Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «34.9. Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации» (0065-2019-0012).

Efficient methods and algorithms to synthesize 360-degree video based on cubemap projection of virtual environment

P.Yu. Timokhin, ORCID: 0000-0002-0718-1436 <webpismo@yahoo.de>
M.V. Mikhaylyuk, ORCID: 0000-0002-7793-080X <mix@niisi.ras.ru>
E.M. Vozhegov, ORCID: 0000-0003-2676-1206 <vozhegovem@icloud.com>
Scientific Research Institute for System Analysis of RAS,
36, build. 1, Nakhimovskiy Avenue, Moscow, 117218, Russia

Abstract. The paper deals with the task of creation and playback of panoramic video with 360-degree overview, which allows the researcher to be immersed in virtual medium outside parent virtual environment system (VES). To solve the task, the extension of the cubemap method is proposed, in which cubemap resolution is determined taking into account viewer camera field of view and screen resolution (Adequate Cubemap Projection, ACMP). The paper studies the influence of the camera orientation inside the cube on the "cubemap pixel / screen pixel" ratio determining panorama visualization quality. Based on this, a method to calculate cubemap resolution for high-quality panorama visualization for all possible camera orientations is proposed. The paper considers an efficient method and algorithm to create ACMP-video on the GPU using render-to-texture technology, which allow to synthesize panoramas with constant orientation or bound to the observer's view direction. In the research efficient methods and algorithms to play ACMP-video are also proposed, which are based on the visualization of visible cube faces and adaptive frame buffering. The obtained methods and algorithms are implemented in ACMP-video synthesis program complex (C++, OpenGL, FFmpeg) including frame capture module (embeddable into VES) and the player. The developed solution was tested in system «Virtual Earth» designed for training to observe Earth objects from the International Space Station (ISS). Using the capture module, an ACMP-video of the flight along the ISS orbit track was created. When playing this video, the trainee flies in orbit above virtual 3D surface of the Earth and can explore it by means of camera rotation. Testing of the complex confirmed the adequacy of the developed methods and algorithms to the task. The obtained scientific and practical results expand the capabilities and scope of application of VES, scientific visualization systems, video simulators and virtual laboratories; provide effective exchange of experience between researchers, etc.

Keywords: 360-degree video; omnidirectional video; cubemap projection; visualization; virtual environment.

For citation: Timokhin P.Yu., Mikhaylyuk M.V., Vozhegov E.M. Efficient methods and algorithms to synthesize 360-degree video based on cubemap projection of virtual environment. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 73–88 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–5

Acknowledgements. The publication is made within the state task on carrying out basic scientific researches (GP 14) on topic (project) «34.9. Virtual environment systems: technologies, methods and algorithms of mathematical modeling and visualization» (0065-2019-0012).

1. Введение

В настоящее время во многих важных научных и прикладных областях востребовано виртуальное окружение, в котором человек (оператор) может обучаться решению реальных задач и проводить исследования. Особенно это актуально для областей, сопряженных с работой в агрессивной и труднодоступной среде (космическая и авиационная отрасли, нефтегазовая сфера, атомная промышленность и др.) [1, 2]. Создание виртуального окружения является достаточно сложной задачей, для решения которой разрабатываются специальные программно-аппаратные комплексы – *системы виртуального окружения* (*СВО*), включающие в себя подсистемы управления, моделирования динамики, визуализации и др. [3, 4]. Часто возникает необходимость воспроизведения виртуального окружения, созданного с помощью СВО, на обычных персональных компьютерах (в том числе, мобильных и портативных устройствах), например, для анализа допущенных при обучении ошибок, обмена опытом между исследователями, демонстрации результатов перед широкой аудиторией и др. [5]. Одним из эффективных подходов является синтез панорамных видеороликов с обзором 360° (*видео 360*), при воспроизведении которых зритель может

поворачивать камеру в произвольном направлении и ощущать эффект присутствия в виртуальной среде [6]. В идеале, 360-градусная панорама является непрерывной и представляет собой сферу, в центре которой расположен наблюдатель. Чтобы получить кадр видео 360, такую «идеальную» сферу необходимо некоторым образом записать в дискретное планарное изображение (панораму 360), с которыми работают все современные видеокодеки. Ввиду того, что поверхность сферы невозможно развернуть на плоскость без искажений, обычно выбирают такие способы проецирования, при которых в областях поля зрения, наиболее важных для зрителя, искажения минимальны.

Существующие методы построения панорамы 360 можно условно разделить на две следующие группы. К *первой группе* относятся методы, в которых панорама проецируется на плоскость с помощью одной или нескольких картографических проекций. Широко распространены решения, основанные на эквидистантной цилиндрической проекции [7] (в настоящее время в данной проекции видео 360 загружается на YouTube). Эквидистантная проекция отображает панораму в истинном масштабе только вдоль экватора, а при приближении к полюсам масштаб увеличивается. Для уменьшения искажений масштаба в работе [8] было предложено проецировать сферу на плоскость горизонтальными полосами приблизительно одинакового разрешения (длины таких полос авторы сокращают при приближении к полюсам), а области полюсов записывать в виде двух круговых изображений.

Ко *второй группе* относятся методы, в которых панорама проецируется на некоторый выпуклый многогранник, который затем разворачивается на плоскость. Благодаря своей простоте и поддержке современными графическими библиотеками широкое распространение получила проекция в кубическую карту (cubemap projection, CMP) [9-12], при которой панорама отображается на 6 граней куба, где каждая грань охватывает угол обзора $90°$. В традиционной CMP все грани равнозначны (отсутствуют области полюсов с сильными искажениями масштаба), однако за счет перспективной проекции при приближении к ребрам и углам куба искажения изображения увеличиваются [10]. Для уменьшения неоднородностей детализации разрабатываются различные модификации CMP: равноугольная кубическая проекция (Equi-Angular Cubemap, EAC) [10], CMP со смещенным от центра положением наблюдателя [11], «умное» ориентирование CMP, при котором объекты переднего плана отображаются на грани, а не на ребра [12] и др. В исследовании [13] авторам удалось уменьшить потери четкости видео 360 (по сравнению с традиционной CMP) за счет проецирования панорамы на ромбододекаэдр. Разработчики из компании Facebook предложили принципиально другой подход, при котором панорама проецируется на грани пирамиды видимости наблюдателя [14]. Чтобы обеспечить охват в $360°$, авторы используют 30 таких пирамид с шагом в $30°$. Несмотря на заявленное значительное сокращение объема видеофайла (до 80% по сравнению с эквидистантной проекцией), авторы впоследствии отказались от данного подхода из-за его сложности в пользу кубической проекции.

Одной из важных задач при реализации методов, основанных на CMP, является выбор разрешения кубической карты, обеспечивающего синтез качественных изображений. При выборе слишком большого разрешения битрейт видеопотока существенно возрастает, что приводит к увеличению времени воспроизведения кадров и появлению вынужденных пауз. Если установить недостаточное разрешение, то ухудшается качество видео (возникают артефакты «блочности» изображения), в результате чего теряется эффект присутствия. В данной работе предлагаются методы и алгоритмы решения этой задачи, основанные на расширении метода кубической проекции, при котором разрешение кубической карты определяется с учетом угла обзора камеры зрителя и разрешения экрана. Такую проекцию мы называем адекватной (Adequate CMP, ACMP), а получаемое с помощью нее видео 360 – ACMP-видео. В разд. 2 предлагается метод вычисления разрешения ACMP-карты, в разд. 3 и 4 рассматриваются методы и алгоритмы создания и воспроизведения ACMP-видео, а в разд. 5 приводятся результаты апробации программного комплекса, созданного на основе предложенных методов и алгоритмов.

2. Метод вычисления разрешения ACMP-карты

Метод кубической проекции состоит в проецировании окружающей наблюдателя виртуальной среды на грани единичного куба с помощью шести перспективных камер, расположенных в центре этого куба и направленных под 90 градусов друг к другу. Каждая камера имеет вертикальный угол обзора (*FOV*), равный 90 градусов и отношение ширины к высоте формируемого изображения (*aspect*), равное единице. Таким образом, каждая камера проецирует видимое ею изображение на соответствующую грань куба. Эта проекция записывается в область вывода с некоторым разрешением d (число пикселов вдоль каждой стороны изображения).

Для просмотра результата кубического проецирования записанные изображения накладываются на грани куба, в центре куба устанавливается камера зрителя C_V, через которую он может наблюдать изображение виртуальной сцены, отображенное на гранях куба. Камера C_V может иметь произвольные *FOV* и *aspect* и ее область вывода может иметь некоторое разрешение D (число пикселов вдоль большей стороны области вывода). Для упрощения рассмотрения мысленно спроецируем области вывода на соответствующие ближние плоскости отсечения камер, так что теперь будем считать, что куб проецирования имеет размер d x d x d пикселов. Наша задача состоит в вычислении такого разрешения d кубической проекции, при котором любой отрезок пикселов из кубической проекции будет отображаться в отрезок не меньшего числа пикселов в области вывода с разрешением D.

Опишем вокруг куба сферу, имитирующую «идеальную» панораму 360, а в центр сферы поместим виртуальную камеру. Рассмотрим некоторую дугу l сферы, которая целиком попадает в поле зрения камеры и на грань куба. Построим проекцию l_{face} дуги l на грань куба (рис. 1а, в силу симметрии, на этом рисунке показана четвертая часть двумерной проекции куба). Рассмотрим также камеру C_V зрителя видео 360, и проекцию l_{cam} дуги l на ближнюю плоскость отсечения этой камеры (рис. 1б). Обозначим через α центральный угол дуги l и рассмотрим функцию $r(\eta,\beta)$ отношения длин проекций l_{face} и l_{cam} для бесконечно малой дуги l (при $\alpha \to 0$):

$$r(\eta,\beta) = \lim_{\alpha\to 0}\left(l_{face}/l_{cam}\right). \quad (1)$$

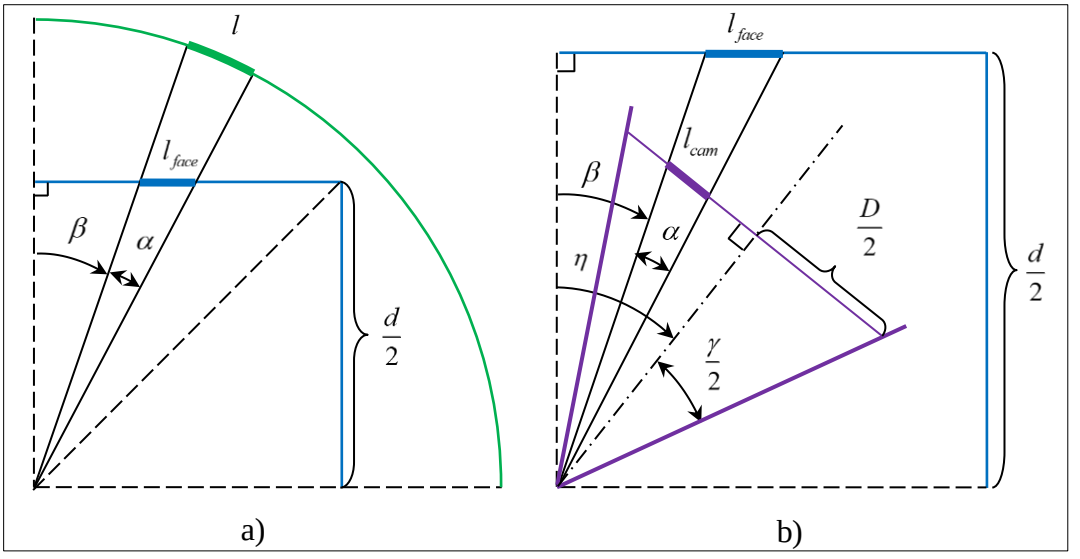


Рис. 1. Проекция дуги l: a) на грань куба; b) на ближнюю плоскость отсечения камеры
Fig. 1. Projection of arc l on: a) the face of the cube; b) near clip plane of the camera C_V

Фактически, функция r показывает, сколько пикселов изображения на грани куба (кубической проекции) будет отображаться в некоторый произвольный пиксел области вывода камеры зрителя. Во избежание образования артефактов «блочности» (когда пиксел грани куба отображается на несколько пикселов области вывода) отношение r в идеале не должно быть меньше 1 для всех пикселов экрана. На практике (например, для экранов с высокой плотностью пикселов), этот порог может быть уменьшен до некоторого значения r_{thres} без заметной потери качества видео 360. Тогда, величину d можно вычислить из равенства $r_{thres} = \min r(\eta, \beta)$.

2.1 Нахождение функции *r*

Чтобы найти функцию r, вычислим длины проекций l_{face} и l_{cam}. Введем следующие обозначения: $\beta \in [0, \pi/4 - \alpha]$ - угол отклонения дуги l от перпендикуляра к грани куба (см. рис. 1а); $\gamma \in (0, \pi)$ - угол наибольшего раствора (горизонтального или вертикального) камеры C_V (см. рис. 1б); $\eta \in [\alpha + \beta - \gamma/2, \beta + \gamma/2]$ - угол отклонения вектора взгляда камеры от перпендикуляра к грани куба.

Из рисунков 1а и 1б запишем следующие выражения для длин l_{face} и l_{cam}

$$\ell_{face} = d\frac{\mathrm{tg}(\alpha+\beta)-\mathrm{tg}\beta}{2}, \quad \ell_{cam} = D\frac{\mathrm{tg}(\eta-\beta)-\mathrm{tg}(\eta-\beta-\alpha)}{2\mathrm{tg}(\gamma/2)}. \tag{2}$$

Подставив эти выражения в формулу (1), получим

$$r(\eta,\beta) = \lim_{\alpha\to 0}\frac{l_{face}}{l_{cam}} = \frac{d}{D}\mathrm{tg}(\gamma/2)\cdot\lim_{\alpha\to 0}\frac{\mathrm{tg}(\alpha+\beta)-\mathrm{tg}\beta}{\mathrm{tg}(\eta-\beta)-\mathrm{tg}(\eta-\beta-\alpha)}. \tag{3}$$

Используя формулу преобразования разности тангенсов, приведем выражение (3) к виду

$$r(\eta,\beta) = \frac{d}{D}\mathrm{tg}(\gamma/2)\cdot\lim_{\alpha\to 0}\frac{\cos(\eta-\beta)\cos(\eta-\beta-\alpha)}{\cos(\alpha+\beta)\cos\beta}. \tag{4}$$

При уменьшении размера дуги l до бесконечно малой величины ($\alpha \to 0$) получим из (4) выражение искомой функции

$$r(\eta,\beta) = (d/D)\mathrm{tg}(\gamma/2)\left(\cos(\eta-\beta)/\cos\beta\right)^2, \text{ где } \beta \in [0,\pi/4],\ \eta \in [\beta-\gamma/2, \beta+\gamma/2]. \tag{5}$$

2.2 Нахождение наименьшего значения r_{min} и размера d

Легко проверить, что на указанных в (5) интервалах функция $f(\eta,\beta) = \cos(\eta-\beta)/\cos\beta$ неотрицательна, поэтому функция $r(\eta,\beta)$ принимает наименьшее значение r_{min} при наименьшем значении f_{min} функции $f(\eta,\beta)$. Чтобы найти f_{min}, попробуем сначала вычислить критические точки функции $f(\eta,\beta)$ в области $\beta \in (0,\pi/4)$, $\eta \in (\beta-\gamma/2, \beta+\gamma/2)$, т.е. те точки, в которых обе частные производные f'_η и f'_β равны нулю (это является необходимым условием наличия экстремума функции). Имеем

$$f'_\eta = -\frac{\sin(\eta-\beta)}{\cos\beta},\ f'_\beta = \frac{\sin\eta}{\cos^2\beta},\ \begin{cases} f'_\eta = 0 \\ f'_\beta = 0 \end{cases} \Rightarrow \begin{cases} \sin(\eta_{crit}-\beta_{crit}) = 0 \\ \sin\eta_{crit} = 0 \end{cases} \Rightarrow \begin{cases} \eta_{crit}-\beta_{crit} = \pi n \\ \eta_{crit} = \pi n \end{cases}. \tag{6}$$

Так как $\beta \in (0, \pi/4)$, а $\gamma \in (0, \pi)$, η не может выходить за границы интервала $(-\pi/2, 3\pi/4)$. Из этого следует, что $\eta_{crit} = 0$, а значит $\beta_{crit} = 0$. Однако это значение $\beta_{crit} \notin (0, \pi/4)$, следовательно, система (6) не имеет решения, а функция $f(\eta, \beta)$ не имеет критических точек в рассматриваемой области. Найдем теперь минимальное значение функции $f(\eta, \beta)$ на границах $\eta_1 = \beta - \gamma/2$, $\eta_2 = \beta + \gamma/2$, $\beta_1 = 0$ и $\beta_2 = \pi/4$. Получаем

$$f_1(\eta_1, \beta) = f_2(\eta_2, \beta) = \cos(\gamma/2)/\cos\beta, \quad f_3(\eta, \beta_1) = \cos\eta, \quad f_4(\eta, \beta_2) = \sqrt{2}\cos(\eta - \pi/4).$$

Функции $f_1(\eta_1, \beta)$ и $f_2(\eta_2, \beta)$ имеют наименьшее значение на отрезке $\beta \in [0, \pi/4]$, когда $\cos\beta$ максимален, т.е. при $\beta = 0$: $f_{1,\min} = f_{2,\min} = \cos(\gamma/2)$. Функция $f_3(\eta, \beta_1) = f_3(\eta, 0) = \cos\eta$ на отрезке $\eta \in [-\gamma/2, \gamma/2]$ принимает минимальное значение при $\eta = \pm\gamma/2$, откуда следует $f_{3,\min} = \cos(\gamma/2)$, а функция $f_4(\eta, \beta_2) = f_4(\eta, \pi/4) = \sqrt{2}\cos(\eta - \pi/4)$ на отрезке $\eta \in [\pi/4 - \gamma/2, \pi/4 + \gamma/2]$ имеет наименьшее значение при $\eta = \pi/4 \pm \gamma/2$, откуда $f_{4,\min} = \sqrt{2}\cos(\gamma/2)$. Так как $\gamma \in (0, \pi)$, то легко видеть, что $f_{\min} = \min_i f_i = \cos(\gamma/2)$. Подставив значение f_{min} в формулу (5), получим искомое выражение для r_{min}:

$$r_{\min} = (d/D)\,\mathrm{tg}(\gamma/2)\, f_{\min}^2 = (d/D)\,\mathrm{tg}(\gamma/2)\cos^2(\gamma/2) = (d/(2D))\sin\gamma. \quad (7)$$

Как отмечалось в начале разд. 2, размер d в данной работе находится из равенства $r_{thres} = r_{min}$. Подставим в него r_{min} из (7) и выразим из полученного равенства искомый d:

$$d = 2Dr_{thres}/\sin\gamma = 2Dr_{thres}\,\mathrm{cosec}\,\gamma. \quad (8)$$

2.3 Оценка качества визуализации кубической проекции

В формуле (8) величина r_{thres} означает наименьшее возможное число пикселов грани куба, которое будет отображаться в любой пиксел камеры зрителя, т.е. фактически r_{thres} определяет уровень качества визуализации кубической проекции в камере зрителя. Мы будем считать значение $r_{thres} = 1$, соответствующим наибольшему уровню качества, т.к. дальнейшее повышение числа отображаемых пикселов грани куба не будет иметь смысла - мы их просто не сможем различить в камере зрителя. Наибольший уровень качества необходим для отображения кубической проекции в камере зрителя с невысоким разрешением D, и, наоборот, для высоких значений D (Full HD, 4K) уровень качества может быть уменьшен. Рассмотрим это на следующем примере.

Вычислим по формуле (8) величину d для $\gamma = 50°$ (угол, близкий к восприятию пространства человеческим зрением), $D = 1024$ пикселов и $r_{thres} = 1$. При этих параметрах мы получим значение $d \approx 2600$ пикселов. Подставим значения d, D и γ в формулу (5) и построим поверхность $r(\eta, \beta)$ (см. рис. 2). Полученная поверхность-«седло» показывает изменение уровня качества визуализации кубической проекции в зависимости от места расположения объекта наблюдения (дуги l сферы 360 градусов) на грани куба (угол β) и в камере зрителя (угол η). На рисунке 2 передний торец «седла» соответствует случаю, когда объект наблюдения располагается в центре грани куба ($\beta = 0$), а задний торец – когда объект наблюдения находится по направлению в ребро куба ($\beta = \pi/4$). При этом самые верхние точки торцевых изгибов соответствуют случаю, когда камера зрителя направлена в объект наблюдения ($\eta = 0$ и $\eta = \pi/4$), т.е. объект находится в центре камеры, а самые нижние

точки - расположению объекта у границ раствора камеры зрителя (для переднего торца - $\eta = \pm\gamma/2 = \pm 25°$, а для заднего - $\eta = \pi/4 - \gamma/2 = 20°$ и $\eta = \pi/4 + \gamma/2 = 70°$).

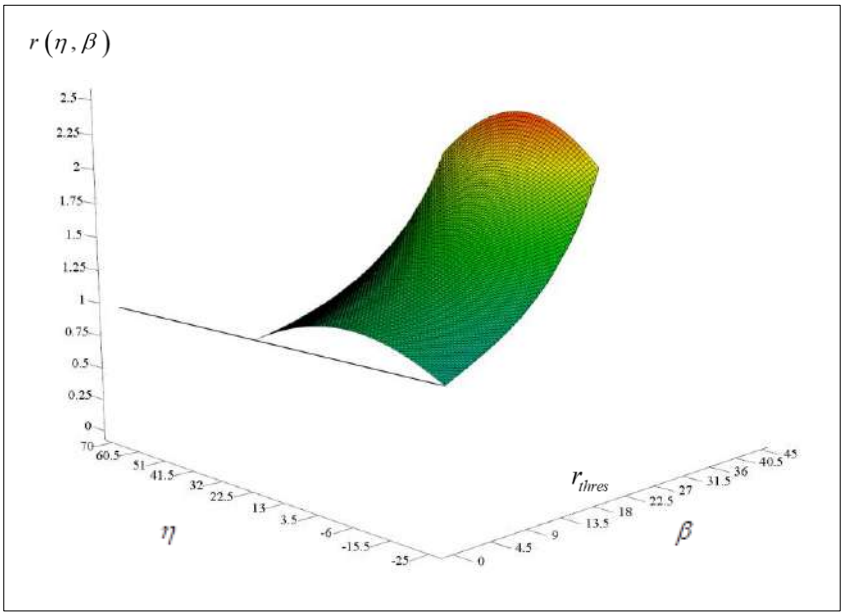


Рис. 2. Пример поверхности $r(\eta, \beta)$ при визуализации кубической проекции с высоким качеством
Fig. 2. Example of $r(\eta, \beta)$ surface at high-quality visualization of cubemap projection

Из рисунка 2 видно, что даже в самом худшем случае (объект в центре грани куба и у границы раствора камеры, см. концы переднего торца «седла») кубическая проекция будет визуализироваться с высоким качеством: в любой пиксел камеры зрителя будет отображаться не менее 1 пиксела грани куба. На практике, построение кубической проекции с разрешением граней d = 2600 пикселов может потребовать больших вычислительных ресурсов, например, если виртуальная сцена содержит сложные высокополигональные текстурированные объекты. В таких случаях можно пойти на уменьшение уровня r_{thres} качества до значения, позволяющего решать задачу, для которой создается видео 360 (на рис. 2 при уменьшении r_{thres} «седло» будет параллельно сдвигаться вниз). Так, например, при r_{thres} = 0.5, объекты в центральной части поля зрения камеры будут визуализироваться с качеством не менее 75%, а в периферийной – с качеством не менее 50% (при этом разрешение граней d уже составит около 1300 пикселов).

3. Метод создания ACMP-видео

Рассмотрим задачу создания видео 360 для наблюдателя, который исследует виртуальное окружение с помощью жестко связанной с ним виртуальной камеры C_{obs}. Наблюдатель может перемещаться и поворачивать камеру. Мы будем синтезировать кадр видео 360 на основе кубической проекции с разрешением d, вычисленным согласно (8), каждые $1/f_v$ миллисекунд, где $f_v \geq 25$ - частота смены кадров видеоролика в кадрах в секунду (d и f_v задаются с помощью диалогового окна до начала записи видео 360). Рассмотрим процесс синтеза кадра для некоторого такого момента времени.

Обозначим через P_{obs}, $\boldsymbol{v}_{obs}$ и $\boldsymbol{up}_{obs}$ позицию, вектор взгляда и вектор «вверх» камеры C_{obs} в мировой системе координат (World Coordinate System, WCS) в этот момент времени. Поместим в точку P_{obs} шесть одинаковых виртуальных камер $C_0, \ldots, C_5$ с FOV = 90°, aspect = 1 и областью вывода d x d пикселов. Направление камеры C_1 совпадает с направлением камеры C_{obs}, а остальные камеры расположены под 90 градусов друг к другу (см. рисунок 3).

Направим эти камеры так, чтобы их ближние плоскости образовывали куб, как показано на рисунке 3 (грани куба названы так, как их видит наблюдатель изнутри).

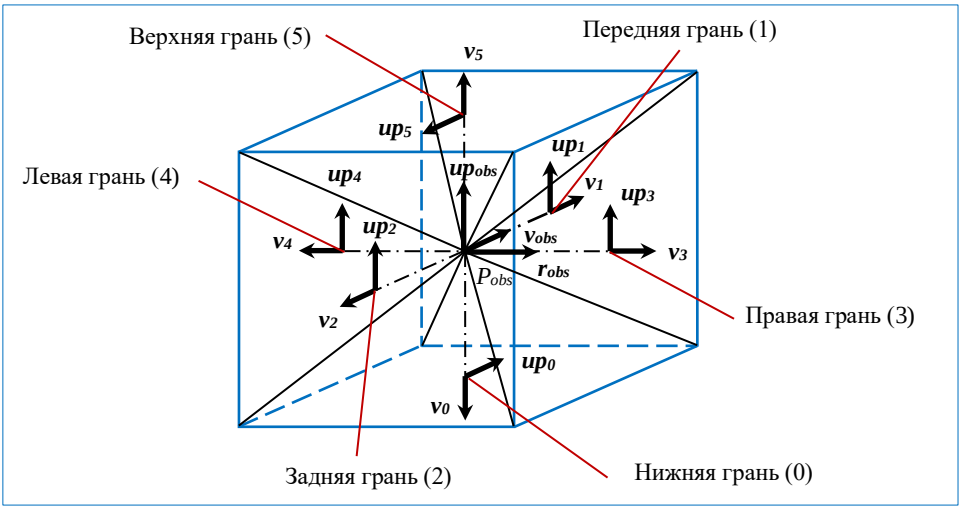


Рис. 3. Построение кубической проекции
Fig. 3. Constructing cubemap projection

Векторы $\boldsymbol{v}_0,\ldots,\boldsymbol{v}_5$ взгляда и векторы $\boldsymbol{up}_0,\ldots,\boldsymbol{up}_5$ «вверх» камер $C_0,\ldots,C_5$ всегда коллинеарны одному из векторов $\boldsymbol{v}_{obs}$, $\boldsymbol{up}_{obs}$, $\boldsymbol{r}_{obs}$, где $\boldsymbol{r}_{obs} = \boldsymbol{v}_{obs} \times \boldsymbol{up}_{obs}$ (вектор «вправо» камеры C_{obs}), а их координаты находятся в соответствии с таблицей 1.

Таблица 1. Координаты векторов камер C_i
Table 1. Coordinates of camera C_i vectors

Грань	***0 (нижняя)***	***1 (передняя)***	***2 (задняя)***	***3 (правая)***	***4 (левая)***	***5 (верхняя)***
$\boldsymbol{v}_i$	$-\boldsymbol{up}_{obs}$	$\boldsymbol{v}_{obs}$	$-\boldsymbol{v}_{obs}$	$\boldsymbol{r}_{obs}$	$-\boldsymbol{r}_{obs}$	$\boldsymbol{up}_{obs}$
$\boldsymbol{up}_i$	$\boldsymbol{v}_{obs}$	$\boldsymbol{up}_{obs}$	$\boldsymbol{up}_{obs}$	$\boldsymbol{up}_{obs}$	$\boldsymbol{up}_{obs}$	$-\boldsymbol{v}_{obs}$

Каждый синтезируемый кадр ACMP-видео будет состоять из 6 изображений, полученных из камер C_i. Синтез такого кадра будем выполнять в RGB-текстуре T размера $3d$ x $2d$ пикселов. Поставим в соответствие каждой C_i-ой камере участок d x d пикселов (тайл) текстуры T, как показано на рис. 4.

1. Активируем FBO и очистим текстуры T и U с помощью операторов *glClearColor*, *glClearDepth* и *glClear*.
2. Зададим матрицу перспективной проекции (общую для камер $C_0,\ldots,C_5$) с FOV = 90° и aspect = 1 с помощью оператора *gluPerspective*.
3. Цикл по i от 0 до 5
   Установим область вывода C_i-ой камеры размера d x d пикселов с координатами левого нижнего угла $x = (i \,\%\, 3)d$, $y = \lfloor i/3 \rfloor d$ с помощью оператора *glViewport*.
   Зададим матрицу видового преобразования C_i-ой камеры с позицией P_{obs} и векторами $\boldsymbol{v}_i$ и $\boldsymbol{up}_i$ с помощью *gluLookAt*.
   Выполним рендеринг виртуальной сцены из камеры C_i.
   Конец цикла.
4. Деактивируем FBO.

Алгоритм 1. Синтез ACMP-кадра
Algorithm 1. ACMP-frame synthesis

С помощью графической библиотеки OpenGL создадим внеэкранный буфер кадра (Framebuffer Object, FBO) [15] с текстурами T (буфер цвета) и U (буфер глубины). Далее

выполним рендеринг (с включенным тестом глубины) виртуальной сцены из каждой C_i-ой камеры в i-ый тайл текстуры T с помощью *Алгоритма 1*.

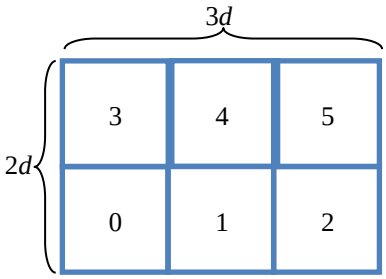


Рис. 4. Тайлы АСМР-кадра
Fig. 4. Tiles of ACMP-frame

После выполнения *Алгоритма 1* полученный кадр АСМР-видео (заполненная текстура T) выгружается из видеопамяти в оперативную память, сжимается и добавляется в видеопоток с помощью набора библиотек FFmpeg [16]. Работа со сжатыми видеопотоками более подробно описана в [5].

4. Метод воспроизведения АСМР-видео

Для воспроизведения АСМР-видео в данной работе создается виртуальная 3D сцена (см. рис. 5), содержащая модель единичного куба с центром в WCS, в который помещается виртуальная камера C_V зрителя (введена в разделе 2). Повороты камеры C_V разрешены только вокруг осей X и Y свой локальной системы координат, что соответствует наклону головы вверх/вниз и влево/вправо. Воспроизведение АСМР-видео включает в себя следующие ключевые шаги: (а) считывание и декодирование АСМР-кадров с частотой f_v; (б) извлечение из очередного АСМР-кадра тайлов-текстур для граней куба, видимых из камеры C_V; (в) плавная визуализация АСМР-кадров. Описание шага (а) можно найти в работе [5], поэтому далее рассмотрим шаги (б) и (в).

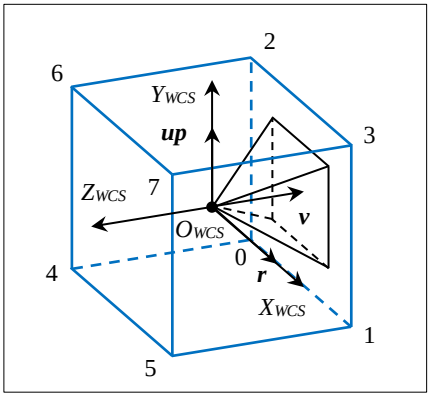


Рис. 5. Единичный куб и камера C_V
Fig. 5. Unit cube and camera C_V

4.1 Отбор текстур видимых граней куба

Для определения тайлов-текстур (граней куба), необходимых для визуализации АСМР-кадра, введем понятие *двухгранника* – пары граней куба с общим ребром. Как можно заметить, у куба 12 вариантов двухгранников (по числу ребер). Пронумеруем вершины куба, как показано на рисунке 5 и запишем массив D ребер куба: {0, 1}, {1, 5}, {5, 4}, {4, 0}, {6, 4}, {7,

5}, {3, 1}, {2, 0}, {2, 3}, {3, 7}, {7, 6}, {6, 2}. Для ребер из массива D запишем массив E вариантов двухгранников, используя нумерацию граней куба из рисунка 3: {0, 1}, {0, 3}, {0, 2}, {4, 0}, {2, 4}, {2, 3}, {3, 1}, {1, 4}, {1, 5}, {3, 5}, {5, 2}, {4, 5}. В зависимости от параметров камеры C_V при просмотре видео в область вывода могут попадать ребра куба, а могут и не попадать. Ребра, попадающие в поле зрения камеры C_V, будут определять двухгранники, которые необходимо использовать при визуализации. Если же в поле зрения не попадает ни одного ребра, то камера C_V будет захватывать какую-то одну грань куба.

Чтобы упростить проверку видимости ребер куба, создадим таблицу H булевских флагов положения каждой вершины куба относительно каждой из 5 отсекающих плоскости камеры (0 – ближняя, 1 – левая, 2 – правая, 3 – нижняя, 4 – верхняя). Дальнюю плоскость отсечения мы не учитываем, т.к. выставляем ее заведомо дальше всех вершин куба. Таблица H состоит из 8 строк, где номер строки совпадает с номером вершины куба. Каждая строка хранит флаги $b_0,\ldots,b_4$, а также флаг $b_5 = (b_0 \,\&\&\, b_1 \,\&\&\, b_2 \,\&\&\, b_3 \,\&\&\, b_4)$. Истинный b_i-ый флаг ($i = 0,\ldots,4$) означает, что вершина куба лежит в i-ой отсекающей плоскости или ее «+» полупространстве, а истинный флаг b_5 - что вершина куба находится в пирамиде видимости камеры C_V. Вычисление флагов описано в работе [17].

С помощью таблицы H будем отбирать для визуализации каждый двухгранник, общее ребро которого пересекает пирамиду видимости камеры C_V. Обозначим через b_{pair} флаг наличия хотя бы одного отобранного двухгранника (*true*/*false* - есть/нет). Грани отобранных двухгранников (или видимую одиночную грань куба) будем отмечать в булевском массиве B_{faces} длины 6 (по числу граней в кубе, *true*/*false* - грань видна/не видна). Это реализует *Алгоритм 2*.

1. Очистим массив B_{faces} значением *false*, b_{pair} = *false*.
2. Цикл по j от 0 до 11, где j – индекс ребра куба
   Если ($H_{D[j][0],\ 5}$ || $H_{D[j][1],\ 5}$), то b_{pair} = *true*; // видима хотя бы одна вершина ребра.
   В противном случае (вершины ребра не видимы), проверим, пересекает ли ребро хотя бы одну отсекающую плоскость, и видима ли точка P_u их пересечения:
   Цикл по i от 1 до 4, где i - отсекающая плоскость камеры C_V
      Если ($H_{D[j][0],\ i}$ ^ $H_{D[j][1],\ i}$), то: // ^ – исключающее ИЛИ.
         Вычислим координаты точки P_u и ее флаг b_5 видимости (см. [17]).
         Если b_5 равен *true*, то: $B_{faces}[E[j][0]]$ = *true*, $B_{faces}[E[j][1]]$ = *true*, b_{pair} = *true*, выходим из цикла.
   Конец цикла.
3. Если в камере C_V не видно ни одного ребра куба (b_{pair} = *false*), то ищем k-ую грань, у которой угол между внешней нормалью и вектором $\boldsymbol{v}$ камеры C_V будет наименьшим:
   k = 0, запишем массив K косинусов углов между внешними нормалями к граням куба и вектором $\boldsymbol{v}$ взгляда: $K = \{-\boldsymbol{v}_y, -\boldsymbol{v}_z, \boldsymbol{v}_z, \boldsymbol{v}_x, -\boldsymbol{v}_x, \boldsymbol{v}_y\}$.
   Цикл по p от 1 до 5, где p - номер грани куба.
      Если $K[i] > K[k]$, то $k = i$.
   Конец цикла.
   $B_{faces}[k]$ = *true*.

Алгоритм 2. Получение номеров видимых граней куба
Algorithm 2. Getting the numbers of visible cube faces

В результате выполнения *Алгоритма 2* в элементах массива B_{faces}, соответствующих видимым граням куба, будут записаны значения *true*. По номеру n каждого такого элемента мы находим смещения $offsetX = (n\%3)d$ и $offsetY = \lfloor n/3 \rfloor d$ (в пикселах) тайла-текстуры n-ой видимой грани относительно левого нижнего угла АСМР-кадра (см. рис. 6) и загружаем эту

тайл-текстуру в видеопамять (в массив из 6 текстурных объектов, созданный перед визуализацией). Чтобы уменьшить время простоя GPU, мы загружаем каждую тайл-текстуру в видеопамять одним непрерывным куском (а не построчно) с помощью связки операторов *glPixelStorei* [15]. Отметим, что если ACMP-кадр не меняется, например, когда видео на паузе, то мы не выполняем повторную загрузку тайлов в текстурные объекты.

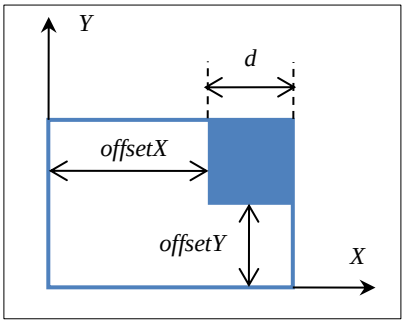


Рис. 6. Чтение тайла ACMP-кадра
Fig. 6. Reading a tile of ACMP-frame

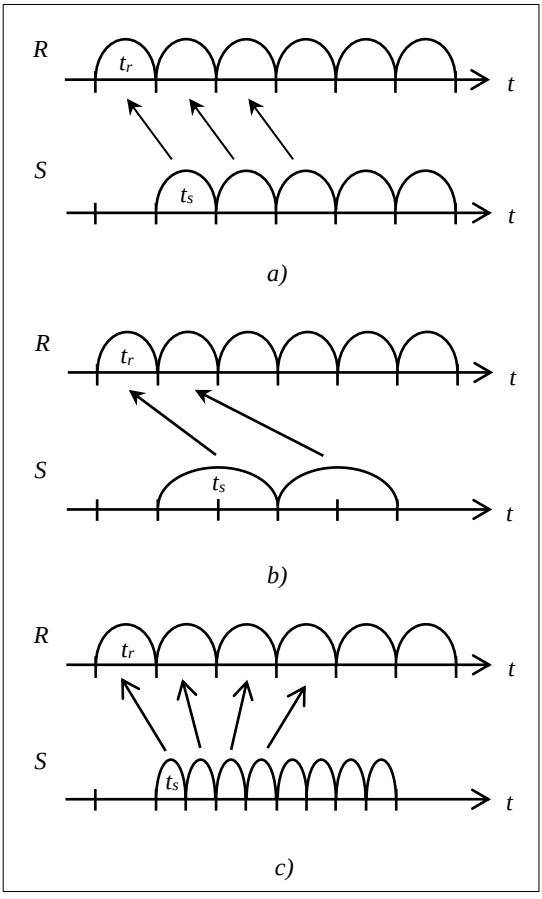


Рис. 7. Совместная работа рендер-потока и потока подкачки при: a) $t_r = t_s$; b) $t_r < t_s$; c) $t_r > t_s$
Fig. 7. Co-working of render-thread and swap-thread at: a) $t_r = t_s$; b) $t_r < t_s$; c) $t_r > t_s$

4.2 Визуализация ACMP-кадров

Чтобы визуализировать ACMP-кадр, необходимо загрузить его сжатый образ из видеофайла в оперативную память (RAM) и декодировать в RGB-изображение. Это занимает определенное время, которое препятствует плавности визуализации ACMP-кадров. Чтобы решить эту задачу в данной работе реализуется подкачка (загрузка в RAM и декодирование) впередистоящих ACMP-кадров в *кольцевой буфер* B_{ring} длины L, рассчитываемой с учетом размеров ACMP-видео, а также производительности используемых CPU и GPU.

Подкачка ACMP-кадров в буфер B_{ring} выполняется в параллельном потоке S (swap-thread), а управление потоком S осуществляет основной поток R визуализации (render-thread) на основе соотношения времен t_s и t_r подкачки и визуализации текущего ACMP-кадра. При этом обрабатываются следующие 3 возможных варианта совместной работы этих потоков. В *первом* варианте ($t_r = t_s$) пока используется текущий элемент (ACMP-кадр) буфера B_{ring}, в его предыдущий элемент подкачивается новый ACMP-кадр (см. рис. 7a). Данный случай является идеальным и не требует корректировки совместной работы потоков. Во *втором* варианте ($t_r < t_s$) рендер-поток выполняется быстрее и в какой-то момент догоняет поток подкачки (см. рис. 7b). В этом случае реализуется приостановка рендер-потока до полного заполнения буфера B_{ring} потоком подкачки (иначе начнут повторно воспроизводиться уже проигранные ACMP-кадры). *В третьем* варианте ($t_r > t_s$) возникает обратная ситуация: поток подкачки выполняется быстрее и в некоторый момент времени догоняет рендер-поток (см. рис. 7c). В этой ситуации реализуется приостановка потока подкачки до появления в буфере B_{ring} свободных для записи элементов (проигранных ACMP-кадров), иначе будут перезаписаны еще не проигранные ACMP-кадры.

5. Результаты

На основе предложенных методов и алгоритмов был разработан программный комплекс синтеза ACMP-видео виртуального окружения. Комплекс включает в себя *модуль захвата* ACMP-кадров, встраиваемый в подсистему визуализации СВО, а также *плеер* созданных ACMP-видео. Данное решение реализовано на языке C++ с применением графической библиотеки OpenGL и инструментария FFmpeg по кодированию/декодированию видео.

a) *b)*

Рис. 8. Апробация программного комплекса синтеза ACMP-видео: a) кадр из ACMP-видео полета вдоль трассы МКС; b) воспроизведенная из кадра панорама Земли

Fig. 8. Approbation of the program complex for ACMP-video synthesis: a) a frame from ACMP-video of the flight along ISS orbit; b) Earth panorama reconstructed from the frame

Созданное решение было апробировано в системе «Виртуальная Земля» [18] тренировки наблюдения объектов земной поверхности с орбиты Международной космической станции (МКС). С помощью модуля захвата было создано ACMP-видео полета вдоль трассы 2-го

суточного витка МКС для средства наблюдения с углом обзора ~ 40° и областью вывода 10^3x10^3 пикселов.

На рисунке 8a показан пример кадра, полученного ACMP-видео. Синтез ACMP-видео выполнялся с разрешением 3000x2000 пикселов и частотой 25 кадров/с на основе стандарта H.264 сжатия видео и контейнера MP4 (оба из международного стандарта MPEG-4). На рисунке 8b показано воспроизведение полученного ACMP-видео с помощью разработанного плеера. В процессе воспроизведения обучаемый совершает полет по орбите МКС, во время которого может исследовать виртуальную трехмерную поверхность Земли, поворачивая камеру в произвольном направлении. Апробация разработанного программного комплекса проводилась на компьютере (Intel Core i5 2.5 ГГц, 8 Гб RAM, 250Гб SSD), оборудованном видеокартой GeForce GTX 1050Ti (4Гб VRAM, 768 ядер).

Апробация показала, что при разрешении до 10^3x10^3 пикселов на грань куба создание ACMP-видео может выполняться в онлайн режиме, т.е. непосредственно в процессе работы СВО. Чтобы создавать ACMP-видео с более высоким разрешением, в будущем планируется развить предложенные методы и алгоритмы в направлении отложенной записи видеороликов [5], при которой в онлайн режиме захватываются только динамические параметры виртуальной сцены (сценарий), а преобразование сценария в видеоролик выполняется уже в офлайн режиме.

6. Заключение

В статье рассмотрена задача создания и воспроизведения качественного панорамного видео с обзором 360 градусов на основе проекции виртуального окружения в кубическую карту. Эвристический выбор разрешения карты является проблематичным, т.к. при недостаточном разрешении ухудшается качество видео и теряется эффект погружения в виртуальную среду, а при избыточном - возрастает битрейт видеопотока и возникают задержки воспроизведения. Для решения этой проблемы в статье предложен ACMP-подход (Adequate Cubemap Projection), при котором разрешение кубической карты определяется с учетом угла обзора камеры зрителя и разрешения экрана. В работе выведена функция зависимости отношения «пиксел карты/пиксел экрана» от ориентации камеры, характеризующая качество визуализации панорамы. На основе исследования этой функции был разработан метод вычисления разрешения кубической карты для качественной визуализации панорамы при всех возможных ориентациях камеры.

В работе предложены эффективные метод и алгоритм создания ACMP-видео на GPU, основанные на технологии рендеринга в текстуру, которые позволяют выполнять построение кубических панорам с привязкой к направлению взгляда наблюдателя. Также в данной статье предложены эффективные методы и алгоритмы воспроизведения ACMP-видео, основанные на отборе для визуализации видимых участков ACMP-кадра и адаптивной буферизации ACMP-кадров в параллельном потоке. Полученные методы и алгоритмы были реализованы в программном комплексе, который включает в себя модуль захвата ACMP-кадров и плеер ACMP-видео.

Разработанный комплекс был апробирован в системе «Виртуальная Земля» тренировки наблюдения объектов земной поверхности с орбиты МКС. Было создано ACMP-видео, с помощью которого обучаемый может вне родительской СВО выполнить полет по орбите МКС над виртуальной трехмерной поверхностью Земли и исследовать ее, поворачивая камеру. Апробация комплекса подтвердила адекватность разработанных методов и алгоритмов поставленной задаче и выявила пути их дальнейшего развития. Полученные научные и практические результаты могут быть применены в системах виртуального окружения, имитационно-тренажерных комплексах, системах научной визуализации и виртуальных лабораториях, образовательных приложениях и др.

Список литературы / References

[1]. Михайлюк М.В., Мальцев А.В., Тимохин П.Ю., Страшнов Е.В., Крючков Б.И., Усов В.М. Системы виртуального окружения для прототипирования на моделирующих стендах использования космических роботов в пилотируемых полетах. Пилотируемые полеты в космос, том 35, № 2, 2020 г., стр. 61-75. / Mikhaylyuk M.V., Maltsev A.V., Timokhin P.Yu., Strashnov E.V., Kryuchkov B.I., Usov V.M. Virtual Environment Systems for Simulating Robots in Manned Space Fligts. Pilotiruemye polety v kosmos. Manned Spaceflight, vol. 35, № 2, 2020, pp. 61-75 (in Russian).

[2]. Барладян Б.Х., Шапиро Л.З., Маллачиев К.А., Хорошилов А.В., Солоделов Ю.А., Волобой А.Г., Галактионов В.А., Ковернинский И.В. Система визуализации для авиационной ОС реального времени JetOS. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 57-70. DOI: 10.15514/ISPRAS-2020-32(1)-3 / Barladian B.Kh., Shapiro L.Z., Mallachiev K.A., Khoroshilov A.V., Solodelov Y.A., Voloboy A.G., Galaktionov V.A., Koverninskiy I.V. Rendering System for the Aircraft Real-Time OS JetOS. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020, pp. 57-70 (in Russian).

[3]. Михайлюк М.В., Торгашев М.А. Система визуализации “GLView” для имитационно-тренажерных комплексов и систем виртуального окружения. Труды 25-й Международной научной конференции GraphiCon, 2015, стр. 96-101 / Mikhaylyuk M.V., Torgashev M.A. The System of Visualization “GLView” for Simulators and Virtual Environment Systems. In Proc. of the 25th International Conference on Computer Graphics and Vision (GraphiCon 2015), 2015, pp. 96-101 (in Russian).

[4]. Страшнов Е.В., Мироненко И.Н., Финагин Л.А. Моделирование режимов полета квадрокоптера в системах виртуального окружения. Информационные технологии и вычислительные системы, № 1, 2020 г., стр. 85-94 / Strashnov E.V., Mironenko I.N., Finagin L.A. Simulation of quadcopter flight modes in virtual environment systems. Informacionnye tekhnologii i vichslitel’nye sistemy (Journal of Information Technologies and Computing Systems), № 1, 2020. pp. 85-94 (in Russian).

[5]. Тимохин П.Ю., Михайлюк М.В., Вожегов Е.М., Пантелей К.Д. Технология и методы отложенного синтеза 4K-стереороликов для сложных динамических виртуальных сцен. Труды ИСП РАН, том 31, вып. 4, 2019 г., стр. 61-72. DOI: 10.15514/ISPRAS-2019-31(4)-4. / Timokhin P.Yu., Mikhaylyuk M.V., Vozhegov E.M., Panteley K.D. Technology and methods for deferred synthesis of 4K stereo clips for complex dynamic virtual scenes. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 4, 2019, pp. 61-72 (in Russian).

[6]. El-Ganainy T., Hefeeda M. Streaming Virtual Reality Content. arXiv:1612.08350, 2016..

[7]. Ray B., Jung J., Larabi M.-C. A Low-Complexity Video Encoder for Equirectangular Projected 360 Video Content. In Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018, pp. 1723-1727

[8]. Li J., Wen Z., Li S., Zhao Y., Guo B., Wen J. Novel tile segmentation scheme for omnidirectional video. In Proc. of the IEEE International Conference on Image Processing (ICIP), 2016, pp. 370-374.

[9]. K.-T. Ng, S.-C. Chan, H.-Y. Shum. Data Compression and Transmission Aspects of Panoramic Videos. IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, № 1, 2005, pp. 82-95

[10]. Brown C. Bringing pixels front and center in VR video. Google AR and VR, March 14, 2017. Available at: https://www.blog.google/products/google-ar-vr/bringing-pixels-front-and-center-vr-video/, accessed 18.03.2020.

[11]. Kuzyakov E., Liu S., Pio D. Optimizing 360 Video for Oculus. Facebook F8 developers conference, 2016. Available at: https://developers.facebook.com/videos/f8-2016/optimizing-360-video-for-oculus/, accessed 18.03.2020.

[12]. Chen Z., Wang X., Zhou Y., Zou L., Jiang J. Content-Aware Cubemap Projection for Panoramic Image via Deep Q-Learning. Lecture Notes in Computer Science, vol. 11962, 2020, pp. 304-315.

[13]. Fu C.-W., Wan L., Wong T.-T., Leung C.-S. The Rhombic Dodecahedron Map: An Efficient Scheme for Encoding Panoramic Video. IEEE Transactions on Multimedia, vol. 11, № 4, 2009, pp. 634-644.

[14]. Kuzyakov E., Pio D. Next-generation video encoding techniques for 360 video and VR. Available at: https://code.facebook.com/posts/1126354007399553/next-generation-video-encodin, accessed 18.03.2020.

[15]. Segal M., Akeley K. The OpenGL Graphics System: A Specification. Version 4.6, Core Profile. The Khronos Group Inc., 2006-2018. Available at: https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf, accessed 18.03.2020.

[16]. FFmpeg. A complete, cross-platform solution to record, convert and stream audio and video. Available at: https://ffmpeg.org/, accessed 18.03.2020.

[17]. Timokhin P.Y., Mikhaylyuk M.V. Effective technology to visualize virtual environment using 360-degree video based on cubemap projection. In Proc. of International Conference on Computing for Physics and Technology (CPT2020), 2020.

[18]. Тимохин П.Ю. Моделирование видимого движения Земли вдоль участков суточной трассы МКС в космических видеотренажерах. Труды НИИСИ РАН, том. 9, № 6, 2019 г., стр. 111-117. DOI: 10.25682/NIISI.2019.6.0014 / Timokhin P.Yu. Simulation of visible Earth motion along daily tracks of ISS orbit in space simulators. Trudy NIISI RAN/Proc. of SRISA RAS, vol. 9, № 6, 2019, pp. 111-117 (in Russian).

Информация об авторах / Information about authors

Петр Юрьевич ТИМОХИН – старший научный сотрудник. Сфера научных интересов: компьютерная графика, визуализация.

Petr Yurievich TIMOKHIN – Senior Researcher. Research interests: computer graphics, visualization.

Михаил Васильевич МИХАЙЛЮК – доктор физико-математических наук, профессор, главный научный сотрудник. Сфера научных интересов: компьютерная графика, визуализация, системы виртуального окружения.

Mikhail Vasilievich MIKHAYLYUK – Doctor of Physical and Mathematical Sciences, Professor, Chief Researcher. Research interests: computer graphics, visualization, virtual environment systems.

Евгений Михайлович ВОЖЕГОВ – ведущий программист. Сфера научных интересов: компьютерная графика.

Evgeniy Mikhaylovich VOZHEGOV – Leading programmer. Research interests: computer graphics.

DOI: 10.15514/ISPRAS–2020–32(4)–6

CASR: анализ coredump файлов в ОС Linux и составление отчётов об ошибках

А.Н. Федотов, ORCID: 0000-0002-8838-471X <fedotoff@ispras.ru>
Ш.Ф. Курмангалеев, ORCID: 0000-0002-0558-2850 <kursh@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Несмотря на то, что при разработке программного обеспечения используются различные технологии и подходы, позволяющие диагностировать ошибки на ранних этапах разработки и тестирования, часть ошибок обнаруживается во время эксплуатации. Для пользователя ошибки часто выглядят как аварийное завершение программы во время работы. Для сбора отчётов об аварийных завершениях программ в операционную систему встраивается специальный компонент анализа. Такой компонент присутствует как в ОС Windows, так и в ОС на базе Linux, в частности в Ubuntu. Важным параметром является степень критичности найденной ошибки, причем данная информация полезна как разработчику дистрибутива, так и пользователю. В частности, пользователи, имея такую диагностику, могут принять организационно-технические меры до выхода исправления ошибки от разработчика программного обеспечения. В статье представлен CASR: инструмент анализа образа памяти в момент завершения процесса (coredump) и составления отчётов об ошибках. Инструмент позволяет проводить оценку критичности обнаруженного аварийного завершения путём анализа образа памяти, а также собирать необходимую информацию для разработчика, которая поможет исправить дефект. В качестве такой информации выступают версия дистрибутива ОС, версия пакета, карта памяти процесса, состояние регистров, значения переменных среды, стек вызовов, номер сигнала, который привёл к аварийному завершению, и т.д. Оценка критичности даёт возможность разработчику программного обеспечения в первую очередь исправить те ошибки, которые являются наиболее опасными. CASR позволяет обнаружить файлы и сетевые соединения, которые были открыты в момент аварийного завершения. Эта информация поможет воспроизвести ошибку, а также принять меры пользователям и администраторам в случае атаки на систему. Инструмент предназначен для работы на ОС Linux, поддерживает архитектуры x86/64, armv7 и может поставляться в виде пакета для дистрибутивов на базе Debian. Инструмент был успешно протестирован на нескольких ошибках, сведения о которых были получены из открытых источниках.

Ключевые слова: оценка критичности; дамп памяти; отчёты об ошибках

Для цитирования: Федотов А.Н., Курмангалеев Ш.Ф. CASR: анализ coredump файлов в ОС Linux и составление отчётов об ошибках. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 89–96 DOI: 10.15514/ISPRAS–2020–32(4)–6

CASR: core dump analysis and severity reporter tool

A.N. Fedotov, ORCID: 0000-0002-8838-471X <fedotoff@ispras.ru>
Sh.F.Kurmangaleev, ORCID: 0000-0002-0558-2850 <kursh@ispras.ru>
Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. Despite the fact that software development uses various technologies and approaches to diagnose errors in the early stages of development and testing, some errors are discovered during operation. To the user, errors often look like a program crash while running. To collect reports on program crashes, a special analysis

component is built into the operating system. Such a component is present in both Windows OS and Linux-based OS, in particular Ubuntu. An important parameter is the severity of the error found, and this information is useful to both the developer of the distribution kit and the user. In particular, users with such diagnostics can take organizational and technical measures before the release of a bug fix from the software developer. The article introduces CASR: a tool for analyzing a memory image at the time of a process termination (coredump) and reporting errors. The tool allows you to assess the severity of the detected crash by analyzing the memory image, as well as collect the necessary information for the developer to help fix the defect. Such information is: OS distribution version, package version, process memory card, state of registers, values of environment variables, call stack, signal number that led to abnormal termination, etc. Severity assessment enables the software developer to correct errors, which are the most dangerous in the first place. CASR can detect files and network connections that were open at the time of the crash. This information will help reproduce the error, and will help users and administrators take action in the event of an attack on the system. The tool is designed to work on Linux OS and supports x86 / 64, armv7 architectures and can be supplied as a package for Debian-based distributions. The tool has been successfully tested with several open source bugs.

Keywords: coredump; crash; error reports; critical estimation of software defects

For citation: Fedotof A.N., Kurmangaleev Sh.F. CASR: core dump analysis and severity reporter tool. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020, pp. 89–96 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–6

1. Введение

Безопасный цикл разработки программного обеспечения активно применяется как в России [1], так и за рубежом. Статический и динамический анализ позволяют обнаруживать ошибки, в том числе и критические ещё на стадии разработки. Тем не менее, уже в готовом программном обеспечении могут быть ошибки, которые будут обнаружены лишь на стадии эксплуатации программного продукта.

Для сбора отчётов об аварийных завершениях программ в операционную систему встраивается специальный компонент анализа. Такой компонент присутствует как в ОС Windows [2], так и в ОС на базе Linux, в частности в Ubuntu [3]. Таким образом, все обнаруженные аварийные завершения на машине пользователя могут быть проанализированы и отправлены разработчикам в Microsoft или сообществом разработчиков дистрибутива Ubuntu. Также компоненты создания и отправки отчётов об ошибках встроены в некоторое программное обеспечение напрямую, например, в браузеры или почтовые клиенты [4].

Таким образом, разработчики стороннего ПО вынуждены либо разрабатывать свои компоненты для сбора информации о сбоях в программном обеспечении, либо совсем отказаться от получения отчётов об ошибках. Кроме того, важную роль играет оценка критичности обнаруженного аварийного завершения [5]. Наиболее критичные аварийные завершения разработчики будут стараться исправить в первую очередь, а пользователи примут меры, чтобы обеспечить защиту своей системы, пока не выйдет исправление.

В работе рассматривается casr – инструмент анализа образа памяти в момент завершения процесса (coredump) в системах ОС Linux, который позволяет проводить оценку критичности найденного дефекта, а также собирать информацию, которая поможет разработчику при анализе аварийного завершения. Метод был протестирован на нескольких известных уязвимостях в системе Astra Linux 1.6 «Орёл».

2. Обзор схожих работ

2.1 Windows Error Reporting (WER)

WER [6] – система сбора отчётов об ошибках в операционных системах на базе Windows.

Миллионы устройств на сегодняшний день работают управлением ОС Windows, и на каждом из них присутствует WER. Основная задача этой системы собирать необходимую информацию для разработчиков компании Microsoft, которую они могут использовать для исправления ошибок: образ памяти, значения переменных среды, файлы логов и т.д. Основная сложность для WER – это огромный поток ошибок, который не возможно обработать человеку без автоматизации. Один из подходов это кластеризация отчётов. Отчёты, которые соответствуют одинаковым ошибкам помещаются в одну «корзину» и разработчик может смотреть на несколько отчётов, разбираясь с одной ошибкой. Кластеризация происходит на основе схожести стеков вызовов. При таком подходе могут возникать неточности, если в одной функции несколько ошибок, а каждая ошибка проявляется с одинаковым стеком вызовов.

По заявлениям авторов [6], разработчики обнаружили и исправили более 5000 ошибок, выявленных из отчётов WER в ОС Windows Vista. Основной недостаток данной системы в том, что она недоступна, а все отчёты, даже о сторонних программах отправляются в Microsoft, а не разработчикам этого ПО.

2.2 Ubuntu Apport

Apport [3] – система сбора отчётов об ошибках дистрибутивах Ubuntu, начиная с 6.10. Как и WER от Microsoft, Apport преследует те же цели – формирование отчётов об ошибках и отправка их разработчикам, в данном случае в поддержку дистрибутива Ubuntu. Использование данной системы имеет ряд преимуществ:

- отчёт формируется сразу при сбое, который не всегда легко воспроизвести;
- вся необходимая информация для отчёта собирается автоматически.

Информация об дистрибутиве, версия пакета, карта памяти процесса, состояние регистров, значения переменных среды, стек вызовов, номер сигнала, который привёл к аварийному завершению и т.д. – всё это входит в состав отчёта об ошибке. Важной особенностью, является то, что отправление отчёта в поддержку дистрибутива Ubuntu является необязательным. Таким образом, пользователь может генерировать отчёты для разработчиков стороннего ПО. Apport – система с открытым исходным кодом на языке Python. Одним из недостатков системы можно считать отсутствие оценки критичности аварийного завершения.

3. Casr

Casr позволяет анализировать образ памяти в момент завершения процесса (coredump), оценивать критичность аварийного завершения, а также формировать отчёты об ошибках. Анализ образа памяти в момент аварийного завершения стал возможен в ОС Linux начиная с ядра 2.6.19. Для этого, системе можно указать полный путь программы-анализатора, и система в момент аварийного завершения какой-либо программы направит содержимое образа памяти на стандартный поток ввода программы-анализатора. Из полученного образа памяти, а также от процесса исследуемой программы (он ещё живой в момент анализа) casr получает всю необходимую информацию для отчёта и оценки критичности.

Оценка критичности основывается проведении классификации аварийного завершения. Каждый класс в свою очередь, может принадлежать одной из трёх групп: эксплуатируемые классы аварийных завершений, возможно эксплуатируемые классы аварийных завершений, неэксплуатируемые классы аварийных завершений.

К эксплуатируемым ситуациям относятся такие аварийные завершения, которые максимально простые для перехвата потока управления программы. Например, аварийное завершение при выполнении инструкции вызова или возврата из функции, или при попытке доступа на выполнение по текущему адресу в указателе инструкций.

К потенциально эксплуатируемым относятся такие ошибки, которые требуют незначительных действий от атакующего для перехвата потока управления. Например, аварийное завершение при загрузке значения из памяти на регистр. Загруженное значение может контролироваться нарушителем и использоваться для передачи управления. К неэксплуатируемым аварийным относятся такие аварийные завершения, эксплуатация которых мало вероятна, например, деление на ноль или разыменование нулевого указателя.

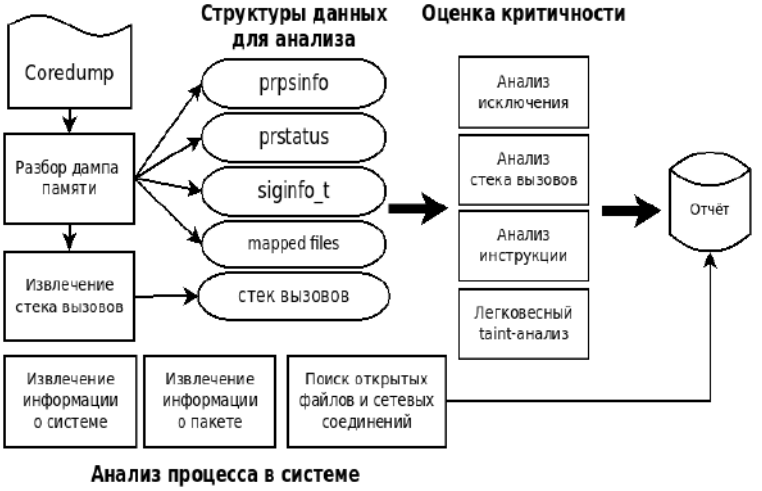


Рис. 1. Схема работы инструмента casr
Fig. 1. Casr workflow scheme

На рис. 1 представлена общая схема работы инструмента. Для оценки критичности casr оценивает состояние программы в момент аварийного завершения. Для этого из образа памяти, который сохранён в формате elf необходимо извлечь карту процесса, сохранённые регистры для всех потоков (prpsinfo), информацию о сигнале из-за которого возникло аварийное завершение (siginfo). Соответствующие структуры расположены в разделе записей (notes) core-файла. Затем анализируется номер сигнала, рассматриваются следующие сигналы: SIGSEGV, SIGABRT, SIGILL, SIGFPE.

При анализе SIGABRT, анализируется стек вызовов, на основе которого делается вывод о классе аварийного завершения.

При анализе SIGSEGV, SIGFPE необходимо провести дизассемблирование инструкции, на которой произошло аварийное завершение, выяснить причины (доступ к памяти чтение/запись/выполнение, семантика инструкции и т.д.). Затем отнести инструкцию к какому-то классу аварийного завершения.

Для аварийного завершения при загрузке значения из памяти в регистр проводится дополнительный анализ помеченных данных. Его цель – уточнить класс аварийного завершения. Рассмотрим работу легковесного анализа помеченных данных на примерах. Ниже приводится фрагмент базового блока для вызова виртуальной функции по таблице.

```
1. mov eax, dword ptr [eax] // аварийное завершение при чтении из памяти
2. mov eax, dword ptr [eax]
3. call eax //потенциальный перехват потока управления
```

Изначально множество помеченных регистров инициализируется загруженным регистром на шаге №1 (*eax*). За тем на шаге №2 происходит доступ к памяти на чтение по помеченному регистру. В этом случае, считается, что значение памяти тоже контролируется. Таким образом, загруженный регистр (*eax*) на шаге №2 тоже становится помеченным. На шаге №3 происходит инструкция вызова. Алгоритм проверяет, есть ли регистры из множества помеченных значений в операндах инструкции. В данном случае регистр *eax* помеченный,

что означает возможность перехвата потока управления при инструкции вызова. Из этого следует, что можно уточнить класс аварийного завершения и присвоить новый: аварийное завершение при инструкции вызова, обнаруженное легковесным анализом помеченных данных, который уже относится к эксплуатируемым классам.

Рассмотрим следующий пример.

```
1. mov esi , dword ptr [ebx]  // аварийное завершение при чтении из памяти
2. add esi , 4
3. mov dword ptr [esi] , eax // запись по контролируемому адресу
```

В этой ситуации алгоритм работает схожим образом. На шаге №3 регистр esi является помеченным, что свидетельствует о потенциально возможной ситуации записи контролируемого значения по контролируемому адресу (CWE-123). В связи с этим, можно уточнить класс аварийного завершения и присвоить новый: аварийное завершение на инструкции записи в память, обнаруженное легковесным анализом помеченных данных, который уже относится к эксплуатируемым классам.

Анализ проводится в рамках базового блока, в котором произошло аварийное завершение.

Сигнал SIGILL однозначно определяется классом аварийного завершения, который соответствует попытке выполнить неправильно сформированную инструкцию.

Кроме классификации аварийных завершений casr, как и apport собирает важную информацию для разработчика: информацию об дистрибутиве, версию пакета, карта памяти процесса, состоянии регистров, значения переменных среды, стек вызовов и т. д.

Важной особенностью casr является то, что в отличии от apport он также собирает дополнительную информацию об открытых файлах и сетевых соединениях в момент аварийного завершения. Эта информация может помочь при попытке воспроизведения аварийного завершения или при ответной реакции на сам факт аварийного завершения (блокировка сетевых соединений указанных адресов).

Вся эта информация формируется в итоговый отчёт, который может быть передан разработчику или специалисту по безопасности.

4. Детали реализации инструмента casr

Инструмент разработан на языке Rust и в своём составе имеет компоненты с открытым исходным кодом. Основными компонентами являются: библиотека для парсинга исполняемых файлов *libgoblin* [7], библиотека для дизассемблирования *capstone* [8,9], библиотека для получения стека вызовов *libunwind* [10]. В рамках разработки для библиотеки libunwind сделаны интерфейсы для вызова функций библиотеки из языка Rust (bindings), которые доступны в открытом доступе [11].

Поддерживаются следующие архитектуры для работы и анализа: x86/64, armv7. Также инструмент может быть собран статически, что упрощает развёртывание, если в системы отсутствуют необходимые библиотеки. Результаты работы (отчёты) хранятся в файлах формата json. Инструмент может поставляться в виде deb-пакета для дистрибутивов на базе Debian. Для просмотра отчётов реализована утилита casr-cli, предоставляющая терминальный пользовательский интерфейс просмотра.

5. Тестирование разработанного инструмента

Для тестирования инструмент был установлен в систему Astra Linux Common Edition 1.6 «Орёл». Для пакетов *imagemagick-6*, *ffmpeg*, *popler-utils* был сформирован набор входных данных, позволяющих воспроизвести ранее известные уязвимости, информация о которых была получена из открытых источников. Также была установлена уязвимая версия сервера nginx для проверки работоспособности инструмента при анализе CVE-2013-2028: уязвимость

переполнения буфера на стеке, которая возникает из-за целочисленного переполнения. На рис. 2 представлен фрагмент отчёта для уязвимости CVE-2013-2028.

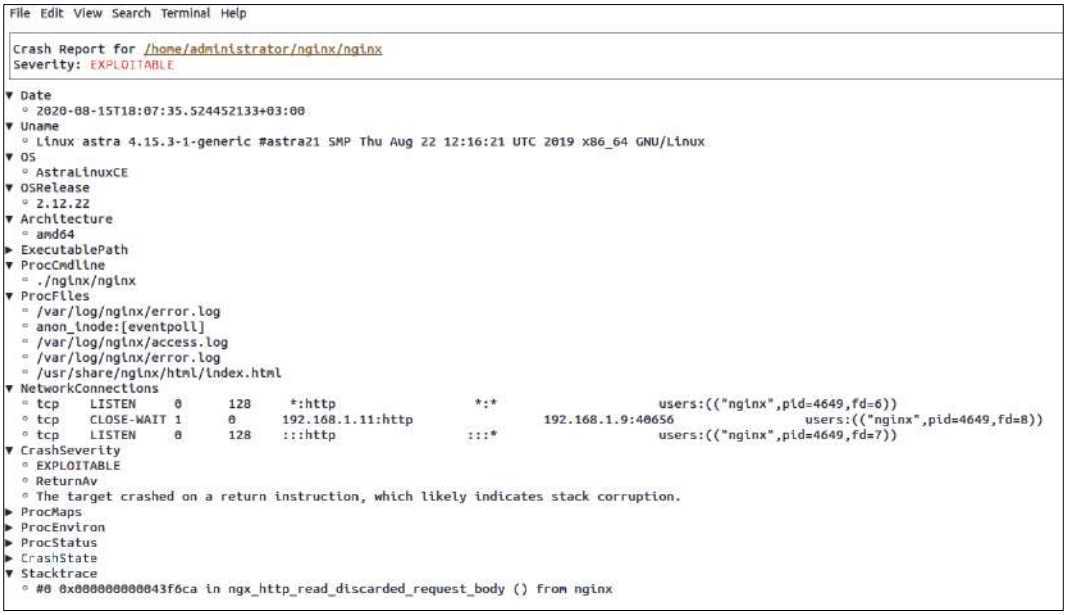


Рис. 2. Пример отчёта об ошибке инструмента casr
Fig. 2. Casr report example

Из основных полей можно отметить поле с оценкой критичности, которое указывает на то, что данное аварийное завершение произошло на инструкции возврата из функции и является эксплуатируемым. Так же стоит обратить внимание на сетевые соединения, активные момент аварийного завершения, из них видно, что с машины по адресу 192.168.1.9 пришёл http-запрос. Из полезных вещей можно отметить строку запуска программы, стек вызовов (в данном случае это одна функция, где произошло аварийное завершение, так как стековые кадры перезаписаны в результате переполнения буфера), открытые файлы.

Ниже приводится краткое описание отчётов об аварийных завершениях из пакетов, которые были установлены в системе.

Ffmpeg (CVE 2019-13390). Ошибка вызвана делением на ноль в функции *avio_enum_protocols.* Инструмент оценил данное завершение как неэксплуатируемое.

Ffmpeg (bug tracker id 8252) аварийное завершение произошло в функции *avfilter_transform* из-за обращения по нулевому указателю. Инструмент оценил данное завершение как неэксплуатируемое.

Imagemagick (bug tracker id 923). Ошибка произошла в функции *ExportQuntomPixels* – программа вызвала функцию *abort()*. Инструмент оценил данное завершение как неэксплуатируемое.

Popler-utils (bugs.fredesktop.org 85276). Ошибка произошла из-за деления на ноль. Инструмент оценил данное завершение как неэксплуатируемое.

Popler-utils (https://gitlab.freedesktop.org/poppler/poppler/-/issues/750). Ошибка возникает из-за нарушения доступа к памяти в функции *Splash::blitTransparent*. Инструмент оценил данное завершение как неэксплуатируемое.

Корректность работы инструмента проверялась путём анализа сохранённых образа памяти через отладчик *gdb*. Инструмент правильно оценил аварийное завершение для уязвимости CVE-2013-2028 и правильно установил адрес откуда пришёл http-запрос. Ошибки системных пакетах также были проверены посредством анализа образа памяти с помощью gdb. Casr

правильно установил класс аварийного завершения, стек вызовов, открытые файлы в момент падения, строку запуска программы.

6. Заключение

Сбор и анализ информации об аварийных завершениях используется как на уровне операционных систем (WER Microsoft, apport Ubuntu), так и на уровне отдельных программ (Firefox Mozila). В этих системах есть два основных недостатка: стороннему разработчику програмного обеспечения невозможно получить отчёт о работе его ПО (WER отправляет отчёты в Microsoft), и отсутствует оценка критичности аварийных завершений (apport Ubuntu).

В статье представлен инструмент Casr, который формирует отчёты об ошибках и оценивает критичность обнаруженных аварийных завершений. Инструмент предназначен для ОС Linux. Поддерживаемые процессорные архитектуры – x86/64, armv7. Отчёт об ошибках ормируется на основе анализа образа памяти (coredump) и информации о процессе в системе.

Сгенерированные отчёты могут быть помочь разработчикам исправить ошибки, возникшие в результате эксплуатации ПО, причём оценка критичности позволяет определить, какие ошибки (экслпуатируемые) нужно исправлять в первую очередь. Кроме того, casr потенциально может быть использован в качестве агента для хостовой системы обнаружения вторжений. Рассмотрим пример уязвимости CVE-2013-2028. Здесь casr определил аварийное завершение как эксплуатируемое и указал открытые сетевые соединения в момент аварийного завершения. Такую ситуацию можно оценить как атаку и автоматически заблокировать все входящие соединения с указанного адреса и порта до дальнейших действий от пользователя системы.

В качестве направления развития инструмента можно отметить интеграцию с хостовой системой обнаружения вторжений.

Список литературы / References

[1]. ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования. / GOST R 56939-2016 Information protection. Secure software development. General requirements (in Russian).
[2]. Dang Y., Wu Rongxin, Zhang H., Zhang D., Nobel P. Rebucket: A method for clustering duplicate crash reports based on call stack similarity. In Proc. of the 34th International Conference on Software Engineering (ICSE), 2012, pp. 1084-1093.
[3]. Apport. URL: https://wiki.ubuntu.com/Apport, accessed 25.08.2020.
[4]. Mozilla crash reporter. URL: https://support.mozilla.org/en-US/kb/mozillacrashreporter, accessed 25.08.2020.
[5]. Gdb ‘exploitable’ plugin. URL: https://github.com/jfoote/exploitable, accessed 25.08.2020.
[6]. Glerum K., Glerum K., Kinshumann K., Greenberg S., Aul G., Orgovan V., Nichols G., Grant D., Loihle G. Debugging in the (very) large: ten years of implementation and experience. In Proc. of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, 2009, pp. 103-116.
[7]. libgoblin. URL: https://github.com/m4b/goblin, accessed 25.08.2020.
[8]. Capstone. URL: https://github.com/aquynh/capstone, accessed 25.08.2020.
[9]. Capstone-rust. URL: https://github.com/capstone-rust, accessed 25.08.2020.
[10]. Libunwind. URL: https://www.nongnu.org/libunwind/, accessed 25.08.2020.
[11]. libunwind-rs. URL: https://github.com/xcoldhandsx/libunwind-rs, accessed 25.08.2020.

Информация об авторах / Information about authors

Шамиль Фаимович КУРМАНГАЛЕЕВ – кандидат физико-математических наук, старший научный сотрудник. Сфера научных интересов: Статический анализ бинарного кода, динамический анализ, фаззинг, обфускация, символьное выполнение.

Shamil Faimovich KURMANGALEEV – Ph.D. of Physical and Mathematical Sciences, Senior

Researcher. Research interests: Static analysis of binary code, dynamic analysis, fuzzing, obfuscation, symbolic execution.

Андрей Николаевич ФЕДОТОВ – кандидат технических наук, младший научный сотрудник. Сфера научных интересов: динамический анализ, символьное выполнение, поиск ошибок в программах.

Andrey Nikolaevich FEDOTOV – Ph.D. in Engineering sciences, junior researcher. Research interests: dynamic analysis, symbolic execution, search for errors in programs.

DOI: 10.15514/ISPRAS–2020–32(4)–7

Отладчик параллельных программ для ОС Linux

А.Б. Киселев, ORCID: 0000-0002-6124-2359 <abkiselev@vniief.ru>
С.Н. Киселев, ORCID: 0000-0002-4236-6516 <snkiselev@vniief.ru>
Российский федеральный ядерный центр –
Всероссийский научно-исследовательский институт экспериментальной физики,
607188, Россия, Нижегородская область, г. Саров, пр. Мира, 37

Аннотация. В статье представлен отладчик параллельных программ, написанных на языке программирования Си/Си++ или Фортране, которые предназначены для выполнения на высокопроизводительных вычислительных системах. В работе раскрывается схема взаимодействия компонентов отладчика параллельных программ, представлен алгоритм обработки результатов профилирования программы с помощью встроенных средств профилирования. Описаны возможности графического интерфейса пользователя и отладчика в целом. В статье рассказано о развитии отладчика параллельных программ, в частности о реализации коммуникационной древовидной схемы соединения его компонентов между собой, о режиме неинтерактивной отладки, о поддержке графических ускорителей корпорации Nvidia.

Ключевые слова: отладка параллельной программы; параллельный отладчик; CUDA

Для цитирования: Киселев А.Б., Киселев С.Н. Отладчик параллельных программ для ОС Linux. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 97-114. DOI: 10.15514/ISPRAS–2020–32(3)–7

A debugger of parallel programs for OS Linux

A.B. Kiselev, ORCID: 0000-0002-6124-2359 <abkiselev@vniief.ru>
S.N. Kiselev, ORCID: 0000-0002-4236-6516 <snkiselev@vniief.ru>
Russian Federal Nuclear Center – All-Russian Research Institute of Experimental Physics,
37 Mira st., Sarov, Nizhny Novgorod region, 607188, Russia

Abstract. The paper presents a debugger for parallel programs in C/C++, or FORTRAN, which are executed in high-performance computers. The debugger's program components and mechanism of their interaction are described. The graphic user's interface capabilities are discussed and the profiling procedure using built-in profiling tools is described. The paper contains of the description of the new parallel debugger capabilities such as a communication treelike scheme of his components connection, and a non-interactive debugging mode, and the support of Nvidia's graphic accelerators. Currently, the debugger provides launching of debug jobs in the systems of batch processing of jobs such as Open PBS / Torque, SLURM, and CSP JAM but it can be configured for other systems. The PD debugger allows to debug program processes and threads, manage breakpoints and watchpoints, logically divide program processes into subsets, manage them, change and view variables, and profile the debugged program using the free Google Performance Tools and mpiP. The PD debugger is written in the Java programming language, intended for debugging programs on Unix / Linux operating systems, and it uses free software components such as SwingX, JHDF5, Jzy3D, RSyntaxTextArea, and OpenGL.

Keywords: parallel program debugging; parallel debugger; CUDA

For citation: Kiselev A.B., Kiselev S.N. A debugger of parallel programs for OS Linux. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 97–114 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–7

1. Введение

Отладка параллельного математического программного комплекса, предназначенного для выполнения на высокопроизводительной вычислительной системе (ВС), – важный этап разработки программного обеспечения, требующий применения специального программного средства – отладчика.

Для отладки на ВС программисты могут использовать как коммерческие, так и свободно распространяемые отладчики. Коммерческие отладчики – это TotalView [1] и Allinea DDT [2]. Их лицензии ограничивают пользователей определенным количеством одновременно отлаживаемых процессов. Свободно распространяемые отладчики не имеют таких ограничений. Среди зарубежных разработок выделяется свободно распространяемый отладчик проекта Eclipse Parallel Tools Platform [3] (PTP), который за несколько лет развития получил необходимый набор возможностей для отладки параллельной программы. Отечественные программные средства отладки – отладчик параллельных программ PDB [4], разработанный и использовавшийся в РФЯЦ-ВНИИЭФ, GEPARD [5] (СО РАН), диалоговый отладчик программ, написанных на непроцедурном языке НОРМА [6] (ИПМ им. М.В. Келдыша, РАН), – судя по отсутствию новых публикаций и исходных кодов, не развиваются.

Описанный в данной статье отладчик PD (parallel debugger) [7] заполняет свободное место в ряду средств отладки параллельных программ, разрабатываемых для отечественных ВС. Отладчик PD обеспечивает отладку программ на Си/Си++ или Фортране. В отладчике PD используются усовершенствованные авторами статьи версии GNU debugger (GDB) [8]. Графический интерфейс отладчика похож на графический интерфейс Allinea DDT, его можно настроить на сочетание *горячих клавиш* и цветовое оформление исходного текста отладчиков Microsoft Visual Studio, Eclipse, IDEA и Allinea DDT. В настоящее время отладчик обеспечивает запуск отладочных заданий в системах пакетной обработки заданий Open PBS/Torque [9], SLURM [10] и СПО JAM [11], но может быть настроен и на другие системы.

Отладчик PD позволяет отлаживать процессы и потоки программы, управлять точками прерывания и наблюдения, логически делить процессы программы на подмножества, управлять ими, изменять и просматривать переменные, а также выполнять профилирование отлаживаемой программы с использованием свободно распространяемых профилировщиков Google Performance Tools [12] и mpiP [13].

Отладчик PD написан на языке программирования Java, предназначен для отладки программ в ОС Unix/Linux, в нем применяются такие свободно распространяемые программные компоненты, как SwingX [14], JHDF5 [15], Jzy3D [16], RSyntaxTextArea [17] и OpenGL [18].

2. Программные компоненты отладчика

Отладчик PD состоит из программы графического интерфейса пользователя, сервера команд и сообщений, агента (рис. 1). В качестве базового отладчика используется GDB. Графический интерфейс и сервер команд и сообщений выполняются в разных потоках одной программы. Вариант отладки на ВС подразумевает, что они запускаются пользователем на инструментальном сервере ВС. Сервер команд и сообщений посылает программным агентам MI-команды [8] (команды машинно-ориентированного текстового интерфейса GDB), а программные агенты, в свою очередь, пересылают их отладчикам GDB. Информацию о результатах выполнения команд (сообщения) отладчики GDB передают программным агентам, которые пересылают её серверу команд и сообщений. Программные агенты не только пересылают информацию, но и контролируют стандартные потоки процессов программы.

В ходе выполнения программы отладчики GDB формируют асинхронные сообщения, которые не связаны с MI-командой, потому что вызваны, например, срабатыванием точки прерывания или наблюдения. Такие сообщения обрабатываются отдельно, а содержащаяся в

них информация отображается во всплывающем графическом окне, чтобы пользователь её не пропустил.

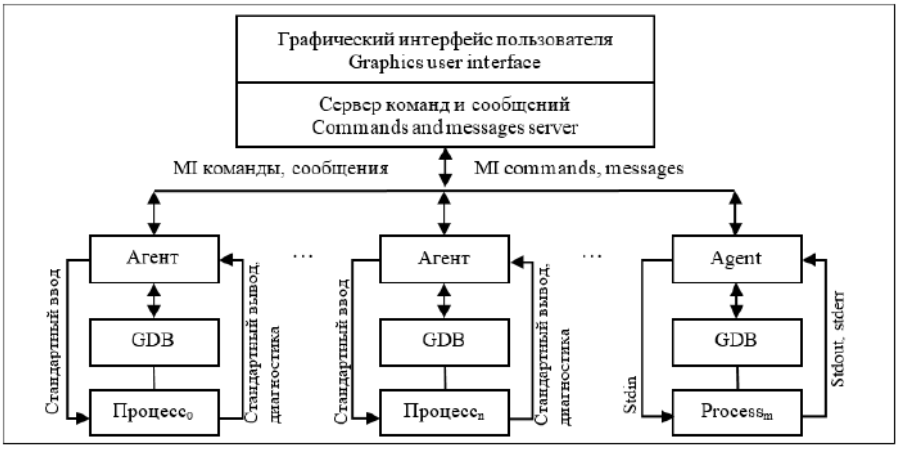


Рис. 1. Схема отладчика PD
Fig. 1. PD debugger schematic

Программные агенты, отладчики GDB и процессы параллельной программы запускаются на вычислительных узлах ВС.

2.1 Программа графического интерфейса пользователя, сервер команд и сообщений

Управление отладкой осуществляется посредством программы графического интерфейса, которая обеспечивает отображение списка названий исходных файлов, значений переменных программы, стандартной выдачи и диагностики процессов, их состояний, загрузку и сохранение параметров сессии - кодировка, шрифт и его размер, точки останова, наблюдения и т.д.

Сервер команд и сообщений выполняется в отдельном программном потоке, он необходим для связи с программными агентами. Сервер формирует текстовые команды машинно-ориентированного командного интерфейса GDB, посылает команды программным агентам, обрабатывает результаты их выполнения и передает их программе графического интерфейса.

2.2 Программный агент

Программный агент реализован для передачи MI-команд отладчику GDB, информации на стандартный ввод отлаживаемого процесса, сообщений с результатами их выполнения, стандартной выдачи и диагностики отлаживаемого процесса серверу команд и сообщений. Взаимодействие программного агента с отладчиком GDB осуществляется посредством псевдотерминалов[1]. Процесс отлаживаемой программы также использует псевдотерминалы для ввода/вывода информации.

Важной функцией программного агента является обработка результатов профилирования программы.

[1] Псевдотерминал – эмулятор терминала в ОС Unix/Linux (PTY), псевдоустройство, используемое для взаимодействия пользователя с локальным или удаленным компьютером.

2.3 Базовый отладчик GDB

Отладчик GDB обеспечивает отладку процессов программы. Управление отладчиком GDB осуществляется с помощью команд его текстового машинно-ориентированного интерфейса (MI).

В отладчике PD используется одна из модифицированных авторами статьи версий GDB – 7.11.1 или 7.12.1. Внесенные в отладчик GDB исправления позволили повысить надежность его функционирования, благодаря им пользователь обеспечен информацией о модулях Фортран-программы, функциях или процедурах, о типах указателей на массивы и многом другом. Для отладки программы, например, на графических ускорителях может быть использована версия отладчика GDB без модификаций[2], но в этом случае перечисленная выше информация отображаться не будет.

3. Опции компилятора

Для отладки программы её необходимо скомпилировать с ключом *-g* и использовать минимальный уровень оптимизации (*-O0*) или вовсе отключить её, поскольку при включенной оптимизации компилятор может переставить инструкции так, что при выполнении программа будет *скакать* по строкам исходного текста. Кроме этого, вместо ключа *-fomit-frame-pointer* лучше применять ключ *-fno-omit-frame-pointer*, который разрешает использование регистра указателя стека процессора.

В случае использования компилятора фирмы Intel не рекомендуется использовать ключ *-ax,* действие которого не позволяет отладчику GDB получить нужную информацию из стека программы. Некоторые компиляторы этой фирмы по умолчанию не добавляют в объектный файл программы расширенную отладочную информацию, поэтому компилятору требуется указать ключ *-debug all*. Компилятор GNU Фортран не включает информацию о модулях в исполняемый файл Фортран-программы, поэтому при отладке скомпилированной им программы вкладка *Fortran modules* в отладчике PD не отображается.

4. Шаблон задания

Отладчик PD позволяет выполнять запуск отладочных заданий в трех системах пакетной обработки заданий – СПО JAM, Open PBS/Torque и SLURM. Каждой системе соответствует отдельный файл с шаблоном задания. Ниже в качестве примера приведен файл с шаблоном задания для SLURM:

```
#!/bin/bash
# submitjob=sbatch
# canceljob=scancel :JOBID
# signaljob=sbatch --signal=:SIGNAL :JOBID
# regexpjob=\D+(\d+)
# showqueue=squeue
#SBATCH -U :COMMENT
#SBATCH -t :WALLTIME
#SBATCH -N :NODES --ntasks-per-node=:PPN
#SBATCH -o dbg.o%j -e dbg.e%j
#SBATCH -J debug
export :ENV
srun ${PD_HOME}/bin/agent :EXEC :ARGS
```

В верхней части примера находятся строки, содержащие служебные директивы (submitjob, canceljob, signaljob, regexpjob и showqueue), с помощью которых осуществляется передача

2 Разработчики GDB реализовали в версии 9.1 аналогичные MI-команды – -symbol-info-functions, -symbol-info-modules, -symbol-info-module-functions и -symbol-info-module-variables. Эти команды будут использованы в следующей версии отладчика PD.

задания SLURM, удаление и другие действия. Регулярное выражение \D+(\d+) позволяет из сообщения о постановке задания в очередь SLURM (например, *Submitted batch job 123456*) выделить идентификатор задания (123456). При взаимодействии с системой пакетной обработки заданий вместо служебных выражений :JOBID, :SIGNAL, :COMMENT, :NODES и т.д. отладчик PD подставит значения, которые пользователь укажет в графическом окне ввода атрибутов задания.

Утилита *srun* сначала запустит на вычислительных узлах ВС программные агенты отладчика PD, а они в свою очередь вызовут исполняемый файл программы, используя параметры - название исполняемого файла отлаживаемой программы (:EXEC) и её аргументы (:ARGS).

Если на ВС функционирует система пакетной обработки заданий, которая не поддерживается отладчиком PD, то администратор достаточно легко сможет сформировать для неё шаблон задания, пользуясь руководством администратора отладчика PD и редактором шаблона задания, который можно вызвать из графического окна отладчика.

5. Алгоритм отладчика

Для отладки параллельной программы на ВС пользователь должен с помощью графического интерфейса отладчика PD сформировать задание и передать его системе пакетной обработки заданий. Для этого в отладчике PD реализовано отдельное графическое окно, в котором пользователь должен указать название исполняемого файла программы, каталог с её исходными файлами, если файлы были перемещены в него после компиляции и сборки, входные параметры программы, количество требующихся для её выполнения вычислительных узлов и процессоров, переменные окружения и другие атрибуты.

После подтверждения ввода информации пользователем программа графического интерфейса считывает содержимое файла с шаблоном задания. Служебные выражения в шаблоне задания заменяются введенными пользователем значениями, в задание добавляется переменная окружения с IP-адресом и сетевым портом компьютера, на котором выполняется сервер команд и сообщений. Затем задание передаётся системе пакетной обработки заданий.

После старта задания на вычислительных узлах первыми запускаются программные агенты. Каждый программный агент создаёт псевдотерминалы для взаимодействия с отладчиком GDB и процессом программы, а также посылает на указанный IP-адрес сообщение о готовности к работе. Отладчик GDB запускает исполняемый файл программы.

Определив, что все программные агенты запущены, программа графического интерфейса с помощью MI-команд *-file-list-exec-source-files* и *-info-modules* получает от отладчика GDB информацию об исходных файлах программы, процедурах/функциях и модулях. Информация обрабатывается и отображается в графическом окне отладчика PD. Далее всем программным агентам посылается команда запуска исполняемого файла *-exec-run --start*. Программные агенты передают её отладчикам GDB. От них агенты принимают информацию о запуске и останове программы, которую пересылают серверу команд и сообщений. Сервер команд и сообщений преобразует её во внутреннее представление и передаёт программе графического интерфейса.

Сообщение с результатом выполнения MI-команды *-exec-run --start* содержит номер строки и название исходного файла программы. Они используются в графическом интерфейсе отладчика для отображения исходного текста и подсветки строки программы, на которой было приостановлено её выполнение. После появления исходного текста программы в графическом интерфейсе отладчика пользователь может расставлять точки прерывания и наблюдения, просматривать и изменять программные переменные, выполнять программу по шагам. Любое перечисленное действие порождает MI-команду, которая с помощью сервера команд и сообщений отсылается программным агентам, отладчикам GDB, а затем результаты её выполнения, пройдя программные компоненты в обратном порядке, передаются программе графического интерфейса.

Поясним вышесказанное на примере отладки трех процессов. Клик мышью по кнопке *Step over* в графическом окне отладчика вызовет обращение к подпрограмме-обработчику данного действия. В ней будет выполнено обращение к серверу команд и сообщений, который сформирует MI-команду *-exec-next*. Сервер пошлет её отладчикам GDB процессов 0-2 (для простоты понимания участие программных агентов опускаем):

```
Process 0 172.17.133.29 ->MI_COMMAND 37-exec-next
Process 1 172.17.133.30 ->MI_COMMAND 37-exec-next
Process 2 172.17.133.31 ->MI_COMMAND 37-exec-next
```

и получит от них ответы, что процессы выполняются:

```
Process 0 172.17.133.29 ->MI_OUTPUT 37^ running
Process 1 172.17.133.30 ->MI_OUTPUT 37^ running
Process 2 172.17.133.31 ->MI_OUTPUT 37^ running
```

Полученная сервером команд и сообщений информация будет передана программе графического интерфейса, которая в графическом окне изменит состояния процессов на *выполняется*. После завершения MI-команды *-exec-next* GDB пришлет серверу команд и сообщений информацию об останове процессов:

```
Process 0 172.17.133.29->MI_OUTPUT *stopped, reason="end-stepping-range",
frame={addr="0x000000000040065f", func="main", args=[{name="argc",
value="3"}], file="hello3.c", fullname="/home/test/hello3.c", line="48"},
thread-id="1", stopped-threads="all", core="1"
Process 1 172.17.133.30 ->MI_OUTPUT *stopped, reason="end-stepping-
range", frame={addr="0x000000000040065f", func="main",
args=[{name="argccc", value="3"}], file="hello3.c",
fullname="/home/test/hello3.c",line="48"}, thread-id="1", stopped-
threads="all", core="0"
Process 2 172.17.133.31 ->MI_OUTPUT *stopped, reason="end-stepping-
range", frame={addr="0x000000000040065f", func="main",
args=[{name="argccc",value="3"}], file="hello3.c",
fullname="/home/test/hello3.c", line="48"}, thread-id="1", stopped-
threads="all",core="1"
```

Сервер команд и сообщений преобразует текстовую информацию ответов в двоичную форму и передаст данные программе графического интерфейса, которая изменит состояния процессов на *остановлен*, отобразит в графическом окне содержимое файла /home/test/hello3.c и выделит цветом 48-ю строку. Кроме этого, для обновления информации о программных переменных, кадрах стека и потоках первому ответившему отладчику GDB будут посланы MI-команды *-thread-info*, *-thread-list-ids*, *-stack-list-locals*, *-stack-list-frames* и *-stack-list-arguments*. Остальным GDB-отладчикам перечисленные MI-команды посланы не будут, поскольку в графическом интерфейсе отладчика PD нет возможности отображать данные всех процессов одновременно. Для установки точек прерывания и наблюдения используются MI-команды *-break-insert* и *-break-watch*.

Перед завершением сессии отладки отладчик PD всегда записывает информацию об установленных точках наблюдения, прерывания, наблюдаемых переменных программы, настройках графического окна в файл с параметрами отладочной сессии, а затем удаляет задание из системы пакетной обработки заданий. При повторном запуске отладчика PD параметры отладочной сессии автоматически восстанавливаются.

6. Профилирование и обработка результатов

Профилирование отлаживаемой программы в отладчике PD осуществляется с помощью Google Performance Tools (GPT) и mpiP. GPT собирает информацию об эффективности

использования процессора и оперативной памяти, а профилировщик mpiP MPI[3]-метрики программы. Результаты профилирования записываются в файлы, которые по запросу пользователя обрабатывает программный агент и пересылает серверу команд и сообщений.

Ниже приведен фрагмент содержимого файла с информацией об использовании памяти и вкратце описан алгоритм его разбора программным агентом.

```
heap profile: 49675: 51865101 [ 49755: 85442155] @ heapprofile       (1)
6384: 8388608 [ 16384: 8388608] @ 0x00403d1b 0x00400e35              (2)
0x347641d994 0x00400cb9
128:  131072  [ 128:   131072 ] @ 0x004040f5 0x00400e35
0x347641d994 0x00400cb9
1:    69      [   1:   69     ] @ 0x347c09b801 0x347c09c305
0x347c09c4b2
MAPPED_LIBRARIES:                                                    (3)
00400000-00405000         r-xp 00000000 00:00 6611789
/home/test/hello3
3475400000-347541c000     r-xp 00000000 00:00 5658721
/lib64/ld-2.5.so
7fc9fc72f000-7fc9fc739000 r-xp 00000000 00:00 24259788
/lib/libunwind.so.8.0.1
```

Информация в файле логически разделена на три секции. В секции (1) находится заголовок, который содержит общее количество программных объектов, использующих память к моменту завершения программы, количество байтов занимаемой памяти, а также интегральные значения, относящиеся ко всем созданным программным объектам. Значения секции (1) используются для вычисления процентных соотношений.

В секции (2) находится информация об использовании памяти с детализацией по строкам исходного текста программы, в которых выделяется память. В левой части (2) находятся числа, в которых зафиксировано количество использующих память программных объектов (к моменту завершения программы), а также количество байтов используемой ими памяти в целом.

Секция (3) содержит информацию о библиотеках, которые использовались в процессе работы программы. В левой части находятся диапазоны адресов, которые связаны с программной библиотекой или исполняемым файлом программы, значения сдвига от начала файла и т.д. Правый столбец содержит названия файлов библиотек или исполняемых файлов.

Обработка файла с результатами профилирования начинается с первой строки секции (2), из которой программный агент считывает первое шестнадцатеричное число после амперсанда. В примере это 0x00403d1b. Число 0x00403d1b – это адрес вызова программной функции, который находится в одном из диапазонов адресов секции (3). По диапазону программный агент находит название файла, в котором следуют искать информацию. Так, значение 0x00403d1b находится в диапазоне адресов от 00400000 до 00405000, оно принадлежит адресному пространству файла /home/test/hello3. С помощью утилиты *addr2line*, учитывая адрес и название файла, программный агент определяет названия исходного файла и функции, а также номер строки исходного текста программы.

Если адрес соответствует динамической библиотеке, то из адреса секции (1) вычитаются адрес текстовой секции файла динамической библиотеки и смещение текстовой секции относительно начала файла, которые вычисляются с помощью утилиты *objdump*. Полученный адрес используется при определении названий функции, исходного файла и номера строки исходного текста программы.

По запросу пользователя программный агент может суммировать информацию, относящуюся к одной и той же исходной строке, функции или файлу.

[3] MPI (message passing interface) – программный интерфейс обмена данными между процессами параллельной программы.

7. Графический интерфейс пользователя

Отладчик PD позволяет отлаживать программу на ВС, на рабочем (локальном) компьютере пользователя, запускать и останавливать процессы программы, расставлять точки прерывания и наблюдения, просматривать и изменять программные переменные; он отображает состояния процессов и потоков, их стандартный вывод и диагностику.

С помощью графического интерфейса обеспечивается:

- отображение списка названий исходных файлов программы и содержащихся в них программных функций;
- логическое деление процессов программы на подмножества для их раздельной отладки;
- сравнение программных переменных в процессах и программных потоках;
- отображение значений элементов массивов, представление значений одно- и двумерных массивов в виде геометрической фигуры, запись значений массивов в файл;
- отображение результатов профилирования программы, запись результатов в файл;
- установка сочетания *горячих клавиш*, отображение исходного текста отлаживаемой программы в соответствии с пользовательскими настройками;
- поиск информации в исходных файлах программы;
- автоматическое сохранение и восстановление параметров отладочной сессии.

Графическое окно параллельного отладчика вызывается с помощью сценария интерпретатора shell *pdx*. Для создания отладочной сессии пользователь вызывает графическое окно ввода атрибутов отладочного задания или окно создания отладочной сессии на локальном компьютере. Например, если пользователь выбрал отладку программы на ВС, то в графическом окне кроме исполняемого файла и аргументов программы он должен указать количество вычислительных узлов и процессоров, время выполнения отладки и передать атрибуты задания системе пакетной обработки заданий. При успешном старте отладочной сессии в окне графического интерфейса появляется содержимое исходного файла программы, отображаются значения локальных переменных, состояния процессов, программных потоков и т.д.

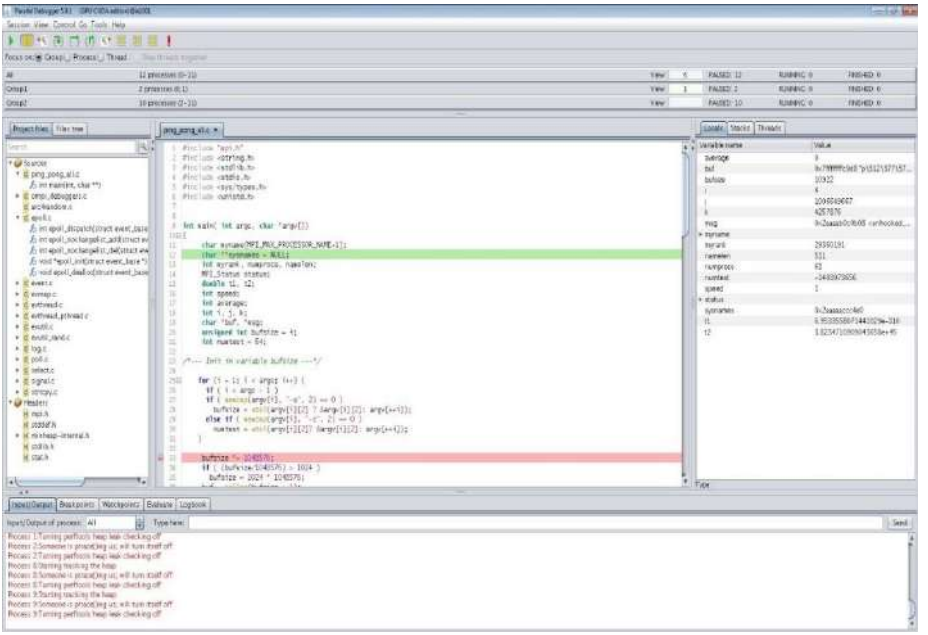


Рис. 2. Графическое окно отладчика PD
Fig. 2. PD debugger graphics window

7.1 Графическое окно отладчика

На рис. 2 показано графическое окно отладчика PD в начале отладки двенадцати процессов MPI-программы ping-pong (тест коммуникационной подсистемы ВС). Строка номер 12 подсвечена зеленым цветом, так как будет выполнена на следующем шаге. В строке 33 установлена точка прерывания: левее номера строки отображается соответствующая пиктограмма, а сама строка выделена красным цветом.

Слева от текста программы показан список исходных файлов и содержащихся в них функций.

Справа от него отображены значения локальных переменных функции main(). Внизу на вкладке *Input/Output* видны сообщения GPT – *Starting tracking the heap* и *Someone is ptrace()ing us; will turn itself off* с номерами процессов, от которых они были получены. На этой вкладке отображаются стандартный вывод и диагностика программы. На рис. 2 показано, что процессы программы разбиты на три подмножества - группы *All* (ранги[4] 0-11), *Group1* (ранги 0 и 1) и *Group2* (ранги 2-11). Группа *All* формируется отладчиком PD автоматически и содержит информацию обо всех отлаживаемых процессах программы. Группы *Group1* и *Group2* сформированы пользователем. *Group1* выбрана для отладки. Двенадцать процессов группы *All* остановлены, выполняющихся и завершившихся процессов нет. В группах *Group1* и *Group2* остановлены два и десять процессов соответственно.

При отладке группы процессов можно указать ранг процесса, информацию о переменных которого отладчик PD должен отображать в графическом окне. Так, в группе *All* указан шестой, а в *Group1* первый ранг.

7.2 Графическое окно просмотра многомерных массивов

Для просмотра значений элементов многомерных массивов реализовано графическое окно (рис. 3), в котором пользователь может отфильтровать значения, указав отображаемый диапазон, увидеть статистические данные – количество ±nan, ±inf, чисел меньше, больше или равных нулю и т.д.

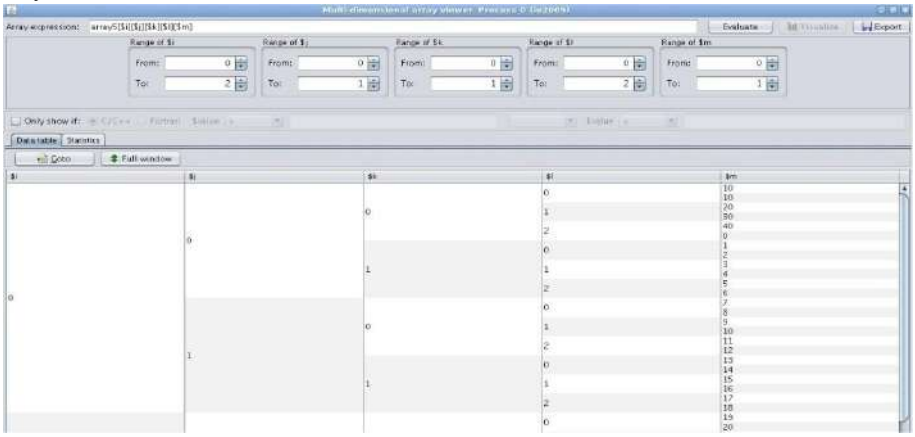


Рис. 3. Графическое окно просмотра значений многомерных массивов
Fig. 3. The graphical window for viewing values of multidimensional arrays

Кроме этого, пользователь имеет возможность записать информацию в файл в формате HDF5 (hierarchical data format – формат файла иерархической структуры) или CSV (comma-separated value – значения, разделенные запятыми). С помощью программного интерфейса OpenGL можно отобразить массив в форму плоской или объемной геометрической фигуры и оценить значения. Например, на рис. 4 изображена фигура, в которой можно визуально обнаружить точки, далеко выходящие за рамки некоторого диапазона.

[4] Ранг – номер MPI-процесса.

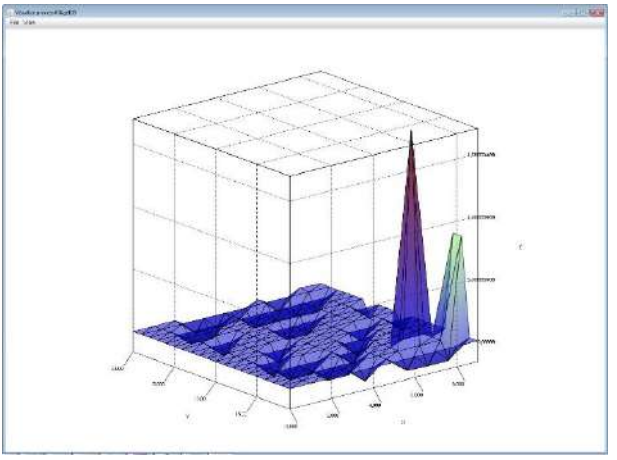
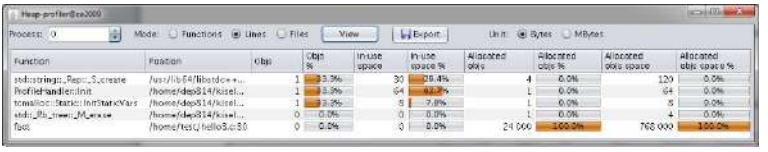


Рис. 4. Графическое представление значений многомерного массива
Fig. 4. A graphic representation of multidimensional array's values

7.3 Графическое окно просмотра результатов профилирования памяти

Результаты профилирования памяти отображаются в графическом окне, пример которого приведен на рис. 5. Значения, представленные в примере, связаны со строками исходного текста, поскольку выбран режим *Lines*. Последняя строка таблицы свидетельствует о том, что в тридцатой строке файла hello3.c было создано 24000 программных объектов, то есть 100% их общего количества, но к моменту завершения программы вся занятая ими память была освобождена, о чем свидетельствует 0 в столбце *In-use space*.



Function	Position	Obj	Obj %	In-use space	In-use space %	Allocated objs	Allocated objs %	Allocated objs space	Allocated objs space %
std::string::_Rep::_S_create	/usr/lib64/libstdc++...	1	3.3%	30	28.4%	4	0.0%	129	0.0%
ProfilerHandler::Init	/home/dep314/kisel...	1	3.3%	64	61.7%	1	0.0%	64	0.0%
tcmalloc::Static::InitStaticVars	/home/dep314/kisel...	1	3.3%	8	7.8%	1	0.0%	8	0.0%
std::_Rb_tree<...,_M_emplace	/home/dep314/kisel...	0	0.0%	0	0.0%	1	0.0%	4	0.0%
foo	/home/test/hello3.c:30	0	0.0%	0	0.0%	24 000	100.0%	768 000	100.0%

Рис. 5. Графическое окно с результатами профилирования программной памяти
Fig. 5. The graphic window with a profile of a program memory

8. Развитие отладчика PD

В процессе использования отладчика PD возникла необходимость доработать его программные компоненты. В результате в отладчике PD были реализованы коммуникационная схема *дерево*, позволившая качественно уменьшить время передачи информации, и новый программный компонент – не интерактивный отладчик. Кроме этого, была обеспечена отладка программ на графических ускорителях корпорации Nvidia.

8.1 Коммуникационная схема дерево

В ранних версиях отладчика PD была реализована схема, в которой каждый программный агент был непосредственно связан с сервером команд и сообщений. Такая схема проста в реализации и хорошо работает, пока с сервером взаимодействуют сотни программных агентов. Как только к серверу подключены тысячи агентов, она становится причиной общего замедления процесса отладки: значительная нагрузка на коммуникационную подсистему инструментального сервера, на котором запущена программа графического интерфейса отладчика PD, препятствует прохождению команд/сообщений к/от программных агентов.

Для распределения нагрузки по узлам ВС в отладчике PD была реализована коммуникационная схема *дерево*, в которой с программой графического интерфейса (верхняя

точка на рис. 6) связаны, например, три программных агента, которые в свою очередь соединены еще с четырьмя программными агентами и т.д. Новая коммуникационная схема позволила уменьшить время формирования соединений между программными агентами PD и сервером команд и сообщений в процессе создания отладочной сессии, а также время передачи информации в процессе отладки программы. В табл. 1 приведено время формирования коммуникационной схемы *дерево* в зависимости от количества программных агентов. В табл. 2 приведены замеры времени фактического выполнения строки программы параллельным отладчиком PD. Время включает передачу MI-команд *выполнить шаг* программным агентам, её обработку отладчиками GDB и передачу результатов серверу команд и сообщений.

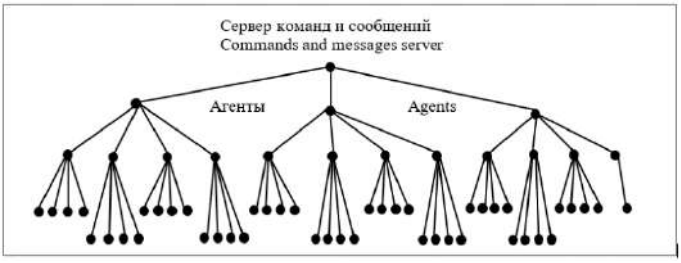


Рис. 6. Пример коммуникационной схемы дерево
Fig. 6. An example of a treelike communication scheme

Табл. 1. Время формирования коммуникационного дерева
Table 1. Formation time of a communication tree

Количество агентов (тысячи)	0,5	1	1,5	2,5	3	4,4	6	8	10
Время (миллисекунды)	19	42	74	119	141	206	343	417	564

Табл. 2. Общее время выполнения строки программы
Table 2. Total execution time of a program line

Количество процессов	500	1000	2000	3000	4000	6000	8000	10000
Время (секунды)	0,14	0,29	0,94	1,39	3,25	4,56	5,54	8,83

На время прохождения команды *выполнить шаг* большое влияние оказывает количество связей у программных агентов. Так, при исследовании этой метрики коммуникационного дерева было замечено, что вслед за увеличением количества связей с единицы до восьми время прохождения команды *выполнить шаг* уменьшается, но по мере дальнейшего роста числа связей оно начинает увеличиваться.

Увеличивая количество связей сервера команд с агентами, можно уменьшить длину пути, который проходят команды, например, при отладке небольшого количества процессов параллельной программы (идеально, если их агенты непосредственно связанны с сервером команд). Однако при отладке всех процессов программы суммарный путь, который пройдут все команды, не уменьшится. Кроме этого, каждая связь потребляет аппаратно-программные ресурсы компьютера, поэтому нет смысла делать значение этой метрики большим, в связи с чем было решено, что у сервера команд будет 32 связи. У программных агентов должно быть не более 8 связей[5]. Представленные в табл. 1 и 2 значения получены с использованием этих параметров коммуникационного дерева.

В целом реализация коммуникационной схемы *дерево* позволила качественно сократить общее время выполнения строки программы. Например, операция *выполнить шаг* при отладке четырех тысяч процессов выполняется более чем в 2,5 раза быстрее.

[5] Параметры коммуникационного дерева содержатся в текстовом конфигурационном файле отладчика PD. При необходимости их можно изменить.

Авторы полагают, что время передачи можно будет уменьшить, изменив формат команд и сообщений: формировать так называемые групповые команды, содержащие, например, команду и список агентов, которым она предназначается, передавать серверу ответ только от наблюдаемого процесса. Реализовав это, можно будет сократить количество информации, передаваемой между сервером команд и сообщений и программными агентами PD.

8.2 Неинтерактивная отладка программы

Новая возможность, реализованная в параллельном отладчике PD, - неинтерактивная (offline) отладка, позволяет отлаживать программу без участия пользователя. Такая отладка может быть применена тогда, когда требуется отладить ресурсоёмкую программу, для которой на ВС нет свободных вычислительных ресурсов, а в отсутствие разработчика они могут появиться.

Для offline-отладки пользователь должен сформировать в графическом интерфейсе отладчика PD пакетное задание, указав название исполняемого файла, количество узлов/процессоров, время выполнения и т.д. Кроме этого ввести, если требуется, конфигурацию профилирования, название журнального файла, метрики точек прерывания/наблюдения, названия глобальных программных переменных и ранг наблюдаемого MPI-процесса. По мере ввода информация о точках прерывания, наблюдения и программных переменных помещается в общую таблицу графического окна ввода параметров offline-сессии, показанную на рис. 7. Из этой таблицы пользователь может удалить строки, а также добавить информацию из файла.

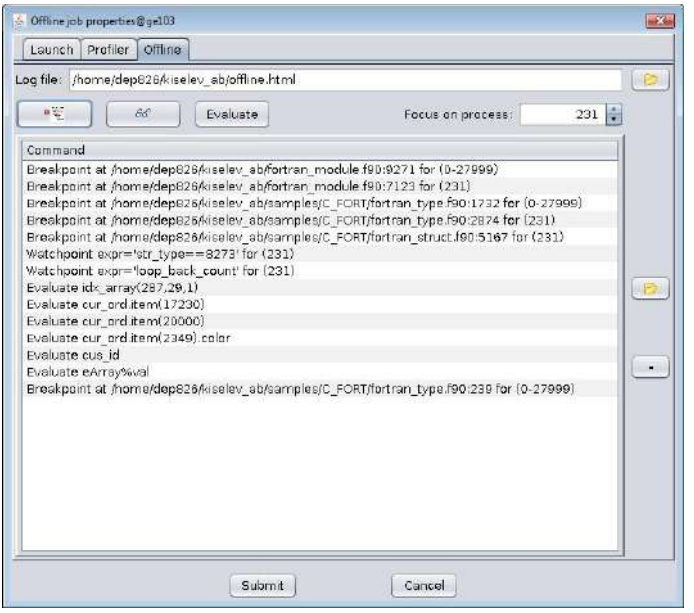


Рис. 7. Графическое окно ввода параметров offline-сессии отладки
Fig. 7. The graphical window for entering parameters of offline debugging session

После щелчка мышью по кнопке *Submit* программа графического интерфейса запускает offline-отладчик, которому она передаёт атрибуты пакетного задания, конфигурацию профилирования и метрики отладочной сессии. Затем offline-отладчик в отдельных программных потоках запускает сервер команд и сообщений, программу контроля завершения задания, обработчик стандартной выдачи и диагностики процессов отлаживаемой программы.

В соответствии со схемой инициализации отладочного задания, которая подробно описана в разд. 5, агенты отладчика PD запускаются системой пакетной обработки заданий на узлах

ВС, между сервером команд и сообщений и программными агентами формируется коммуникационное *дерево*. Сервер команд и сообщений посылает агентам PD команду *-exec-run --start* (запустить программу и остановиться на точке входа в программу). Все программные агенты с помощью отладчика GDB выполняют её, а потом оповещают сервер о готовности к отладке. После этого offline-отладчик устанавливает точки прерывания/наблюдения и продолжает выполнение программы.

Если в контролируемом процессе сработала точка прерывания/наблюдения, то offline-отладчик с помощью соответствующих MI-команд считывает значения локальных переменных процедуры/функции, значения указанных пользователем глобальных переменных, данные о программных потоках и стеке программы. Эту информацию offline-отладчик записывает в журнальный файл HTML-формата. Затем он продолжает выполнение прерванного процесса.

Точки прерывания, которые срабатывают в других процессах, игнорируются. Однако если какой-либо процесс получает программный сигнал, то offline-отладчик всегда считывает и записывает в журнальный файл всю перечисленную выше информацию этого процесса.

По завершении программы offline-отладчик записывает в журнальный файл результаты её профилирования и данные, считанные из временного файла стандартной выдачи и диагностики. После этого он завершается.

8.2.1 Формат журнального файла

Журнальный файл разбит на три части. Первая часть (Messages) содержит таблицу (табл. 3) с информацией о событиях, которые произошли в процессе отладки (старт программы, установка точек прерывания/наблюдения и т.д.).

При срабатывании точки прерывания или получении программного сигнала offline-отладчик записывает в таблицу информацию, содержащую название исходного файла и номер строки программы, в которой произошло прерывание. Рядом записываются данные о стеке (Stack), локальных переменных процедуры/функции (Locals), программных потоках (Threads), наблюдаемых глобальных переменных (Evaluate) и резюме (Summary), скрытые за символом ▼.

Табл. 3 Пример таблицы Messages
Table 3 A sample Messages *table*

Time	Processes	Type	Message
00:00:00	0-23		Launching job 141899.ce100 at Fri Feb 01 10:09:03 GMT+03:00 2019
00:00:00	0-23		Startup complete
00:00:01	0-23		Add breakpoint at /home/dep826/0226/tests/fort/mod.f90:25
00:00:01	0-23		Add breakpoint at /home/dep826/0226/tests/fort/mod.f90:38
00:00:02	0		Process stopped at breakpoint in prom() at /home/dep826/0226/tests/fort/mod.f90:24 ▼Stack Stack arguments #3 _start() #2 __libc_start_main() at /lib64/libc.so.6 #1 main() #0 prom() at /home/dep826/0226/tests/fort/mod.f90:24 ▼Locals ▼Threads ▼Evaluate ▼Summary

Кликнув мышью по данному символу, можно раскрыть запись (в табл. 3 раскрыта запись Stack). Пункт Summary содержит сводную информацию о срабатывании точек прерывания/наблюдения в процессах программы в момент возникновения прерывания или срабатывании точки наблюдения у контролируемого процесса в течение некоторого короткого промежутка времени.

Во вторую часть журнального файла (Profiles) записывается информация профилирования программы (табл. 4), которая содержит данные, получаемые с помощью профилировщиков GPT и mpiP. Если пользователь не расставил галочки на вкладке *Profiles* при создании offline-сессии, то секции Profiles в журнальном файле не будет.

Табл. 4. Пример информации использования процессора [12]
Table 4. An example of CPU-use information [12]

Function	Call	Call %	Full path
cycles	201813	97.2	/home/dep826/0226/tests/c/threads.c
doSomething	5898	2.8	/home/dep826/0226/tests/c/threads.c

Третья, последняя часть (Output) журнального файла, содержит информацию из стандартного вывода и диагностики процессов программы. Диагностическая информация выделяется красным цветом.

8.3 Поддержка CUDA

В отладчике PD была реализована возможность отлаживать программу на GPU (graphics processing unit). Собственно отладка программы на GPU выполняется проприетарным отладчиком GDB корпорации Nvidia (Nvidia-GDB), который входит в состав CUDA SDK [19]. Для формирования нестандартных MI-команд Nvidia-GDB, разбора сообщений и управления отладкой CUDA-программы были доработаны графический интерфейс отладчика PD и сервер команд и сообщений.

Надо отметить, что в составе CUDA SDK имеется отладчик nsight, который позволяет отлаживать программы, использующие графические ускорители Nvidia, но nsight не обеспечивает отладку MPI-программ на BC. Кроме него, только отладчики TotalView и Allinea DDT обеспечивают отладку MPI- и CUDA-программ, но они большинству отечественных программистов недоступны.

8.4 MI-команды для Nvidia-GDB

В процессе отладки программы на компьютерах семейства Intel x86 или, например, Эльбрус отладчик PD посылает стандартные MI-команды и принимает стандартные сообщения – ответы. Однако в момент выполнения программы в GPU Nvidia-GDB выдаёт специализированную информацию, например:

```
CudaFocus={device="0",sm="0", warp="0", lane="0", kernel="0", grid="1",
blockIdx="(0,0,0)", threadIdx="(255,0,0)"}
```

Данное сообщение свидетельствует, что код отлаживаемой программы выполняется в GPU с индексом 0, индекс блока (0,0,0) и нити (255,0,0) и т.д. Дальнейшее управление отладкой может выполняться только с помощью команд с приставкой -cuda: *-cuda-focus-switch, -cuda-info-kernels, -cuda-info-blocks, -cuda-info-threads* и т.д.

Для выполнения переключений и отображения информации о GPU в графическом интерфейсе отладчика PD значения метрик device, warp, kernel, blockIdx и threadIdx записываются в характеристику процесса, а команды с префиксом -cuda посылаются Nvidia-GDB до тех пор, пока программный код использует GPU. Рассмотрим этапы переключения отладчика PD на программную нить с индексом (53,0,0). Графический интерфейс PD посылает Nvidia-GDB MI-команду

```
-cuda-focus-switch kernel 0 block (0,0,0) thread (53,0,0)
```

Ответ отладчика Nvidia-GDB с результатом её выполнения может, например, содержать такую информацию:

```
CudaFocus={device="0", sm="0", warp="1", lane="21", kernel="0", grid="1",
blockIdx="(0,0,0)", threadIdx="(53,0,0)"}, frame = {addr =
"0x0000000000dbd720", func = "bitreverse", args = [{name = "data", value
= "0x7fffce600000"}], file="bitreverse.cu", fullname =
"/home/CUDA/bitreverse.cu", line="12"
```

Соответствующий программный класс сервера команд и сообщений обрабатывает информацию об успешном переключении фокуса, а затем графический интерфейс отладчика PD выполняет смену индексов в переключателе (рис. 8).

Рис. 8 Переключатель индексов ядра, блока и нити при отладке программы на GPU
Fig. 8 The switch for changing of a kernel, a block and a thread indexes at the debugging on GPU

8.5 Графические компоненты для отладки программы на GPU

8.5.1 Переключатель ядра, блока и программной нити

Переключатель (рис. 8) позволяет переключаться на ядро, блок или нить в режиме отладки программных потоков. Он появляется в графическом интерфейсе отладчика PD только тогда, когда код программы выполняется в GPU. Чтобы пользователь мог использовать доступные на момент отладки индексы, при каждом останове программы отладчик PD считывает из GPU граничные значения индексов Kernel, Block и Thread, а затем переконфигурирует переключатель. Операция переключения на ядро, блок и нить выполняется после клика мышью по кнопке Ok.

В случае завершения отлаживаемой нити отладчик автоматически переключается на *живую* нить.

8.5.2 Информация о программных нитях

При выполнении кода программы в GPU в графическом интерфейсе отладчика PD на закладке *Thread* появляется таблица *CUDA Thread*, содержащая список GPU-нитей (рис. 9).

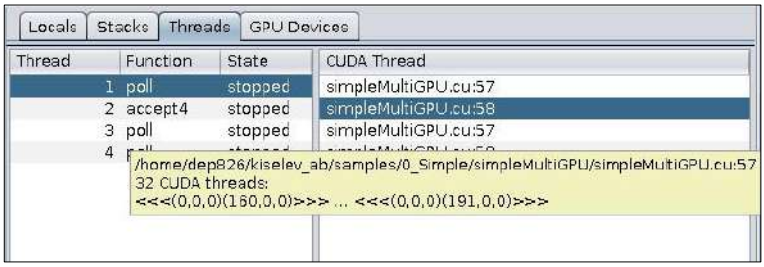


Рис.9. Пример таблицы CUDA Thread *и всплывающей подсказки*
Fig. 9. An example of the CUDA Thread *table and a tooltip*

Если пользователю важен диапазон индексов blockIdx и threadIdx, то достаточно некоторое время удерживать указатель мыши над таблицей *CUDA Thread*, чтобы появилась всплывающая подсказка с интересующей его информацией. Подсказка содержит полный путь к исходному файлу, номер исходной строки (цифра 57), общее количество программных нитей (32), а также их начальный и конечный индексы (строка с символами меньше и больше).

Графический интерфейс отладчика PD скрывает таблицу *CUDA Thread*, если процесс более не использует GPU.

8.5.3 Информация о GPU

На вкладке *GPU Devices* (рис. 10) отображается сводная информация о доступных/используемых GPU-устройствах. Так, например, на рисунке показано, что процессам с рангами 0-180 доступны по два устройства с указанными характеристиками, остальные процессы не используют GPU, об этом свидетельствует фраза *No device*.

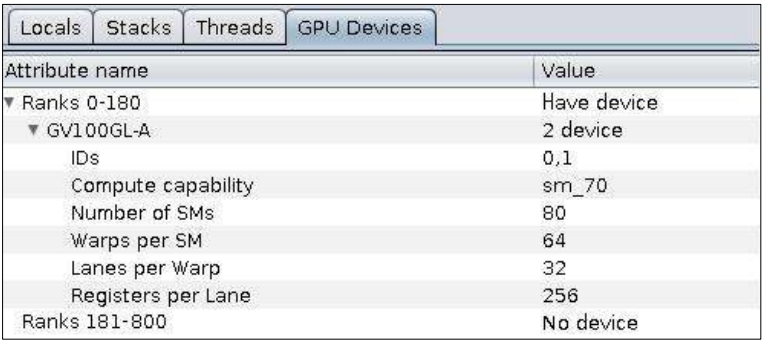


Рис. 10. Пример содержимого таблицы GPU Devices
Fig. 10. An example of contents of the GPU Devices *table*

9. Заключение

Параллельный отладчик применяется в РФЯЦ-ВНИИЭФ для отладки программных комплексов моделирования физических процессов. Он входит в дистрибутивы защищенных ОС *системное программное обеспечение супер-ЭВМ со встроенными средствами защиты информации от несанкционированного доступа* [20] и *Арамид* [21] и в составе этих ОС используется на вычислительных системах ФГУП РФЯЦ-ВНИИЭФ.

По своим возможностям отладчик PD очень близок к Allinea DDT и TotalView, но в отличие от них у него нет лицензионных ограничений: PD позволяет отлаживать и профилировать любое количество процессов параллельной программы на любом количестве процессоров.

Отладчик PD написан на *чистой* Яве, поэтому по сравнению, например, с Eclipse PTP он использует гораздо меньше оперативной и дисковой памяти компьютера. Кроме этого, обладая достаточно большим опытом использования Eclipse PTP, авторы статьи уверенно заявляют, что пользовательский интерфейс отладчика PD более удобный.

Отладчик PD функционирует не только на компьютерах фирмы Intel, но и на отечественной платформе *Эльбрус*.

Последняя версия отладчика PD имеет качественные отличия от предыдущих версий за счет реализации древовидной схемы передачи информации, в ней появился новый программный компонент – offline-отладчик, обеспечена отладка CUDA-программ.

Список литературы / References

[1]. TotalView. URL: https://totalview.io/free-trial, 12.03.2020.
[2]. Distributed Debugging Tool. URL: https://www.arm.com/products/development-tools/server-and-hpc/forge/ddt, 12.03.2020.
[3]. Eclipse Parallel Tools Platform. URL: http://eclipse.org/ptp, 12.03.2020.
[4]. Федоров В.К., Киселев С.Н. Отладчик параллельных приложений (PDB). Вопросы атомной науки и техники. Серия «Математическое моделирование физических процессов», вып. 3, 2013 г., стр. 65-71. // Fedorov V.K, Kiselev S.N. A debugger of parallel applications (PDB). Voprosy Atomnoy Nauki i Tekhniki. Series «Mathematical modelling of physical processes», issue 3, 2013, pp.65-71 (in Russian).
[5]. Malyshkin V. E, Romanenko A.A. GEPARD – General Parallel Debugger for MVS-1000/M. Lecture Notes in Computer Science, vol. 2763, 2003, pp. 519-523.
[6]. Андрианов А.Н., Базаров С.Б., Бугеря А.Б., Колударов П.И., Набоко И.М. Применение отладчика параллельных программ при решении задачи о фокусировке ударных и взрывных волн на

многопроцессорных ЭВМ. Препринты ИПМ им. М. В. Келдыша, № 50, 2004 г., 15 стр. / Andrianov A.N., Bazarov S.B., Bugerya A.B., Koludarov P.I., Naboko I.M. Application of parallel programs debugger for multiCPU modeling of shock and blast waves focusing. Keldysh Institute preprints, № 50, 2004, 15 p. (in Russian).

[7]. Киселев А.Б., Киселев С.Н., Семенов Г.П. Отладчик параллельных программ для кластеров на базе ОС Linux. Вопросы атомной науки и техники. Серия «Математическое моделирование физических процессов», 2018 г., вып. 2, стр. 72-80. // Kiselev A.B., Kiselev S.N. A parallel debugger for clusters on the base of OS Linux. Voprosy Atomnoy Nauki i Tekhniki. Series «Mathematical modelling of physical processes», issue 2, 2018, pp. 72-80 (in Russian).

[8]. GDB. URL: http://www.gnu.org/software/gdb, 12.03.2020.

[9]. Torque. URL: http://www.adaptivecomputing.com/products/open-source/torque, 12.03.2020.

[10]. SLURM. URL: https://slurm.schedmd.com/documentation.html, 12.03.2020.

[11]. Киселев А.Б, Киселев С.Н. Система пакетной обработки заданий JAM. Вопросы атомной науки и техники. Серия «Математическое моделирование физических процессов», 2009 г., вып. 4, стр. 60-66. / Kiselev A.B., Kiselev S.N. JAM batch jobs system. Voprosy Atomnoy Nauki i Tekhniki. Series «Mathematical modelling of physical processes», issue 4, 2009, pp. 60-66 (in Russian).

[12]. Google Performance Tools. URL: https://github.com/gperftools/gperftools, 12.03.2020.

[13]. mpiP: Lightweight, Scalable MPI Profiling. URL: http://mpip.sourceforge.net, 12.03.2020.

[14]. SwingX. URL: https://github.com/arotenberg/swingx, 12.03.2020.

[15]. JHDF5. URL: http://www.hdfgroup.org, 12.03.2020.

[16]. Jzy3D. URL: http://www.jzy3d.org, 12.03.2020.

[17]. RSyntaxTextArea. URL: http://www.sorceforge.org/rsyntaxtextarea, 12.03.2020.

[18]. OpenGL. URL: https://www.opengl.org/sdk/libs/, 12.03.2020.

[19]. CUDA Toolkit Documentation. URL: http://docs.nvidia.com/cuda/eula/index.html, 12.03.2020.

[20]. Кульнев Д.В., Модянов Р.В., Петрик А.Н. Защищенная ОС. Открытые системы. СУБД, № 4, 2015 г. / Kulnev D.V., Modjanov R. V, Petrik A.N. Secured OS. Open systems. DBMS, № 4. 2015 (in Russian).

[21]. Петрик А.Н. Защищенная операционная система Арамид для супер-ЭВМ. Сборник тезисов докладов Национального суперкомпьютерного форума (НСКФ-2019), 2919 г. / Petrik A.N. Secured OS Aramid for the super-computer. In Proc. of the National Supercomputer Forum (NSCF-2019), 2019 (in Russian).

Информация об авторах / Information about authors

Алексей Борисович КИСЕЛЕВ, кандидат физико-математических наук, начальник лаборатории разработки системного программного обеспечения. Сфера научных интересов: системное программное обеспечение высокопроизводительных вычислительных систем, облачные вычисления, методы защиты информации.

Aleksey Borisovich KISELEV, Ph.D in physics and mathematics, head of the laboratory for system software development. Research interests: system software for high-performance computing systems, cloud computing, information security methods.

Сергей Николаевич КИСЕЛЕВ, старший научный сотрудник. Сфера научных интересов: системное программное обеспечение высокопроизводительных вычислительных систем, облачные вычисления, методы защиты информации в распределенных вычислительных системах.

Sergey Nikolaevich KISELEV, senior researcher. His research interests include system software for high-performance computing systems, cloud computing, methods of information's protection in distributed computing systems.

DOI: 10.15514/ISPRAS–2020–32(4)–8

Протокол сертификации целостности облачных вычислений

[1] *Е.С. Шишкин, ORC-ID: 0000-0002-6221-5651 <evgeny.shishkin@infotecs.ru>*
[1,2] *Е.С. Кислицын, ORC-ID: 0000-0003-0373-9866 <evgeny.kislitsyn@infotecs.ru>*
[1] *ОАО «ИнфоТеКС»,*
127287, Россия, Москва, Старый Петровско-Разумовский проезд, д. 1/23, стр. 1
[2] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1

Аннотация. В статье сформулирована задача сертификации целостности вычислений, проводимых стороной, которой мы не обязательно доверяем. Предложен интерактивный многопользовательский протокол решающий эту задачу при заданных ограничениях. По сравнению с ближайшим аналогом, предложенный протокол упрощает процедуру построения доказательства с *O(nlogn)* до *O(n)*, а сложность коммуникации сводит к одному раунду при сопоставимой длине сертификата.

Ключевые слова: целостность вычислений; интерактивные доказательства; вычислительные сертификаты

Для цитирования: Шишкин Е.С., Кислицын Е.С. Протокол сертификации целостности облачных вычислений. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 115–132. DOI: 10.15514/ISPRAS–2020–32(4)–8

Благодарности. Благодарим Андрея Рыбкина и Михаила Бородина за конструктивную критику первой версии протокола: благодаря их замечаниям удалось обнаружить и устранить уязвимость в логике протокола. Мы также благодарим Джэйсона Тетча (Jason Teutsch) за конструктивную критику черновика статьи и за объяснения деталей протокола TrueBit.

Protocol for Certifying Cloud Computations Integrity

[1] *E.S. Shishkin, ORC-ID: 0000-0002-6221-5651 <evgeny.shishkin@infotecs.ru>*
[1, 2] *E.S. Kislitsyn, ORC-ID: 0000-0003-0373-9866 <evgeny.kislitsyn@infotecs.ru>*
[1] *InfoTeCS, Advanced Research Department*
1/23, bld.1, Stariy Petrovsko-Razumovskiy proezd, Moscow, 127287, Russia
[2] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia

Abstract. We define a problem of certifying computation integrity performed by some remote party we do not necessarily trust. We present a multi-party interactive protocol that solves this problem under specified constraints. The protocol rely on a computing entity called weak reliable computing device that is able to produce certificate for a tiny computation. A complex computation of a user can be certified by relying on such weak entity if we translate the user program into an iterative form that converges to the result, a style resembling continuation-passing style programming paradigm. Each iteration of the computation can then be certified by the weak reliable computing device. To ensure the user that computation result was computed fairly and correctly, we put this mechanism into a multi-party incentivized context of several computing providers competing with each other for publishing the result and taking the reward, or finding an error in the published result of other parties. Together with computation result, the computation certificate is published. The certificate is constructed in such a way that other fair parties (auditors) are able to find divergent computation step in

O(logn) steps, where n is a number of computing steps taken before reaching the result. If error is found, the auditor sends proof of the error (called refutation) in the trusted weak computing device that plays the role of autonomous transparent arbiter able to check refutation by replaying a tiny fraction of the computation. Comparing to the nearest related work, our protocol reduces a proof construction complexity from *O(nlogn)* to *O(n)*, turning a communication complexity to exactly one round using a certificate of a comparable length. We also give the formal specification for the protocol and prove some of its security properties in a specified threat-model. Experimental evaluation of the protocol in a model setting is provided.

Keywords: computation integrity, interactive proofs, computation certificates.

For citation: Shishkin E.S., Kislitsyn E.S. Protocol for Certifying Cloud Computations Integrity. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 115–132 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–8

Acknowledgements. We are grateful to Andrey Rybkin and Mikhail Borodin for constructive criticism of the first version of the protocol: thanks to their comments, a vulnerability in the protocol logic was discovered and eliminated. We also thank Jason Teutsch for his constructive criticism of the draft article and for explaining the details of the TrueBit protocol.

1. Введение

Предположим, пользователю требуется произвести какое-то сложное вычисление $C(x)$. Для такого вычисления может потребоваться большой вычислительный ресурс, например, кластер из множества вычислительных узлов, и такого ресурса у пользователя может и не быть.

Для решения своей задачи, пользователь обращается за услугой к облачному сервису – исполнителю вычислений. Пользователь передает исполнителю задачу $C(x)$ и начальные данные d, желая получить результат $C(d)$.

Бывают такие вычисления, результат которых легко валидировать, например, результат функции сортировки списка целых чисел. Факт отсортированности проверяется простой однопроходной функцией и совпадением состава элементов отсортированного и исходного списков.

Напротив, есть вычисления, результат которых в общем случае валидировать трудно без проведения повторного вычисления: например, вычисление, проверяющее *отсутствие* гамильтонова цикла в графе[1]. Если исполнителю удаётся найти хотя бы один гамильтонов цикл, то этот цикл легко проверить. Но что делать, если исполнитель отвечает, что в графе *нет таких циклов*: как проверить достоверность такого ответа?

Возьмем другой пример: автоматическая верификация моделей программ – известная вычислительно сложная задача. Предположим, пользователь передал исполнителю модель программы и список свойств, которым она должна удовлетворять, и исполнитель отвечает пользователю, что в переданной программе найти отклонений от заявленных свойств не удалось. Но как проверить, что: 1) поиск действительно производился 2) был произведен в полном объеме?

В текущей практике пользователь вынужден полагаться на добросовестность исполнителя, на его авторитет, при этом результат проведенного вычисления заверяется цифровой подписью исполнителя.

В этой статье рассматривается распределенный протокол, позволяющий решить описанную задачу технически, без необходимости привнесения дополнительного доверия исполнителю вычисления.

Протокол опирается на наличие доверенного линейного вычислителя, например, смарт-контракта в публичном блокчейне, выполняющего роль авторитетного автономного арбитра в разрешении разногласий по поводу правильности вычисления.

[1] Гамильтонов цикл - цикл в графе, проходящий через каждую вершину ровно один раз.

По сравнению с ближайшим аналогом – протоколом TrueBit [17] – наш протокол существенно уменьшает оценку сложности построения доказательства и минимизирует расходы на коммуникацию сторон в случае разногласий, сводя их количество к одному раунду. Такой эффект достигается за счёт иного механизма конструирования доказательного сертификата и другого способа представления пользовательского вычисления.

Статья устроена следующим образом: в разд. 2 приводится строгая формулировка решаемой задачи. Разд. 3 описывает предлагаемое нами решение в простых терминах. Разд. 4 посвящен описанию метода представления пользовательского вычисления в виде, пригодном для итеративного вычисления. Далее, в разд. 5 приведена формальная запись всего протокола. Разд. 6 посвящен анализу надёжности протокола в рамках заданной модели угроз, кратко обсуждается экономическая составляющая протокола. Разд. 7 описывает возможные атаки на протокол и способы противодействия. Первые экспериментальные результаты приведены в разд. 8. Статью заключает разд. 9, в котором приведены краткие сведения о схожих работах и их технических результатах по сравнению с нашей работой.

2. Формулировка задачи

Мы ищем эффективный способ перепроверки правильности результата вычисления, проведенного исполнителем, которому мы не обязательно доверяем. Вместе с результатом вычисления исполнитель предоставляет некое доказательство - сертификат, размер которого должен разумно коррелировать со сложностью проводимого вычисления. По сертификату мы хотим иметь возможность однозначно установить правдивость полученного результата и сделать это обязательно существенно быстрее, чем повторное вычисление.

Определение 1. (Задача сертификации вычислений) Для произвольной вычислимой в точке $d \in \mathbb{N}$ функции $C\colon \mathbb{N} \to Result$, задать вычислимые функции:

$$proof(C, d) \to Result \times Proofs,$$
$$verify(C, d, r, prf) \to \{True, False\},$$

где $r \in Result$ называют проверяемым результатом вычисления, т.е. $r \stackrel{?}{=} C(d)$, значение $prf \in Proofs$ – доказательство целостности вычисления, называемое также *сертификатом*. При этом выполняются требования:

- значение $verify(C, d, r, prf) = True$ тогда и только тогда, когда $C(d) = r$, и $False$ в противном случае; **[1.1]**
- вычислительная сложность $proof(C, d)$ линейно зависит от сложности C;
- вычислительная сложность $verify$ существенно меньше сложности $proof$ для любой фиксированной задачи C, т.е.

  $$\lim_{n\to\infty} \frac{v_C(n)}{e_C(n)} = 0,$$

  где $v_C(n), e_C(n)$ - асимптотические оценки вычислительной сложности функций $verify$ и $proof$ для задачи C в зависимости от условного размера входа n;
- с ростом сложности $e_C(n)$, размер доказательства prf растёт не быстрее, чем линейно, т.е.

  $$\exists k \forall n\,.\ |prf(n)| \le k \cdot e_C(n).$$

2.1 Вероятностный критерий проверки сертификата

У описанной задачи есть вариации, ослабляющие либо усиливающие некоторые из приведенных требований.

Далее, мы намеренно ослабляем требование **[1.1]**, т.к. оказывается чрезмерно строгим на практике.

Определение 2. (Вероятностный критерий проверки сертификата) Рассмотрим такие события:

$$E_0(\lambda): proof_\lambda(C,d) = \langle r, prf_\lambda \rangle$$
$$E_1(\lambda): verify_\lambda(C,d,r,prf_\lambda) = True$$
$$E_2: C(d) = r.$$

Если для функций $proof_\lambda$, $verify_\lambda$ выполняется такой критерий:

$$\lim_{\lambda\to\infty} \Pr[E_0(\lambda)\cdot E_1(\lambda)\cdot E_2] = 1,$$

где λ – это некий параметр метода, величина которого может влиять на вычислительную сложность функций и размер сертификата, в этом случае утверждаем, что алгоритмы удовлетворяют вероятностному критерию проверки сертификата.

Другими словами, мы описываем не детерминированный метод, а вероятностный, где вероятность нужного исхода регулируется специальным параметром.

2.2 Интерактивность

Для некоторых способов решения описанной задачи свойственна *интерактивность*, т.е. процесс проверки решения может происходить в несколько раундов обмена сообщениями между исполнителем и проверяющим. Оценку на количество необходимых раундов мы называем *коммуникационной сложностью* протокола. Интерактивный способ решения описанной задачи также называют *протоколом сертификации целостности вычислений*.

2.3 Доверенный линейный вычислитель

Одна из разновидностей протоколов сертификации вычислений полагается на наличие доверенной стороны, которая хотя и не способна провести *всё вычисление* $C(d)$ *целиком*, но помогает в проверке более простых вычислительных действий.

Например, устройство, способное доказательно выполнить операцию сложения пары натуральных чисел, пусть и не способно посчитать факториал, но если разбить задачу на множество операций сложения, мы бы могли провести всё вычисление достоверным образом.

Определение 3. (Доверенный Линейный Вычислитель) Устройство, решающее задачу сертификации вычислений для вычислимых функций f, таких, что вычислительная сложность $f(n)$ растёт с ростом n не быстрее, чем линейно, т.е. $f(n) \in O(n)$, мы называем *доверенным линейным вычислителем*.

К таким вычислителям относятся, например, безопасные анклавы [12] и смарт-контракты, выполняемые участниками блокчейн-протокола [4].

2.4 Многопользовательский протокол

В классической интерпретации задачи сертификации вычисления подразумевается, что исполнитель вычисления доказывает заказчику вычисления корректность полученного им результата, а заказчик проверяет доказательство, взаимодействие происходит напрямую.

Наша же модель подразумевает иной сценарий взаимодействия: заказчик вычисления и множество исполнителей взаимодействуют посредством устройства доверенного линейного вычислителя (ДЛВ). Несколько исполнителей может выполнять задачу одновременно. Результат вместе с доказательством записывается в ДЛВ, давая возможность другим исполнителям выполнить перепроверку. В случае разногласий относительно результата, исполнители через ДЛВ доказывают ошибочность опубликованного результата. В противном случае, по прошествии периода времени, вычисление считается выполненным корректно, и заказчик получает ответ. Такую модель взаимодействия мы называем *многопользовательским протоколом*.

2.5 Решаемая задача

В этой статье рассматривается многопользовательский протокол сертификации целостности вычислений с вероятностным критерием проверки сертификата, при условии наличия доверенного линейного вычислителя, выполненного с возможностью аутентификации пользователей.

Для простоты изложения *в качестве устройства доверенного линейного вычислителя мы используем смарт-контракт, размещенный в публичном блокчейне*, но можно представить и другие варианты устройств.

3. Схема работы протокола

В этом разделе описывается устройство работы протокола без использования формальных определений.

1. Заказчик преобразует вычислимую функцию $C(x)$ в другую вычислимую функцию $f(x)$, такую, что после n-кратного её применения к точке d, значение сходится к $C(d)$, при этом $f(m) \in O(m)$. Далее будет показано, что такое преобразование всегда можно выполнить.
2. Заказчик публикует функцию f и точку d в смарт-контракте, доступ к которому имеет множество потенциальных исполнителей.
3. Исполнитель проводит вычисление $r = f(f \dots f(d))$, продолжая итерации до тех пор, пока результат не сойдется к неподвижной точке. После каждой итерации $f(x)$, дополнительно вычисляется хэш-значение $c_i \coloneqq H(x \circ c_{i-1})$ Полученный таким образом список значений $cert \coloneqq \langle c_1, \dots, c_n \rangle$ образует *сертификат*. Исполнитель вычисляет слепок сертификата: $hc \coloneqq H(H(cert))$ и *проекцию сертификата* $cp \coloneqq \langle \pi(c_1), \pi(c_2), \dots, \pi(c_n) \rangle$.

   Смысл формирования слепка в том, чтобы, опубликовав это значение, дать возможность другим исполнителям сравнить их сертификат с исходным, но при этом не публиковать само значение в открытом виде.

   Проекция сертификата нужна для того, чтобы быстро (за $\log n$) отыскать ту часть сертификата, с которой начинается разногласие, не публикуя последовательность $cert$ в открытом виде.

   Сам сертификат играет роль секрета, по которому в дальнейшем протокол установит всех добросовестных исполнителей; поэтому это значение не публикуется в открытом виде.
4. Исполнитель публикует решение задачи r, слепок сертификата hc и проекцию cp в смарт-контракте.
5. Другие исполнители, также решавшие задачу, но получившие решение позже, могут перепроверить опубликованное решение сравнив слепок сертификата с тем, которое получилось у них. Эту категорию исполнителей мы называем *проверяющими*.
6. В случае появления разногласий относительно результата, проверяющий отправляет в смарт-контракт номер i той части проекции сертификата, в которой наблюдается расхождение, вместе с точкой r_{i-1} и предыдущим значением c_{i-1}
7. Смарт-контракт по заданным значениям i, r_{i-1}, c_{i-1} убеждается либо в ошибочности опубликованного решения, либо в его верности, выполняя одну итерацию вычисления (детали приведены в разделе **5**).
8. Если в течение некоторого времени для опубликованного решения не предъявлено опровержения, решение считается корректным.
9. Проверяющие отправляют в смарт-контракт значение $H(H(cert) \circ id)$, где id – уникальный идентификатор проверяющего. Это значение играет роль доказательства проделанной работы по перепроверке решения.

10. Исполнитель отправляет в смарт-контракт значение s, такое, что $hc = H(s)$, и по этой информации формируется список добросовестных проверяющих и тех, кто попытался нарушить правила протокола.

4. Итеративная вычислимость

Описываемый протокол требует представления пользовательского вычисления в итеративной форме. Пример на Рис. 1а демонстрирует реализацию функции факториала в стандартной функциональной записи, и на Рис. 1б – в итеративной форме. Если последовательно вычислять значения функции $factFP(\{N, 1\})$, передавая в качестве очередной точки значение, полученное на предыдущем шаге, то функция сойдется к значению $\{0, N!\}$.

```
fact(0) ->
    1;
fact(N) when N > 0 ->
    N * fact(N-1).
```

(a) Стандартная рекурсивная реализация (Recursive implementation)

```
factFP({0, Acc}) ->
    {0, Acc};
factFP({N, Acc}) when N > 0 ->
    {N - 1, Acc * N}.
```

(b) Итеративная реализация (Iterative implementation)

Рис. 1. Варианты записи функции факториала на языке Erlang
Fig. 1. Recursive vs. Iterative factorial implementation (Erlang)

Принципиальная разница между двумя реализациями в том, что первая реализация (Рис.1а) производит сразу всё вычисление, когда вторая (Рис.1б) производит часть вычисления и возвращает промежуточный результат, который можно продолжить вычислять дальше до достижения неподвижной точки.

Можно усомниться в том, что любая программа представима в итеративной форме. Следующая теорема показывает, что это всегда можно сделать, множеством способов.

Теорема 1. Для любой вычислимой в точке d функции $C: \mathbb{N} \to \mathbb{N}$ существуют такие вычислимые функции

$$inj: \mathbb{N} \to T, proj: T \to \mathbb{N}, F: T \to T$$

такие, что

$$proj(\text{fix } F\ inj(d)) = C(d)$$

где fix: $(T \to T) \times \mathbb{N} \to T$ – оператор, вычисляющий неподвижную точку функции, стартуя из заданной точки, т. е.

$$\text{fix}\, F\, p = F(\text{fix}\, F\, p)$$

Здесь T – некоторое конечное множество.

Доказательство. Доказательство вынесено в Приложение 1.

5. Описание протокола

Протокол опирается на использование доверенного линейного вычислителя – в данном случае это смарт-контракт в блокчейне – к которому участники осуществляют запросы.

Назовём состоянием смарт-контракта набор переменных определенного типа.

Каждый пользовательский запрос потенциально может изменять состояние смарт-контракта. В описании протокола мы используем понятие текущего состояния переменных и нового измененного состояния переменных, которое получается в процессе обработки запроса.

Используемые обозначения.

$\mathbb{N}_p = \{0 \dots 2^p - 1\}$ – множество натуральных чисел с заданной верхней границей;

X, X' – значение переменной X на момент получения запроса и значение, на момент окончания обработки запроса соответственно.

Обозначения переменных.

$Id \in 2^{\mathbb{N}} \cup \{0\}$ – множество идентификаторов пользователей системы; условимся использовать число 0 для обозначения *неопределённого* идентификатора.

$r \in \mathbb{N}$ – решение задачи,

$s \in \mathbb{N}$ – производное значение от сертификата,

$solver \in Id$ – идентификатор пользователя, первым представившего правильное решение задачи,

$V \in 2^{Id}$ – множество идентификаторов пользователей, представивших доказательство проведенной перепроверки опубликованного сертификата,

$L \in 2^{Id}$ – множество идентификаторов пользователей, осуществлявших попытку компромитировать работу протокола каким-либо способом,

$P \subseteq Id \times \mathbb{N}$ – множество пар – идентификатор и доказательство перепроверки – присланных пользователями, предположительно проводившими перепроверку, до раскрытия секрета, позволяющего установить правдивость перепроверки.

Вход: Пользовательская задача $f: \mathbb{N} \to \mathbb{N}$ и входные данные $d \in \mathbb{N}$.

Выход: $\langle r, s, solver, V, L \rangle$

Предварительные действия.

1. Пользователь задаёт функцию f и начальную точку d. Про требования, накладываемые на функцию f и её взаимосвязи с пользовательской задачей C было сказано ранее.
2. Выбирают криптографически стойкую хэш-функцию $H: \mathbb{N} \to \mathbb{N}_q$, параметр q выбирают по рекомендациям к хэш-функции.
3. Пользователь конструирует вычислительную функцию
$$F(\langle x, c \rangle) \coloneqq \boldsymbol{IF}\ x == f(x)\ \boldsymbol{THEN}\ \langle x, c \rangle\ \boldsymbol{ELSE}\ \langle f(x), H(x \circ c) \rangle$$
Здесь $\circ: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ – какая-либо бинарная операция. Вычисление начинается из точки $\langle d, H(d) \rangle$
4. Выбирают семейство функций $\pi_k: \mathbb{N}^k \to \mathbb{N}_p^k$ для $k \geq 1$, такое, что
$$\pi_1: \mathbb{N} \to \mathbb{N}_p - \text{какая-либо хэш-функция (не обязана совпадать с } H(x));$$
$$\pi_k(\langle c_1, \dots, c_k \rangle) \coloneqq \langle \pi_1(c_1), \dots, \pi_1(c_k) \rangle, \text{ для } k > 1.$$
Далее индекс k мы будем опускать, так как он понятен из контекста – длина кортежа, поданного на вход.
5. В блокчейне публикуется смарт-контракт, выполненный с возможностью:
   - Публиковать заявку на вычисление, указывая функцию F и начальное значение d; заявке назначается статус “опубликована”.
     На каждую новую заявку заводится отдельный набор переменных состояния $\langle r, s, solver, V, L, P \rangle$, изначально пустые.
   - Получать актуальную информацию про заявки: F, d, статус заявки;
     Если заявка находится в статусе “исполнена”, то дополнительно получать значение проекции сертификата cp, решение задачи r и соответствующее ему значение c_n.
     Если заявка находится в статусе “валидирована”, то дополнительно получать набор значений $\langle r, s, solver, V, L \rangle$
   - Для неисполненных заявок, помечать заявку как исполненную, указывая значения r_n, c_n, cp, hc , такие, что:
$$F(\langle r_n, c_n \rangle) = \langle r_n, c_n \rangle,$$
$$cp \coloneqq \pi(\langle c_1, \dots, c_n \rangle),$$
$$hc \coloneqq H(H(\langle c_1, \dots, c_n \rangle)).$$
     Пусть u – идентификатор пользователя, приславшего решение.
     Смарт-контракт проверяет, что предъявленная пара $\langle r_n, c_n \rangle$ является неподвижной

точкой F, в этом случае заявка помечается как “исполненная” и

$$solver' = u \wedge r' = r_n.$$

В случае, если $\langle r_n, c_n \rangle$ не является неподвижной точкой, решение отклоняется, статус заявки не меняется. В этом случае,

$$L' = L \cup \{u\}.$$

- Для исполненных заявок, опровергать предъявленное решение: для опубликованной проекции сертификата cp предьявлять значения i, c_{i-1}, c_i, r_{i-1}, такие, что:

$$\pi(c_{i-1}) = cp[i-1] \wedge F(\langle r_{i-1}, c_{i-1} \rangle) = \langle r_i, c_i \rangle \wedge \pi(c_i) = cp[i], \quad (5.1)$$

$$c_{i+1} \coloneqq H(r_i \circ c_i), \quad (5.2)$$

$$\pi(c_{i+1}) \neq cp[i+1]. \quad (5.3)$$

Здесь $cp[i]$ – элемент кортежа cp, стоящий на позиции i.

Пусть u – идентификатор пользователя, приславшего опровержение.

В случае, если смарт-контракт устанавливает, что предъявленное опровержение – корректное, заявка переводится в статус “опубликована”. В этом случае

$$L' = L \cup \{solver\} \wedge V' = V \cup \{u\} \wedge solver' = 0.$$

- Для заявок, находящихся в статусе “исполнена”, предъявлять доказательство проверки решения prf, такое, что:

$$prf := H(H(\langle c_1, \dots, c_n \rangle) \circ id),$$

где id – уникальный идентификатор предьявляющей стороны.

При этом

$$P' = P \cup \{(id, prf)\}.$$

- Для заявок, находящихся в статусе “исполнена”, по прошествии периода времени T, предьявлять секрет s пользователем с идентификатором $solver$, такой что

$$hc = H(s).$$

Если проверка не проходит, заявка возвращается в статус “опубликована” и

$$L' = L \cup \{solver\},$$

$$solver' = 0.$$

Иначе статус заявки меняется на “валидирована”. В этом случае

$$V' = V \cup \{x : (x, p) \in P \wedge H(hc \circ x) = p\},$$

$$L' = L \cup \{x : (x, p) \in P \wedge H(hc \circ x) \neq p\}.$$

Шаги протокола

1. Пользователь размещает заявку на вычисление в смарт-контракте, указывая F, d.
2. Участники вычитывают из смарт-контракта заявки со статусом “опубликована”, и проводят поиск неподвижной точки функции F, начиная поиск из точки $\langle d, H(d) \rangle$.
3. После получения результата, исполнитель публикует значения r_n, c_n, cp, hc вычисленые по описанным выше правилам в смарт-контракте.
4. Другие участники, выполнявшие вычисление, перепроверяют опубликованный результат, сравнивая значение cp с тем, которое получилось у них.
   - В случае обнаружения несовпадения, отправляют в смарт-контракт опровержение решения.
   - В случае совпавшего решения, отправляют в смарт-контракт доказательство проверки решения.
5. Исполнитель отправляет в смарт-контракт значение s – хэш-значение, посчитанное от сертификата – делая возможным валидировать проверяющих.
   - В случае успешной валидации секрета смарт-контрактом, для данной заявки получен

выход протокола: $\langle r, s, solver, V, L\rangle$

- Иначе заявка возвращается в статус “опубликована” и работа протокола продолжается с пункта 2.

6. Безопасность протокола

Под *нарушителем* мы понимаем участника протокола, желающего нарушить функциональные свойства протокола с какой-либо целью. Под степенью безопасности протокола мы понимаем вероятность наступления нежелательного события, при котором нарушаются функциональные свойства протокола в рамках заданной модели нарушителя.

Определение 4. (Модель нарушителя M_1)

- Для нарушителя, поиск прообраза по заданному образу выбранной хэш-функции $H(x)$ является вычислительно трудной задачей.
- Нарушитель обладает знанием устройства функции $f(x)$ не меньшим, чем заказчик вычисления.

Определение 5. (Угрозы безопасности протокола)

- Событие E_1 : Опровержение корректного решения.
- Событие E_2 : Ложное доказательство перепроверки решения.

Теорема 2. Опровержение опубликованного кем-то корректного решения (событие E_1) является вычислительно трудной задачей для нарушителя из M_1.

Доказательство. Для опровержения опубликованного решения, протокол требует предоставить такие i, c_{i-1}, c_i, r_{i-1}, такие что:

$$\pi(c_{i-1}) = cp[i-1] \wedge \pi(c_i) = cp[i] \wedge H(r_{i-1} \circ c_{i-1}) = c_i \quad (6.1)$$

$$F(\langle r_i, c_i\rangle) = (r_{i+1}, c_{i+1}) \wedge \pi(c_{i+1}) \neq cp[i+1] \quad (6.2)$$

Предположим, что $F: \mathbb{N} \times \mathbb{N}_q$ всюду определена (на практике это далеко не так, но чтобы облегчить работу злоумышленнику, положим). Заметим, что в этом случае выполнить требование (6.1) просто, оно будет выполняться почти для любой точки $\langle r, c\rangle$. Поэтому основная нагрузка ложится на предикат (6.2).

Пусть хэш-функции имеют сигнатуры: $H: \mathbb{N} \to \mathbb{N}_q$, $\pi: \mathbb{N}_q \to \mathbb{N}_p$, и известно, что H является криптографически стойкой, тогда задача нахождения подстановки в предикат (1) сводится к задаче поиска прообразов указанных функций по заданным значениям образов – известная вычислительно сложная задача, которая при выборе оптимальных p, q является трудноосуществимой для нарушителя из M_1.

Теорема 3. Предоставление ложного доказательства перепроверки опубликованного решения (событие E_2) является вычислительно трудной задачей для нарушителя из M_1.

Доказательство. Для доказательства перепроверки опубликованного кем-то решения, протокол требует предоставить такое значение:

$$prf = H(s \circ id)$$

Здесь id – уникальный идентификатор участника, предоставляющего доказательство. На каждый уникальный id допускается публиковать только одно такое prf в качестве доказательства. Из полезной информации, злоумышленнику известны значения:

$$hc = H(s) \quad (3)$$

$$prf = H(s \circ id_i) \quad (4)$$

при этом идентификаторы id_i также известны. Здесь $s = H(\langle c_1, c_2, \dots c_n\rangle)$, т.е. хэш от сертификата вычисления. Вероятность отыскать искомый прообраз для s:

$$P_s = \frac{1}{2^{q \cdot n}}, \text{где } , q \geq 64, n \gg 10^3$$

то есть событие маловероятное. Нарушитель может попробовать отыскать коллизию в (3), т.е. найти s', такое что $H(s') = hc$, но в этом случае, при удачно выбранной операции $\circ$, вероятность одновременного выполнения всех равенств из (4) (для всех id) с найденным s' также будет незначительной.

6.1 Экономические стимулы

Ранее мы обсудили некоторые аспекты криптографической безопасности протокола. Учитывая, что описанный протокол в качестве доверенного линейного вычислителя использует смарт-контракт, выполняемый в публичной блокчейн-сети, есть возможность введения дополнительных мер, существенно повышающих мотивацию рациональных участников следовать протоколу – *экономические стимулы*.

В этой парадигме, стабильность протокола зависит не только от безопасности криптографических механизмов, но и от выбранной системы штрафов и поощрений для участников. Мы не будем подробно останавливаться на этой теме, опишем общую идею возможных вариантов такой системы в виде поправок к описанию протокола.

1. Пользователь размещает заявку на вычисление в смарт-контракте, указывая F, d, снабжая свой запрос криптовалютным депозитом размером D_r.
2. Участники вычитывают задачу и начинают поиск решения.
3. После получения результата кем-то из вычислителей, первый, кто нашел решение, публикует результат, снабжая свое решение криптовалютным депозитом D_s.
4. Другие участники перепроверяют результат.
   - В случае расхождения, публикуют опровержение, снабжая опровержение депозитом D_p.
   - В случае совпадения, отправляют в смарт-контракт доказательство проверки решения, с депозитом D_w.

В конце работы протокола, смарт-контракт имеет на своём счету количество криптовалюты, равное $s = D_r + D_s + m \cdot D_p + n \cdot D_w$, где n – количество участников, доказавших факт перепроверки решения, m – количество раз, когда решение опровергалось. Напомним, что выходом протокола является набор: $\langle r, s, solver, V, L \rangle$, который позволяет перераспределить криптовалютный фонд размера s между исполнителем $solver$ и добросовестными проверяющими V, штрафуя нарушителей из L.

7. Особенности протокола

У предложенного протокола есть несколько мест, которые могут вызвать трудности в практической реализации, а некоторые моменты могут представлять вектора атак для злоумышленников. Ниже мы обсуждаем возможные пути их преодоления.

7.1 Преобразование вычисления в итеративный вид

Как было показано в Теореме 1, вычисление всегда можно привести к итеративному виду, и существует множество способов это сделать.

Например, в работе [17] используется такой подход: пользовательское вычисление компилируется из языка высокого уровня в байткод некоторой виртуальной машины. В качестве одного шага вычисления берется n шагов интерпретатора этого байткода, достигая какого-то состояния, т.е. значений регистров и памяти. Это состояние берется за промежуточный результат вычисления, который подается на вход на следующей итерации, и так далее, до достижения результата.

Мы используем другой подход: сама вычислительная программа из её оригинального вида преобразуется в вид, напоминающий т.н. *стиль передачи продолжений* (Continuation Passing Style) [14], но вместо хвостового вызова, аргумент вместе с указанием какой вызов делать следующим возвращается в качестве вычисленного значения. Такое представление, достигая того же технического результата, может приводить к более компактным представлениям состояний и не требует наличия специального интерпретатора. Существуют способы автоматической трансляции функциональных программ в подобное представление [1].

7.2 Расходящаяся функция F

Представим атаку, при которой в смарт–контракт записывается расходящееся вычисление. В этом случае, исполнители потратят ресурс, но решения не найдут - налицо убыток.

Есть несколько способов преодолеть эту трудность:

- Оформлять вычисление на языке, *гарантирующем* завершаемость. Например, *язык примитивно-рекурсивных функций* даёт такую гарантию и позволяет реализовывать большинство практически интересных алгоритмов [21]. При этом, в смарт-контракт вместе с байткодом функции f передавать ее исходный код P. Вычислитель, компилируя P в f', выполняет проверку $f = f'$. В случае, если проверка успешна, вычислитель убеждается в завершаемости f и принимается за работу.
- Расширить протокол, добавив возможность сертифицировать частичный результат вычислений, т.е. результат, который не сошелся к неподвижной точке, но выполнен корректно с точки зрения сертификата. В случае, если никто из проверяющих не представит более полного (длинного) решения, представленное решение считается удовлетворительным.

7.3 Размер байткода F

Может оказаться так, что размер вычислительной функции или начальных данных слишком велик для размещения в смарт-контракте или ином выбранном ДЛВ. На текущий момент, это принципиальное ограничение рассматриваемого способа: публикуемые F и d должны удовлетворять критерию: $|F| + |d| \leq T_{max}$, где T_{max} – максимальный размер операции для выбранного ДЛВ.

В случае смарт-контрактов, если это условие не выполняется, то такая транзакция отклоняется.

7.4 Размер точки r

В случае разногласий по решению, протокол обязывает предъявить точки c_{i-1}, c_i, r_{i-1}, доказывающие несостоятельность опубликованного решения.

Точка r_{i-1} – это по сути промежуточное состояние вычисления. Может быть так, что размер r_{i-1} становится слишком большим для загрузки в смарт-контракт. На текущий момент, это принципиальное ограничение рассматриваемого способа. Публикуемое вычисление должно удовлетворять критерию: $\forall r, |f(r)| \leq T_{max}$, где T_{max} – максимальный размер операции для выбранного ДЛВ.

7.5 Размер сертификата

Может быть так, что проекция сертификата по размеру не помещается в допустимые размеры операции выбранного ДЛВ. В случае использования смарт-контракта, возможно реализовать хранение проекции во внешнем неизменяемом хранилище, например, в распределенной файловой системе IPFS [2], доступ к ней можно реализовать с помощью технологии оракулов [10].

8. Экспериментальная часть

Для испытания протокола на практике мы запрограммировали логику смарт-контракта и одну пользовательскую задачу.

В силу временных ограничений, мы не стали использовать блокчейн-платформу как среду для испытаний, а реализовали логику смарт-контракта и прикладную задачу для BEAM-машины функционального языка программирования Erlang. [2]

В качестве пользовательской задачи, подаваемой на вход смарт-контракту, мы выбрали задачу UNSAT – доказательство отсутствия набора присваиваний для булевых переменных заданной формулы КНФ, такой, что после подстановки значений в формулу, формула вычисляется в истину. Задача UNSAT сопряжена с известной NP-полной задачей SAT [5], но, в отличии от SAT, её решение в общем случае не может быть проверено за полиномиальное время.

Наш выбор объясняется тем, что алгоритм решения задачи UNSAT имеет прямое практическое применение – верификация моделей программ, закодированных в виде системы КНФ.

Существует множество алгоритмов решения SAT/UNSAT, и известно, что никакой из них не гарантирует время работы меньшее, чем $O(2^n)$, где n – число булевых переменных КНФ, хотя на практике оказываются весьма эффективны.

Мы выбрали один из простейших таких алгоритмов – алгоритм DPLL [6]. Перед передачей его в смарт-контракт, необходимо было преобразовать его в вид, описанный в разд. **4.**

В качестве данных, передаваемых в алгоритм DPLL, мы выбрали задачу доказательства эквивалентности двух программ, реализующих логику очереди FIFO (задача взята из [3]). Задача закодирована в виде КНФ-формулы таким образом, что, в случае, если $DPLL(d) = Unsat$, то это означает их эквивалентность. В противном случае, алгоритм вернет различия двух поведений.

В данном эксперименте, нас интересовало следующее:

- Насколько изменится время вычисления функции f (обозначим за T_1) после представления её в итеративной форме (T_2)?
- Насколько изменится размер байткода функции f (обозначим за S_1) после представления её в итеративной форме (S_2)?
- Каким окажется количество итераций вычисления f до получения неподвижной точки (обозначим за n), и полный размер сертификата (C_f)?
- Каким окажется размер наибольшей промежуточной точки $f^i(d)$ (обозначим за d_{max}), и размер исходных данных d_0?

Эксперименты проводились на серверном узле Intel Xeon Gold 6254 CPU 3.10GHz x 4 Cores, 16GB RAM; среда исполнения: Erlang 20, ERTS 9.3.3.11, под управлением ОС Linux Fedora 29 (виртуальная машина). Результаты приведены в Табл. 1.

Табл. 1. Результаты экспериментов
Table 1. Experimental results.

	d_0	S_1	S_2	T_1	T_2	d_{max}	C_f	n
$QueueInvar_2$	5703	2064	2272	0.1	0.4	42028	52992	1656
$QueueInvar_4$	13707	2064	2272	193	423	170758	13145664	410802

[2] Первоначальные попытки реализовать код прикладного алгоритма на языке Solidity (блокчейн-платформа Ethereum) показали, что это крайне изматывающая деятельность, в силу ограниченной выразительной силы этого языка и ограничений виртуальной машины EVM.

T_1, T_2 измеряется в секундах, $d_0, d_{max}, S_1, S_2, C_f$ – в байтах, n – в штуках
T_1, T_2 are measured in seconds, $d_0, d_{max}, S_1, S_2, C_f$ – in bytes, n – number of iterations

Кроме этого, был реализован тестовый сценарий, проверяющий работу шага опровержения неверного решения, состоящий из таких шагов:

1. Публикуется задача DPLL(x), представленная в итеративной форме, и начальные данные d. Размеры пользовательской программы S_2 и данных d_0 позволяют отправить их в доверенный вычислитель – смарт-контракт.
2. Один из исполнителей намеренно публикует *неверное* решение задачи, состоящее из значений r_n, cp, hc. Размер значения cp может оказаться больше, чем допускает смарт-контракт. В этом случае, содержимое cp публикуется на внешнем распределенном неизменяемом хранилище IPFS, и вместо cp отправляется ссылка на объект в IPFS. При этом, обеспечивать доступность этого объекта – обязанность исполнителя.
3. Другой исполнитель проверяет опубликованное решение и, обнаружив ошибку, конструирует опровержение, отправляя в смарт-контракт значения i, c_{i-1}, c_i, r_{i-1}
4. Смарт-контракт проверяет корректность опровержения, осуществляя проверки (5.1), (5.2), (5.3) из разд. **5**. Если вместо значения cp была отправлена ссылка в IPFS, смарт-контракт запрашивает содержимое этой ссылки по адресу $i-1$ и i через доверенного оракула, получая значения c_{i-1}, c_i. В случае если ссылка недоступна, решение отклоняется.

Мы выложили программные тексты макета протокола и прикладной задачи в открытый доступ [3]. Всё, что касается работы с хранилищем IPFS и оракулом, моделируется с помощью обычного программного кода, без обращений к реальным сервисам.

9. Обзор работ

Верификация результатов вычислений, проведенных третьей стороной – это одна из актуальных задач в теории современной криптографии. Было предложено большое количество разных подходов, каждый со своим набором компромиссов.

Известно семейство способов решения описанной задачи, основанное на применении PCP теоремы для вычислений из класса NP. Напомним, что класс NP – класс задач разрешимости, у которых есть решения, проверяемые за полиномиальное время.

PCP теорема говорит о том, что решение любой задачи в классе NP может быть проверено полиномиальным алгоритмом с помощью фиксированного числа случайно выбранных структурных элементов решения. Более формально, $NP = PCP[O(\log n), O(1)]$, где n – размер входных данных, $\log n$ – количество случайных бит на 1 запрос к решению.

Для способов из этого семейства характерна такая схема взаимодействия сторон: 1) Заказчик оформляет свою задачу в виде булевой схемы (схемы функциональных элементов [22]), по схеме строит многочлен – алгебраическую нормальную форму (АНФ; количество слагаемых в АНФ зависит от количество функциональных элементов в схеме [19]); посылает исполнителю схему и данные для задачи 2) Исполнитель из булевой схемы выводит соответствующую ей булеву функцию, из которой получает АНФ, кодирует выход каждого из узлов схемы в АНФ; далее формирует полученный результат вычисления и сертификат, при этом сертификат может храниться у исполнителя, в облаке, либо передаваться с результатом заказчику 3) Заказчик использует один из подходов для проверки сертификата, при этом ему достаточно случайно выбрать ограниченное количество узлов сертификата для проверки в каждом из подходов.

[3] https://bitbucket.org/unboxed_type/safecomp/src

Известны такие подходы к проверке сертификата: 1) Интерактивный подход без договоренностей заключается в осуществлении запросов к сертификату, находящемуся у исполнителя (или в облаке) произвольных значений узлов – элементов схемы – до тех пор, пока заказчик не удостоверится в правильности вычисления [8] [20] [18] 2) Интерактивный подход с договорённостями заключается в формировании договорённостей посредством криптографических протоколов, проверяющих целостность сертификата. Ключевым отличием от предыдущего подхода является значительное расширение класса проверяемых задач [15] [16] [13].

Табл..2. Асимптотические оценки для некоторых алгоритмов сертификации
Table 2. Asymptotic estimates for several certification algorithms

	$inter(n)$	$proof(n)$	$verify(n)$	$cert(n)$
Libra Non-ZKP	$O(d(n)\cdot \log s(n))$	$O(s(n))$	$O(d(n)\cdot \log s(n))$	$O(d(n)\cdot \log s(n))$
TrueBit	$O(\log n)$	$O(n\log n)$	$O(1)$	$O(n)$
SafeComp	1	$O(n)$	$O(1)$	$O(n)$
$proof(n)$ – сложность построения доказательства, $verify(n)$ – проверка доказательства, $inter(n)$ – количество раундов, $cert(n)$ – размер сертификата, $s(n)$ – размер схемы функциональных элементов в зависимости от размера входа задачи. $d(n)$ – диаметр схемы, т.е. максимальная длина пути в схеме $proof(n)$ – complexity of proof construction, $verify(n)$ – complexity of proof verification, $inter(n)$ – number of interaction rounds, $cert(n)$ – size of certificate, $s(n)$ – size of functional element circuit as a function of task input size, $d(n)$ – depth of a circuit, i.e. a maximum path length in the circuit				

Подходы из семейства PCP накладывают существенные ограничения на проверяемые вычисления. Например, должны быть известны точные верхние оценки на количество итераций циклов и размеры структур данных.

Протокол Libra [20] относится к категории способов, опирающихся на PCP теорему. Он не сопоставим напрямую с нашим протоколом, так как имеет ряд ограничений, например, на структуру пользовательских вычислений. Тем не менее, в некоторых случаях, он может быть применен для получения того же технического результата, поэтому мы привели оценки и для него.

В Табл. 2 сравнивается асимптотика ближайших известных аналогов.

9.1 Сравнение с TrueBit

С появлением блокчейнов стал возможен подход, использующий смарт-контракт в качестве доверенного арбитра для разрешения разногласий, при этом экономические стимулы мотивируют участников выполнять протокол добросовестно.

Пионерской работой в этой области стал протокол TrueBit [17]. Протокол также использует смарт-контракт в качестве доверенного арбитра в случае разногласий исполнителей относительно результата, но имеет существенные отличия от нашего протокола.

Форма вычислительной задачи. В случае с TrueBit, пользовательская задача оформляется в виде байткода некоторой виртуальной машины, при этом интерпретатор такого байткода должен быть размещен в смарт-контракте. Данный подход теоретически позволяет компилировать существующие тексты программ без изменений, однако необходим интерпретатор, реализованный на языке смарт-контрактов блокчейна – механизм довольно сложный, и до сих пор, насколько нам известно, не существует реализаций такого интерпретатора для какой-либо популярной вычислительной архитектуры. В нашем способе, напротив, пользовательское вычисление должно быть представлено в виде итеративной сходящейся к ответу функции, оформленной на существующем языке смарт-контрактов выбранной платформы. Это можно считать неудобством, так как требуется представить

программу в ином виде, но за то делает её пригодной для использования без развертывания дополнительного интерпретатора.

Процедура разрешения разногласий. Для разрешения разногласий протокол TrueBit опирается на интерактивную игру между исполнителями. Игра проходит в несколько раундов: количество раундов оценивается как $O(\log n)$, где n – количество шагов вычисления. На каждом раунде, оба исполнителя-соперника вычисляют какое-то количество корней дерева Меркель: в качестве листьев берутся промежуточные состояния вычисления. Сложность получения одного такого корня имеет оценку $O(n)$. Совокупная сложность построения доказательства, за все раунды, оценивается как $O(n \log n)$. Такое устройство процедуры разрешения разногласий позволяет оптимизировать объем данных, отправляемых в смарт-контракт, однако ценой увеличения совокупной сложности получения доказательства.

В нашем протоколе используется другой подход: исполнитель вместе с решением публикует проекцию сертификата, по которой можно быстро – за $O(\log n)$ отыскать место разногласия и построить доказательство-опровержение. Длина и вычислительная сложность получения сертификата имеют оценку $O(n)$. В случае “длинных” вычислений, для хранения проекции сертификата мы используем внешнее неизменяемое хранилище, например, IPFS. В качестве связующего звена между смарт-контрактом и хранилищем может использоваться доверенный оракул. Таким образом, процедура разрешения разногласий проходит ровно за 1 раунд, понижая оценку сложности с $O(n\log n)$ до $O(n + \log n) = O(n)$, однако ценой записи дополнительных O(n) байт данных в сеть.

10. Заключение

В статье предложен вариант решения задачи сертификации целостности облачных вычислений при заданных ограничениях на вычислительную задачу. Приведено формальное описание протокола, проведен анализ безопасности протокола в рамках описанной модели нарушителя. По сравнению с ближайшим аналогом, наш протокол существенно снижает сложность построения доказательства, а количество раундов коммуникации сводит до одного. Приведены первые экспериментальные результаты, доказывающие, что протокол реализуем на практике с приемлемыми издержками.

В будущих работах по данной теме хотелось бы 1) глубже исследовать свойства протокола, влияющие на его стабильность, например, предложить доказуемо надёжную модель экономических стимулов для участников; 2) реализовать протокол на каком-то публичном блокчейне, и, например, провести верификацию актуальных моделей программ через решение задачи UNSAT.

Список литературы / References

[1] Appel A. W., Jim T. Continuation-passing, closure-passing style. In Proc. of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 1989, pp. 293-302.
[2] Benet J. Ipfs-content addressed, versioned, p2p file system. arXiv preprint, arXiv:1407.3561, 2014.
[3] Biere A., Cimatti A., Clarke E., Zhu Y. Symbolic model checking without BDDs. Lecture Notes in Computer Science, vol. 1579, 1999, pp. 193-207.
[4] Buterin V. What is Ethereum? Ethereum Official webpage. Available at: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html, accessed 14.08.2020.
[5] Cook S. A. The complexity of theorem-proving procedures. In Proc. of the Third Annual ACM Symposium on Theory of Computing, 1971, pp. 151-158.
[6] Davis M., Logemann G., Loveland D. A machine program for theorem-proving. Communications of the ACM, vol. 5, no. 7, 1962, pp. 394-397.
[7] Dershowitz N., Hanna Z., Nadel A. A scalable algorithm for minimal unsatisfiable core extraction. Lecture Notes in Computer Science, vol. 4221, 2006, pp. 36-41.

[8] Goldwasser S., Kalai Y.T., Rothblum G.N. Delegating computation: interactive proofs for muggles. Journal of the ACM, vol. 62, no. 4, 2015, pp. 1-64.
[9] Hopcroft J.E., Motwani R., Ullman J.D. Introduction to Automata Theory, Languages, and Computation: Pearson New International Edition. Pearson, 2013, 496 p.
[10] Kochovski P., Gec S., Stankovski V., Bajec M., Drobintsev P.D. Trust management in a blockchain based fog computing platform with trustless smart oracles. Future Generation Computer Systems, vol. 101, 2019, pp. 747-759.
[11] Luu L., Teutsch J., Kulkarni R., Saxena, P. Demystifying incentives in the consensus computer. In Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 706-719.
[12] Mandt T., Solnik M., Wang D. Demystifying the secure enclave processor. Available at: http://mista.nu/research/sep-paper.pdf, accessed 14.08.2020.
[13] Parno B., Howell J., Gentry C., Raykova M. Pinocchio: Nearly practical verifiable computation. In Proc. of the IEEE Symposium on Security and Privacy, 2013, pp. 238-252.
[14] Reynold, J. C. The discoveries of continuations. Lisp and symbolic computation, vol. 6, no. 3-4, 1993, pp. 233-247.
[15] Setty S., Braun B., Vu V., Blumberg A. J., Parno B., Walfish M. Resolving the conflict between generality and plausibility in verified computation. In Proc. of the 8th ACM European Conference on Computer Systems, 2013, pp. 71-84.
[16] Setty S., Vu V., Panpalia N., Braun B., Blumberg A. J., Walfish M. Taking proof-based verified computation a few steps closer to practicality. In Proc. of the 21st USENIX Security Symposium, 2012, pp. 253-268.
[17] Teutsch J., Reitwießner C. A scalable verification solution for blockchains. arXiv preprint arXiv:1908.04756, 2019.
[18] Thaler J. Time-optimal interactive proofs for circuit evaluation. Lecture Notes in Computer Science, vol. 8043, 2013, pp. 71-89.
[19] Vollmer, H. Introduction to circuit complexity: a uniform approach. Springer Science & Business Media, 2013, 272 p.
[20] Xie T., Zhang J., Zhang Y., Papamanthou C., Song D. Libra: Succinct zero-knowledge proofs with optimal prover computation. Lecture Notes in Computer Science, vol. 11694, 2019, pp. 733-764.
[21] Верещагин Н.К., Шень, А. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. Учебное пособие. МЦНМО, 2013, 160 стр. / Vereshchagin N.K., Shen A. Lectures on mathematical logic and theory of algorithms. Part 3. Computable functions. MCCME, 2013, 160 p. (in Russian).
[22] Ложкин С. А. Лекции по основам кибернетики. Издательский отдел ф-та ВМиК МГУ, 2004 г., 251 стр. / Lozhkin S.A. Lectures on the basics of cybernetics. Publishing department of the Faculty of Computational Mathematics and Cybernetics, Moscow State University, 2004, 251 p. (in Russian).

Приложение 1

Теорема. Для любой вычислимой в точке d функции $C\colon \mathbb{N} \to \mathbb{N}$ существуют такие вычислимые функции

$$inj\colon \mathbb{N} \to T, proj\colon T \to \mathbb{N}, F\colon T \to T$$

такие, что

$$proj(\text{fix } F\, inj(d)) = C(d)$$

где fix: $(T \to T) \times \mathbb{N} \to T$ – оператор, вычисляющий неподвижную точку функции, стартуя из заданной точки, т. е.

$$\text{fix}\, F\, p = F(\text{fix}\, F\, p)$$

Здесь T – некоторое конечное множество.

Доказательство. Положим

$$inj(d) = \langle \mathrm{d}, 0 \rangle \qquad proj(\langle x, y \rangle) = x$$
$$F(\langle x, 0 \rangle) = \langle \mathrm{C}(x), 1 \rangle \qquad F(\langle x, 1 \rangle) = \langle x, 1 \rangle$$

Выбранные функции удовлетворяют условиям теоремы. Такое представление F можно назвать *тривиальным* – оно никак не помогает разделить функцию $C(x)$ на составные более

простые части. Можно ли убедиться в существовании представления F, отличного от тривиального? Да, можно.

Согласно тезису Тьюринга, любую вычислимую в точке функцию $C(x)$ можно представить в виде машины Тьюринга. Пусть M – машина, соответствующая функции $C(x)$, т.е.

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, Q_f),$$

$$Q - \text{множество состояний машины}, \Gamma - \text{алфавит символов ленты},$$

$$\Sigma \subseteq \Gamma - \text{входной алфавит}, Q_f \subseteq Q - \text{множество принимающих состояний},$$

$$q_0 \in Q - \text{начальное состояние},$$

$$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\} - \text{ функция шага машины.}$$

Тогда функцию F можно определить следующим образом:

$$F(\langle M, I, q, p\rangle) = \begin{cases} \langle M, I', q', p'\rangle, \text{ если } q \notin Q_f \\ \langle M, I, q, p\rangle, \text{ иначе} \end{cases},$$

где

$$(q', X, p') = \delta(q, I_p), \text{ где } I_p - \text{содержимое ленты на позиции } p,$$

$$I' = I[X/p].$$

Делая один шаг из состояния q, находясь на ленте на позиции p, машина M меняет состояние ленты, получая новое состояние ленты I' (заменяя содержимое клетки p на X) и новую позицию головки p'. Если достигнуто принимающее состояние, шагов больше не производится – достигается неподвижная точка F.

По условию теоремы функция C является вычислимой в точке d, поэтому через какое-то количество итераций машина обязательно перейдет в принимающее состояние.

Заметим, что элементы вида $\langle M, I, q, p\rangle$ могут быть закодированы, например, конечным числом натуральных чисел. Пример такой кодировки можно найти в [9].

Сама функция F вычислима по той причине, что опирается на функцию шага $\delta(q, X)$, функцию проверки принадлежности элемета к множеству $q \in Q_f$ и функцию выбора $if - then - else$. Все эти функции вычислимы в случае конечности множеств Q и Γ.

Пусть

$$inj(d) := \langle M, I_0, q_0, 1\rangle,$$

где I_0 – начальное состояние ленты, на которой записано в том числе d.

Функция $proj(\langle M, I, q, p\rangle)$ извлекает из ленты I ответ $C(d)$.

Такие функции можно построить, потому что на ленте всегда возможно зафиксировать область для начального значения и для ответа. Ответ $C(d)$ представим в виде конечного числа клеток ленты, так как вычислимость функции в точке гарантирует конечное число шагов машины.

Если размер области для ответа заранее неизвестен, мы можем зафиксировать на ленте позицию начала, и уникальный маркер, наличие на ленте которого означает конец области, отведенной для ответа.

Описанное представление тройки $F, inj, proj$ удовлетворяют требованиям теоремы, и не является тривиальным представлением.

Информация об авторах / Information about the authors

Евгений Сергеевич ШИШКИН – ведущий исследователь научного отдела. Сфера научных интересов: дедуктивная формальная верификация программ, автоматическая верификация моделей программ, логические дедуктивные системы, языки формальных спецификаций, интерактивные среды построения доказательств, распределенные системы, блокчейны и смарт-контракты.

Evgeny Sergeevich SHISHKIN – senior researcher in the research department. Research interests: deductive formal verification of computer programs, automatic verification of program models, logical deductive systems, formal specification languages, interactive proof assistants, distributed systems and algorithms, blockchains, smart-contracts.

Евгений Сергеевич КИСЛИЦЫН – аспирант кафедры высшей алгебры механико-математического факультета. Научные интересы включают исследование неассоциативных структур, теорию колец, гомоморфное шифрование, формализацию распределенных протоколов.

Evgeny Sergeevich KISLITSYN – Postgraduate student of the Department of Higher Algebra, Faculty of Mechanics and Mathematics. Research interests include the study of non-associative structures, ring theory, homomorphic encryption, and formalization of distributed protocols.

DOI: 10.15514/ISPRAS–2020–32(4)–9

Метрики эффективности и производительности при использовании эволюционного алгоритма на грид-системах из персональных компьютеров

Н.П. Храпов, ORCID: 0000-0002-8038-7625 <nkhrapov@gmail.com>
Институт проблем передачи информации им. А.А. Харкевича РАН,
127051, Россия, г. Москва, Большой Каретный пер., д. 25, стр. 1

Аннотация. Область применения систем добровольческих вычислений постоянно расширяется. Существует много научных работ по адаптации различных вычислительных алгоритмов к ГСПК. Темой представленной работы является эффективная адаптация эволюционного алгоритма к системам добровольческих вычислений. Рассматриваются причины потерь производительности, предлагаются критерии и метрики оценки качества работы алгоритма. Рассматриваемые метрики могут быть использованы для сравнительного анализа различных политик планирования заданий при проведении вычислений. Вводимые метрики могут быть посчитаны как на имитационных моделях, так и в процессе проведения практических вычислений.

Ключевые слова: грид-системы; распределённые вычисления; эволюционный алгоритм; эффективность; производительность

Для цитирования: Храпов Н.П. Метрики эффективности и производительности при использовании эволюционного алгоритма на грид-системах из персональных компьютеров. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 133–140. DOI: 10.15514/ISPRAS–2020–32(4)–9

Благодарности: Работа выполнена при поддержке гранта РФФИ №19-07-00911

Metrics of efficiency and productivity when using the evolutionary algorithm on desktopgrid

N.P. Khrapov, ORCID: 0000-0002-8038-7625 <nkhrapov@gmail.com>
Kharkevich Institute for Information Transmission Problems of the RAS,
19, Bolshoy Karetny per., Moscow, 127051, Russia

Abstract. The scope of voluntary computing systems is constantly expanding. The BOINC system is currently the most famous for organizing volunteer computing. There are many scientific papers on the adaptation of various computational algorithms to the BOINC. The topic of the presented work is the effective adaptation of the evolutionary algorithm to voluntary computing systems. The reasons for productivity losses are considered, criteria and metrics for evaluating the quality of the algorithm are introduced. The content of the article consists of two main parts. The first part of the article discusses general metrics that can be used to assess the performance of various computational algorithms within BOINC. The problem of adaptation of the evolutionary algorithm is considered in the second part of the article. Two main problem-specific causes of productivity loss are considered: the effect of waiting for the last job and the effect of a common queue. Methods are proposed for quantitative assessment of the influence of various systemic effects on the performance of an evolutionary algorithm. The proposed metrics can be used in a comparative analysis of various job scheduling policies when performing calculations. The metrics can be calculated both during the actual and simulated calculations.

Keywords: grid systems; distributed computing; evolutionary algorithm; efficiency; performance

For citation: Khrapov N.P. Metrics of efficiency and productivity when using the evolutionary algorithm on desktopgrid. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 133–140 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–9

Acknowledgements. The author thanks Russian Foundation for Basic Research (grant №19-07-00911) for financial support.

1. Введение

Технологии грид-систем из персональных компьютеров (далее – ГСПК) существенно повышают доступность вычислительных ресурсов для научных коллективов за счёт привлечения компьютеров добровольцев. Принцип организации вычислений посредством ГСПК состоит в том, что вычислительный узел запрашивает задание у сервера проекта, скачивает и запускает его на исполнение. По окончанию расчётов результат вычислений отправляется обратно на сервер.

Существует множество реализаций программного обеспечения для организации добровольческих вычислений. Наибольшую популярность к настоящему моменту приобрёл программный пакет BOINC (Berkeley Open Infrastructure for Network Computing) [1]. Проект разработки ПО BOINC успешно развивается, начиная с 2004-го года. Технические возможности системы BOINC постоянно расширяются. Существуют расширения, позволяющие объединять в единую инфраструктуру компьютеры добровольцев с кластерами и суперкомпьютерами [2]. Широкое распространение получают модификации системы BOINC, предназначенные для интеграции вычислительных ресурсов на корпоративном уровне [3].

Многие вычислительные алгоритмы, изначально предназначенные для функционирования на параллельных системах, адаптируются к системе BOINC. Проведён ряд исследований по адаптации метода ветвей и границ к ГСПК [4]. Грид-системы из персональных компьютеров успешно используются для решения задач криптоанализа [5]. Описание решения задачи о графах при помощи системы BOINC приведено в [6].

При портировании вычислительного алгоритма на ГСПК необходимо учитывать как специфику самого алгоритма, так и особенности функционирования инфраструктуры. Практика адаптации существующих алгоритмов и ПО к ГСПК показывает, что алгоритм, эффективно функционирующий на вычислительном ресурсе с системой очередей, может неэффективно работать на ГСПК. Для повышения эффективности функционирования эволюционного алгоритма необходимо предварительно выявить его особенности, связанные с организацией вычислений, а также рассмотреть, как эти особенности соотносятся со спецификой функционирования ГСПК.

Практика решения задач оптимизации посредством эволюционного алгоритма (далее – ЭА) ранее описывалась в [7]. Особенности работы ЭА, влияющие на эффективность выполнения на ГСПК рассмотрены в [8]. Темой данной работы является более глубокий анализ причин потери производительности и введение оценивающих метрик для дальнейших работ по выработке оптимальных стратегий управления ресурсами.

2. Метрики производительности и эффективности в ГСПК в общем случае

При анализе источников потерь производительности необходимо учитывать критерии производительности и эффективности вычислительной системы. Для различных решаемых прикладных задач различные критерии является более важными. Рассмотрим два основных критерия производительности:

2.1 Время решения прикладной задачи полностью

Прикладная задача разбивается на некоторое количество независимых друг от друга подзаданий. В процессе вычислений дополнительные подзадания могут генерироваться на основе результатов посчитанных подзаданий. Временем решения прикладной задачи в данном случае будет время между началом расчётов на стороне эволюционного алгоритма и получением результата последнего подзадания.

Рассмотрим возможные метрики в рамках данного подхода. В системах параллельных вычислений Makespan – время между отправкой первого подзадания и получением результата последнего подзадания:

$$M_S = t_{stop} - t_{strart}.$$

Практика проведения вычислений на BOINC показывает, что между началом расчётов эволюционным алгоритмом и отправкой первого задания на вычислительный узел может пройти несколько дней, если очередь заданий на сервере достаточно велика. Поэтому использование параметра *Ms* для системы BOINC потребует важного уточнения, какой момент времени считать началом вычислений: момент поставки первого задания в очередь на сервере или момент отправки первого задания на вычислительный узел. Сложность в детерминировании времени решения задачи обусловлена различием принципов работы системы очередей и ГСПК.

В системе очередей прикладной задаче выделяется необходимый ресурс сразу для всех подзаданий полностью при достижении прикладной задачей своей очереди. Таким образом, время между началом генерации подзаданий эволюционным алгоритмом и получением последнего результата в системе очередей состоит из времени ожидания в очереди и временем выполнения вычислений. Время ожидания в очереди зависит от состояния очереди. А время выполнения вычислений зависит от специфики решаемой задачи и характеристик предоставляемого ресурса.

Отличительной особенностью системы BOINC является единая очередь для всех подзаданий. Это означает, что новые подзадания, генерируемые на основе результатов предыдущих, будут поставлены в конец очереди аналогично первому заданию. Поэтому в системе BOINC состояние очереди заданий влияет как на время отправки первого задания, так и на время проведения непосредственных вычислений. Важно учитывать, что в инфраструктуре ГСПК время решения прикладной задачи не является статической характеристикой. Это означает, что если для одной и той же задачи начать расчёты в различные моменты времени, то продолжительность вычислений может быть различной. Время проведения вычислений будет зависеть от готовности вычислительных узлов, предоставляемых добровольцами, и от загруженности очереди подзаданиями других вычислительных задач. Учитывая, что состояние очереди влияет как на момент начала вычислений, так и на их продолжительность, то наиболее разумным будет за t_{start} считать время постановки первого задания в очередь на BOINC-сервере.

На основе параметра Makespan, можно дать определение ускорения системы (Speedup):

$$Sp = \frac{\sum t_i}{Ms},$$

t_i – время расчёта i-го подзадания.

Применимо к инфраструктуре ГСПК использование параметра Speedup также требует двух важных уточнений.

Первое уточнение связано с использованием репликации. В рамках данной работы, будем считать, что если канонический результат получен из n реплик, то ti – это среднее арифметическое время вычислений.

Второе уточнение связано с тем, какую величину считать временем вычисления отдельного подзадания – время непосредственных вычислений центрального процессора или время между отправкой задания на вычислительный узел и получением результата. В рамках

инфраструктуры ГСПК вычисления на узле могут приостанавливаться и перезапускаться. Поэтому для многих вычислительных узлов время использования центрального процессора и время выполнения подзадания могут сильно отличаться. Параметр ускорения, посчитанный на основе CPU time (далее – Speedupcpu или Spcpu), характеризует численное преимущество инфраструктуры BOINC перед идеально стабильной последовательной машиной с усреднёнными характеристиками вычислительной мощности подключённых вычислительных узлов. Этот параметр актуален при вычислении ускорения гибридной инфраструктуры или при сравнении ускорений, полученных на параллельной системе и на ГСПК.

Рассмотрим параметр ускорения, посчитанный на основе времени пребывания задания на вычислительном узле (далее – Speedup или Sp). Speedup характеризует преимущество системы BOINC перед последовательной вычислительной системой с производительностью, подобной усреднённой *фактической* производительности персонального компьютера добровольца (с учётом перезагрузок, сбросов и отключений). Посчитанная таким образом величина Sp является метрикой ускорения, которое даёт распараллеливание при использовании доступных в ГСПК реальных вычислительных ресурсов. Этот параметр актуален при сравнении различных политик планирования в рамках ГСПК.

Аналогично параметру Makespan, параметр Speedup является динамическим параметром, зависящим от состояния инфраструктуры для конкретного периода проведения вычислений.

Критерий оценки производительности по времени решения прикладной задачи является важным в рамках концепции высокопроизводительных вычислений (High Performance Computing – HPC). На практике, в рамках данной концепции специалист, поставивший прикладную задачу, желает получить результат вычислений в течении периода от нескольких часов до нескольких недель.

2.2 Способность системы выполнить очень большие объёмы вычислений

В рамках данного критерия объём проделанной работы играет более важную роль, чем быстрота получения результата прикладной задачи. Один из способов получения количественных метрик производительности проекта ГСПК состоит в подсчёте суммарного процессорного времени (или количества операций с плавающей точкой) всех выполненных заданий за некоторый достаточно продолжительный период. Эта характеристика может быть отнормирована на единицы времени. Например, общий объём вычислений можно разделить на количество секунд, дней или месяцев.

Данный критерий является важным в рамках концепции высокомасштабных вычислений (High-Throughput Computing – HTC). Согласно этой концепции, специалист, поставивший прикладную задачу, желает получить результат вычислений в течении периода от нескольких недель до нескольких лет.

Для сравнения концепций HPC и HTC можно провести аналогию с двумя транспортными задачами. Рассмотрим два портовых города (напр. Новороссийск и Буэнос-Айрес), расстояние между которыми 12 тыс. км. Первая задача состоит в перевозке одного пассажира между этими городами. Вторая задача состоит в перевозке 10 тысяч тонн груза из одного города в другой. Очевидно, что для транспортировки одного человека важным критерием будет время перевозки. И в соответствии с этим критерием, гражданская авиация будет более производительна, чем морские перевозки. Но для перевозки грузов большой массы наиболее важным критерием является принципиальная возможность такой транспортировки. В соответствии с данным критерием, использование грузовых судов является более производительным.

По каждому из двух критериев можно сравнивать производительность различных систем. Для отдельных вычислительных систем и прикладных задач данные два критерия могут как дополнять друг друга, так и противоречить друг другу.

Под эффективностью понимается способность системы оптимальным образом использовать имеющиеся ресурсы. В параллельной вычислительной системе эффективность определяется как средняя доля выполнения алгоритма, в течении которого процессоры выделены для решения задачи:

$$E_p = \frac{\sum T_i}{p \cdot M_s} = \frac{S_p}{p}.$$

Здесь T_i – время использования i-го процессора при решении задачи, p – количество процессоров. Для системы параллельных вычислений эффективность равна отношению реального ускорения к максимально возможному для данной системы в теории (т.е. к количеству процессоров).

Использование этой формулы для инфраструктуры BOINC требует ряда уточнений.

Первое уточнение связано с тем, какой параметр можно считать количеством процессоров. В рамках данной работы будем считать, что p – это суммарное количество процессоров компьютеров, которые посчитали минимум одно подзадание. Посчитанная таким образом эффективность основана на средней стабильности вычислительных узлов, не зависящей от политики управления ресурсами.

Второе уточнение связано со способом вычисления ускорения при оценке эффективности. Аналогично двум способам вычисления ускорения, можно составить два способа вычисления эффективности E_p и E_{cpu}.

Также возможен специфический для ГСПК способ оценки эффективности работы инфраструктуры, основанный на анализе количества запросов. Вычисления могут быть организованы таким образом, что в некоторые промежутки времени сервер может не содержать заданий. Если вычислительный узел запрашивает задания у сервера задания, а сервер таковых заданий не имеет, то узел будет некоторое время бездействовать. Чем больше суммарное время бездействия вычислительных узлов, тем ниже эффективность работы инфраструктуры. Запросы, в ответ на которые отправляется вычислительное задание будем называть обслуженными, а запросы, переводящие вычислительный узел в состояние бездействия – необслуженными. Дополнительная метрика оценки эффективности работы инфраструктуры равна отношению числа обслуженных запросов со стороны вычислительных узлов к их общему числу. При равномерном распределении запросов во времени данный параметр эффективности будет стремиться к отношению времени, когда очередь содержит какое-либо количество заданий к общему времени рассмотрения:

$$E_{req} = \frac{N_{handle}}{N} \approx \frac{T_{queue}}{T},$$

N_{handle} – количество обслуженных запросов со стороны вычислительных узлов; N – общее количество запросов; T_{queue} – время, в течении которого очередь содержала одно или более заданий; T – общее время проведения вычислений.

При рассмотрении влияния различных системных эффектов на производительность и эффективность необходимо учитывать по какому именно критерию оцениваются указанные величины. Принципиально важными является ожидания специалистов в прикладной области, использующих ресурсы инфраструктуры ГСПК.

3. Анализ причин потери производительности и эффективности при реализации эволюционного алгоритма на ГСПК

Рассмотрим основные причины снижения производительности и эффективности работы ЭА на инфраструктуре ГСПК и введём метрики, характеризующие степень влияния конкретных причин. Данные метрики могут использоваться при сравнении различных политик

управления ресурсами для выработки оптимальной стратегии. Также система вычисления метрик может быть интегрирована в инфраструктуру в качестве системы поддержки принятия решений для оператора вычислительной системы в режиме реального времени.

3.1 Ожидание всех результатов поколения (проблема “семеро одного не ждут”)

Логика эволюционного алгоритма предполагает несколько поколений вычислений. Последовательность проведения вычислений ЭА USPEX с использованием решателя VASP отображена на рис. 1.

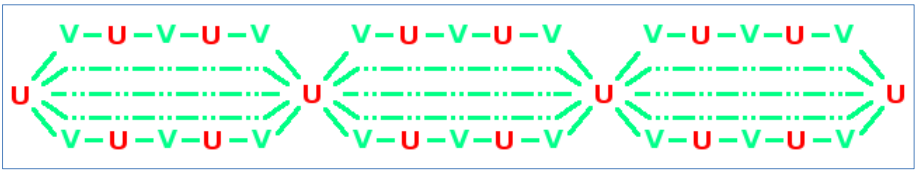


Рис. 1. Пример нитей трёх поколений вычислений. V – решатель VASP, выполняется на вычислительном узле. U – эволюционный алгоритм USPEX, выполняется на сервере
Fig. 1. An example of threads of three generations of computation. V - VASP solver, executed on the computational node. U - evolutionary USPEX algorithm, executed on the server

Рисунок показывает схему использования ресурсов в эволюционном алгоритме для 3-х поколений. Каждое поколение состоит из *N* независимых друг от друга нитей. Каждая нить представляет собой последовательность действий, выполняемых поочерёдно эволюционным алгоритмом на стороне сервера (*U – эволюционный алгоритм USPEX*) и решателем на стороне вычислительного узла (*V – VASP*). В пределах нити новые подзадания генерируются эволюционным алгоритмом на основе результатов предыдущих. Для типовых задач моделирования структуры вещества одна нить состоит из 3-х или 4-х участков работы VASP, генерируемых эволюционным алгоритмом USPEX. В пределах одной нити различные VASP-задания могут выполняться на различных вычислительных узлах. Число нитей в поколении является постоянным для решаемой научной задачи и может принимать значения от 1 до нескольких сотен. Число поколений также может достигать нескольких сотен. Новое поколение вычислений генерируется после того как все нити вычислений предыдущего поколения полностью посчитаны.

Рассмотрим типовой сценарий возникновения задержки одного из заданий поколения. На практике задержки возникают в 5-10% случаев. Допустим, в рамках решаемой задачи поколение состоит из восьми нитей вычислений, семь из которых были посчитаны инфраструктурой BOINC за первые сутки, а восьмая нить вычислений обрабатывалась 5 суток по причине нестабильности отдельного вычислительного узла. Таким образом, расчёт всего поколения займет пять суток. Последние 4 суток поколение будет ожидать оставшееся последнее задание.

Пример поколения, в котором сбой на отдельной нити вычислений задерживает генерацию нового поколения, показан на рис. 2:

Рис. 2. Пример нитей последовательности вычислений в поколении после возникновения технического сбоя
Fig. 2. An example of threads of a sequence of calculations in a generation after a technical failure occurs

Эффект задержки последнего задания способен в несколько раз увеличить время решения научной задачи. Но этот эффект ряде случает не будет снижать параметр эффективности E_{req}. Например, если в инфраструктуре BOINC одновременно решается несколько научных задач (возможно, поставленных разными специалистами), то во время ожидания поколения одной научной задачи инфраструктура будет успешно выполнять решение других задач. Преодоление эффекта снижения времени выполнения вычислительного задания возможно посредством применения различных научных методов к планированию вычислениями и предоставления потребителю ресурсов возможности настройки параметров вычислений в соответствии с его приоритетами.

Для количественной оценки влияния ожидания всех результатов поколения при решении конкретной USPEX-задачи необходимо ввести ряд величин.

a) Суммарное время ожидания для всех нитей всех поколений. Временем ожидания для отдельной нити поколения является время между окончанием вычислений данной нити и окончанием расчётов всех нитей поколения. Для последней посчитанной нити в поколении время ожидания равно нулю.

$$T_w = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij},$$

n – количество поколений, m – количество нитей в поколении, t_{ij} – время ожидания j-й нити в i-м поколении.

b) Коэффициент ожидания последней нити в поколении, равный отношению суммарного полного времени всех поколений к суммарному времени ожидания. Под полным временем поколения понимается произведение числа нитей в поколении на время самого поколения.

$$K_w = \frac{T_w}{\sum_i^n m t_i},$$

T_w – суммарное время ожидания всех нитей, n – количество поколений, m – количество нитей в поколении, t_i – время выполнения i-го поколения.

3.2 Влияние очерёдности для подзаданий

Данный источник потерь производительности аналогичен проблеме «семеро одного не ждут». Как уже отмечалось выше, в системе BOINC каждое подзадание отправляется в общую очередь. Предположим, что одно подзадание требует час процессорного времени для расчётов, а среднее время ожидания в очереди – 10 часов. Таким образом, USPEX получит результат задания через 11 часов после отправки. Аналогичным образом последующие подзадания вычислительной задачи будут отправлены на вычислительный узел после прохождения очереди.

При отсутствии репликации наилучшими количественными характеристиками влияния очереди будут суммарное и среднее время пребывания задания в очереди, а также отношение суммарного времени пребывания в очереди к суммарному времени всех заданий.

Совокупное влияние единой очереди и эффекта ожидания последнего результата способно привести к снижению уровня производительности в десятки и сотни раз. Рассмотрим пример. Пусть в поколении 10 нитей, каждая из которых предполагает 1 час расчётов на стороне вычислительного узла. 10 заданий отправляются на 10 вычислительных узлов. Через час времени возвращается 9 результатов вычислений. 10-й результат не был получен по причине выхода вычислительного узла из строя. Тогда по истечении времени дедлайна (предположим, 10-ти часов) будет сгенерировано дублирующее задание и поставлено в общую очередь. По прошествии ещё 10-ти часов пребывания в очереди дублирующее задание отправится на вычислительный узел, где будет успешно посчитано в течении часа.

Таким образом, время расчёта всего поколения составит 21 час, 20 из которых составит ожидание последнего задания. Из этих 20-ти часов 10 часов составит дедлайн, и 10 часов составит ожидание дублирующего задания в очереди. На практике проведения вычислений

при большом количестве нитей в поколении наблюдались эффекты многократной задержки дублирующих заданий, сильно снижающих производительность.

4. Заключение

Представленная работа является результатом анализа практики применения эволюционного алгоритма USPEX для решения задач оптимизации. Следующий этап исследований состоит в применении методов имитационного моделирования процесса проведения вычислений для выработки оптимального алгоритма управления ресурсами. Полученную на моделируемой инфраструктуре стратегию предполагается применять при проведении практических вычислений. Предложенные в этой работе метрики могут быть использованы для количественной оценки эффективности и производительности системы на дальнейших этапах исследований.

Список литературы / References

[1]. Berkeley Open Infrastructure for Network Computing. Available at: https://boinc.berkeley.edu/. Accessed 19.08.2020.

[2]. Манзюк М.О., Заикин О.С., Посыпкин М.А. CluboRUN: программный комплекс для использования свободных ресурсов вычислительных кластеров в boinc-расчетах. Информационные технологии и вычислительные системы, № 4, 2014 г., стр. 3-11. / Manzyuk M.O., Zaikin O.S., Posypkin M.A. CluBORun: tool for utilizing idle resources of computing clusters in BOINC computing. Informacionnye tekhnologii I vichslitel'nye sistemy (Journal of Information Technologies and Computing Systems, № 4, 2014, pp. 3-11 (in Russian).

[3]. Ivashko E., Golovin A., Partition algorithm for association rules mining in BOINC-based enterprise desktop grid. Lecture Notes in Computer Science, vol. 9251, 2015, pp. 268-272.

[4]. Posypkin M., Khrapov N. Branch and bound method on desktop grid systems. In Proc. of the 2017 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference, 2017, pp. 526-528.

[5]. Богачкова И.А., Заикин О.С., Кочемазов С.Е., Отпущенников И.В., Семёнов А.А. Применение алгоритмов решения проблемы булевой выполнимости к криптоанализу хэш-функций семейства MD. Прикладная дискретная математика. Приложение, № 8, 2015 г., стр. 139–142 / Bogachkova I.A., Zaikin O.S., Kochemazov S.E., Otpuschennikov I.V., Semenov A.A. Application of algorithms for solving the problem of Boolean satisfiability to cryptanalysis of hash functions of the MD family. Applied discrete mathematics. Application, № 8, 2015, pp. 139–142 (in Russian).

[6]. Vatutin E. Comparison of Decisions Quality of Heuristic Methods with Sequential Formation of the Decision in the Graph Shortest Path Problem. In Proc. of the Third International Conference BOINC-based High Performance Computing, 2017, pp. 67-76.

[7]. Khrapov N., Rozen V., Samtsevich A., Posypkin M., Sukhomlin V., Oganov A. Using virtualization to protect the proprietary material science applications in volunteer computing. Open Engineering, vol. 8, issue 1, 2018, pp. 57-60.

[8]. Храпов Н. П. Анализ причин потери производительности при адаптации эволюционного алгоритма к системам добровольных вычислений. Сборник научных трудов международного конгресса «Современные проблемы компьютерных и информационных наук», 2019 г., стр. 21-26. / Khrapov N.P. Analysis of the performance reasons for adapting the evolutionary algorithm to voluntary computing systems. In Proc of the International Congress on Modern Problems of Computer and Information Sciences], 2019, pp. 21-26 (in Russian).

Информация об авторе / Information about the author

Николай Павлович ХРАПОВ – и.о. научного сотрудника. Научные интересы: распределённые вычисления, виртуализация, грид-системы из персональных компьютеров.

Nikolay Pavlovich KHRAPOV – researcher. Research interests: distributed computing, virtualization, desktopgrid.

DOI: 10.15514/ISPRAS–2020–32(4)–10

Диагностика гипертрофий левых отделов сердца с помощью глубокой нейронной сети

[1,2] *П.К. Андреев, ORCID: 0000-0003-4907-9808 <andreev.pk@phystech.edu>*
[1,3] *В.В. Ананьев, ORCID: 0000-0002-5070-8117 <survial53@gmail.com>*
[1,3] *В. А. Макаров, ORCID: 0000-0001-6744-4130 <vladimir.makarov@novsu.ru>*
[1,2,4] *Е. А. Карпулевич, ORCID: 0000-0002-6771-2163 <karpulevich@ispras.ru>*
[1,5] *Д. Ю. Турдаков, ORCID: 0000-0001-8745-0984 <turdakov@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, 141701, Россия, г. Долгопрудный, Институтский пер., 9*
[3] *Новгородский государственный университет имени Ярослава Мудрого, 173003, Россия, г. Великий Новгород, Большая Санкт-Петербургская ул., 41*
[4] *Национальный исследовательский центр "Курчатовский институт", 123182, г. Москва, пл. Академика Курчатова, 1*
[5] *Московский государственный университет имени М.В. Ломоносова, 119991, г. Москва, ул. Ленинские Горы, 1*

Аннотация. В настоящей работе представлены результаты применения сверточной нейронной сети для диагностики гипертрофий левых отделов сердца посредством анализа электрокардиограмм (ЭКГ) в 12 стандартных отведениях. В ходе исследования был собран и обработан новый уникальный набор данных, содержащий 64 тысячи записей ЭКГ. На основе сопутствующих записям заключений были сформированы метки принадлежности к двум рассматриваемым классам: гипертрофия левого желудочка и гипертрофия левого предсердия. Набор сигналов и выделенные метки были использованы для обучения глубокой сверточной нейронной сети с остаточными блоками, получившаяся модель способна детектировать гипертрофию левого желудочка с качеством по F-мере свыше 0.82 и гипертрофию левого предсердия с качеством свыше 0.78. Кроме того, был осуществлен поиск оптимальной архитектуры нейросети, произведена экспериментальная оценка эффекта от включения в модель метаданных пациентов и предобработки сигнала, а также сделан сравнительный анализ трудности детектирования гипертрофий левых отделов по отношению к двум другим часто встречающимся нарушениям сердечной активности – мерцательной аритмии и блокады левой ножки пучка Гиса.

Ключевые слова: нейронные сети; ЭКГ; электрокардиография; машинное обучение; гипертрофия

Для цитирования: Андреев П.К., Ананьев В.В., Макаров В.А., Карпулевич Е.А., Турдаков Д. Ю. Диагностика гипертрофий левых отделов сердца с помощью глубокой нейронной сети. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 141–154. DOI: 10.15514/ISPRAS–2020–32(4)–10

Благодарности: Исследование поддержано грантом РФФИ 19-57-06004 МНТИ_а.

Diagnosis of left atrial and left ventricular hypertrophies using a deep neural network

[1,2] *P.K. Andreev, ORCID: 0000-0003-4907-9808 <andreev.pk@phystech.edu>*
[1,3] *V.V. Ananev, ORCID: 0000-0002-5070-8117 <survial53@gmail.com>*
[1,3] *V. A. Makarov, ORCID: 0000-0001-6744-4130 <vladimir.makarov@novsu.ru>,*
[1,2,4] *E. A. Karpulevich, ORCID: 0000-0002-6771-2163 <karpulevich@ispras.ru>*
[1,5] *D. Y. Turdakov, ORCID: 0000-0001-8745-0984 <turdakov@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, 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia*
[3] *Yaroslav-the-Wise Novgorod State University, 41, Bolshaya Saint-Petersburgskaya st., Veliky Novgorod, Novgorod region, 173003, Russia*
[4] *National Research Center «Kurchatov Institute», 1, Academician Kurchatov sq., Moscow, 123182, Russia*
[5] *Lomonosov Moscow State University, 1, Leninskie Gory st, Moscow, 119991, Russia*

Abstract. This paper presents the results of the application of a convolutional neural network to diagnose left atrial and left ventricular hypertrophies by analyzing 12-lead electrocardiograms (ECG). During the study, a new unique dataset containing 64 thousand ECG records was collected and processed. Labels for the two classes under consideration, left ventricular hypertrophy and left atrial hypertrophy, were generated from the accompanying medical reports. A set of signals and obtained labels were used to train a deep convolutional neural network with residual blocks; the resulting model is capable of detecting left ventricular hypertrophy with F-score more than 0.82 and left atrial hypertrophy with F1-score over 0.78. In addition, the search for optimal neural network architecture was carried out and the experimental evaluation of the effect of including patient metadata into the model and signal preprocessing was conducted. Besides, the paper provides a comparative analysis of the difficulty of detecting left ventricular and left atrial hypertrophies in relation to the other two frequently occurring heart activity disorders, namely atrial fibrillation and left bundle branch block.

Keywords: neural networks; ECG; electrocardiography; machine learning; hypertrophy

For citation: Andreev P.K., Ananev V.V., Makarov V.A., Karpulevich E.A., Turdakov D.Y. Diagnosis of left atrial and left ventricular hypertrophies using a deep neural network. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 141–154 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–10

Acknowledgements. The reported study was funded by RFBR according to the research project № 19-57-06004.

1. Введение

Электрокардиография как эффективный метод инструментальной диагностики в кардиологии непрерывно совершенствуется. На основе исследований создаются новые подходы к интерпретации электрокардиограмм. Для синтеза непротиворечивого диагностического заключения врачу зачастую приходится проводить многофакторный глубокий анализ с проверкой большого числа критериев, рассмотрением сочетания специфических признаков и внешних факторов. На качество диагностического заключения сильно влияет индивидуальная подготовка врача. Для существенного сокращения трудозатрат и повышения качества заключения, формулируемого врачом, постоянно совершенствуются и внедряются в практику автоматизированные инструменты.

В настоящей статье представлены результаты применения методов машинного обучения для диагностики гипертрофии левых отделов сердца. Гипертрофия желудочков и предсердий относятся к числу сложно диагностируемых заболеваний, которые часто встречаются в сочетании с другими патологиями сердца, например, блокадами. Известно не менее десятка

различных критериев диагностики гипертрофий, зависящих от конкретного набора ЭКГ-признаков и учитывающих такие факторы как пол, возраст, раса, вес пациента, площадь поверхности тела [1, 2].

Гипертрофия левого желудочка (aleft ventricular hypertrophy, LVH) – достаточно распространенное заболевание. Оно выявляется примерно у 20% обследуемых в возрасте 50 лет и этот процент растет с увеличением возраста пациентов. Поскольку существуют препараты лечения LVH, эффективность которых подтверждена клинической практикой, задача своевременного выявления этой патологии представляется весьма актуальной [1].

Как известно, патологии предсердий выявляется на основе анализа формы и длительности зубца p, для их обозначения используются термины: увеличение предсердия (atrial enlargement), патология предсердия (atrial abnormality), гипертрофия предсердия (atrial hypertrophy). В диагностических заключениях набора ЭКГ, на котором построено исследование, чаще используется термин «гипертрофия», его будем использовать в тексте статьи.

Гипертрофия левого предсердия (left atrial hypertrophy, LAH) является, как правило, важным первичным индикатором нарушения работы сердца. Изменение функционирования левого предсердия может стать причиной фибрилляции предсердий, сердечной недостаточности, инсульта, транзиторной ишемической атаки, острого инфаркта миокарда [3, 4]. Раннее выявление патологий левого предсердия и своевременная терапия позволяют предотвратить необратимые негативные последствия.

Электрокардиография не является наилучшим методом выявления гипертрофии. Гораздо лучшее качество диагностики дает эхокардиография. У пациентов с клинически значимой гипертрофией, которая обнаруживается по результатам эхокардиографии, ЭКГ может не содержать диагностических признаков заболевания. Однако этот факт не исключает выявления указанных патологий методом электрокардиографии. Во-первых, чувствительность и специфичность алгоритмов выявления гипертрофий на ЭКГ достаточно высоки и уступают ЭхоКГ на 20-35% [1]. Для людей старшей возрастной группы (> 40 лет) эти показатели находятся на вполне приемлемом уровне. Во-вторых, электрокардиография является более дешевым и наиболее часто применяемым методом скрининга, не требующим участия высококвалифицированного специалиста непосредственно для снятия ЭКГ. Использование портативных инструментов в сочетании с техниками телемедицины позволяет пациенту самостоятельно снять ЭКГ и предоставить ее врачу.

За последнее десятилетие были достигнуты значительные успехи в создании алгоритмов машинного обучения, основанных на глубоких нейронных сетях. Их главной отличительной чертой является способность определять сложные закономерности в данных, так, с помощью обучения глубоких нейронных сетей были достигнуты существенные успехи в компьютерном зрении [5], распознавании речи [6], обработке естественного языка [7], а также ряде медицинских приложений [8, 9], в том числе в автоматической интерпретации ЭКГ [10, 11]. В частности, было показано, что качество диагностики некоторых состояний при использовании данных алгоритмов может превышать уровни, соответствующие врачам среднего уровня квалификации, а также классическим коммерческим программам анализа ЭКГ [12]. В силу описанных выше причин исследование применимости глубоких нейронных сетей для диагностики гипертрофий левых отделов сердца приобретает особую актуальность.

Так, в рамках работы [10] для определения ритма ЭКГ была разработана сверточная нейронная сеть с остаточными блоками. Модель была обучена предсказывать тип ритма по отрезку сигнала длительностью 1.3 секунды с одного электрокардиографического отведения. Для обучения модели был собран и аннотирован новый набор данных, содержащий 91,232 тридцатисекундных ЭКГ сигналов, соответствующих 53,549 пациентов. Было продемонстрировано, что качество автоматического определения каждого из 12 рассматриваемых типов ритма (фибрилляция предсердий, атриовентрикулярная блокада I

степени, эктопический предсердный ритм, бигеминия, тригеминия, идиовентрикулярный ритм, узловой ритм, синус ритм, наджелудочковая тахикардия, желудочковая тахикардия, атривентрикулярная блокада II степени тип 1 и зашумленный ритм) сравнимо или превосходит усредненное качество диагностики нескольких практикующих кардиологов. Несмотря на многообещающие результаты, рассмотрение в работе лишь одного из двенадцати стандартных отведений и только ритмических типов патологий ограничивает применение данного исследования в клинической практике.

В работе [11], напротив, использовался набор данных, содержащий сигналы ЭКГ в двенадцати отведениях. Набор состоит из 2,470,424 записей, соответствующих 1,676,384 пациентов. Длина каждой записи составляет от 7 до 10 секунд. Данные использовались для обучения нейронной сети подобной [10], но с большим количеством входных каналов и меньшей глубиной. В работе рассматривались следующие пересекающиеся классы отклонений сердечной активности от нормы: атриовентрикулярная блокада I степени, фибрилляция предсердий, блокада левой ножки пучка Гиса, блокада правой ножки пучка Гиса, синусовая тахикардия и синусовая брадикардия. Так же, как и в [10], полученные в данной работе результаты позволяют с оптимизмом смотреть на возможность практического применения нейронных сетей для диагностики заболеваний сердца, так как качество анализа признаков ЭКГ по F-мере в данной работе оказалось выше, чем у ряда врачей специалистов.

Таким образом, несмотря на существенный прогресс в вопросе применимости нейронных сетей для интерпретации электрокардиограмм, использование данных алгоритмов для диагностики гипертрофий левых отделов сердца в настоящий момент не является полностью изученным.

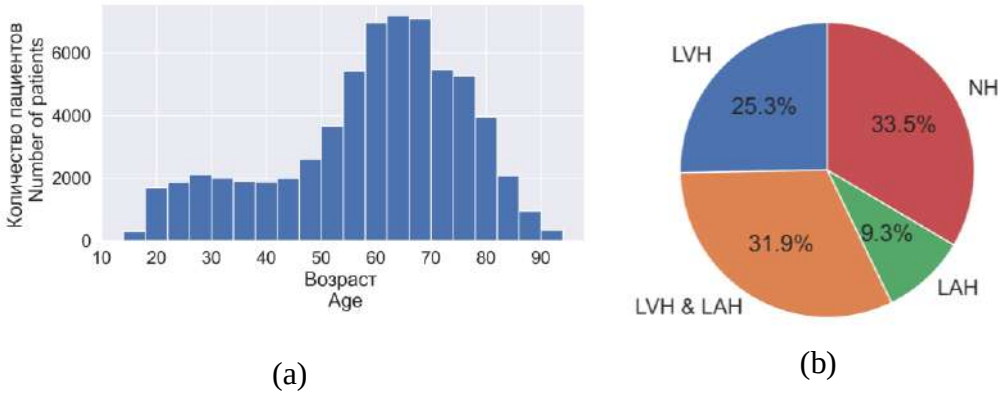
2. Данные

В рамках настоящей работы в результате взаимодействия с несколькими медицинскими центрами в Великом Новгороде был получен новый, уникальный набор данных, содержащий записи ЭКГ в 12 стандартных отведениях. В наборе содержатся 64 тысячи анонимизированных записей ЭКГ, соответствующие 36 тысячам пациентов в возрасте от 13 до 95 лет (распределение пациентов по возрасту показано на Рис. 1 (a)). Длительность каждой записи составляет 4 секунды.

Каждый экземпляр ЭКГ сопровождался описанием заключения, которое было сформулировано врачом клиники. Из текста заключений по ключевым словам (табл. 1) были выделены бинарные метки принадлежности к двум рассматриваемым классам (гипертрофия левого желудочка и гипертрофия левого предсердия), при этом данные классы могут пересекаться, что и проиллюстрировано на круговой диаграмме рис. 1 (b).

Табл. 1. Словосочетания, покрываемые регулярными выражениями, использованными для получения меток принадлежности к классам
Table 1. Phrases covered by regular expressions used to get class membership labels

Класс	Словосочетания
Гипертрофия левого желудочка	Гипертрофия левого желудочка, Гипертрофия левых отделов, Гипертрофия обоих желудочков
Гипертрофия левого предсердия	Гипертрофия левого предсердия, Гипертрофия левых отделов, Гипертрофия обоих предсердий



(a) (b)

Рис 1. (a): Гистограмма распределения пациентов по возрасту. (b): Круговая диаграмма распределения ЭКГ записей по следующим группам: LVH – гипертрофия левого желудочка (при этом отсутствует гипертрофия левого предсердия), LVH & LAH – гипертрофия левых отделов (гипертрофия левых желудочка и предсердия одновременно), LAH – гипертрофия левого предсердия (при этом отсутствует гипертрофия левого желудочка), NH – отсутствие гипертрофий левых отделов

Fig 1. a): Distribution of patients by age. (b): Distribution of ECG records into the following groups: LVH - left ventricular hypertrophy only (left atrial hypertrophy is absent), LVH & LAH - left ventricular and atrial hypertrophy at the same time, LAH - left atrial hypertrophy only (left ventricular hypertrophy is absent), NH - left ventricular hypertrophy and left atrial hypertrophy are absent

3. Модель

В качестве алгоритма классификации электрокардиограмм была использована сверточная нейронная сеть с остаточными блоками, схожая с использованной в работе [11], архитектура сети приведена на рис. 2.

Входные данные представляют собой ЭКГ сигнал с 8 отведений (*I, II, V1, V2, V3, V4, V5, V6*) в виде матрицы с размерностями 768 × 8, где 768 – число измерений (при частоте дискретизации 250 ГЦ соответствует длине записи в 3.072 секунды), 8 – число отведений (ввиду линейной зависимости остальных стандартных отведений *III, aVR, aVF, aVL* от *I* и *II*, их включение в модель не имеет смысла), а также метаданные пациента: возраст – натуральное число от 13 до 95, пол – число 0 (мужской) или 1 (женский).

ЭКГ-сигнал подается на вход нескольким (N) последовательным остаточным блокам [13], выделяющим релевантные для классификации признаки. Каждый остаточный блок представляет собой два сверточных модуля, состоящих из последовательных слоев батч нормализации [14], нелинейности (ReLU), дропаута (англ. dropout [15]) и свертки, начало и конец остаточного блока соединены остаточной связью (англ. skip connection), кроме того, для понижения пространственной размерности каждые k (= 3) блоков в остаточной связи применяется операция пулинга (соответственно, в одном из сверточных блоков шаг фильтра больше на единицу).

Выход сверточной части модели конкатенируется с нормализованными метаданными пациентов и подаются на вход двум последовательным полносвязным слоям, сопровождающимися сигмоидной функцией активации. В результате для каждого образца модель выдает 2 числа в промежутке от 0 до 1, которые могут интерпретироваться как вероятности принадлежности к 2 рассматриваемым классам. Таким образом, в настоящей работе использовалась одна модель для решения обеих задач бинарной классификации, альтернативный подход мог бы заключаться в использовании двух независимых моделей для каждой из задач в отдельности, однако гипертрофии левых отделов часто встречаются в

комбинации, кроме того, признаки релевантные для детекции данных состояний могут пересекаться.

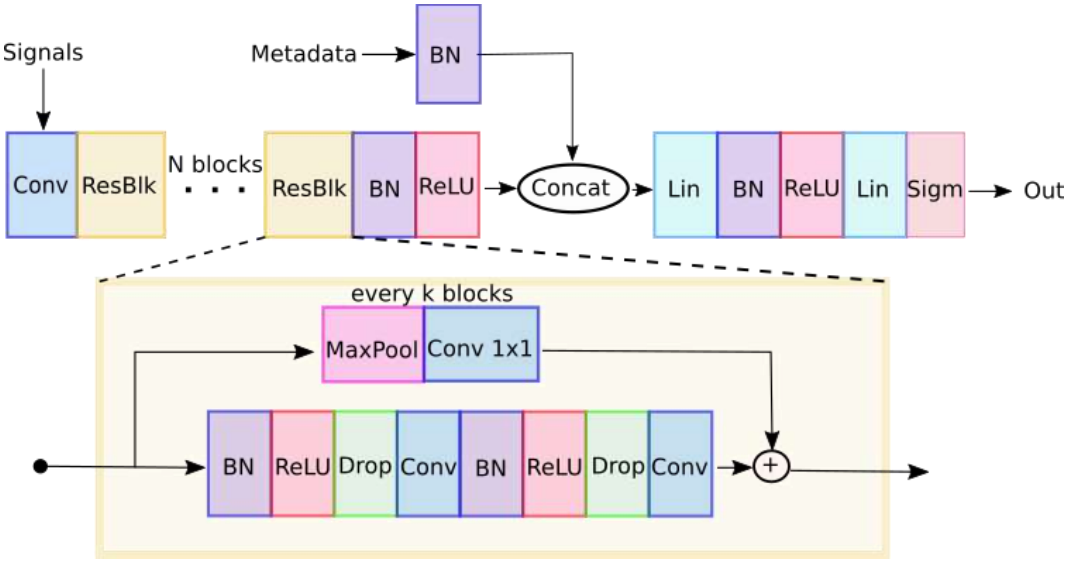


Рис 2. Архитектура использованной нейросети. Conv – операция свертки, ResBlk – остаточный блок, BN – батч-нормализация, ReLU – функция нелинейности, Drop – операция дропаута, MaxPool – операция пулинга функцией максимума, Lin – полносвязный линейный слой, Concat – конкатенация векторов, Sigm – сигмоидная функция активации

Fig 2. Neural network architecture. Conv - unidimensional convolution operation, ResBlk - residual block, BN - batch normalization, ReLU - nonlinearity function, Drop - dropout operation, MaxPool - pooling by the maximum function, Lin - fully connected linear layer, Concat - vector concatenation, Sigm - element-wise sigmoid function

4. Методология и результаты экспериментов

Рассматриваемая модель обучалась на описанном в разделе 2 наборе данных с помощью алгоритма обратного распространения ошибки [16], при этом на каждой итерации обучения («эпохе») из записей ЭКГ длиной 4 секунды вырезался случайный участок длительностью 3 секунды, что позволяет значительно снизить эффект переобучения. В качестве функции потерь использовалась средняя по классам бинарная кросс-энтропия, которая оптимизировалась на обучающей выборке по методу Adam [17] с параметрами $\beta_1 = 0.9$, $\beta_2 = 0.999$, размером батча 64, начальной скоростью обучения (learning rate), равной 0.001, и её понижением с мультипликативным фактором 0.1 при отсутствии значимых улучшений на валидационной выборке в течение нескольких эпох. Во всех представленных ниже экспериментах, если не оговорено иначе, вероятность дропаута была фиксирована 0.5, начальное число каналов – 64, размер сверточных ядер – 7, число блоков – 9.

4.1 Поиск оптимальной архитектуры

Для поиска оптимальной архитектуры модели был произведен перебор ряда гиперпараметров: числа остаточных блоков, начального числа каналов в нейросети (её «толщины») и размера сверточного ядра. Перебор осуществлялся с помощью кросс-валидации на 5 частей, при этом при подборе каждого из рассматриваемых гиперпараметров остальные оставались фиксированными заданным значениям: число блоков – 9, начальное число каналов – 64, размер сверточных ядер – 7. Кросс-валидация проводилась по следующей

методологии: исходный набор данных был разделен на 5 приблизительно равных частей, каждая из которых содержит по 12 тысяч ЭКГ, шаг кросс-валидации заключался в последовательном выборе каждого из пяти наборов, разделения его на две равные части (тестовую и валидационную), объединения остальных четырёх частей в тренировочную выборку, запуске процесса обучения на тренировочной выборке с ранним остановом по значению метрики на валидионной выборке и оценке качества модели на тестовой части набора. Результаты экспериментов приведены на рис. 3.

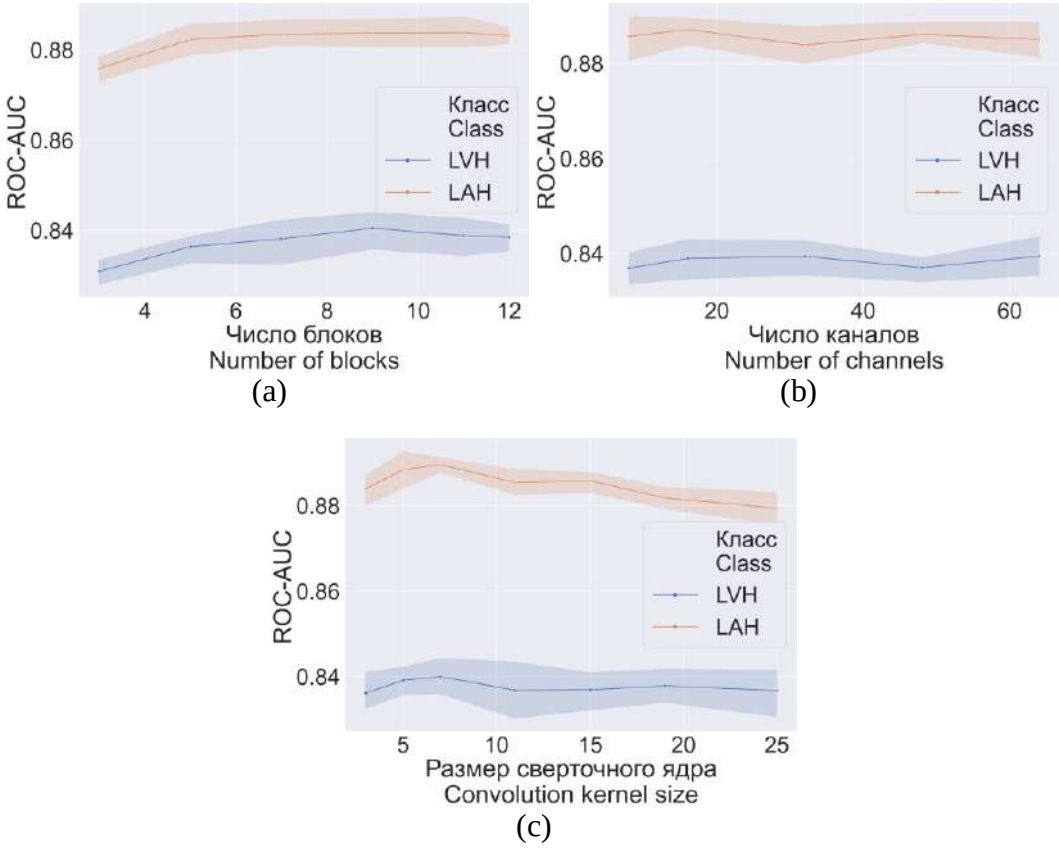


Рис 3. Зависимость площади под ROC-кривой от гиперпараметров (для каждой точки проведена кросс-валидация на 5 частей, доверительные интервалы оценены с помощью бутстрапа)

Fig 3. Dependence of the area under the ROC curve on hyperparameters (for each point, 5-fold cross-validation was carried out, the confidence intervals were estimated using bootstrap)

Как видно по рис. 3 (a), оптимальные результаты применения модели получены при использовании примерно 9 остаточных блоков и дальнейшее увеличение глубины приводит к ухудшению качества (на исследованном участке от 3 до 12 блоков), что находится в согласии с результатами работы [11] (с точностью до нескольких блоков, что связано с неполной идентичностью архитектур). Кроме того, по рис 3 (b) можно видеть, что качество классификации незначительно зависит от количества входных каналов на исследованном диапазоне, таким образом, «глубина» сети играет более значительную роль чем её «толщина». Зависимость качества модели от размера сверточного ядра приведена на рис 3 (c); видно что оптимальное значение достигается при размере ядра, равном 7, при этом зависимость довольно слабая.

4.2 Зависимость качества классификации от размера обучающей выборки

Оценка характера зависимости качества классификации от размера обучающей выборки проводилась по методологии схожей с использованной в пункте 4.1. На каждом шаге кросс-валидации алгоритм обучался (начиная со случайной инициализации) на разном количестве образцов в обучающей выборке, при этом размеры валидационной и тестовых выборок оставались фиксированными. Результаты приведены на рис. 4, на котором также для сравнения представлены результаты аналогичного эксперимента для двух часто рассматриваемых (например, в [11]) классов нарушений – фибрилляция предсердий и блокада левой ножки пучка Гиса.

Из сравнения рис. 4(a) и рис. 4(b) можно сделать вывод о том, что диагностика гипертрофий на основе ЭКГ является более сложной задачей, чем выявление фибрилляции предсердий и блокады левой ножки пучка Гиса. Действительно, качество диагностики данных состояний гораздо раньше выходит на высокое значение (аналогичные соотношения наблюдаются и для других метрик, например, средней точности и F-меры). Также из графиков можно видеть, что качество диагностики рассматриваемых состояний монотонно увеличивается при увеличении тренировочной выборки, таким образом, можно ожидать значительного прироста качества при дальнейшем пополнении обучающего набора.

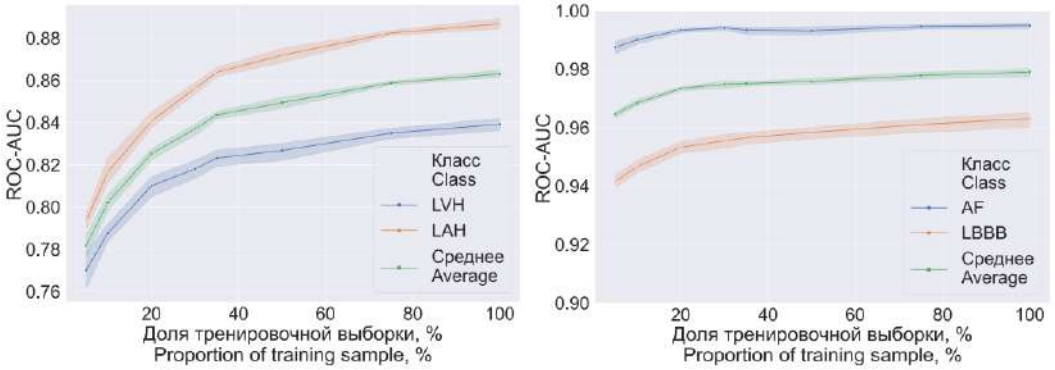


Рис 4. Зависимость площади под ROC-кривой от размера использованной обучающей выборки выраженного в долях от 48 тысяч образцов (максимально доступный размер обучающей выборки на кросс-валидации) (a): для классификации гипертрофий левых отделов (ГЛЖ и ГЛП), (b): для фибрилляции предсердий (англ. Atrial Fibrillation, AF) и блокады левой ножки пучка Гиса (англ. Left Bundle Branch Block, LBBB)

Fig 4. Dependence of the area under the ROC curve on the size of the training sample expressed in fractions of 48 thousand samples (the maximum available size of the training sample for cross-validation) (a): for left ventricular hypertrophy (LVH) and left atrial hypertrophy (LAH), (b): for atrial fibrillation (AF) and left bundle branch block (LBBB)

4.3 Влияние предобработки сигнала и включения в модель метаданных

Для повышения качества предсказаний модели в настоящей работе были использованы техники предобработки сигнала: коррекция изолинии с помощью локального взвешенного сглаживания (LOWESS [18]) и удаление высокочастотного шума с помощью дискретного вейвлет-преобразования [19] (с мягким порогом VisuSrink [20], материнская вейвлет Symlet – 8, уровень декомпозиции – 4). Кроме того, исследовалась зависимость качества предсказаний от факта включения в модель метаданных пациентов, пола и возраста. Влияние данных факторов проиллюстрировано на рис. 5 в виде диаграмм размаха.

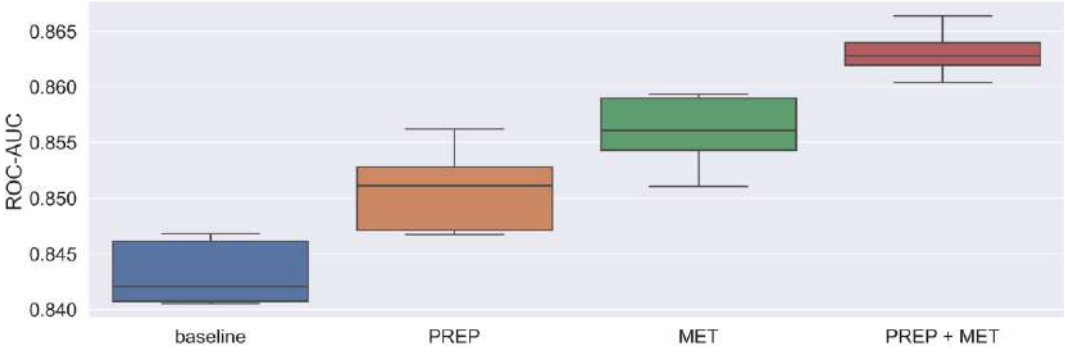


Рис 5. Средняя по классам площадь под ROC-кривой в зависимости от методологии (кросс-валидация на 5 частей). baseline – обучение на непредобработанных сигналах и без включения в модель информации о метаданных пациентов, PREP – обучение на предобработанных сигналах, MET – обучение с включением в модель информации о метаданных пациентов

Fig 5. Class-average area under the ROC curve, depending on the methodology (5-fold cross-validation). baseline - training on unprocessed signals and without including patient metadata information in the model, PREP - training on preprocessed signals, MET - training with patient metadata information included in the model

Предобработка сигнала и включение в модель метаданных пациентов статистически значимо увеличивают качество детекции рассматриваемых состояний как в совокупности, так и по отдельности (критерий Уилкоксона, $p < 0.05$). В целом, средняя площадь под ROC-кривой увеличилась почти на 2%, что согласно рис. 4(а) соответствует аналогичному приросту в качестве при увеличении тренировочной выборки в 3 раза. Также было выяснено, что основной вклад в прирост качества, при удалении шума из сигнала, привносит коррекция изолинии, в то время как удаление высокочастотного шума не влияет на результаты значимо. Аналогично при добавлении в модель метаданных прирост качества главным образом обусловлен информацией о возрасте пациентов.

4.4 Выводы

Итоговые метрики для модели с оптимальным (из рассмотренных) набором гиперпараметров (количество блоков, начальное число каналов и размер сверточного ядра), при использовании предобработки сигнала и включении в модель метаданных приведены в табл. 2. Оценка данных значений также проводилась на кросс-валидации на 5 частей, при этом для подсчета метрик, требующих предсказания модели в виде индикаторных меток (F-мера, Точность, Чувствительность, Специфичность) была произведена бинаризация предсказаний модели с порогом, подобранным на валидационной выборке по максимуму F-меры.

Табл. 2. Доверительные интервалы (по t-распределению Стьюдента) для метрик качества обнаружения рассматриваемых состояний с помощью разработанной нейронной сети (кросс-валидация на 5 частей)

Table 2. Confidence intervals (by Student's t-distribution) for metrics of detection of the abnormalities under consideration using the developed neural network (5-fold cross-validation).

Класс	ROC-AUC	Чувствительность Sensitivity	Специфичность Specificity	Точность Precision	F-мера F1-score
ГЛЖ	0.840±0.006	0.927±0.016	0.559±0.04	0.738±0.017	0.821±0.006
ГЛП	0.889±0.007	0.864±0.02	0.766±0.012	0.717±0.013	0.783±0.011

Таким образом, в процессе проведения настоящего исследования были решены следующие задачи:

1. Собран и обработан уникальный набор данных, содержащий записи ЭКГ в двенадцати стандартных отведениях.
2. Разработана сверточная нейронная сеть, способная детектировать гипертрофию левого желудочка с качеством по F-мере свыше 0.82 и гипертрофию левого предсердия с качеством свыше 0.78.
3. Осуществлен поиск оптимальной архитектуры и произведена экспериментальная оценка эффекта от включения в модель метаданных пациентов и предобработки сигнала.
4. Проведен сравнительный анализ трудности детектирования гипертрофий левых отделов по отношению к двум другим часто встречающимся нарушениям сердечной активности – мерцательной аритмии и блокады левой ножки пучка Гиса.

5. Заключение

Электрокардиографические критерии, используемые на практике для выявления гипертрофий левых отделов сердца, как правило, имеют высокую специфичность (около 90%) и умеренную чувствительность (30-60%), например, критерий напряжения Корнелла для определения гипертрофии левого желудочка имеет специфичность равную 85-95% при чувствительности 30-50% [1]. Как видно из рис. 6, показатели качества диагностики с помощью разработанного алгоритма находятся примерно на том же уровне (при специфичности 90%, чувствительности определения LAH и LVH равны 48% и 62% соответственно), при этом не требуют участия специалиста в процессе интерпретации. Более того, появляется естественная возможность для балансирования между чувствительностью и специфичностью посредством варьирования порогов на присвоенные моделью вероятности обнаружения рассматриваемых состояний (см. рис. 6).

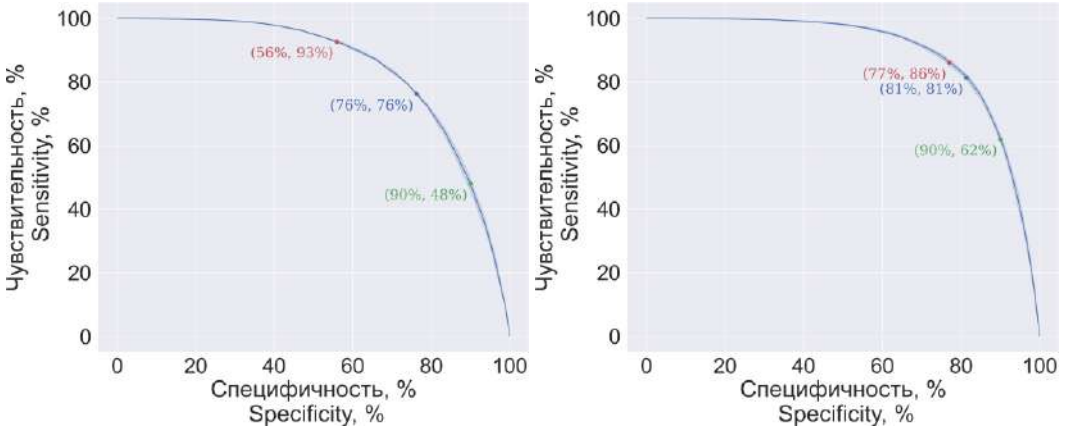


Рис 6. Кривые чувствительность-специфичность и их характерные точки: красная – точка, соответствующая максимуму F-меры на валидационной выборке (см. Табл. 2), синяя – точка равенства чувствительности и специфичности, зеленая – точка кривой со специфичностью равной 90%. (a): кривая для диагностики гипертрофии левого желудочка, (b): кривая для диагностики гипертрофии левого предсердия

Fig 5. Sensitivity-specificity curves and their characteristic points: red - the point corresponding to the maximum of F1-score on the validation sample (see Table 2), blue - the point of equality of sensitivity and specificity, green - the point of the curve with specificity equal to 90%. (a): curve for the diagnosis of left ventricular hypertrophy, (b): curve for the diagnosis of left atrial hypertrophy

Существенным ограничением настоящего исследования является способ получения разметки использованного набора данных. Полученная разметка по нашим оценкам содержит значительное количество ошибок как ввиду способа выделения меток принадлежности к классам (по ключевым словам), так и ввиду ошибок в самих заключениях. Количество

ошибок, возникающих из-за несовершенства метода анализа заключений, может быть снижено посредством использования более продвинутых алгоритмов выделения меток, так, в работе [11] был разработана методология выделения меток на основе ленивой ассоциативной классификации (lazy associative classifier) [21] по выделенным из заключений n-граммам. Ошибки же в самих заключениях не могут быть достоверно исправлены без привлечения нескольких экспертов для переразметки всего набора данных, однако возможно снизить влияние неправильно размеченных образцов на процесс обучения [22, 23], что является одним перспективных направлений дальнейших исследований.

Вероятно, наиболее значимым направлением дальнейшего изучения применения нейронных сетей в контексте интерпретации ЭКГ является обобщение полученных результатов на более широкий спектр аномалий сердечной активности, в особенности на состояния, угрожающие жизни пациента. Многие из таких состояний крайне редко встречаются в клинической практике, и в частности в наборе данных, использованном в настоящей работе, что делает затруднительным применение к их диагностике рассмотренных методик. В этой связи особенно актуальным может быть дальнейшее развитие техник аугментаций [24] и самообучения (self-supervised learning) [25] в ЭКГ домене.

Таким образом, в настоящей работе продемонстрирована эффективность применения глубоких нейронных сетей для диагностики гипертрофий левых отделов сердца. Данное исследование, совместно с рядом других работ [10, 11, 12], показывает преимущества парадигмы построения систем поддержки принятия решений в клинической практике на основе алгоритмов глубокого обучения. Дальнейшее развитие данной парадигмы потенциально может привести к значительному снижению затрат человеческих ресурсов на интерпретацию ЭКГ, и вместе с тем снизить количество врачебных ошибок.

Список литературы / References

[1] Gertsch Marc. The ECG: a two-step approach to diagnosis. Springer-Verlag Berlin Heidelberg, 2004, 615 p.

[2] Kozłowski D. Method in the Chaos – a step-by-step approach to ECG interpretation. European Journal of Translational and Clinical Medicine, vol. 1, no. 1, pp. 74-90.

[3] Donal Erwan et al. EACVI/EHRA Expert Consensus Document on the role of multi-modality imaging for the evaluation of patients with atrial fibrillation. European Heart Journal - Cardiovascular Imaging, vol. 17, no. 4, 2016, pp. 355–383.

[4] Tsang T.S., Abhayaratna W.P., Barnes M.E. et al. Prediction of cardiovascular outcomes with left atrial size: is volume superior to area or diameter? Journal of the American College of Cardiology, vol. 47, no. 5, 2006, pp. 1018-1023.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proc. of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026-1034.

[6] Amodei Dario et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In Proc. of the International conference on machine learning, 2016, pp. 173–182.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. In Proc. of the 31st Conference on Neural Information Processing Systems, 2017, pp. 5998-6008.

[8] Bejnordi Babak Ehteshami et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. Journal of the American Medical Association, vol. 318, no. 22, 2017, pp. 2199-2210.

[9] De Fauw Jeffrey, Ledsam Joseph R., Romera-Paredes Bernardino et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. Nature medicine, vol. 24, no. 9, 2018, pp. 1342–1350.

[10] Hannun Awni Y., Rajpurkar Pranav, Haghpanahi Masoumeh et al. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature medicine, vol. 25, no. 1, 2019, pp. 65-69.

[11] Ribeiro Antônio H., Ribeiro Manoel Horta, Paixão Gabriela M.M. et al. Automatic diagnosis of the 12-lead ECG using a deep neural network. Nature communications, vol. 11, no. 1, 2020, pp. 1–9

[12] Smith Stephen W., Walsh Brooks, Grauer Ken et al. A deep neural network learning algorithm outperforms a conventional algorithm for emergency department electrocardiogram interpretation. Journal of electrocardiology, vol. 52, 2019, pp. 88-95.
[13] He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian. Deep residual learning for image recognition. In Proc. of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. 0
[14] Ioffe Sergey, Szegedy Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.
[15] Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex et al. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, vol. 15, no. 1, 2014, pp. 1929-1958.
[16] Rumelhart David E., Hinton Geoffrey E., and Williams Ronald J. Learning representations by back-propagating errors. Nature, vol. 323, 1986, pp. 533-536.
[17] Kingma Diederik P., Ba Jimmy. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
[18] Cleveland William S. Robust locally weighted regression and smoothing scatterplots. Journal of the American Statistical Association, vol. 74, no. 368, 1979, pp. 829-836.
[19] Kania Michał, Fereniec Małgorzta, and Maniewski Roman. Wavelet denoising for multi-lead high resolution ECG signals. Measurement Science Review, vol. 7, no. 4, 2007, pp. 30-33.
[20] Donoho David L., and Johnstone Iain M. Threshold selection for wavelet shrinkage of noisy data. In Proc. of the 16th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 1994, pp. A24–A25.
[21] Veloso Adriano, Wagner Meira, and Zaki Mohammed J. Lazy associative classification. In Proc. of the Sixth International Conference on Data Mining (ICDM'06), 2006, pp. 645–654.
[22] Nguyen Duc Tam et al. Self: Learning to filter noisy labels with self-ensembling. arXiv preprint arXiv:1910.01842, 2019.
[23] Huang Lang, Zhang Chao, and Zhang Hongyang. Self-Adaptive Training: beyond Empirical Risk Minimization. arXiv preprint arXiv:2002.10319, 2020.
[24] Guryanova Valeriia. Online Augmentation for Quality Improvement of Neural Networks for Classification of Single-Channel Electrocardiograms. In Proc. of the International Conference on Analysis of Images, Social Networks and Texts, 2019, pp. 37–49.
[25] Kiyasseh Dani, Zhu Tingting, and Clifton David A. CLOCS: Contrastive Learning of Cardiac Signals. arXiv preprint arXiv:2005.13249, 2020.

Информация об авторах / Information about authors

Павел Константинович АНДРЕЕВ – выпускник Московского физико-технического института, сотрудник отдела «Информационные системы». Сфера его научных интересов включает биомедицинский анализ данных, глубокое обучение, робастные методы машинного обучения, а также обучение с частичным привлечением учителя.

Pavel Konstantinovich ANDREEV is a graduate of the Moscow Institute of Physics and Technology, an employee of the Information Systems Department. His research interests include biomedical data science, deep learning, robust machine learning, and semi-supervised learning.

Владислав Валерьевич АНАНЬЕВ является магистрантом и ассистентом кафедры информационных технологий и систем НовГУ, сотрудник ИСП РАН. Сфера научных интересов: анализ и разметка данных из различных сфер деятельности, глубокое обучение, компьютерное зрение и обработка изображений.

Vladislav Valerievich ANANEV is a graduate of the magistracy and assistant of the Department of Information Technologies and Systems, Novgorod State University, an employee of ISP RAS. Area of research interests: data labeling and analysis for various fields of activity, deep learning, computer vision and image processing.

Владимир Алексеевич МАКАРОВ – кандидат технических наук, старший научный сотрудник. Сфера научных интересов: анализ бинарного кода, искусственный интеллект для персонифицированной медицины.

Vladimir Alexeevich MAKAROV – PhD, Senior scientist. Research interests: binary code analysis, artificial intelligence for personalized medicine.

Евгений Андреевич КАРПУЛЕВИЧ является специалистом отдела «Информационные системы». Сфера научных интересов: применение алгоритмов анализа данных к биомедицинскому домену, разработку систем распределенного хранения и анализа данных.

Evgeny Andreevich KARPULEVICH is a specialist of the Information Systems Department. Research interests: application of data analysis algorithms to the biomedical domain, development of systems for distributed data storage and analysis.

Денис Юрьевич ТУРДАКОВ – к.ф.-м.н., заведующий отделом «Информационные системы» ИСП РАН, доцент МГУ. Сфера научных интересов: машинное обучение, интеллектуальный анализ данных, извлечение информации, обработка естественного языка, сложные сети, анализ социальных сетей, большие данные.

Denis Yurievich TURDAKOV – Ph.D. head of the Information Systems Department at ISP RAS, associated professor at MSU. Research interests: machine learning, data mining, information extraction, natural language processing, complex networks, social network analysis, big data.

DOI: 10.15514/ISPRAS–2020–32(4)–11

Использование аппарата свёрточных нейронных сетей для стегоанализа цифровых изображений

А.А. Полунин, ORCID: 0000-0002-5870-5439 <polunin2002@mail.ru>
Э.А. Яндашевская, ORCID: 0000-0003-1050-9137 <elenayanda@yandex.ru>
Академия Федеральной службы охраны Российской Федерации,
302015, Россия, г. Орел, ул. Приборостроительная, д. 35

Аннотация. В статье дается обоснование актуальности задачи стегоанализа, как определения факта наличия скрытого канала в инфокоммуникационных системах, узлы которых обмениваются цифровыми изображениями. Рассматриваются вопросы применения аппарата свёрточных нейронных сетей для решения этой задачи. Предполагается, что вероятность правильной классификации изображений с помощью хорошо обученной свёрточной нейронной сети будет сопоставима с показателями статистических алгоритмов или RM-модели или даже окажется лучше них. Дается представление о принципах построения и возможностях свёрточных нейронных сетей в рамках их применимости к решению задачи стегоанализа. Для повышения оперативности и результативности процесса распознавания стегоконтейнеров предложен вариант модели классификации изображений для свёрточной нейронной сети, в которой используется комбинация нескольких свёрточных и полносвязных слоев. Разработана программная реализация варианта этой модели с возможностью обучения нейронной сети и оценивания качества классификации. Проведен анализ существующих программных продуктов, предназначенных для задачи определения факта использования стеганографии в цифровых изображениях. Обосновано преимущество классификаторов на основе нейронных сетей по сравнению со статистическими классификаторами. С использованием разработанной программной реализации проведено экспериментальное исследование модели классификации на наборах цифровых изображений, содержащихся в открытых источниках. В статье приведены результаты обучения нейронной сети, а также анализ сильных и слабых сторон выбранной модели.

Ключевые слова: стеганография; стегоанализ; цифровые изображения; нейронная сеть; свёрточный слой; полносвязный слой; машинное обучение

Для цитирования: Полунин А.А., Яндашевская Э.А. Использование аппарата свёрточных нейронных сетей для стегоанализа цифровых изображений. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 155–164. DOI: 10.15514/ISPRAS–2020–32(4)–11

Using of convolutional neural networks for steganalysis of digital images

A.A. Polunin, ORCID: 0000-0002-5870-5439 <polunin2002@mail.ru>
E.A. Yandashevskaya, ORCID: 0000-0003-1050-9137 <elenayanda@yandex.ru>
Russian Federation Security Guard Service Federal Academy,
25, Priborostroitelnaya st., Oryol, 302015, Russia

Abstract. The article substantiates the relevance of steganalysis, as a determination of the presence of a hidden channel in telecommunication systems, whose nodes exchange digital images. The article deals with the application of convolutional neural networks to solve this problem. It is assumed that the probability of correct

image classification using a well-trained convolutional neural network will be comparable or even better than characteristics of statistical algorithms or the RM model. We introduce principles of construction and capabilities of convolutional neural networks in the framework of their applicability to solving the problem of steganalysis. To improve the efficiency and effectiveness of the stegocontainer recognition process, a version of the image classification model for a convolutional neural network is proposed. It is based on combination of several convolutional and fully connected layers. We have developed software for this model version with the ability to train a neural network and evaluate the quality of classification. The analysis of existing software products designed for the task of determining the fact of using steganography in digital images is carried out. The advantage of classifiers based on neural networks in comparison with statistical ones is proved. Using the developed software, an experimental study of classification model on sets of digital images contained in open sources has been carried out. The article presents the results of neural network training, as well as an analysis of the strengths and weaknesses of the selected model.

Keywords: steganography; steganalysis; digital images; neural network; convolutional layer; fully connected layer; machine learning

For citation: Polunin A.A., Yandashevskaya E.A. Using of convolutional neural networks for steganalysis of digital images. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 155–164 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–11

1. Введение

Одним из актуальных направлений в предметных областях безопасности и мониторинга инфокоммуникационных систем являются вопросы формирования и эксплуатации скрытых каналов на основе стеганографических методов преобразования информации. Такие каналы используют функциональные возможности этих систем с целью порождения несанкционированных информационных потоков или потоков удаленного управления сервисами. Одним из наиболее распространенных видов стеганографических контейнеров (далее – стегоконтейнеров), применяемых для организации подобных скрытых каналов, являются цифровые изображения. В связи с этим являются актуальными задачи стегоанализа, формулируемые, как определение факта наличия скрытого канала в инфокоммуникационных системах, узлы которых обмениваются цифровыми изображениями.

В настоящее время большинство средств стегоанализа, предназначенных для обнаружения факта стеговложения, базируется на статистических методах [1], которые основываются на формировании модели изображения, позволяющей определить вектор признаков наличия стеговложения. К таким моделям относится, например, RM-модель (Rich Model), применяемая для обучения статистических классификаторов [2].

Их существенными недостатками являются: требовательность аналитических моделей, на которых основываются методы стегоанализа, к ряду параметров анализируемых контейнеров, что влияет на чувствительность моделей, и необходимость реализации полученных решений численными методами, что приводит к снижению оперативности и результативности процесса стегоанализа. В последнее время ряд исследований в области стегоанализа, в частности, цифровых изображений посвящен применению методов машинного обучения, основанных на применении искусственных нейронных сетей [1, 3, 4]. С использованием этих методов решается задача бинарной классификации цифровых изображений – разделения их множества на подмножества, содержащие и не содержащие стеговложения.

В рамках статьи рассматривается модель классификации, основанная на искусственной нейронной сети, которая содержит комбинацию свёрточных и полносвязных слоев, с целью определения значений показателя точности процесса классификации.

2. Особенности применения аппарата свёрточных нейронных сетей для решения задачи стегоанализа цифровых изображений

Аппарат свёрточных нейронных сетей (СНС) находит широкое применение при решении задачи обнаружения пространственных зависимостей в цифровых изображениях, а особенности формирования СНС позволяют уменьшить количество параметров и улучшить качество определения признаков [5].

В общем случае СНС состоят из следующих базовых блоков:

- свёрточные слои;
- слои подвыборки (пулинга);
- полносвязные слои.

Как правило, первый свёрточный слой отвечает за распознавание низкоуровневых признаков, а последующие слои объединяют их, переходя к более высокоуровневым признакам. Исследование возможностей СНС показывает, что на достаточно большом количестве цифровых изображений процесс обучения СНС позволяет оптимально настроить значения весов свёрточных слоев, необходимых для преобразования цифровых изображения, в пригодный для вычислительной системы вектор признаков, обеспечивающий повышение результативности процесса распознавания при минимизации его вычислительной сложности. Кроме того, особенностью СНС является возможность обнаружения каких-либо характеристик не в целом цифровом изображении, а во многих его частях, что достигается сегментацией изображения во время прохождения ядра СНС.

Функцией свёрточного слоя СНС является двумерная свертка – операция, используемая для уменьшения размера матрицы. Ядро K – матрица, которая состоит из коэффициентов, называемых весами, по сути является фильтром. При перемещении ядра по двумерному изображению I – другой матрице, выполняется умножение весов на значения пикселей, над которыми находится ядро, с их последующим суммированием, результатом которого является значение нового элемента (пикселя) следующего слоя (рис. 1). Таким образом, на выходе свёрточного слоя получается новая матрица (изображение) меньшего размера [6].

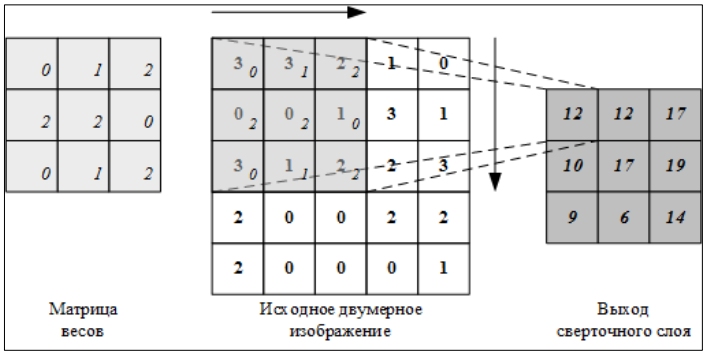


Рис. 1. Процесс двумерной свертки черно-белого цифрового изображения
Fig. 1. Two-dimensional convolution of a black-and-white digital image

Как и свёрточный, слой подвыборки (пулинговый) необходим для уменьшения размера изображения с целью минимизации количества требуемых вычислительных операций. В настоящее время используется несколько типов пулинга: максимальный (Max Pooling), средний (Average Pooling) и пулинг суммы (Sum Pooling). Первый тип применяется для вычисления максимального значения из части изображения, покрываемой ядром, второй – для вычисления среднего среди всех значений анализируемой области, третий – для определения их суммы (рис. 2).

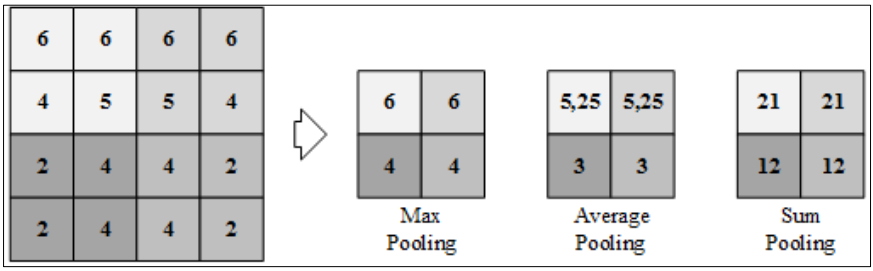


Рис. 2. Процесс обработки изображения слоем подвыборки
Fig. 2. Image processing with a pooling layer

Основной задачей полносвязного слоя (рис. 3) является моделирование нелинейной функции, используемой непосредственно для классификации, в то время как предшествующие слои являются средствами предобработки цифрового изображения. Применение нескольких слоев обученной СНС позволяет находить закономерности во входных данных и, анализируя их, формировать результат (одно или несколько решений), являющийся решением задачи. После обучения СНС на её выходе (в случае классификации) можно будет установить вероятность принадлежности изображения к тому или иному классу, что актуально для решения задачи распознавания стегоконтейнеров [8].

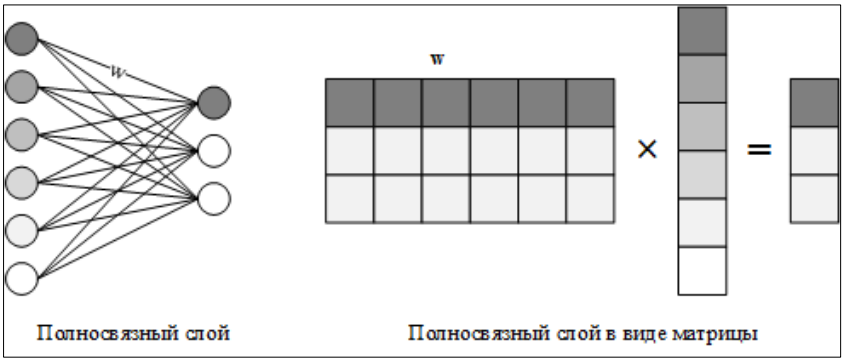


Рис. 3. Представление полносвязного слоя нейронной сети
Fig. 3. Neural network fully connected layer representation

3. Разработка модели свёрточной нейронной сети

Для синтеза свёрточного слоя СНС необходимо определить следующие параметры [9]:

- f (filters count) – количество фильтров в слое.
- K (kernel size) – размер (высота и ширина) ядра (обычно является нечётным числом, часто используются фильтры размером 3 или 5);
- s (stride) – шаг свёртки (количество пикселей, на которое перемещается матрица фильтра по входному изображению);
- p (padding) – дополнения нулями (количество пикселей, которые добавляются с каждого края изображения).

Таким образом, входными параметрами свёрточного слоя являются:

- собственно входное изображение в виде тензора $I_{in}(W_{in}, H_{in}, D_{in})$, где W_{in}, H_{in} – ширина и высота изображения соответственно, D_{in} – количество каналов;

- гиперпараметры: f, K, s, p.

Выходными данными слоя является тензор $I_{out}(W_{out}, H_{out}, D_{out})$, где:

$$W_{out} = \frac{W_{in} - K + 2p}{s} + 1, \quad (1)$$

$$H_{out} = \frac{H_{in} - K + 2p}{s} + 1, \quad (2)$$

$$D_{out} = f. \quad (3)$$

Слою подвыборки требуется всего один гиперпараметр – шаг пулинга, т.е. число раз, в которое нужно сократить пространственные размерности. Обычно используется слой пулинга с уменьшением размера входного тензора в два раза. Единственным гиперпараметром для полносвязного слоя является количество выходных значений.

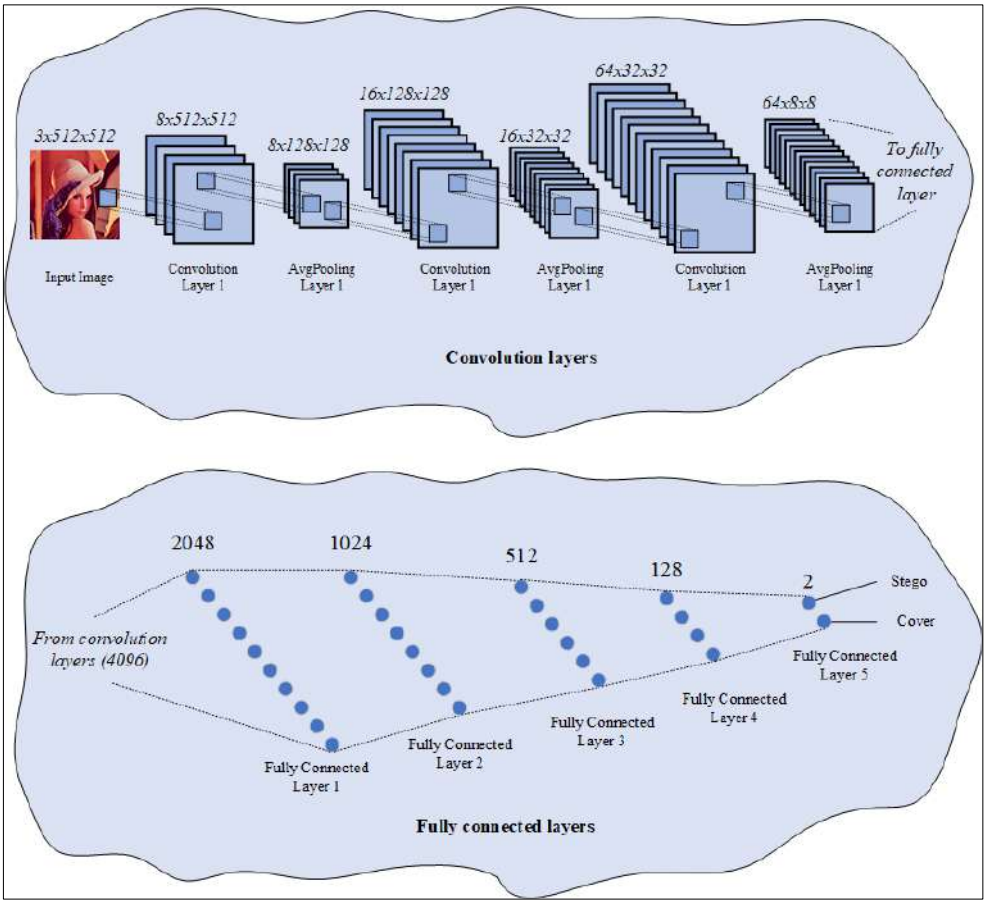


Рис. 4. Модель свёрточной нейронной сети
Fig. 4. Convolutional neural network model

Для решения стоящей в работе задачи стегоанализа цифровых изображений, была разработана модель СНС (рис. 4) со следующими слоями и их параметрами, рассчитанными в соответствии с выражениями (1–3):

- сверточный слой №1 (in_channels=3, out_channels=8, kernel_size=5, padding=2, stride=1);
- слой подвыборки №1(kernel_size=5, padding=2, stride=4);
- сверточный слой №2 (in_channels=8, out_channels=16, kernel_size=5, padding=2, stride=1);

- слой подвыборки №2 (kernel_size=5, padding=2, stride=4);
- сверточный слой №3 (in_channels=16, out_channels=64, kernel_size=5, padding=2, stride=1);
- слой подвыборки №3 (kernel_size=5, padding=2, stride=4);
- полносвязный слой №1 (in_features=4096, out_features=2048);
- полносвязный слой №2 (in_features=2048, out_features=1024);
- полносвязный слой №3 (in_features=1024, out_features=512);
- полносвязный слой №4 (in_features=512, out_features=128);
- полносвязный слой №4 (in_features=128, out_features=2).

После полной обработки входного цифрового изображения на основании выходных данных последнего слоя модели вычисляется вероятность принадлежности объекта к тому или иному классу.

4. Программная реализация модели свёрточной нейронной сети и ее экспериментальные исследования

Разработанная модель СНС была реализована в виде специального программного обеспечения (СПО) обнаружения стеганографических вложений в цифровых изображениях. Для решения задачи бинарной классификации были введены два класса цифровых изображений: «stego» и «cover». К первому относятся цифровые изображения со вложенной информацией – стегоконтейнеры, ко второму классу относятся цифровые изображения без вложений – пустые контейнеры. Тестирование СПО производилось на ресурсе Google Colaboratory. Данный сервис позволяет бесплатно использовать вычислительные мощности на удаленных серверах и быстро импортировать различные наборы данных с других ресурсов. СПО написано на языке Python, при этом для построения СНС использовалась библиотека с открытым исходным кодом PyTorch.

5. Результаты исследования

Наборы цифровых изображений для обучения СНС, реализованной в СПО, были взяты с ресурса kaggle.com. В качестве тренировочного набора данных были взяты изображения из digital-steganography (с применением стеганографических алгоритмов LSB, FFT, DCT – 15,7 тысячи изображений) и image-dataset (обычные изображения – 21,3 тысячи изображений). На входе в нейронную сеть каждое изображение масштабировалось к размеру 512x512. Обучение производилось по наборам из 64 изображений, после каждого набора высчитывалась функция потерь (loss) и точность (accuracy), их динамика в процессе обучения представлена на рис. 5.

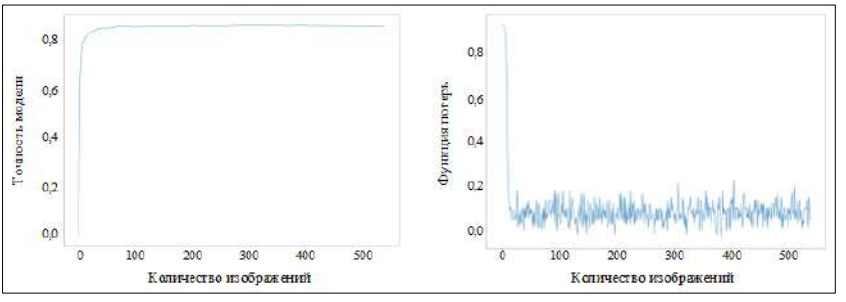


Рис. 5. Точность и функция потерь модели свёрточной нейронной сети в процессе обучения
Fig. 5. Convolutional neural network model training (accuracy and loss function)

Тестирование СПО производилось на наборе данных steghide-images, содержащем по 1,4 тысячи обычных изображений и содержащих стеганографические вложения. Точность и

функция потерь также определялись для каждого набора из 64 изображений. Их динамика приведена на рис. 6.

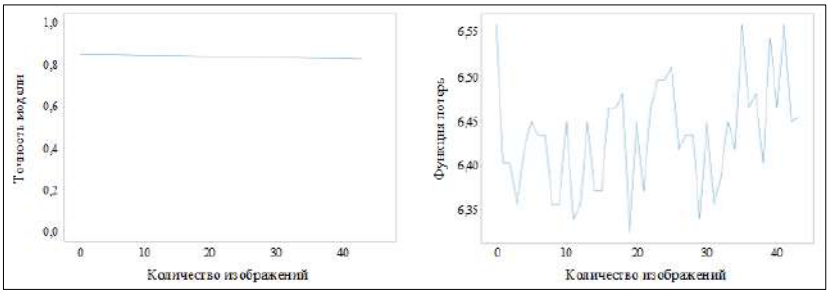


Рис. 6. Точность и функция потерь модели свёрточной нейронной сети в процессе тестирования
Fig. 5. Convolutional neural network model testing (accuracy and loss function)

Анализ полученных зависимостей (рис. 5-6) демонстрирует возможность правильной классификации в 85% случаев.

В [10] приведены результаты тестирования статистических классификаторов на изображениях, сформированных с использованием различных стеганографических алгоритмов (табл. 1). Из таблицы видно, что точность разработанной модели в целом превосходит средние показатели точности статистических классификаторов. Дополнительно следует отметить, что существенным недостатком статистических классификаторов, отсутствующим в методах на основе нейронных сетей, является их узкая специализация на строго определенные методы формирования стегоконтейнеров. Так в [3] представлены результаты анализа использования свёрточных нейронных сетей для решения задачи стегоанализа для алгоритмов встраивания WOW и S-UNIWARD при отношении количества вложенной информации на пиксель 0,2 bpp и 0,4 bpp (табл. 2).

Табл. 1. Сравнение вероятности ошибок при тестировании статистических классификаторов на различных стеганографических алгоритмах (вероятность ошибки)
Table 1. Error probability comparison during statistical classifiers testing with various steganographic algorithms

Алго-ритм	bpp	CHEN	CC-CHEN	LIU	CC-PEV	CDF	CC-300	CF	JRM	CC-JRM	J+SRM
nsF5	0,05	0,4153	0,3816	0,3377	0,369	0,3594	0,3722	0,3377	0,3407	0,3298	0,3146
	0,1	0,3097	0,247	0,1732	0,2239	0,202	0,2207	0,1737	0,1782	0,1616	0,1375
	0,15	0,2094	0,1393	0,0706	0,1171	0,0906	0,1127	0,072	0,0793	0,0663	0,0468
	0,2	0,1345	0,0708	0,0273	0,0549	0,036	0,0486	0,0273	0,0338	0,0255	0,015
MBS	0,01	0,407	0,3962	0,3826	0,3876	0,3786	0,4038	0,371	0,3478	0,3414	0,326
	0,02	0,3178	0,2962	0,278	0,2827	0,2684	0,312	0,256	0,2156	0,2122	0,1832
	0,03	0,2395	0,21	0,1925	0,1965	0,1795	0,2241	0,1684	0,1266	0,1195	0,0983
	0,04	0,177	0,1437	0,1288	0,1298	0,1135	0,1594	0,1087	0,0751	0,067	0,0494
	0,05	0,1243	0,0946	0,0812	0,0833	0,0704	0,1176	0,0684	0,0427	0,0373	0,0282
YASS	0,077	0,2009	0,1825	0,2324	0,2279	0,1268	0,093	0,0532	0,0324	0,0303	0,0173
	0,114	0,1989	0,1585	0,2118	0,1573	0,0718	0,0701	0,0437	0,0349	0,0227	0,0111
	0,138	0,252	0,1911	0,1886	0,1827	0,0742	0,05	0,0271	0,0287	0,0178	0,0104
	0,159	0,2334	0,1476	0,1793	0,1341	0,0507	0,037	0,0164	0,021	0,0103	0,0054
	0,187	0,1277	0,0876	0,1301	0,0723	0,0224	0,035	0,0146	0,0165	0,0081	0,0045
MME	0,05	0,4678	0,4546	0,4479	0,4492	0,434	0,4427	0,4443	0,4424	0,4307	0,4194

Алго-ритм	bpp	CHEN	CC-CHEN	LIU	CC-PEV	CDF	CC-300	CF	JRM	CC-JRM	J+ SRM
	0,1	0,3001	0,2611	0,2574	0,2613	0,2501	0,3026	0,2466	0,2286	0,2091	0,1891
	0,15	0,2165	0,1735	0,1677	0,1721	0,1586	0,2299	0,1608	0,1404	0,1221	0,1027
	0,2	0,0217	0,0104	0,0127	0,0127	0,0124	0,0726	0,0153	0,0112	0,008	0,0059
BCH	0,1	0,4599	0,4496	0,4448	0,4426	0,439	0,4497	0,429	0,4305	0,4229	0,406
	0,2	0,3594	0,3124	0,3087	0,2974	0,2752	0,2958	0,2629	0,2707	0,2369	0,1946
	0,3	0,1383	0,0889	0,0862	0,0779	0,0697	0,0912	0,0663	0,0715	0,0536	0,039
BCHopt	0,1	0,4726	0,4683	0,4558	0,4618	0,4595	0,4684	0,455	0,4515	0,448	0,4306
	0,2	0,4032	0,3712	0,3583	0,3548	0,3368	0,3517	0,3265	0,3253	0,303	0,2582
	0,3	0,24	0,1711	0,1719	0,1605	0,1356	0,1681	0,1289	0,1389	0,1102	0,083
Средняя точность		**73,22%**	**77,05%**	**77,81%**	**77,88%**	**80,77%**	**78,63%**	**82,19%**	**82,98%**	**84,19%**	**85,93 %**

Табл. 2. Сравнение вероятности ошибок стеганоанализа Yedroudj-Net, Xu-Net, Ye-Net и SRM+EC для алгоритмов встраивания WOW и S-UNIWARD при 0,2 bpp и 0,4 bpp
Table 2. Steganalysis error probability comparison of Yedroudj-Net, Xu-Net, Ye-Net, and SRM+EC for embedding algorithms WOW and S-UNIWARD at 0.2 bpp and 0.4 bpp

Модель	BOSS 256x256			
	WOW		S-UNIWARD	
	0.2bpp	0.4bpp	0.2bpp	0.4bpp
SRM+EC	36.5 %	25.5 %	36.6%	24.7 %
Yedroudj	27.8%	14.1%	36.7%	22.8%
Xu-net	32.4 %	20.7 %	39.1 %	27.2 %
Ye-Net	33.1 %	23.2 %	40.0 %	31.2 %

Дополнительное использование RM-модели для обучения статистических классификаторов [2] позволяет увеличить точность классификации. Однако в среднем, при учете отношения количества вложенной информации на пиксель (bpp), значение показателя точности классификации ненамного превосходит точность разработанной модели. Из табл. 2 видно, что разработанная модель не уступает представленным в [3]. В то же время по представленным значениям показателя точности можно сделать вывод о том, что использование нейронных сетей для обнаружения стегоконтейнера, дает лучшие результаты по сравнению с RM-моделью. При этом значение показателя точности разработанной модели может быть увеличено за счет увеличения объема выборки цифровых изображений, используемой на этапе обучения нейронной сети.

6. Заключение

В статье рассмотрена возможность применения аппарата свёрточных нейронных сетей для обнаружения стеганографических вложений в цифровых изображениях. Результаты исследования демонстрируют возможность обнаружения до 85% фактов наличия стеганографических вложений. В ходе проведенного исследования были решены следующие задачи:

- разработана модель свёрточной нейронной сети для обнаружения факта применения стеганографии в цифровых изображениях;
- на основании разработанной модели реализована программа для стегоанализа, реализующая бинарную классификацию цифровых изображений;
- проведен сравнительный анализ использования разработанной модели, статистических классификаторов и других моделей нейронных сетей, используемых для решения

задачи стегоанализа.

К достоинствам предложенного способа обнаружения стеговложений можно отнести достаточную точность и простоту реализации. Реализованная модель позволяет находить скрытые зависимости, не применяя сложных статистических алгоритмов. Из недостатков стоит отметить необходимость решения задачи формирования представительной выборки цифровых изображений, используемой на этапе обучения нейронной сети.

Направлением дальнейших исследований является совершенствование разработанной модели с целью повышения вероятности обнаружения стеганографических вложений и снижения вычислительной сложности ее программной реализации, а также создание собственной базы изображений с различными параметрами вложений.

Список литературы / References

[1]. Boroumand M, Chen M, Fridrich J. Deep Residual Network for Steganalysis of Digital Images. IEEE Transactions on Information Forensics and Security, vol. 14, issue 5, 2019, pp. 1181-1193.

[2]. Kodovsky J, Fridrich J. Rich models for steganalysis of digital images. IEEE Transactions on Information Forensics and Security, vol. 7, issue 3, 2012, pp. 868-882.

[3]. Yedroudj M, Comby F, Chaumont M. Yedrouj-Net: An Efficient CNN for Spatial Steganalysis. In Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, 2018, pp. 2092-2096.

[4]. Lerch-Hostalot D, Megias D. Detection of Classiffer Inconsistencies in Image Steganalysis. In Proc. of the ACM Workshop on Information Hiding and Multimedia Security. 2019, pp. 222-229.

[5]. Свёрточная нейронная сеть с нуля. Часть 0. Введение [Электронный ресурс] – Режим доступа: URL: https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-zero-introduction (01.07.2020) / Convolutional neural network from scratch. Part 0. Introduction. URL: https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-zero-introduction (in Russian).

[6]. Shafkat I. Intuitively Understanding Convolutions for Deep Learning. URL: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1 (accessed 06.06.2020).

[7]. Saha S. A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed 06.06.2020)

[8]. Yousfi Y, Butora J, Fridrich J, Giboulot Q. Breaking ALASKA: Color Separation for Steganalysis in JPEG Domain. In Proc. of the ACM Workshop on Information Hiding and Multimedia Security, 2019, pp. 138-149.

[9]. Convolution arithmetic tutorial. URL: http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html (accessed 01.07.2020)

[10]. Kodovsky J, Fridrich J. Steganalysis of JPEG images using rich models. In Proc. of the SPIE International Conference on Media Watermarking, Security, and Forensics, 2012, paper 8303-8.

Информация об авторах / Information about authors

Александр Александрович ПОЛУНИН – сотрудник. В его научные интересы входят: машинное обучение, искусственный интеллект на основе нейронных сетей, компьютерное зрение.

Alexander Alexandrovich POLUNIN is an employee. His research interests include machine learning, artificial intelligence based on neural networks, and computer vision.

Элина Андреевна ЯНДАШЕВСКАЯ – сотрудница. К её интересам в научной сфере можно отнести: криптографические методы защиты информации, стеганография, стегоанализ, скрытые каналы передачи данных.

Elina Andreevna YANDASHEVSKAYA is an employee. Her research interests: cryptographic methods of information security, steganography, steganalysis, hidden data transmission channels.

DOI: 10.15514/ISPRAS–2020–32(4)–12

Двухшаговый метод объединения новостей в сюжеты

[1,2] *К.А. Скорняков, ORCID: 0000-0002-8218-4258 <kirill.skorniakov@ispras.ru>*
[1,2] *А.С. Ласкина, 0000-0003-0878-7023 <laskina.as@ispras.ru>*
[1,3] *Д.Ю. Турдаков, ORCID: 0000-0001-8745-0984<turdakov @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*
141701, Россия, Москва, Керченская, д.1 А, корп. 1
[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1.

Аннотация. Работа посвящена разработке метода выделения сюжетов в новостях на русском языке. Сюжетом мы считаем группу новостей про одно событие реального мира. Предлагается двухэтапная схема кластеризации, при которой результаты первого «грубого» шага уточняются с помощью бинарного классификатора на парах новостей. В рамках работы создан размеченный на принадлежность сюжетам корпус новостей на русском языке, доступный для скачивания. На этом наборе данных показывается, что предложенный метод превосходит существующие решения по основным внешним метрикам кластеризации.

Ключевые слова: выделение сюжетов; кластеризация; новости

Для цитирования: Скорняков К. А., Ласкина А. С., Турдаков Д. Ю. Двухшаговый метод объединения новостей в сюжеты. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 165–174. DOI: 10.15514/ISPRAS–2020–32(4)–12

Благодарности: Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта No18-07-01059.

Two Step Method for Grouping News with Similar Topics

[1,2] *K.A. Skorniakov, ORCID: 0000-0002-8218-4258 <kirill.skorniakov@ispras.ru>*
[1,2] *A.S. Laskina, 0000-0003-0878-7023 <laskina.as@ispras.ru>*
[1,3] *D.Yu. Turdakov, ORCID: 0000-0001-8745-0984<turdakov@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, MIPT,*
Kerchenskaya, Moscow, 141701, Russian Federation
[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia

Abstract. Amount of news is rapidly growing up in recent years. People cannot handle them effectively. This is the main reason why automatic methods of news stream analysis have become an important part of modern science. The paper is devoted to the part of the news stream analysis which is called “event detection”. “Event” is a group of news dedicated to one real-world event. We study news from Russian news agencies. We consider this task as clusterization on news and compare algorithms by external clusterization metrics. The paper

introduces a novel approach to detect events at news in Russian language. We propose a two-staged clustering method. It comprises “rough” clustering algorithm at the first stage and clarifying classifier at the second stage. At the first stage, a combination of shingles method and naive named entity based clusterization is used. Also we present a labeled dataset of news event detection based on «Yandex News» service. This manually labeled dataset can be used to estimate event detection methods performance. Empirical evaluation on these corpora proved the effectiveness of the proposed method for event detection at news texts.

Keywords: event detection; clustering; news

For citation: Skorniakov K.A., Laskina A.S., Turdakov D.Yu. Two Step Method for Grouping News with Similar Topics. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 165–174 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–12

Acknowledgements. This work was supported by a grant from the Russian Foundation For Basic Research №18-07-01059

1. Введение

Объем новостных изданий и количество новостей в день растет с каждым годом. Это усложняет задачи мониторинга актуальной информации и анализа новостных потоков. Поэтому автоматизация этих процессов интересует исследователей последние 20 лет ([1–6]). Статья посвящена одной из подзадач анализа новостного потока – построению системы выделения сюжетов. Сюжетом мы будем называть группу новостей про одно событие реального мира ([1]). Мы будем рассматривать новости от русскоязычных новостных изданий, без дополнительных ограничений на их тематику. Такие тексты посвящены какому-либо событию в реальном мире (в отличие от постов в социальных сетях/блогах), написаны профессионалами с соблюдением правил орфографии и пунктуации. Мы рассматриваем задачу выделения сюжетов как задачу кластеризации и сравниваем ее с существующими решениями, используя внешние метрики кластеризации. Такая система может использоваться как отдельная сущность (новостной аггрегатор), так и как вспомогательный модуль, позволяющий проводить анализ распространения информации на уровне сюжетов, а не отдельных новостей.

На наш взгляд, большинство работ про выделение сюжетов имеют один общий недостаток – недостаточно строгую оценочную часть. В данной работе мы исправим этот недочет и покажем, что разработанная система превосходит существующие решения относительно основных метрик внешней оценки кластеризации для новостей на русском языке.

2. Методы выделения сюжетов

Методы выделения сюжетов различаются по наличию заранее заданных событий реального мира. Например, А.-М. Попеску (A.-M. Popescu) и М. Пеннаккиотти (M. Pennacchiotti) [4] считают, что сюжеты связаны с заданным набором знаменитостей. В своей работе мы считаем, что у нас нет никакой априорной информации о связях новостей.

Методы можно разделить на 2 группы: без использования учителя и комбинированные методы (использующие методы обучения с учителем вместе с методами без учителя).

2.1 Выделение сюжетов без учителя

К группе методов обучения без учителя относятся работы, в которых авторы решают задачу выделения сюжетов как задачу кластеризации. В этой группе можно выделить два основных направления – потоковая и «групповая» кластеризации. При потоковой обработке каждое сообщение проверяется на принадлежность одному из существующих кластеров-сюжетов. При «групповой» на массиве сообщений запускается алгоритм кластеризации и полученные кластеры считаются сюжетами.

Типичным примером потоковой кластеризации является работа Дж. Шанкаранараян (J. Sankaranarayanan) и др. [7]. Авторы предлагают поддерживать список актуальных кластеров и проверять каждую входящую новость на принадлежность одному из них.

Р. Лонг (R. Long) и др. в [8] предложили метод «групповой» кластеризации, основанный на группировке новостей по «тематическим словам» (topical words).

В работе С. Петровича (S. Petrović) и др. [5] была предложена модификация алгоритма Local Sensetive Hashing (LSH, стандартный метод для «групповой» кластеризации) для выявления сюжетов в потоке сообщений Twitter.

2.2 Комбинированные методы

К комбинированным методам относятся методы, использующие классификацию «поверх» кластеризации для повышения качества ее работы. Т. Сакаки (T. Sakaki) и др. [9] использовали классификацию сообщений поверх поисковика Twitter, чтобы оставить сообщения, посвященным землетрясениям. Х. Беккер (H. Becker) и др. [6] предлагают объединять алгоритм потоковой кластеризации с классификатором, определяющим относятся ли выявленные кластеры к сюжетам или это набор случайных сообщений. В работе Дж. Г. Гонрада (J.G. Conrad) и М. Бендера (M. Bender) [10] авторы используют двухэтапную кластеризацию, где на втором этапе они уточняют грубые кластеры с помощью извлеченных векторов именованных сущностей.

2.3 Признаки

Для построения качественного алгоритма машинного обучения очень важен выбор подходящего признакового пространства. Рассмотрим подробнее, какие варианты признаков используют для решения задачи выделения сюжетов другие исследователи. Многие методы ([2, 3, 11]) выделения сюжетов основаны на представлении документа в виде «мешка слов» (bag of words) и его вариациях: мешок самых популярных слов, мешок именованных сущностей и т.д.

Часть методов использует различные виды графов (термин-текст, текст-текст и т.д.) для кластеризации, например, Т. Хуа (T. Hua) и др. в работе [12].

Другой популярный подход – использование тематического моделирования. При таком подходе сначала фиксируется число тем t, каждый текст представляется в виде вектора размера t. Элементы этого вектора – вероятности принадлежности текста к соответствующей теме. Так в [13], [14] для кластеризации сообщений в сюжеты используется популярный метод тематического моделирования – латентное размещение Дирихле (LDA) и его модификации.

2.4 Наборы данных

В значительном числе статей используется Twitter как основной источник для данных и тестирования. Для этого есть несколько причин: популярность в англоязычном Интернет, удобное API для скачивания, наличие хештегов (по ним удобно группировать сообщения).

К сожалению, они не подходят нам по нескольким причинам.

- Основная причина в том, что исследователи не выкладывают оригиналы текстов твиттов (из-за правил Twitter[1]). В редких случаях доступны id твиттов, но их технически сложно получить из-за ограничений API Twitter, при этом твитты заблокированных/удаленных пользователей получить и вовсе невозможно.
- Твитты – это короткие тексты, в основном на английском языке, со своим языком. Такие тексты содержат сленг, опечатки. Тексты новостных изданий, напротив, средней длины,

[1] https://developer.twitter.com/en/developer-terms/policy

написаны с соблюдением правил языка, без использования узкоспециализированной лексики.

Однако сотрудники исследовательского подразделения Thomson Reuters (крупная медиакомпания) Дж. Г. Гонрад (J.G. Conrad) и М. Бендер (M. Bender) в работе [10] используют англоязычный набор новостей от Reuters для решения задачи объединения в сюжеты новостей Thomson Reuters. Также некоторые исследователи ([2, 3]) используют набор данных TDT4[2], содержащий в себе распределение англоязычных и арабских новостей по сюжетам. Набор данных закрыт и предоставляется за отдельную плату.

2.5 Варианты постановки задачи

Определение сюжета – «набор новостей/текстов про одно событие реального мира» – достаточно общее, поэтому в различных работах можно встретить сужение этого определения на конкретную область. Кроме того, есть различные вариации выделения сюжетов.

По теме сюжетов работы делятся на:

- сюжеты о знаменитостях ([4]);
- сюжеты о землетрясениях ([9]);
- сюжеты в новостях из различных областей (политика, спорт, экономика и т.д., [2, 3, 5, 6, 8, 10, 11]);
- сюжеты, привязанные к гео-координатам/конкретному городу ([7, 12 – 14]).

По типу выделения сюжетов:

- определение новых сюжетов в новостном потоке (new event detection, [1 – 3, 5, 7, 14]);
- для фиксированного сюжета определение его продолжения в новостном потоке (event tracking, [1, 8, 11]);
- группировка всех новостей в сюжеты ([4, 6, 10, 12, 13]).

В нашей работе нас интересуют сюжеты о новостях из различных областей жизни с группировкой всех новостей в сюжеты.

2.6 Кандидаты на сравнение

Как видно из обзора, «ландшафт» методов выделения сюжетов достаточно обширен. Различаются как варианты постановок задачи, так и используемые в решениях признаки.

Ближе всего к нашей постановке находится подход, описанный в [10].

- Сюжет – группа новостей про событие реального мира
- В качестве признаков доступны только текст, заголовок и время публикации новости. Не используются специфичные для платформы признаки (гео-локация, хештеги, упоминания пользователей и т.д.).
- Используется кластеризация на группе новостей.

3. Двухэтапная кластеризация

Для сохранения баланса между скоростью и качеством работы мы используем двухэтапный комбинированный метод. На быстром, но «грубом» этапе формируются блоки из кандидатов на принадлежность одному сюжету. На следующем этапе происходит уточнение полученного разбиения с помощью классификации на парах объектов.

В качестве предварительной кластеризации мы использовали комбинацию двух методов:

- Local Sensetive Hashing (LSH) метод для сравнения текстов -- алгоритм «шинглов» ([15]).

[2] https://catalog.ldc.upenn.edu/LDC2005T16

- «наивную» кластеризацию по именованным сущностям: в один кластер попадают новости, рассказывающие про одни именованные сущности.

За ними следует попарная классификация объектов каждого кластера с помощью алгоритмов машинного обучения с учителем.

Общая схема работы приведена на рис. 1.

Итоговый алгоритм выглядит следующим образом:

1. предобработка текста;
2. применение алгоритма шинглов; в один кластер попадают новости, имеющие хотя бы один общий супершингл (мы использовали 84 хеш-функции и супершинглы размером 4); подробнее см. в 3.1;
3. выделение именованных сущностей в тексте;
4. группировка текстов, имеющих общие именованные сущности; подробнее см. в 3.1;
5. объединение результатов 2 и 4 шагов; подробнее см. в разделе 3.2;
6. уточняющая классификация, подробнее см в 3.3;
7. объединение новостей в сюжеты после предыдущего шага.

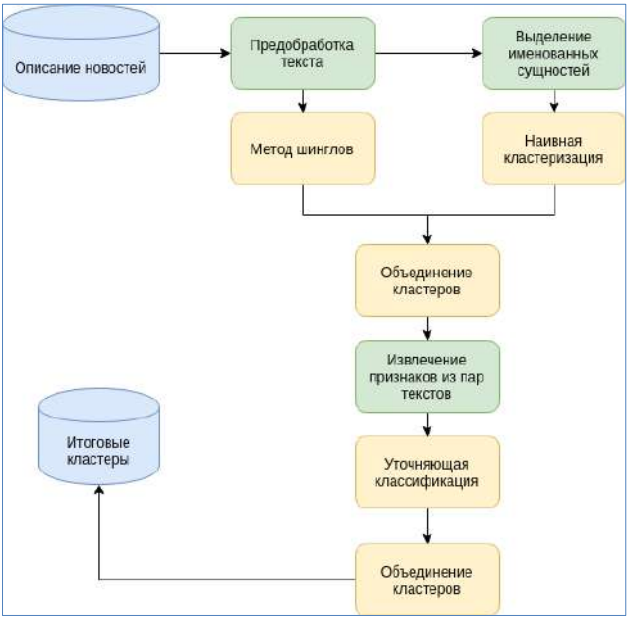


Рис. 1. Общая схема
Fig. 1. General scheme

3.1 «Грубая» кластеризация

Перед кластеризацией проводится предобработка данных: удаление html тегов и других артефактов сборщика данных, удаление стоп-слов и пунктуации, лемматизация слов.

«Грубая» кластеризация состоит из объединения двух методов: шинглов и «наивной» кластеризации. Методы запускаются независимо.

Для методов шинглов мы использовали 84 хэш-функции, супершинглы размером 4, шинглы размером 1. Мы относили новости к одном сюжету, если у них совпал один супершингл.

При «наивной» кластеризации новости принадлежат одному сюжету, если имеют общую именованную сущность за исключением «шумовых сущностей». «Шумовыми» мы считаем такие именованные сущности, которые встречаются в большой доле новостей в корпусе. Для их отсечения мы использовали пороговое значение такой доли в 0.8.

3.2 Объединение сюжетов

В процессе кластеризации одна и та же новость может попасть в несколько кластеров.

Мы считаем, что каждая новость должна принадлежать не более, чем одному сюжету и отношение «новости принадлежат одному сюжету» должно быть транзитивным, поэтому такие кластеры нужно объединить в один сюжет.

Для решения этой проблемы мы предлагаем построить граф связности новостей, где вершины – это новости, а ребра – отношение «принадлежат одному сюжету». Для этого достаточно пройтись по всем парам внутри каждого кластера, тем самым получив список ребер такого графа.

Тогда для формирования итогового разбиения на сюжеты нам нужно найти все компоненты связности в таком графе. Мы использовали реализацию метода Д. Пирса (D.J. Pearce) [16]. Таким образом мы группируем новости в сюжеты так, что отношение «новости принадлежат одному сюжету» транзитивно.

3.3 Уточняющая классификация

Следующим шагом идет уточнение полученного разбиения новостей.

Для каждого блока мы будем образовывать всевозможные пары текстов, принадлежащих данному блоку.

Далее для каждой пары новостных текстов определим, принадлежат ли они одному сюжету или нет, т.е. решим задачу бинарной классификации, отмечая каждую пару метками «0» или «1».

Для этого мы используем логистическую регрессию на следующих признаках:

- коэффициент Жаккара (P. Jaccard) [17], Дайса (L.R. Dice) [18] между «мешком слов»;
- косинус между tf-idf представлениями документов;
- расстояние Левенштейна [19] относительно слов и букв;
- нормализованное расстояние сжатия ([20]).

После уточняющей классификации ребра, получившие метку «0», удаляются из графа и повторяется шаг 5, описанный в 3.2.

4. Эксперименты

4.1 Данные

В качестве данных для экспериментов мы использовали тексты новостей и информацию о разбиении новостей на сюжеты с сервиса «Яндекс Новости»[3]. Данные собирались с помощью краулера на основе фреймворка scrapy, аналогично работе [21].


```
'_id': {'$oid': '5cdb5ba73619bf7289fcbd47'},
'title': 'Ученый КФУ: Из-за сильной магнитной бури татарстанцы могут увидеть полярное сияние',
'url': 'https://www.tatar-inform.ru/news/2019/05/14/650928/?utm_source=yxnews&utm...',
'story_url': 'https://news.yandex.ru/story/Na_Zemle_zafiksirovana_moshhnejshaya_magnitnaya...,
'text': 'Мощная магнитная буря, которая началась сегодня примерно в 6 часов утра по
московскому времени, может вызвать полярное сияние, сбои в работе систем связи...',
'agency': 'Татар-информ',
'category': 'Общество|society',
'platform': 'yandex',
'type': 'news',
'datetime': 1557878460
```


Рис. 2. Пример новостной статьи
Fig. 2. News example

[3] https://yandex.ru/news/

Скачанные данные представляют собой json-объект с информацией о названии, тексте, времени размещения, идентификаторе сюжета и прочих служебных полях для новостей. Пример приведен на рис. 2.

4.2 Формирование обучающей выборки

Для использования уточняющей классификации нам нужно построить набор пар новостей с метками «1» (принадлежат одному сюжету) и «0» (принадлежат разным сюжетам).

Для формирования положительных примеров мы использовали разбиение новостей на сюжеты в «Яндекс Новостях» (поле *story_url*) -- пара получает метку «1», если принадлежит одному сюжету в «Яндекс Новостях».

Основная проблема при создании подобных обучающих наборов из пар объектов – генерация отрицательных примеров. Стандартные подходы ([22]) предполагают отбор пар текстов с высоким коэффициентом похожести, но размеченные человеком как «0».

Однако наши эксперименты показали (мы не описываем эти предварительные эксперименты в этой статье), что в нашей задаче эффективней строить выборку иначе.

Так как основная цель нашего классификатора – исправлять ошибки кластеризации, то обучающая выборка формировалась по следующему алгоритму.

1. Получаем набор кластеров из алгоритма предварительной кластеризации.
2. Для каждого кластера строим все возможные пары новостей из этого кластера.
3. Пары, в которых новости принадлежат одному сюжету в «Яндекс Новостях» получают метку «1», остальные метку «0».

Такой подход к созданию негативных примеров показал себя лучше, чем отбор новостей из разных сюжетов с коэффициентом Жаккара выше порога.

4.3 Формирование тестовой выборки

Новости на «Яндекс Новостях» объединяются в сюжеты с помощью автоматических алгоритмов, поэтому в таком разбиении бывают ошибки.

Для исправления этих ошибок при построении тестовой выборки использовалась ручная разметка.

Идеальный вариант разметки новостей на сюжеты – попросить аннотаторов из массива новостей выделить группы сюжетов. К сожалению, он не исполним в реальности, т.к. требует огромной работы аннотаторов уже на сотнях новостей. Поэтому мы просили аннотаторов не группировать новости в сюжеты «с нуля», а исправлять предварительное разбиение, полученное в «Яндекс Новостях».

Процесс разметки был устроен следующим образом.

- Каждому аннотатору показывались по очереди новости из сюжетов «Яндекс Новостей».
- Для каждого сюжета аннотатору нужно было исключить новости, не подходящие к основному событию сюжета.
- Параллельно аннотатор отмечал номера сюжетов, описывающих одно событие.

Такая разметка позволяет построить такой набор кластеризованных новостей, в котором новости принадлежат одному кластеру, только когда они принадлежат одному сюжету (за счет шага 2). И такой набор кластеров, что новости из разных кластеров принадлежат разным сюжетам (шаг 3).

В результате разметки получился корпус из 547 новостей, размеченных на принадлежность сюжетам, доступный здесь[4]. Разметка проводилась двумя аннотаторами, коэффициент согласия каппа Кохена (J. Cohen) [23]: 0.99, что говорит о высокой степени уверенности

[4] http://talisman.ispras.ru/wp-content/uploads/2020/09/news_events.json_.gz

разметчиков. В спорных случаях привлекался третий аннотатор, и решение принималось большинством голосов.

В результате разметки из 51 исходного кластера получилось 70 новых (большая часть новых – кластеры размера 1, которые попали в изначальные кластеры по ошибке).

4.4 Метрики качества

Задача выделения сюжетов – это кластеризация, для которой известны метки настоящих кластеров на тесте. Поэтому мы будем использовать внешние метрики качества кластеризации ([24]), такие как: Adjusted Mutual Information (AMI), homogeneity, completeness, v-measure.

Дадим качественные характеристик этих метрик. Гомогенность (Homogeneity) максимальна, если все объекты в выделенных кластерах принадлежат одному сюжету, полнота (Completeness) максимальна, если все новости из сюжета попали в один кластер. V-мера (v-measure) равна среднему гармоническому гомогенности и полноты.

Скорректированная взаимная информация (AMI) основана на энтропии пар объектов, в зависимости от их попадания в разные кластеры. Принимает значение 1, когда выделенные кластеры совпадают с сюжетами, значение 0, когда разбиение случайно.

4.5 Сравнение с существующими работами

Основную сложность при тестировании систем выделения сюжетов представляет сравнение с существующими методами. Многие из них используют специфичные для конкретной площадки признаки (хештеги в Twitter), закрытые наборы данных. Код подавляющего большинства методов нельзя найти в открытом доступе. Также разные авторы по-разному ставят саму задачу выделения сюжетов (см. подраздел 2.5}, что делает сравнение между такими работами некорректным.

Мы будем сравнивать наш метод с работой [10], по нескольким причинам:

- авторы группируют все новости в сюжеты;
- авторы рассматривают новости из различных областей;
- авторы не используют признаки, специфичные для конкретной площадки;
- авторы также используют комбинированный метод.

Введем следующие обозначения: Shingles – кластеризация на основе шинглов, clf – бинарный классификатор на втором шаге, Naive – «наивная» кластеризация, SSEC – Semi-Supervised Events Clustering [10].

Замеры проводились на 547 новостях, размеченных на принадлежность сюжетам.

Табл. 1. Сравнение методов выделения сюжетов
Table 1. Comparison of plot extraction methods

Метод	**AMI**	**Homogeneity**	**Completeness**	**V-measure**
Shingles	0.45	**0.98**	0.74	0.85
Naive	0.14	0.26	0.79	0.39
Shingles + Naive	0.12	0.22	0.80	0.34
Shingles + Naive + clf	**0.81**	0.91	**0.89**	**0.90**
SSEC	0.42	**0.98**	0.73	0.84

Из таблицы видно, что предложенный метод превосходит по трем из четырех метрик существующие решения. Это достигается за счет комбинации высокой гомогенности (Homogeneity) метода шинглов с полнотой (Completeness) «наивной» кластеризации.

Как мы видим, эта комбинация не работает без уточняющей классификации, т.к. из-за «наивной» кластеризации сильно падает гомогенность выделения сюжетов. Гомогенность

максимальна, если все объекты в выделенных кластерах принадлежат одному сюжету, полнота максимальна, если все новости из сюжета попали в один кластер.

5. Заключение

В ходе работы был разработан метод выделения сюжетов, превосходящий по качеству существующие решения. Была предложена двухэтапная схема, объединяющая методы кластеризации и классификации на парах новостей. Показано, что второй этап классификации пар позволяет комбинировать различные методы кластеризации, достигая нужного баланса между гомогенностью и полнотой. Был предложен способ генерации данных для качественного обучения классификатора для уточняющей классификации.

Также мы попытались исправить пробел в методике оценки качества методов выделения сюжетов и построили корпус новостей, позволяющий оценивать точность и полноту методов. Размеченные данные позволили нам оценить различные методы выделения сюжета, используя внешние метрики кластеризации.

В дальнейшем мы планируем расширить тестовый корпус и проверить применимость такого подхода на менее формальных текстах из социальных сетей. Также интерес представляет совместное выделение сюжетов в текстах новостных изданий и постах/комментариях обычных пользователей.

Список литературы / References

[1]. J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report, In Proc. of the DARPA Broadcast News Transcription and Understanding Workshop, 1998, pp, 194-218.

[2]. T. Brants, F. Chen, and A. Farahat. A system for new event detection. In Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003, pp. 330-337.

[3]. G. Kumaran and J. Allan. Text classification and named entities for new event detection. In Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2004, pages 297-304.

[4]. A.-M. Popescu and M. Pennacchiotti. Detecting controversial events from twitter. In Proc. of the 19th ACM International Conference on Information and Knowledge Management, 2010, pp. 1873-1876.

[5]. S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In Proc. of the Annual Conference of the North American Chapter of the Association for Computational linguistics, 2010, pp. 181-189.

[6]. H. Becker, M. Naaman, and L. Gravano. Beyond trending topics: real-world event identification on twitter. In Proc. of the Fifth International AAAI Conference on Weblogs and Social Media, 2011, pp. 438-441.

[7]. J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In Proc. of the 17th ACM Sigspatial International Conference on Advances in Geographic Information Systems, 2009, pp. 42-51.

[8]. R. Long, H. Wang, Y. Chen, O. Jin, and Y. Yu. Towards effective event detection, tracking and summarization on microblog data. Lecture Notes in Computer Science, vol. 6897, 2011, pp. 652-663.

[9]. T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In Proc. of the 19th International Conference on World Wide web, 2010, pp. 851-860.

[10]. J.G. Conrad and M. Bender. Semi-supervised events clustering in news retrieval. In Proc. of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval, 2016, pp. 21–26.

[11]. M. Mohd. Named entity patterns across news domains. In Proc. of the 1st BCS IRSG Conference on Future Directions in Information Access, 2007, 5 p.

[12]. T. Hua, F. Chen, L. Zhao, C.-T. Lu, and N. Ramakrishnan. Sted: semi-supervised targeted-interest event detectionin in twitter. In Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 1466-1469.

[13]. K. N. Vavliakis, F. A. Tzima, and P. A. Mitkas. Event detection via lda for the mediaeval 2012 sed task. In Proc. of the Multimedia Benchmark Workshop, 2012, 2 p.

[14]. X. Zhou and L. Chen. Event detection over twitter social media streams. The VLDB journal, vol. 23, no. 3, 2014, pp. 381-400.

[15]. A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. Computer Networks and ISDNS, vol. 29, no. 8-13, 1997, pp. 1157-1166.

[16]. D. J. Pearce. An improved algorithm for finding the strongly connected components of a directed graph. Victoria University, Wellington, NZ, Tech. Rep, 2005.

[17]. P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin del la Société Vaudoise des Sciences Naturelles, vol. 37. 1901, pp. 547–579 (in French).

[18]. L. R. Dice. Measures of the amount of ecologic association between species. Ecology, vol. 26, no. 3, 1945, pp. 297-302.

[19]. В.И. Левенштейн. Двоичные коды, способные исправлять удаления, вставки и обращения. Доклады Академии Наук СССР, том 163, no. 4, 1966 г., стр. 845-848. / V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Soviet physics doklady, vol. 10, no. 8, 1966, pp. 707-710.

[20]. R. Cilibrasi and P. M. Vitányi. Clustering by compression. IEEE Transactions on Information Theory, vol. 51, no. 4, 2005, pp. 1523–1545.

[21]. А.К. Яцков, М.И. Варламов, Д.Ю. Турдаков. Сбор и извлечение данных с веб-сайтов СМИ. Программирование, том 44, no. 5, 2018 г., стр. 68-80 / A.K. Yatskov, M.I. Varlamov, and D.Yu. Turdakov. Extraction of data from mass media web sites. Programming and Computer Software, vol. 44, no.5, 2018, pp. 344-352.

[22]. E. Pronoza, E. Yagunova, and A. Pronoza. Construction of a russian paraphrase corpus: unsupervised paraphrase extraction. In Proc. of the Russian Summer School in Information Retrieval, 2015, pp. 146–157.

[23]. J. Cohen. A coefficient of agreement for nominal scales. Educational and Psychological Measurement, vol. 20, no. 1, 1960, pp. 37–46.

[24]. П.А. Пархоменко, А.А. Григорьев, Н.А. Астраханцев. Обзор и экспериментальное сравнение методов кластеризации текстов. Труды ИСП РАН, том 29, вып. 2, 2017. DOI: 10.15514/ISPRAS-2017-29(2)-6 / P.A. Parhomenko, A.A. Grigorev, N.A Astrakhantsev. A survey and an experimental comparison of methods for text clustering: application to scientific articles. Trudy ISP RAN/Proc. ISP RAS, 2017, vol.29, issue 2, pp.161-200 (in Russian).

Информация об авторах / Information about the authors

Кирилл Андреевич СКОРНЯКОВ – аспирант. Научные интересы: анализ нормативных документов, обработка графов и методы получения их векторных представлений небольшой размерности, методы адаптации существующих алгоритмов к новому домену, распределенные алгоритмы машинного обучения, обнаружение дубликатов текстов, обработка текстов на естественном языке.

Kirill Andreevich SKORNYAKOV – postgraduate student. Research interests: analysis of regulatory documents, graph embedding, domain adaptation, transfer learning, distributed machine learning algorithms, detection of duplicate texts, natural language processing.

Анна Сергеевна ЛАСКИНА – студентка магистратуры. Научные интересы: обработка текстов на естественном языке, машинное обучение.

Anna Sergeevna LASKINA – Master's student. Research interests: natural language processing, machine learning.

Денис Юрьевич ТУРДАКОВ – к.ф.-м.н., заведующий отделом «Информационные системы» ИСП РАН, доцент МГУ. Сфера научных интересов: машинное обучение, интеллектуальный анализ данных, извлечение информации, обработка естественного языка, сложные сети, анализ социальных сетей, большие данные.

Denis Yurievich TURDAKOV – Ph.D. head of the Information Systems Department at ISP RAS, associated professor at MSU. Research interests: machine learning, data mining, information extraction, natural language processing, complex networks, social network analysis, big data.

DOI: 10.15514/ISPRAS–2020–32(4)–13

Извлечение логической структуры из сканированных документов

[1]*А.О. Богатенкова, ORCID: 0000-0001-8679-1568 <nastyboget@ispras.ru>*
[2] *И.С. Козлов, ORCID: 0000-0002-0145-1159 <kozlov-ilya@ispras.ru>*
[2] *О.В. Беляева, ORCID: 0000-0002-6008-9671 <belyaeva@ispras.ru>*
[1] *А.И. Перминов, ORCID: 0000-0001-8047-0114 <perminov@ispras.ru>*

[1] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*
[2] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*

Аннотация. В статье предложен конвейер обработки сканированных документов, а также разработан метод извлечения структуры из них. Данный метод основан на многоклассовой классификации строк документа, в том числе классификации на заголовки и списки. Конвейер состоит из извлечения текста и рамок строк документов с помощью методов OCR, формирования признаков и обучения классификатора на данных признаках. Кроме того, размечен и доступен для изучения корпус документов, проведена экспериментальная проверка реализованного метода на данном корпусе и описаны возможности для дальнейшей работы и исследований.

Ключевые слова: машинное обучение; структура документа; обработка естественного языка; OCR

Для цитирования: Богатенкова А.О., Козлов И.С., Беляева О.В., Перминов А.И. Извлечение логической структуры из сканированных документов. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 175–188. DOI: 10.15514/ISPRAS–2020–32(4)–13

Logical structure extraction from scanned documents

[1]*Bogatenkova A.O.*, ORCID: *0000-0001-8679-1568 <nastyboget@*ispras.ru>
[2] *Kozlov I.S., ORCID: 0000-0002-0145-1159 <kozlov-ilya@ispras.ru>*
[2] *Belyaeva O.V., ORCID: 0000-0002-6008-9671 <belyaeva@ispras.ru>*
[1] *Perminov A.I., ORCID: 0000-0001-8047-0114 <perminov@ispras.ru>*

[1] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation*
[2] *Ivannikov Institute for System Programming of the RAS, 25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia*

Abstract. Logical structure extraction from various documents has been a longstanding research topic because of its high influence on a wide range of practical applications. A huge variety of different types of documents and, as a consequence, the variety of possible document structures make this task particularly difficult. The purpose of this work is to show one of the ways to represent and extract the structure of documents of a special type. We consider scanned documents without a text layer. This means that the text in such documents cannot be selected or copied. Moreover, you cannot search for the content of such documents. However, a huge number of scanned documents exist that one needs to work with. Understanding the information in such documents may be useful for their analysis, e. g. for the effective search within documents, navigation and summarization. To cope with a large collection of documents the task should be performed automatically. The paper describes the pipeline for scanned documents processing. The method is based on the multiclass classification of

document lines. The set of classes include textual lines, headers and lists. Firstly, text and bounding boxes for document lines are extracted using OCR methods, then different features are generated for each line, which are the input of the classifier. We also made available dataset of documents, which includes bounding boxes and labels for each document line; evaluated the effectiveness of our approach using this dataset and described the possible future work in the field of document processing.

Keywords: machine learning; document structure; natural language processing; OCR

For citation: Bogatenkova A.O., Kozlov I.S., Belyaeva O.V., Perminov A.I. Logical structure extraction from scanned documents. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 175–188 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–13

1. Введение

Документы имеют определённую логическую структуру. Например, законы делятся на главы, статьи, разделы. Научные статьи состоят из аннотации, введения, обзора существующих работ и других секций. Информация о логической структуре полезна для автоматического анализа документа.

Задача автоматического извлечения логической структуры из документов осложняется следующими факторами:

Во-первых, большое количество документов представляет из себя сканированные изображения с бумажных носителей. Такие изображения не содержат текстовый слой, для его извлечения необходимо использовать методы обработки изображений.

Во-вторых, зачастую логические части документа выделяются с помощью форматирования: увеличенного размера шрифта, жирности, отступов. Такая информация помогает читателю лучше понимать структуру документа, автоматическая система также должна учитывать эти признаки.

В-третьих, разные типы документов организованы отличным образом. Например, научные статьи, финансовые отчёты, законы могут состоять из разных структурных элементов (законы имеют главы, статьи, пункты, подпункты; научные статьи состоят из введения, аннотации, списка литературы). Форматирование и язык документов также могут быть различны (законы, как правило, пишут в 1 колонку, научные статьи – в 1-2 колонки). Задача создать систему, способную обработать любой тип документа может оказаться слишком сложной, задача построения своей системы для каждого типа документов «с нуля» может оказаться слишком трудоёмкой. Таким образом, наша система должна позволять добавлять поддержку новых типов документов, причём такое добавление не должно быть слишком трудоёмким.

Многие типы документов имеют требования по оформлению, однако эти требования не являются полностью формализованными, кроме того составители документов могут от них отклоняться. В связи с описанным выше разнообразием можно сделать вывод, что для выделения логической структуры лучше подходят методы машинного обучения.

В данной статье описан метод извлечения структуры документа в виде заголовков, элементов списков и текстовых строк. Каждая строка документа относится к одному из этих трех типов на основе определённых признаков. Для выделения признаков может быть необходима метаинформация, такая как размер и тип шрифта, отступы, междустрочные интервалы и т. д. Поэтому извлечение логической структуры целесообразно делать на этапе анализа изображений.

Данный анализ предполагает следующее: с помощью методов OCR из изображений извлекаются строки документа с текстом и координатами их рамок на изображении. Следующим шагом конвейера является выделение признаков на основе извлечённых данных. Далее выбранным алгоритмом машинного обучения проводится многоклассовая классификация строк.

Статья организована следующим образом: разд. 2 содержит обзор различных подходов, с помощью которых решается задача выделения структуры документа; в разд. 3 раскрывается процесс составления обучающего набора данных, в частности описывается набор документов, используемый при реализации и проверке метода и манифест для разметки данных; в разд. 4 рассматривается реализованный метод; в разд. 5 показаны результаты экспериментальной проверки метода, сравнение различных методов машинного обучения, анализ ошибок и анализ важности признаков, а в разд. 6 представлены краткие выводы и предлагаются возможности для дальнейшей работы и исследований.

2. Обзор аналогичных работ

Применяется множество разнообразных подходов [2-4], которые позволяют выделять в тексте заголовки и распознавать логическую структуру документов.

Среди подходов можно выделить следующие:

- на основе оглавления;
- на основе правил;
- на основе машинного обучения.

2.1 Извлечение структуры из документов на основе оглавления и правил

По анализу содержимого и структуре изображений документов проводятся соревнования ICDAR [5-7]. В одном из таких соревнований [5] производилось извлечение структуры из книг, содержимое которых было получено с помощью оптического распознавания символов. Структура книг в виде разбиения на страницы, параграфы, главы извлекалась с использованием оглавления, которое присутствовало в большинстве книг.

В 2019 году проводились соревнования FinTOC [8], где из финансовых документов извлекалась структура в виде иерархии уровней заголовков документов. Максимальная глубина уровней равна пяти. Одна из команд-участниц [2] извлекала необходимую структуру используя оглавление документов, а также систему правил, которые применялись для определения иерархии заголовков.

Сначала идентифицировались страницы, содержащие текст оглавления, затем в документе находились страницы, соответствующие заголовкам, указанным в оглавлении.

Последним шагом являлось выделение иерархии найденных заголовков, основанное на применении правил: анализировались такие признаки, как междустрочный интервал, отступ, шрифт, символы нумерации.

Использованный подход позволил получить достаточно высокую точность, но низкую полноту, так как некоторые оглавления документов были неполными.

Извлечение структуры документов на основе оглавления имеет ряд недостатков. Во-первых, невозможно обрабатывать документы, в которых нет оглавления. Во-вторых, при использовании этого метода в структуру документа не будут включаться заголовки, которые не вошли в оглавление, например, заголовки более низкого уровня. В-третьих, данный метод не позволяет извлекать элементы маркированных и нумерованных списков, которые не включаются в оглавление документа.

2.2 Извлечение структуры из документов на основе машинного обучения

В соревнованиях [8], кроме извлечения иерархической структуры документов, решалась задача определения, является ли конкретный блок документа заголовком. Командам был дан набор pdf-документов, xml-файлов с выделенными блоками документов, а также набор признаков для каждого блока: является ли шрифт блока жирным, курсивом, состоит ли текст

из заглавных букв, начинается с заглавной буквы или с нумерации. Кроме данных признаков, каждая из команд использовала различные дополнительные морфологические, семантические, лингвистические признаки. На основе этих признаков обучались различные классификаторы: SVM, MNB, Extra Tree, Decision Tree, Gradient Boosting. Для оценки результатов использовалась F1-мера, максимальный score в соревновании – 0,982.

Победители соревнования [3] создали новый датасет для обучения с помощью аугментации данных, перевели новые сгенерированные текстовые блоки в векторное представление, а затем использовали рекуррентные нейронные сети LSTM и BiLSTM для решения задачи классификации.

Для того, чтобы классифицировать строки документа, логично использовать такие признаки, как жирность шрифта, отступы, высоту текста и т. д. Эти признаки сильно отличаются друг от друга диапазонами значений, поэтому нейронные сети LSTM и BiLSTM, как правило, плохо подходят для решения данной задачи. Кроме того, описанный подход применялся для определения заголовков и не распространялся на элементы списков.

В статье [4] структура документа извлекалась с использованием методов машинного обучения, включая глубокое обучение. Цель данной работы -- автоматически идентифицировать и классифицировать различные секции документов и понять их смысл в рамках документа (назначить семантическую метку).

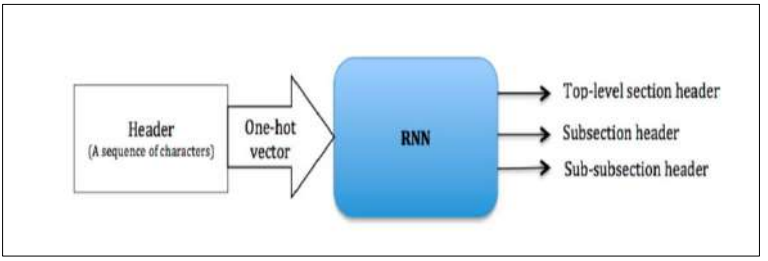


Рис. 1. Вход и выход классификатора заголовков
Fog. 1. Input and output of the section classifier

Классификатор (рис. 1), который был использован при решении задачи, состоит из нескольких частей. Сначала строки документа подаются на вход классификатору (классификатор строк), который определяет, является ли строка заголовком, затем строки-заголовки классифицируются точнее другими классификаторами (классификаторы секций). В этом решении структура документа имела вложенность 3, то есть предполагалось выделение секций, подсекций, подподсекций.

Кроме того, в данной работе был размечен датасет, на котором происходило обучение модели. Метрика качества – F1-мера, при идентификации заголовков итоговый score – 0,96; при классификации секций средний F1-score – 0,81.

Данный подход ограничивает число уровней вложенности извлекаемой структуры и так же не позволяет извлекать элементы списков. Разработанный нами метод позволяет извлекать структуру без ограничения уровней вложенности.

3. Набор данных и манифест

3.1 Описание данных

Датасет представляет собой набор документов в виде изображений в формате JPEG, скачанный с сайта zakupki.gov.ru [9]. Набор данных доступен для изучения [1].

Анализируемые документы являются сканированными копиями страниц текстов государственных закупок. Каждое изображение будем считать отдельным документом. Из

рассмотрения удалены документы, содержащие таблицы, рисунки, рамки и прочие нетекстовые элементы.

Данный корпус имеет ряд специфических особенностей. Текст расположен в одной колонке, не выделен цветом, шрифт не меняется (меняется только его начертание или размер). Большая часть всех текстов написана на русском языке, редко встречаются латинские буквы. Так как содержимое документов представляет собой в основном договоры предприятий, в текстах встречается большое количество элементов списков (нумерованных и маркированных), зачастую списки имеют очень глубокий уровень вложенности (четвертый, пятый). Заголовков относительно немного, они могут быть пронумерованы и также иметь глубокий уровень вложенности, поэтому их можно спутать с элементами списков.

Поскольку сканированная копия может быть сделана с любой страницы документа, а не только первой, на некоторых страницах (которые мы считаем отдельным документом) могут отсутствовать заголовки или элементы списков.

3.2 Разметка данных

Для создания обучающего набора данных часто используют ручную разметку -- предлагают выполнить задачу классификации человеку-аннотатору.

В книге [10] предлагается выполнять разметку обучающего корпуса в следующем порядке.

1. Спецификация задания – формальное определение задания и формата данных, используемое ПО и так далее.
2. Составление Манифеста – инструкции для аннотаторов.
3. Разметка данных. Непосредственно разметка данных с учётом Манифеста
4. Измерение согласованности аннотаторов. На этом шаге проверяется то, что размечающие понимают задание одинаково. Если это не так, то производится возврат к шагу 2.
5. Вынесение решения по аннотациям – если производилась разметка с перекрытием (один и тот же пример размечался более чем одним анотатором), возникает необходимость объединить их результаты. Мы пропустили этот шаг, так как аннотаторы производили разметку без перекрытия.

Опишем эти шаги подробнее применительно к нашей задаче.

3.2.1 Спецификация задания

Мы поставили нашу задачу как многоклассовую классификацию строк. Аннотаторам последовательно показывались изображения сканированного документа, одна из строк которого обведена в красную рамку. Аннотатору было необходимо отнести текст в рамке к одному из наперёд заданных классов. В нашей задаче выполнялась классификация на следующие классы:

- заголовок;
- элемент списка;
- простой текст;
- другое (не текст).

Таким образом, аннотатору ставилась задача классификации изображения. Для выполнения этого задания использовалась система собственной разработки [11]. После разметки данные сохранялись в формате JSON, формат описан в Приложении (рис. 4).

3.2.2 Манифест

В предыдущей секции мы определили задание для разметки как классификацию изображения на один из нескольких классов. Теперь необходимо определить, чем должны руководствоваться аннотаторы, относя изображение к тому или иному классу, в этом им

помогает *манифест*, или инструкция для разметки (основную часть манифеста можно посмотреть в Приложении, в [1] содержится полный текст манифеста).

Как правило, невозможно полностью формально описать правила классификации (в противном случае нам нет необходимости использовать машинное обучение). Поэтому в манифесте допустимо использование нестрого определённых понятий и правил (например, *жирный текст, выравнивание по центру, изображение синей печати*). При этом важно, чтобы все аннотаторы понимали задание одинаково. Для того, чтобы убедиться в этом, вычисляется согласованность. Низкая согласованность может означать то, что манифест написан недостаточно хорошо/понятно и требует доработки.

3.2.3 Согласованность

Для проверки одинакового и правильного понимания задания аннотаторами необходимо измерить их согласованность:

1. предложить нескольким аннотаторам независимо выполнить разметку одного и того же множества заданий;
2. вычислить специальную статистику, показывающую, насколько согласована разметка;
3. в случае низкой согласованности рекомендуется разобрать спорные ситуации и обновить манифест, лучше прописав правила для спорных ситуаций и добавив примеров.

Хороший обзор о методах вычисления согласованности можно найти в статьях [12] и [13]. Для проверки правильности разметки была посчитана специальная статистика *kappa (κ)* Коэна (Cohen), принимающая значения ≤ 1. Чем ближе κ к 1, тем выше согласованность. После разметки десяти документов (407 строк) двумя аннотаторами значение статистики κ оказалось равным 0.975, что считается высоким уровнем согласованности.

После чего было решено размечать остальной корпус документов. В результате было размечено 600 документов (21350 строк) и отдельные JSON файлы были объединены в один (рис. 4 в Приложении).

Стоит отметить, что высокая согласованность ещё не означает, что задание выполнено «правильно». Так, если все аннотаторы будут всегда относить каждое изображение к классу *Простой текст*, не обращая внимания на признаки и манифест, то κ будет равна 1, но вряд ли это можно назвать правильной разметкой. Такая проблема актуальна при использовании краудсорсинга. Обзор методов краудсорсинга можно найти в статье [14].

4. Описание решения

4.1 Описание реализованного метода

Первым шагом в решении задачи является выделение текста и рамок для строк документа с помощью методов OCR. Вторым шагом является составление вектора признаков для каждой строки документа. С помощью выделенной текстовой информации извлекаются различные текстовые признаки, описанные в подразделе 4.2. Для визуальных признаков используется информация из координат рамок (для отступов и высоты). В определении жирности шрифта используется само изображение документа и координаты рамок строк внутри изображения.

Таким образом, каждый документ представлен набором векторов признаков для строк, а тренировочные данные являются объединением данных для документов.

Следующим шагом в решении задачи является применение алгоритма машинного обучения, который на основе выделенных признаков распределит строки по классам. Схема конвейера показана на рис. 2.

Рис. 2. Конвейер обработки документов
Fig. 2. Pipeline for documents processing

4.2 Выделение признаков

Среди признаков, характеризующих строки документа, можно выделить следующие группы:

- *Признаки, основанные на регулярных выражениях*.

  Данная группа признаков основывается на анализе начала и конца каждой строки. Такие признаки очень важны для выявления элементов списков различных типов, а также могут сигнализировать о конце заголовка или начале списка.

  Регулярные выражения позволяют выделить следующие признаки:
  - начинается ли строка с цифры или буквы со скобкой или точкой (также анализируются иерархические выражения вида 1.1.1);
  - начинается ли строка с тире (и других символов, характерных для маркированного списка);
  - состоит ли строка целиком из заглавных букв (характерно для некоторых заголовков);
  - начинается ли строка с заглавной (строчной) буквы;
  - начинается ли строка с конкретных слов типа «Раздел», «Секция», «Глава» и т. д.;
  - оканчивается ли строка символами вида «. , ; :»;
  - оканчивается ли строка строчной буквой.
- *Текстовые признаки.*

  Данная группа признаков связана с подсчетом некоторых строковых характеристик, а именно:
  - количество букв в первом и втором словах строки;
  - количество слов в строке (строка разбивается на слова по пробелам);
  - количество символов в строке (длина строки).
- *Визуальные признаки.*

  Данная группа признаков связана с графическим представлением текста в документе. То есть при анализе строки рассматривается не ее текст, а следующие признаки (первые три признака измеряются в пикселях):
  - отступ от левого края страницы;
  - высота текста строки (точнее высота ограничивающей ее рамки);
  - отступ от верхнего края страницы;
  - жирность шрифта различных уровней (подробнее см. в разд. 4.2.1).

Важно рассматривать не одну строку, а ее окрестность для увеличения точности классификатора строк. Поэтому к признакам каждой строки добавляются признаки четырех предыдущих и последующих строк.

Для строк, которые начинаются с нумерации, определяется, есть ли в документе строка, предшествующая данной с нумерацией, меньшей данной на единицу.

И, наконец, для каждого документа вычисляется средний отступ от левого края страницы, средняя высота шрифта, средняя длина строки, среднее число слов в строках, среднее значение для жирности шрифта (ядро свертки 5), среднее число букв в первом слове каждой строки. Данные значения добавляются к признакам каждой строки документа.

4.2.1 Определение жирности шрифта

Рассмотрим более подробно способ определения жирности шрифта. Для этого использовались морфологические операции Dilation и Erosion [15] из библиотеки OpenCV [16]. Данные операции с помощью комбинаций увеличения и сужения границ на изображении позволяют детектировать жирность текста.

При работе с текстом в качестве исходного изображения используется bounding box конкретной строки, полученный с помощью [17]. В данной работе функции erode и dilate запускались с параметром kernel, равным от 2 до 7 включительно (6 раз).

5. Экспериментальная проверка метода

5.1 Подбор классификатора

При решении задачи было опробовано множество методов машинного обучения, для лучших из них проведен анализ результатов. В анализе участвовало 4 классификатора:

- алгоритм k ближайших соседей (KNeighborsClassifier);
- логистическая регрессия (LogisticRegression};
- градиентный бустинг (GradientBoostingClassifier);
- «extreme» градиентный бустинг (XGBClassifier).

Набор размеченных документов тремя способами был разбит на тренировочное и тестовое множества. Разбиение производилось по документам, то есть группа строк, относящихся к одному документу попадала целиком либо в тренировочное, либо в тестовое множество. На каждом разбиении было проведено обучение классификаторов и вычисление F1-score (с макро-усреднением). Усредненные значения F1-score для каждого классификатора указаны в табл. 1.

Табл. 1. Сравнение классификаторов
Table 1. Classifier comparison

Классификатор	**F1-score**
Nearest Neighbors	0.89
Logistic Regression	0.9
Gradient Boosting	0.92
XGBoost	**0.95**

В целом, все рассмотренные классификаторы показали хороший результат (табл. табл. 1), наилучший результат показал XGBClassifier, поэтому было решено выбрать его.

5.2 Анализ значимости признаков

Анализ значимости признаков был проведен с помощью библиотеки xgbfir [18]. В табл. 2 представлены первые 10 признаков с наивысшей значимостью (information gain). Это признаки, которые имеют наибольший вес при вычислении предсказания классификатора.

С использованием данных о важности признаков, в признаковое пространство были добавлены новые признаки. Например, вместо одного признака, отвечающего за жирность шрифта, была добавлена целая группа признаков, отвечающая за различные уровни жирности шрифта. Для самых важных признаков к вектору признаков каждой строки документа были добавлены усреднённые значения данных признаков по документу. В табл. 2 представлен итоговый список признаков после добавления и значения их важности.

5.3 Результаты

После настройки параметров XGBClassifier способом, описанным в [9], на валидационном датасете (*learning_rate=0.1, n_estimators=1000, max_depth=7, min_child_weight=2, gamma=0,*

subsample=1, colsample_bytree=1, alpha=0.01) итоговый F1-score, полученный в результате кросс-валидации (разбиение данных на 3 части), оказался равным 0.98995.

Табл. 2. Значимость признаков
Table 2. Features importances

Признак	**Information gain**
Число символов первого слова в строке	22089
Индикатор, является ли строка продолжением списка	2400
Жирность шрифта	2368
Число слов в строке	2263
Отступ от левого края страницы	1557
Признак начала строки с выражения вида 1.1.1 (произвольный уровень вложенности, вместо цифр могут быть буквы)	1519
Жирность шрифта (менее жирный шрифт)	811
Число букв в начале строки	361
Тире в начале строки	359

5.4 Анализ ошибок

На рис. 3 показана матрица ошибок для полученного классификатора. Матрица получена на валидационном датасете. По вертикальной оси расположены правильные классы, по горизонтальной – классы, которые предсказал классификатор. В клетках на пересечении расположены значения количества строк, у которых совпали данные классы.

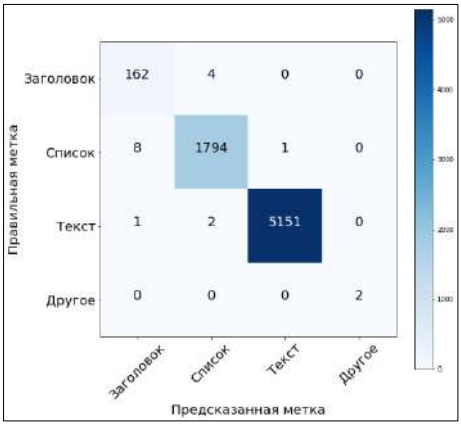


Рис. 3. Матрица ошибок без нормализации
Fig. 3. Confusion matrix without normalization

На рис. 3 видно, что 8 строк, относящихся к типу «Список», были проклассифицированы как «Заголовок», 4 строки; напротив, вместо метки «Заголовок» получили метку «Список». Аналогичную статистику можно посмотреть и для других пар классов.

Таким образом, больше всего классификатор путает классы «Заголовок» и «Список». Это можно объяснить тем, что некоторые признаки данных классов очень похожи, например, многие заголовки начинаются с нумерации, а элементы списков имеют большой отступ от левого края страницы.

6. Заключение

В данной статье разработан метод выделения логической структуры документа, основанный на классификации строк документа, определяющий в документах заголовки и списки разных уровней вложенности.

Реализован пайплайн, состоящий из обработки документов с помощью программы Tesseract [17] и извлечения текста и рамок строк, выделения векторов признаков и обучения классификатора. Кроме того, с помощью ручной разметки получен размеченный датасет, доступный для изучения [1]. Для эффективной классификации списков и заголовков помогает извлечение признаков, указанных в табл. 2 и использование XGBClassifier [20]. Итоговый F1-score на данных кросс-валидации, полученный при настройке параметров классификатора, равен 0.98995. Однако, в силу особенностей датасета классификатор может путать заголовки и элементы списков.

Дальнейшие исследования могут быть направлены на выделение более подробной структуры. Помимо классификации каждой строки можно определять её уровень вложенности по отношению к документу.

Список литературы / References

[1] A. Bogatenkova. Dataset. URL: https: //github.com/NastyBoget/document_structure_extraction (accessed 27.05.2020).

[2] E. Giguet and G. Lejeune. Daniel@ fintoc-2019 shared task: toc extraction and title detection. In Proc. of the Second Financial Narrative Processing Workshop (FNP 2019), 2019, pp. 63-68,

[3] K. Tian and Z. Peng. Finance document extraction using data augmentation and attention. In Proc. of the Second Financial Narrative Processing Workshop (FNP 2019), 2019, pp. 1-4.

[4] M.M. Rahman and T. Finin. Deep understanding of a document's structure. In Proc. of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, 2017, pp. 63-73.

[5] A. Doucet, G. Kazai, S. Colutto, and G. Mühlberger. Icdar 2013 competition on book structure extraction. In Proc. of the 12th International Conference on Document Analysis and Recognition, 2013, pp. 1438-1443.

[6] L. Gao, X. Yi, Z. Jiang, L. Hao, and Z. Tang. Icdar2017 competition on page object detection. In Proc. of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, 2017, pp. 1417-1422.

[7] C. Clausner, A. Antonacopoulos, and S. Pletschacher. Icdar2017 competition on recognition of documents with complex layouts-rdcl2017. In Proc. of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, 2017, pp. 1404-1410.

[8] R. Juge, I. Bentabet, and S. Ferradans. The fintoc-2019 shared task: financial document structure extraction. In Proc. of the Second Financial Narrative Processing Workshop (FNP 2019), 2019, pp. 51–57.

[9] Единая информационная система в сфере закупок, ЕИС. URL: https://zakupki.gov.ru/ (дата обращения 27.05.2020) / Unified information system in the field of procurement, EIS. URL: https://zakupki.gov.ru/ (in Russian).

[10] J. Pustejovsky and A. Stubbs. Chapter 6. Annotation and Adjudication. In Natural Language Annotation for Machine Learning: A guide to corpus-building for applications. O'Reilly Media, 2012, pp. 105-139.
[11] A. Permonov. Paragraph labeler application, URL: https://github.com/dronperminov/ParagraphLabelerApp (accessed 10.06.2020).
[12] R. Artstein and M. Poesio. Inter-coder agreement for computational linguistics. Computational Linguistics, vol. 34, no. 4, 2008, pp. 555-596.
[13] P.S. Bayerl and K.I. Paul. What determines inter-coder agreement in manual annotations? A meta-analytic investigation. Computational Linguistics, vol. 37, no. 4, 2011, pp. 699–725.
[14] Р.А. Гилязев, Д.Ю. Турдаков Активное обучение и краудсорсинг: обзор методов оптимизации разметки данных. Труды ИСП РАН, том 30, вып. 2, 2018 г., стр. 215-250. DOI: 10.15514/ISPRAS-2018-30(2)-11 / R.A. Gilyazev, D.Y. Turdakov. Active learning and crowdsourcing: a survey of data markup optimization methods. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 2, 2018, pp. 215-250 (in Russian).
[15] J. Liang, J. Piper, and J.-Y. Tang. Erosion and dilation of binary images by arbitrary structuring elements using interval coding. Pattern Recognition Letters, vol. 9, no. 3, 1989, pp. 201–209.
[16] Opencv, Intel Corporation. URL: https://opencv.org (accessed 27.05.2020).
[17] R. Smith. An overview of the tesseract ocr engine. In Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, 2007, pp. 629-633.
[18] B. Kostenko. XGBoost feature interactions reshaped. URL: https://github.com/limexp/xgbfir (accessed 27.05.2020).
[19] A. Jain. Complete guide to parameter tuning in xgboost with codes in python. URL: https: //www.analyticsvidhya.com/blog/2016/03/completeguide-parameter-tuning-xgboost-with-codes-python (accessed 27.05.2020).
[20] XGBoost documentation, The XGBoost Contributors. URL: https://xgboost.readthedocs.io/en/latest/index.html (accessed 27.05.2020).

Приложение / Appendix

```
[
    {
    "name": name of the first document,
    "width": image width (pixels),
    "height": image height,
    "entities": [
        {
            "label": first line label,
            # bounding box for the first line
            "x": first line left indent,
            "y": first line top indent,
            "width": first line width,
            "height": first line height,
            "text": first line text
        }, ...
    }, ...
]
```

Рис. 4. Результат разметки
Fig. 4. Labeling result

Манифест

Выделяются следующие типы строк: заголовок, элемент списка, текст.

1. Заголовок – это название главы, секции, подглавы, параграфа. Строка помечается заголовком, если:
   - текст визуально (полностью) выделяется жирностью (рис. 5);

Рис. 5. Пример заголовка №1
Fig. 5. Header example №1

- текст полностью выделяется шрифтом (курсив, подчеркнутый, другой шрифт, другой размер шрифта) (рис. 6);

  при этом если текст строки выделен шрифтом частично, то заголовком это не считается;
- текст выделяется отступом (расположен по центру) (рис. 6);

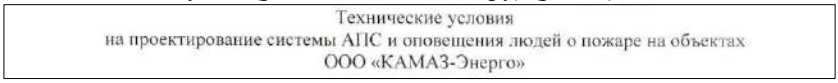


Рис. 6. Пример заголовка №2
Fig. 6. Header example №2

- если заголовок занимает несколько строк, остальные строки тоже относятся к типу «заголовок».

2. Элемент списка – это начало нумерованного или маркированного списка. Строка помечается как элемент списка, если:
   - строка наряду с несколькими другими строками пронумерована («1. 1) а) 1.1» и т. д. в начале строки) или выделена некоторым маркером (точка, тире и т. д.);
   - если элемент списка визуально занимает несколько строк, все строки кроме первой помечаются как текст. Также к списку не относятся строки, помеченные как заголовок (выделенные шрифтом, жирностью и т. д.). На рис. 7 как элементы списка будут помечены только две строки, остальные помечаются как текст.

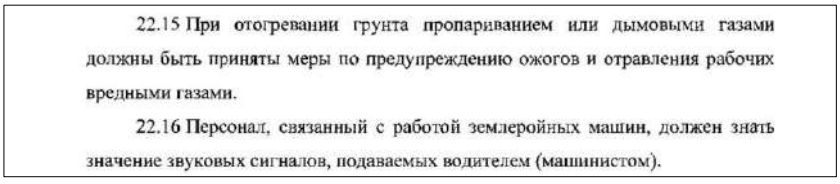


Рис. 7. Пример элементов списка
Fig. 7. List example

3. Текстовые строки – это все остальные строки, содержащие текст документа.
4. Other – при разметке могут попадаться выделенные области, не содержащие текста, такие области помечаются как «Other».

Информация об авторах / Information about authors

Анастасия Олеговна БОГАТЕНКОВА является студенткой бакалавриата кафедры системного программирования. Научные интересы: распознавание структуры документов, цифровая обработка изображений.

Anastasiya Olegovna BOGATENKOVA is a bachelor student of the Department of System Programming. Research interests: document layout analysis, digital image processing.

Илья Сергеевич КОЗЛОВ является стажером-исследователем. Научные интересы: распознавание структуры документов, цифровая обработка изображений, нейросетевая обработка данных, распознавание образов.

Ilya Sergeevich KOZLOV – researcher. Research interests: document layout analysis, digital image processing, neural network data processing, image pattern recognition.

Оксана Владимировна БЕЛЯЕВА – аспирантка. Научные интересы: распознавание структуры документов, цифровая обработка изображений, нейросетевая обработка данных, распознавание образов.

Oksana Vladimirovna BELYAEVA – a PhD Student. Research interests: document layout analysis, digital image processing, neural network data processing, image pattern recognition.

Андрей Игоревич ПЕРМИНОВ является студентом магистратуры кафедры системного программирования Научные интересы: цифровая обработка сигналов, нейросетевая обработка данных, создание искусственных данных.

Andrey Igorevich PERMINOV is a master student of the Department of system programming. Research interests: digital signal processing, neural network data processing, generation of artificial data.

DOI: 10.15514/ISPRAS–2020–32(4)–14

Использование синтетических данных для тонкой настройки моделей сегментации документов

[1] О.В. Беляева, ORCID: 0000-0002-6008-9671 <belyaeva@ispras.ru>
[2] А.И. Перминов, ORCID: 0000-0001-8047-0114 <perminov@ispras.ru>
[1] И.С. Козлов, ORCID: 0000-0002-0145-1159 <kozlov-ilya@ispras.ru>

[1] Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1

Аннотация. В рамках задачи автоматического анализа документов мы решаем задачу сегментации изображений документов DLA (Document Layout Analysis). Целью работы является сегментация изображений документов в условиях ограниченного набора реальных данных и использование для обучения искусственно созданных данных. В качестве данных рассматривается PDF-документы сканированных договоров, коммерческих предложений и технических заданий без текстового слоя. В работе мы обучаем известную высокоуровневую модель FasterRCNN сегментировать текстовые блоки, таблицы, печати и подписи на изображениях рассматриваемых данных. Работа направлена на генерацию синтетических данных схожих с реальными. Это обусловлено потребностью модели в большом наборе данных для обучения и высокой трудозатратностью их подготовки. В работе приведено описание этапа постобработки для устранения артефактов, полученных в результате сегментации. В работе приводится тестирование и сравнение качества модели, обученной на разных наборах данных (с/без синтетических данных, малом/большом наборе реальных данных, с/без этапа постобработки). В итоге мы показываем, что генерация синтетических данных и использование постобработки увеличивает качество модели при малом обучающем наборе реальных данных.

Ключевые слова: анализ физической структуры документа; сегментация документа; анализ макета документа; обнаружение объектов на изображении; тонкая настройка модели; активное обучение

Для цитирования: Беляева О.В., Перминов А.И., Козлов И.С. Использование синтетических данных для тонкой настройки моделей сегментации документов. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 189–202. DOI: 10.15514/ISPRAS–2020–32(4)–14

Synthetic data usage for document segmentation models fine-tuning

[1] Belyaeva O.V., ORCID: 0000-0002-6008-9671 <belyaeva@ispras.ru>
[2] Perminov A.I., ORCID: 0000-0001-8047-0114 <perminov@ispras.ru>
[1] Kozlov I.S., ORCID: 0000-0002-0145-1159 <kozlov-ilya@ispras.ru>

[1] Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia
[2] Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

Abstract. In this paper, we propose an approach to the document images segmentation in a case of limited set of real data for training. The main idea of our approach is to use artificially created data for training and post-

processing. The domain of the paper is PDF documents, such as scanned contracts, commercial proposals and technical specifications without a text layer is considered as data. As part of the task of automatic document analysis, we solve the problem of segmentation of DLA documents (Document Layout Analysis). In the paper we train the known high-level FasterRCNN \cite{ren2015faster} model to segment text blocks, tables, stamps and captions on images of the domain. The aim of the paper is to generate synthetic data similar to real data of the domain. It is necessary because the model needs a large dataset for training and the high labor intensity of their preparation. In the paper, we describe the post-processing stage to eliminate artifacts that are obtained as a result of the segmentation. We tested and compared the quality of a model trained on different datasets (with / without synthetic data, small / large set of real data, with / without post-processing stage). As a result, we show that the generation of synthetic data and the use of post-processing increase the quality of the model with a small real training data.

Keywords: Document Layout Analysis; Document Segmentation; Physical Document Structure; Image Object Detection; Model fine-tuning; Active Learning

For citation: Belyaeva O.V., Perminov A.I., Kozlov I.S. Synthetic data usage for document segmentation models fine-tuning. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 189–202 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–14

1. Введение

В настоящее время большое число документов представляют из себя файлы в формате PDF (Portable Document Format). Как правило, они удобны и просты в использовании. Но в частных случаях, встречаются PDF-документы, содержащие текстовый слой низкого качества, который был автоматически распознан. В худшем случае, документ представляет из себя скан бумажного носителя и не содержит вовсе текстового слоя. К таким документам, могут быть применимы только методы анализа изображения для извлечения его содержимого и структуры.

Выделяют *физическую* и *логическую* структуры документа. Первая задается такими классами объектов, как изображение, текст, таблицы, подписи и так далее. Во второй объекты разделяют на заголовки, параграфы и другие логические элементы.

Например, в статье [2] рассматривается первый вид структуры, который характеризуется геометрическим расположением объектов на странице.

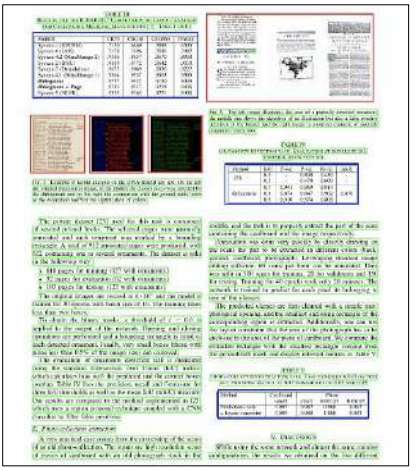


Рис. 1. Пример сегментации документа [3]
Fig. 1. An example of document layout analysis [3]

Геометрическая организация документа: шрифт, размер текста, отступы содержат важную информацию, помогающую человеку понять смысл документа. Отсюда, важно сохранить такую информацию при автоматическом его анализе. Таким образом, одним из важных этапов автоматического анализа документа является сегментация страницы (Document

Layout Analysis, DLA, рис. 1). Задача сегментации изображения документа является частным случаем задачи сегментации изображения.

Сегментация страниц отсканированных документов проводится на первых этапах анализа и задает дальнейший характер распознавания документа в целом. В ходе DLA документ разбивается на области, каждая из которых должна содержать однородную информацию (например, только текст, заголовки или только одну таблицу).

Для сегментации изображений активно применяются методы на основе глубоких нейронных сетей, таких как [1]. Методы глубокого машинного обучения в рамках решения задачи DLA появились недавно [3-5] и уже нашли широкое применение для анализа сложных макетов документов. Методы сегментации на основе глубоких нейронных сетей демонстрируют высокое качество работы, но требуют огромного объёма данных для обучения, разметка которых требует высокой внимательности и много человеко-часов. Решением данной проблемы является автоматическая генерация данных. Например, в статье [3] используются данные с сайта [6] для автоматического получения обучающей выборки сегментации научных статей, в результате авторам удалось создать огромный обучающий датасет.

В нашей работе мы исследуем возможность применения глубокого обучения к задаче DLA на данных другой области: технических заданиях и юридических документах. Мы решаем задачу многоклассовой сегментации, которая комбинирует в себе задачи Object Detection и классификации.

В условиях ограниченного набора реальных данных, нами решается задача активного обучения [7]. Существует несколько подходов активного обучения с использованием отбора существующих данных [8] или синтеза новых данных [9]. Мы достигаем адаптации модели к новому множеству за счёт её тонкой настройки на новом наборе данных. Набор создается автоматически путем генерации искусственных документов близких к нашему домену. В рамках данной работы тонкая настройка представляет собой обучение весовых коэффициентов модели на новом наборе данных. Таким образом, тонкая настройка производится на искусственно сгенерированных документах, похожих на реальные. Далее предобученная модель дообучается на ограниченном наборе имеющихся реальных данных. Таким образом, нам удалось достичь высокого качества сегментации документов в ограниченном наборе реальных данных.

Мы использовали сложную сеть Faster RCNN [1] для сегментации документа в рамках задачи Object Detection с выделением нескольких классов на изображении: текстовый блок (Text), таблица (Table), картинка (Picture). Первоначальное обучение производится на датасете PubLayNet [3], мощностью 360 тысяч размеченных изображений медицинских статей.

Для улучшения точности сегментации мы используем постобработку для устранения артефактов, образованных в результате предсказания модели. Постпроцессинг представляет собой расширение или сужение границ обнаруженных объектов с целью избежать обрезки текста, таблиц; устранить наложения объектов друг на друга и т.д. Всё вышеперечисленное влияет на качество распознавания сегментируемых объектов в дальнейшем.

2. DLA решения

Развитие DLA берет начало с ранних известных эвристических методов Smearing [10], Recursive XY-cut [11], Docstrum [12], а также сегментация методом наибольших белых прямоугольников [13]. И в настоящее время активно проводятся исследования в области DLA с использованием машинного обучения. Выделяются два типа работ в этой области: бинаризации изображений и многоклассовая сегментация. Работы [4-5] сегментируют документы путем бинаризации изображений на None-Text и Text классы (текстовые/нетекстовые области). Как правило, это нужно для анализа исторических документов, на которых важно отфильтровать только текстовую информацию. В [4-5] предлагают использование свёрточной сети для сегментации страниц исторических

документов. Авторы [4] используют простую сеть только из одного свёрточного слоя для распознавания исторических рукописных документов и сравнивают результаты с более глубокими сложными сетями. В [5] для сегментирования исторических документов предлагают Fully Convolution Network (FCN) сеть, использующую метрику, которая учитывает только пиксели переднего плана на бинаризованной странице и игнорирует фоновые пиксели.

Работы [1, 3, 14] занимаются многоклассовой сегментацией документов. В работе [15] авторы обнаруживают и распознают структуру таблиц на изображениях. В [3] создали огромный датасет на основе данных с сайта PubMed и обучили FasterRCNN на новых данных. В [14] решают задачу детекции таблиц и даграмм с помощью глубокой свёрточной сети с CRF (conditional random field) слоем.

Как показывает практика, нельзя выбрать одну конкретную сеть и сказать, что она лучше остальных. В зависимости от задачи и требований к производительности (скорости обработки изображений в секунду) и качеству сегментации, выбор может быть сделан в пользу любой из них. Ниже приведены основные характеристики данных архитектур, влияющие на их качество:

- детектор признаков (VGG16, ResNet, Inception, MobileNet, рис. 2);
- размер выхода детектора признаков;
- разрешения входного изображения;
- стратегия соответствия и порог IoU (Intersection over Union);
- количество предложений или прогнозов;
- увеличение данных путём аугментации;
- набор данных для обучения;
- какой слой или слои карты признаков используются для обнаружения объектов;
- функция потерь;
- конфигурации обучения, включая размер пакета, изменение размера входного изображения, скорость обучения и снижение скорости обучения (табл. 1).

Табл. 1. Скорость работы различных архитектур [16]
Table 1. Speed of different architectures [16]

Архитектура	**Минимальный FPS**	**Максимальный FPS**
Fast R-CNN	3	10
Faster R-CNN	5	17
SSD	22	59
YOLO	40	91

Чтобы сравнивать модели между собой, необходимо выбрать единый набор данных, на котором будет происходить сопоставление. Обычно для этого используется набор соревнования COCO [17], в котором выполняется сегментация по 80 различным классам. Ниже приводится сравнительная таблица качества сегментации архитектур (табл. 2).

Табл. 2. Точность сегментации различных архитектур (на наборе данных COCO)
Table 2. Accuracy of layout analysis n of various architectures (on the COCO dataset)

Архитектура	**Точность**
Fast R-CNN	21.9
Faster R-CNN	34.9
SSD300	23.2
SSD512	26.8
YOLO	33.0
RetinaNet	40.8

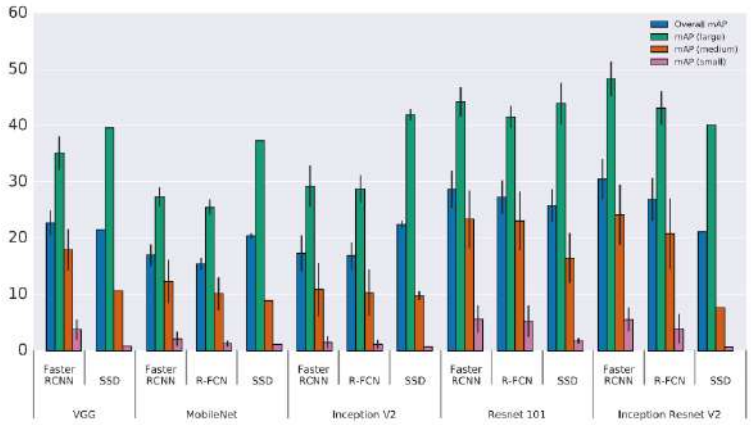


Рис. 2. Сравнение точности архитектур по детекторам признаков [16]
Fig. 2. Comparison of the accuracy of architectures by feature detectors [16]

На основе работ [1-3] в качестве модели был взят FasterRCNN с опорной сетью ResNet101. Согласно рис. 2, архитектура ResNet101 имеет высокую точность детекции больших объектов mAP^{large}, которых в наших данных большинство, FasterRCNN опережает большинство других моделей по точности сегментации на COCO данных (табл. 2).

Согласно работе [3], FasterRCNN достигает state-of-the-art качества обнаружения таблиц на изображениях.

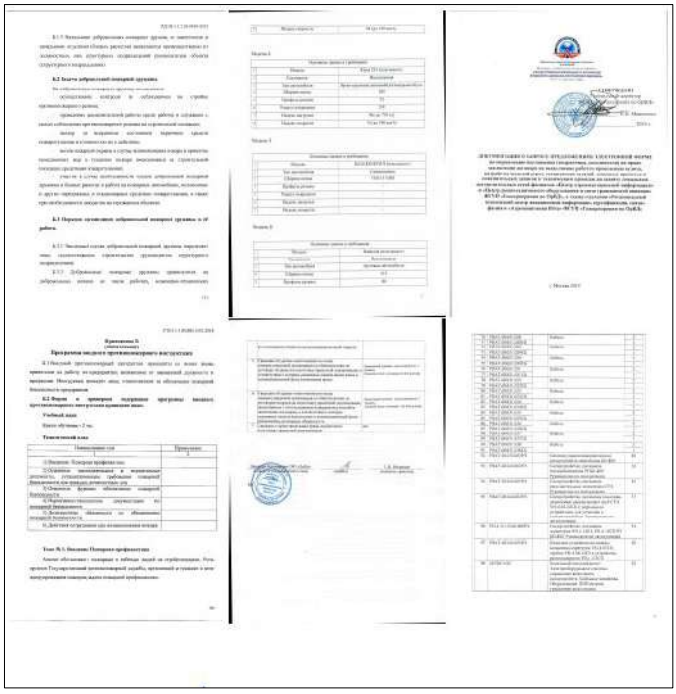


Рис. 3. Примеры изображений реальных документов
Fig. 3. Examples of images of real documents

3. Входные данные

Входные данные состоят из документов технических заданий и нормативно-правовых актов. Рассматриваемые документы находятся в открытом доступе и доступны по ресурсу [18]. Размеченный датасет изображений сканированных документов из [18] выложен и доступен по ссылке [19]. Сканированные документы, как правило, характеризуются высоким качеством символов, белым фоном, низким уровнем шума, манхэттенским стилем оформления, одноколоночностью. На рис. 3 представлены примеры входных данных.

3.1 Описание классов сегментирования

В рассматриваемых выше документах мы выделяем 3 класса объектов на изображениях:

- Text – текстовые блоки, содержащие однородный текст с единым форматированием (размером, жирностью, шрифтом, отступами между строк);
- Table – класс таблиц с границами, которые могут содержать объединенные ячейки по вертикали или по горизонтали. Заголовок таблиц может отличаться от тела другим форматированием;
- Picture – класс, содержащий печати, подписи, изображения в документах.

Выделение в отдельные классы элементов *Список, Заголовок* лучше производить на более низком уровне с использованием регулярных выражений и семантики.

4. Генерация документов

Для обучения модели сегментации необходимо иметь большой набор реальных данных, разметка которых, как было сказано ранее – процесс дорогостоящий. Поэтому было принято решение создать набор из искусственных данных путём самостоятельной генерации. Для этого реализован модуль генерации, позволяющий создавать случайные документы схожими визуально с реальными по текстовым шрифтам, межстрочным интервалам, жирности и стилям текста (рис. 4). Модуль предоставляет возможность извлечения ограничивающих прямоугольников для всех объектов извлекаемой логической структуры. Для создания документов выбрана стратегия случайного создания различных блоков и добавления их на страницу с последующим изменением форматирования. Сгенерированные блоки формируют единый docx документ, используемый впоследствии для создания PDF версии для упрощения извлечения изображений страниц.

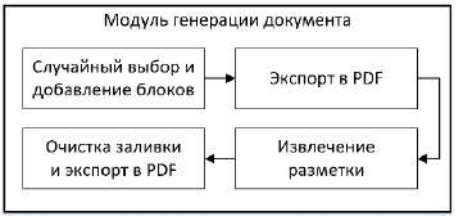


Рис. 4. Алгоритм работы модуля генерации
Fig. 4. Algorithm of the generation module

4.1 Особенности создаваемых документов

Для максимальной близости генерируемых документов к предметной области модуль генерации использует следующие методы:

1. создаваемые документы одноколоночны;
2. используются различные семейства шрифтов;
3. для заголовков используются жирные шрифты и/или шрифты большего размера;
4. таблицы имеют все границы, а также объединённые ячейки;
5. создаются многостраничные таблицы с одинаковыми заголовками;

6. колонтитулы и нумерация страниц выравнивается случайным образом;
7. межстрочные интервалы и отступы имеют высокую вариативность.

4.2 Генерация текстовых блоков

Для генерации текстовой составляющей программный модуль содержит в себе текстовые фрагменты в виде текстовых файлов, разбитых по количеству слов в предложении, а также по типу блока – в обычном тексте, в списке или таблице. Для получения очередного предложения модуль случайным образом выбирает файл и строку в нём, а затем добавляет её содержимое в создаваемый документ.

4.3 Генерация списков

Алгоритм генерации списков очень схож с созданием обычных текстовых блоков, однако для списков ещё выбираются тип списка (маркированный или нумерованный) и количество пунктов. В качестве содержимого элемента списка выбираются строки файла, содержащего элементы различных списков.

4.4 Генерация таблиц

Модуль генерации выбирает случайным образом количество строк и столбцов и заполняет ячейки. Текстовое содержимое ячеек извлекается из файлов с текстовой информацией ячеек таблиц реальных данных. После заполнения ячеек содержимым, алгоритм выбирает случайное количество ячеек и тип объединения (горизонтальное или вертикальное) и объединяет их. Для некоторых ячеек случайным образом изменяется направление текста и его форматирование.

4.5 Генерация других блоков

Помимо обычного текста, списка и таблиц, модуль генерации добавляет в документы верхний и/или нижний колонтитул, а также нумерацию страниц в случайном месте (слева/по центру/справа, снизу/сверху).

4.6 Получение координат и классов блоков сегментации

Из-за отсутствия в docx информации о местоположении объектов на странице документов прямое получение разметки невозможно. Поэтому для каждого типа блока выбираются уникальные контрастные цвета, которыми они заливаются во время генерации.

По завершении генерации документ экспортируется в PDF формат. Затем заливка блоков удаляется и документ также экспортируется в PDF. Очищенные документы конвертируются в набор изображений страниц и используются в качестве входных обучающих данных для моделей сегментации.

Для получения разметки необходимо выделить контуры созданных блоков и получить их координаты. Для этого модуль извлечения разметки выполняет следующие действия:

- получает изображение с залитыми блоками;
- применяет маски, выполняющие фильтрацию изображения для определённого диапазона цветов;
- выделяет контуры с полученных изображений;
- находит пустое место средствами OpenCV [20] и уменьшает контуры с последующей нормализацией координат;
- добавляет полученные границы в список разметки.

После обработки всех изображений разметка сохраняется в формате COCO [17].

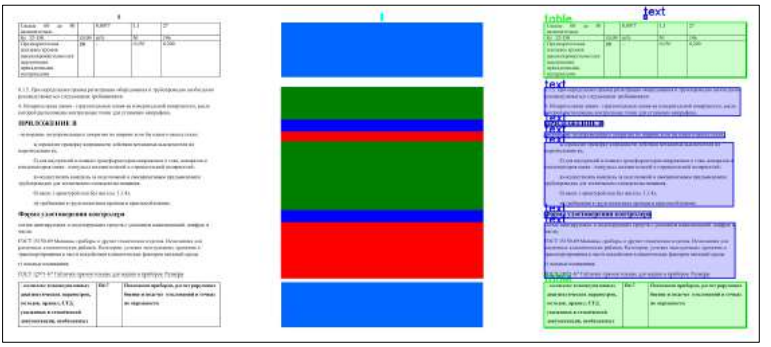


Рис. 5. Примеры сгенерированной страницы, её залитой версии и извлечённые контуры разметки
Fig. 5. Examples of the generated page, its flooded version and the extracted markup contours

5. Обучение модели

Мы обучали модель FasterRCNN [1] с помощью фреймворка TensorFlow Object Detection [21]. Веса и конфигурационные файлы выложены по ресурсу [19]. Тренировочный процесс состоял из трех этапов.

- Модель с претренированными весами на COCO датасете [17] тренировалась на 125 тысячах данных PubLayNet в течение 20 эпох [22];
- Модель доучивалась 4 эпохи на данных мощностью в 18 тысяч изображений, полученных с помощью модуля генерации;
- Далее модель обучалась на множестве реальных данных. В рамках эксперимента мы обучали на малом и большем наборах реальных данных (мощностью в 100 и 500 экземплярах соответственно).

Обучение модели на каждом этапе производилось со следующими параметрами:

- входные изображения формата A4 приводились к размеру 800 на 600 пикселей;
- значения learning rate устанавливались 0.001;0.0001;0.00001 начиная с 0, 1, 900 000-го шага соответственно;
- использовался SGD (Stochastic Gradient Descent) алгоритм с коэффициентом момента 0.9 и градиентным отсечением (gradient clipping) 10.0.

6. Постобработка

В результате предсказания обученной модели и невнимательной разметки данных могут образовываться артефакты, негативно влияющие на качество дальнейшего распознавания каждого сегментируемого блока. В качестве артефактов можно выделить следующее:

- обрезка текстового блока;
- обрезка таблиц;
- наложение блоков друг на друга (приводит к дублированию информации);
- избыточный захват фоновых пикселей по границам выделенного блока.

Для устранения артефактов сегментирования в пайплайн был включен модуль постобработки, включающий следующие этапы.

- На первом этапе постобработки, мы используем изменённый алгоритм None-Maximum-Supression (NMS) [23]. Мы хотим максимально сохранять текстовую информацию на изображениях, поскольку ее потеря критична для распознавания структуры документа в целом. С этой целью мы изменили классический подход NMS. Мы объединяем дубли с меньшей уверенностью для каждого объекта в один большой, вместо их удаления. В качестве дублей выступают объекты одного класса, пересечение которых Intersection over Union (IoU) превышает задаваемого порога $IoU > \delta$. Мы объединяем дубли,

пересечение которых IoU превосходит $\delta = 0.4$, в один объект и вычисляем score для нового объекта как средневзвешенное между старыми дублями.

- На втором этапе, изображение бинаризируется и инвертируется.
- На третьем этапе мы расширяем границы сегментируемых объектов на предварительно бинаризованном изображении, анализируя объекты от большей уверенности к меньшей. Объекты расширяются до тех пор, пока не пересекут соседние объекты или не превысят порога расширения. Порог расширения по горизонтали/вертикали задается как (ширина изображения / 30) и (высота изображения / 30) соответственно. Данный этап является подготовительным для следующего.
- На четвертом этапе границы сегментируемых объектов сужаются (уменьшается размер объектов). Сужение каждой границы к центру объекта проводится до пересечении с текстовым содержимым, со значениями пикселей > 0.
- На последнем этапе, устраняются пересекающиеся текстовые объекты, которые не были объединены на первом этапе.

а) б) в)

Рис. 6. Результаты работы постобработки. В первой строке расположены результаты предсказания модели до постобработки, во второй с постобработкой

Fig. 6. Results of post-processing work. The first line contains the results of model prediction before post-processing, the second with post-processing

Результаты работы постобработки представлены на рис. 6. На изображении (а) видно, без применения постобработки теряется верхняя строка основного текстового блока, на точности сегментации это бы не сильно отразилось, в отличии от точности распознавания текстовой информации на странице. На изображении (б) сегментатор обрезает часть второй таблицы без использования постобработки. На изображении (в) сегментатор дополнительно выделяет блок с текстом «7.4», и 2 раза подпись снизу. При применении 1-го шага постобработки, мы устранили бы дублирующую информацию. Вдобавок постобработка устраняет избыточный захват фоновых пикселей на краях выделенных объектов.

7. Результаты

Был подготовлен тестовый набор реальных данных мощностью 278 изображений, на котором проводились замеры качества моделей: **PLN** – набор данных научных статей PupLayNet, 125 тысяч изображений [22], **GEN** – набор сгенерированных данных, 18 тысяч изображений, **NPA-small** – набор реальных данных, 100 изображений, **NPA-big** – набор реальных данных 500 изображений [19].

Табл. 3. Количество объектов каждого класса в разных наборах данных
Table 3. The number of objects of each class in different datasets

	Текст	**Таблица**	**Картинка**	**Количество изображений**
GEN	104514	12413	0	18 000
NPA-small	474	53	31	100
NPA-big	2146	196	96	500
NPA test	1379	74	6	278

В табл. 4, 6 представлены результаты замеров качества модели FasterRCNN, обученной на разных наборах данных. В качестве метрик оценки качества использовались average precision (AP) $IoU = 0.5$ (PASCAL VOC метрика [24]) и mean average recall (MAR) со значениями порогов IoU $[0.5; 0.95]$ с учетом large объектов, вычисленные с помощью coco api [17].

Табл. 4. Результаты сегментации (F_1-мера)
Table 4, Results of layout analysis

	Текст	**Таблица**	**Картинка**	**Итого**
PLN+GEN+NPA-big	**0.810**	**0.937**	**0.696**	**0.820**
PLN+GEN+NPA-small	0.502	0.824	0.336	0.559
PLN+NPA-small	0.489	0.846	0.346	0.565
PLN	0.045	0.065	0.004	0.039

В табл. 5, 7 отражена точность сегментирования с этапом постобработки. Измерения качества проводились на тестовом наборе с постобработкой.

Табл. 5. Результаты сегментации (F_1-мера) с постобработкой
Table 5, Results of layout analysis with post-processing

	Текст	**Таблица**	**Картинка**	**Итого**
PLN+GEN+NPA-big	0.810	0.937	0.696	0.820
PLN+GEN+NPA-big + Post	**0.840**	**0.968**	**0.755**	**0.855**
PLN+GEN+NPA-small	0.502	0.824	0.336	0.559
PLN+GEN+NPA-small + Post	0.652	0.888	0.448	0.663

Из табл. 4 видно, что генерация данных позволила повысить качество распознавания только класса «Текст» (строка 2) на 0.04, с применением постобработки результат заметно лучше – изменение на 0.15 (табл. 5). Качество сегментации класса «Картинка» не улучшилось по причине его отсутствия в генерированных данных GEN. Качество сегментации класса «Таблица» ухудшилось по причине переобучения на генерированных данных (в наборе данных GEN 12413 таблиц, в NPA-small – 53 таблицы) согласно табл. 3. Дообучение на большем наборе реальных данных из 500 изображений позволило достигнуть существенного увеличения качества по всем классам (строка 1) табл. 4. Модель, дообученная на малом наборе данных, показывает высокие результаты только с постобработкой согласно строкам 3-4 табл. 5. Удовлетворительное качество класса «Картинка» во 2-3 строках табл. 4 обусловлено его отсутствием в сгенерированном наборе данных GEN и малым количеством в наборе NPA-small.

В результате выбор синтетических данных оправдался для класса «Текст». Сегментатор переобучился на синтетических данных класса «Таблица», по этой причине точность сегментации этого класса возрасла только на наборе NPA-big.

Таким образом, можно достигнуть высоких результатов сегментирования на новых наборах данных NPA-small, NPA-big в условиях их ограниченности с использованием генерации дополнительных данных и применения методов постобработки.

Табл. 6. Результаты сегментации (точность / полнота)
Table 6. Results of layout analysis (precision / recall)

	Текст	**Таблица**	**Картинка**	**Итого**
PLN+GEN+NPA-big	0.941 0.711	0.968 0.907	0.899 0.568	0.936 0.729
PLN+GEN+NPA-small	0.609 0.427	0.915 0.749	0.509 0.251	0.678 0.476
PLN+NPA-small	0.570 0.428	0.926 0.778	0.502 0.264	0.666 0.490
PLN	0.029 0.099	0.053 0.085	0.002 0.017	0.028 0.067

Табл. 7. Результаты сегментации (точность / полнота) с постобработкой
Table 7. Results of layout analysis (precision / recall) with post-processing

	Текст	**Таблица**	**Картинка**	**Итого**
PLN+GEN+NPA-big	0.941 0.711	0.968 0.907	0.899 0.568	0.936 0.729
PLN+GEN+NPA-big + Post	0.886 0.798	0.968 0.969	0.811 0.706	0.888 0.824
PLN+GEN+NPA-small	0.609 0.427	0.915 0.749	0.509 0.251	0.678 0.476
PLN+GEN+NPA-small + Post	0.708 0.604	0.912 0.866	0.490 0.412	0.678 0.627

8. Заключение

В работе была выбрана модель сегментации и проведена ее тонкая настройка в условиях ограниченного объёма реальных данных. Несмотря на ограниченность набора данных, нам удалось достичь высоких результатов сегментации с постобработкой в 0.663 и 0.855 f-меры для 100 и 500 тренировочных экземпляров соответственно. Это достигается за счет генерации

искусственных данных с распределением схожим с реальными и постобработкой для устранения артефактов после сегментации изображения. Обученные веса модели и наборы данных выложены и публично доступны [19].

Исходя из полученных результатов, можно утверждать, что использование синтетических данных для дообучения моделей сегментации на малом количестве реальных данных и использование методов постобработки позволяет достичь высокой точности и не требует ручной разметки большого количества данных.

Список литературы / References

[1] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, issue 6, 2017, pp. 1137-1149.

[2] G.M. Binmakhashen and S.A. Mahmoud. Document layout analysis: a comprehensive survey. ACM Computing Surveys (CSUR), vol. 52, issue 6, 2019, pp. 1-36.

[3] X. Zhong, J. Tang, and A.J. Yepes. Publaynet: largest dataset ever for document layout analysis. arXiv:1908.07836, 2019.

[4] K. Chen, M. Seuret, J. Hennebert, and R. Ingold. Convolutional neural networks for page segmentation of historical document images. In Proc. of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, 2017, pp. 965-970

[5] C. Wick and F. Puppe. Fully convolutional neural networks for page segmentation of historical document images. In Proc. of the 13th IAPR International Workshop on Document Analysis Systems (DAS), 2018, pp. 287-292.

[6] Pubmed. national library of medicine. URL: https://pubmed.ncbi.nlm.nih.gov. Accessed: 2020-21-07.

[7] G. Csurka. Domain adaptation for visual applications: a comprehensive survey. arXiv:1702.05374, 2017.

[8] М.А. Рындин, Д.Ю Турдаков. Проактивная разметка примеров для адаптации к домену. Труды ИСП РАН, том 31, вып. 5, 2019 г., стр. 145-152. DOI: 10.15514/ISPRAS-2019-31(5)-11 / M.A. Ryndin, D.Y. Turdakov. Domain adaptation by proactive labeling. Trudy ISP RAN/Proc. ISP RAS, vol.31, issue 5, 2019, pp. 145-152 (in Russian).

[9] C. R. De Souza, A. Gaidon, Y. Cabon, and A. M. López. Procedural Generation of Videos to Train Deep Action Recognition Networks. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2594-2604.

[10] L. Angeline, K. Teo, and F. Wong. Smearing algorithm for vehicle parking management system. In Proc. of the 2nd Seminar on Engineering and Information Technology, 2009, pp. 331-337.

[11] J. Ha, R. M. Haralick, and I. T. Phillips. Recursive xy cut using bounding boxes of connected components. In Proc. of the 3rd International Conference on Document Analysis and Recognition, vol. 2, 1995, pp. 952—955.

[12] L. O'Gorman. The document spectrum for page layout analysis. IEEE Transactions on pattern analysis and machine intelligence, vol. 15, issue 11, 1993, pp. 1162-1173.

[13] T.M. Breuel. An algorithm for finding maximal whitespace rectangles at arbitrary orientations for document layout analysis. In Proc. of the Seventh International Conference on Document Analysis and Recognition, 2003, pp. 66—70.

[14] I. Kavasidis, C. Pino, S. Palazzo, F. Rundo, D. Giordano, P. Messina, and C. Spampinato. A saliency-based convolutional neural network for table and chart detection in digitized documents. Lecture Notes in Computer Science, vol. 11752, 2019, pp. 292—302.

[15] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed. Deepdesrt: deep learning for detection and structure recognition of tables in document images. In Proc. of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, 2017, pp. 1162—1167.

[16] Object detection: speed and accuracy comparison (faster r-cnn, r-fcn, ssd, fpn, retinanet and yolov3). URL: https://medium.com/@jonathan_hui/object-detectionspeed-and-accuracy-comparison-faster-r-cnn-r-fcnssd-and-yolo-5425656ae359. Accessed: 2020-18-07.

[17] Coco, common objects in context. URL: https://cocodataset.org/#home. Accessed: 2020-18-07.

[18] Единая информационная система в сфере закупок, ЕИС. URL: https://zakupki.gov.ru/ (дата обращения 27.05.2020) / Unified information system in the field of procurement, EIS. URL: https://zakupki.gov.ru/ (in Russian).

[19] Dla-dataset. EIS. URL: https://disk.yandex.ru/d/XVjQf20BVsElKA (accessed: 2020-18-07).

[20] Open source computer vision library. URL: https://opencv.org (accessed: 2020-18-07).
[21] Tensorflow object detection api. URL: https://github.com/tensorflow/models/tree/master/research/object_detection. Accessed: 2020-18-07.
[22] Publaynet dataset. URL: https://github.com/ibm-aurnlp/PubLayNet/tree/master/pre-trained-models. Accessed: 2020-18-07.
[23] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-nms−improving object detection with one line of code. In Proc. of the IEEE International Conference on Computer Vision, 2017, pp. 5561-5569.
[24] Pascal voc evaluation. URL: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/htmldoc/devkit_doc.html#SECTION00064000000000000000 (accesed: 08.09.2020).

Информация об авторах / Information about authors

Оксана Владимировна БЕЛЯЕВА – аспирантка. Научные интересы: распознавание структуры документов, цифровая обработка изображений, нейросетевая обработка данных, распознавание образов.

Oksana Vladimirovna BELYAEVA – a PhD Student. Research interests: document layout analysis, digital image processing, neural network data processing, image pattern recognition.

Андрей Игоревич ПЕРМИНОВ является студентом магистратуры кафедры системного программирования. Научные интересы: цифровая обработка сигналов, нейросетевая обработка данных, создание искусственных данных, цифровая обработка изображений.

Andrey Igorevich PERMINOV – master student of the Department of System Programming. Research interests: digital signal processing, neural network data processing, generation of artificial data, digital image processing.

Илья Сергеевич КОЗЛОВ является стажером-исследователем. Научные интересы: распознавание структуры документов, цифровая обработка изображений, нейросетевая обработка данных, распознавание образов.

Ilya Sergeevich KOZLOV – researcher at ISP RAN. Research interests: document layout analysis, digital image processing, neural network data processing, image pattern recognition.

DOI: 10.15514/ISPRAS–2020–32(4)–15

Использование доменно-состязательного обучения для распознавания текстовых капч

[1] *Кущук Д.О., ORCID: 0000-0002-8778-8608 <dkuschuk@ispras.ru>*
[2] *Рындин М.А., ORCID: 0000-0002-7504-3975 <mxrynd@ispras.ru>*
[2] *Яцков А.К., ORCID: 0000-0002-1312-1675 <yatskov@ispras.ru>*
[2] *Варламов М.И., ORCID: 0000-0002-1083-6210 <varlamov@ispras.ru>*

[1] *Московский физико-технический институт*
141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9
[2] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Несмотря на появление более продвинутых вариантов публичных тестов Тьюринга, в настоящее время текстовая капча является достаточно распространённой, поэтому создание методов ее автоматического решения актуальны и сегодня. Современные алгоритмы успешно справляются с этой задачей, однако, обладают рядом ограничений, таких как: неспособность работать с изменяющейся длиной текста на изображении, медленное и сложное обучение. В данной работе представлен алгоритм атак на текстовые капчи, не требующий априорного знания длины текста на изображении. Экспериментально показано, что использование данного алгоритма совместно с методом состязательного обучения позволяет добиваться высокого качества на реальных данных, используя 200-500 размеченных примеров для обучения. Экспериментальное сравнение разработанного метода с современными аналогами показало, что при использовании одинакового числа реальных примеров для обучения наш алгоритм показывает сравнимое или более высокое качество, при этом он имеет более высокую скорость работы и обучения.

Ключевые слова: машинное обучение; решение капчи; OCR; состязательное обучение

Для цитирования: Кущук Д.О., Рындин М.А., Яцков А.К., Варламов М.И. Использование доменно-состязательного обучения для распознавания текстовых капч. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 203–216. DOI: 10.15514/ISPRAS–2020–32(4)–15

Using Domain Adversarial Learning for Text Captchas Recognition

[1] *Kushchuk D.O., ORCID: 0000-0002-8778-8608 <dkuschuk@ispras.ru>*
[2] *Ryndin M.A., ORCID: 0000-0002-7504-3975 <mxrynd@ispras.ru>*
[2] *Yatskov A.K., ORCID: 0000-0002-1312-1675 <yatskov@ispras.ru>*
[2] *Varlamov M.I., ORCID: 0000-0002-1083-6210 <varlamov@ispras.ru>*

[1] *Moscow Institute of Physics and Technology*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russian Federation
[1] *Ivannikov Institute for System Programming of the RAS,*
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia

Abstract. Nowadays the problem of legal regulation of automatic collection of information from sites is being actively solved. This means that interest in tools and programs for automatic data collection is growing and that's why interest in automatic programs for solving CAPTCHA («Completely Automated Public Turing test to tell Computers and Humans Apart») is increasing too. In spite of creation of more advanced types of captcha, nowadays text captcha is quite common. For instance, such large services as Yandex, Google, Wikipedia, VK

continue to use them. There are many methods of breaking text captchas in literature, however, it should be noted that most of them have a limitation to priori know the length of the text on the image, which is not always the case in the real world. Also, many methods are multi-stage, which brings additional inconvenience to their implementation and application. Moreover, some methods use a large number of labeled pictures for training, but in reality, to collect data one has to be able to solve captchas from different sites. Respectively, manually labeling dozens of thousands of examples for training for each new type of captcha is an unrealistically difficult action. In this paper we propose a one-step algorithm of attack on text captchas. This approach does not require a priori knowledge of the text's length on the image. It has been shown experimentally that the usage of this algorithm in conjunction with the adversarial learning method allows one to achieve high quality on real data, using the low number (200-500) of labeled examples for training. An experimental comparison of the developed method with modern analogs showed that using the same number of real examples for training, our algorithm shows a comparable or higher quality, while it has a higher speed of working and training.

Keywords: machine learning; captcha solving; OCR; adversarial learning

For citation: Kushchuk D.O., Ryndin M.A., Yatskov A.K., Varlamov M.I. Using Domain Adversarial Learning for Text Captchas Recognition. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 203–216 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–15

1. Введение

На сегодняшний день в мире происходит процесс правового урегулирования автоматического сбора информации с сайтов, а значит интерес к автоматическим программам решения капчи возрастает. Капча (Captcha) – это аббревиатура от фразы «Completely Automated Public Turing test to tell Computers and Humans Apart» (Полностью автоматизированный публичный тест Тьюринга, который отличает людей от компьютеров).

Благодаря автоматическому сбору информации исследователи и аналитики получают доступ к большим объемам актуальных данных для работы. Так как решение капчи вносит дополнительную латентность в процесс сбора, владельцы сайтов продолжают иметь дополнительный слой защиты от атаки на свои сервисы.

Так, например, в статье [1] во время краулинга сайта Яндекс.Новости, из-за большого количества запросов к серверу, авторы сталкивались с блокировкой сбора информации и были вынуждены распознавать капчу самостоятельно. Это создавало трудности для свободного сбора информации.

Несмотря на появление более продвинутых вариантов капчи, на сегодняшний день текстовая капча является достаточно распространённой. Например, такие крупные сервисы как Яндекс, Google, Wikipedia, Вконтакте продолжают использовать именно их (см. рис. 2). Текстовые капчи легки в разработке и привычны для пользователей. Поэтому задача развития методов распознавания и совершенствования текстовой капчи всё еще актуальна.

На данный момент существует множество методов распознавания текстовой капчи. Например, в статье [2] используется свёрточная сеть VGG, а в статье [3] используется предварительная сегментация картинки. Эти алгоритмы являются алгоритмами обучения с учителем и требуют большого объема размеченных обучающих данных, в то время как методы [4-6] основаны на генеративно-состязательном подходе и могут быть отнесены к методам обучения с частичным привлечением учителя. Данные алгоритмы состоят из двух основных фаз: в первой фазе происходит генерация примеров для обучения с использованием большого количества неразмеченных данных и, возможно, небольшого количества размеченных примеров. Во второй фазе осуществляется сам процесс обучения на полученном наборе данных. Этот подход позволяет значительно сократить количество примеров для обучения, что в задаче распознавания капч необходимо, ведь собирать и размечать реальные данные с сайтов достаточно трудоемкий процесс. Лучший в этой области метод представлен в статье [7], он достигает высокой точности на многих типах капч с различных сайтов. Однако, данный метод умеет работать только с капчами фиксированной длины.

Недавняя статья [8], в которой авторы используют идеи обучения представлениями, обходит ограничение априорного знания длины текста, но показывает более низкое качество распознавания. Метод доменно-состязательного обучения, в котором к основной модели добавляется дискриминатор [9], помогает устранить многофазность предыдущего подхода. Но в данной статье авторы используют основную архитектуру, способную к разгадыванию капч только с фиксированной длиной.

В данной работе мы представляем новый метод, совмещающий названные выше подходы и обладающий такими достоинствами как: простота архитектуры, точность, высокая скорость обучения и работы, требование малого числа примеров для обучения. Но главное преимущество – возможность работы алгоритма на капчах с произвольной длиной текста.

Во втором разделе данной работы будет приведен обзор современных методов. В первой части разд. 3 будет описан алгоритм распознавания текста на изображении. Во второй части представлен алгоритм DA-CRNN, основной идеей которого является использование состязательного обучения для снижения числа примеров, необходимого для обучения. В разд. 4 будет проведено экспериментальное тестирование разработанного метода.

2. Обзор существующих решений

2.1 Свёрточные нейронные сети

CNN (сверточная нейронная сеть) -- тип нейросетей, используемый для анализа изображений. Примеров сверточных архитектур множество – VGG [10], ResNet [11], LeNet-5 [12]. Логичным является применение сверточных архитектур для расшифровки шумных изображений. Авторы статьи [2] так и поступили, и использовали архитектуру, похожую на архитектуру VGG, для распознавания текстовой капчи.

Данный метод имеет ряд ограничений. Во-первых, длина текста на картинке должна быть строго фиксированной, для каждого вида капчи обучается своя модель. Например, в данной статье капча состоит из 5 символов. Во-вторых, все обучающие примеры должны быть размечены, авторы используют генератор капч BotDetect captcha [15] для генерации 100000 картинок для обучения и 1000 картинок для тестирования. В реальности, нужно уметь разгадывать капчи со сторонних сайтов, соответственно, вручную разметить 100000 примеров для обучения для каждого нового типа капчи является трудновыполнимым действием.

Похожие методы используются в и в других статьях с некоторыми дополнениями и особенностями. Например в [14] используется активное обучение CNN, то есть в процессе обучения модели она сама выбирает примеры, на которых продолжать обучение, и эксперты их размечают.

2.2 Сегментация и распознавание

Другая статья [3] иллюстрирует иной метод распознавания текстовой капчи. Идея состоит в том, чтобы сначала производить сегментацию картинки, а потом на основе этой сегментации предсказывать отдельные символы.

Весь алгоритм состоит из 4 частей – сегментации, слайсера, маркера и судьи. Сначала производится множественная сегментация, то есть картинка разделяется на части, отделяя каждый символ друг от друга, причем таких вариантов разделения несколько. После этого с помощью слайсера выбираются сегменты и строится граф, далее граф анализируется маркером и присваивается значение каждому сегменту, после чего судья выбирает самый вероятный ответ.

Так как задача множественной сегментации картинки (первые два слоя) в данном случае решается без учителя, то целесообразно использовать метод обучения с подкреплением. При

правильной и удачной сегментации, модель не спрашивает ничего человека, а при неправильной эксперту задается вопрос, в каком шаге была произведена ошибка: в сегментации или в распознавании? Именно так и происходит обучение модели.

2.3 Генеративные методы

В 2014 году была опубликована статья [15], в которой авторы представили генеративно-состязательные нейронные сети, в которых происходит обучение двух частей - дискриминатора и генератора. Такие сети в основном используются для генерации синтетических изображений, очень похожих на реальные.

В 2017 году опубликовалась статья [4], которая использует этот подход для распознавания текстовой капчи. В ней сначала генерировались искусственные картинки с заранее известным текстом, похожие на настоящие капчи. Далее на них обучалась модель распознавания текста на изображении. После чего получившуюся модель использовали для предсказания текста на реальных капчах. Тем самым исчезает необходимость собирать и вручную размечать большое количество реальных картинок для последующего обучения модели на них, ведь теперь на основе реальных неразмеченных изображений генерируются похожие на них искусственные размеченные изображения, которые и используются впоследствии для обучения.

Модель, которая использовалась для предсказания, состоит из нескольких сверточных слоев, после которых используются рекуррентные слои.

Также в 2017 году был опубликован метод [5], который полностью основан на генеративно-состязательном методе и решает задачу распознавания автомобильных номеров. А метод [6] решает задачу анализа фотографий уличных вывесок и знаков путем адаптации размеченных синтетических изображений на реальные уличные вывески. Причем в этой работе авторы используют не только попиксельную адаптацию пространства представления, но также и всю геометрию фотографий, углов, ракурсов.

После этого был опубликован метод расшифровки капчи [7], также основанный на GAN (генеративно-состязательная сеть), который на данный момент является одним из самым эффективных в задаче распознавания капчи с частичным привлечением учителя.

Данный метод, предложенный Гуйсинь Е (Guixin Ye) в 2018 году, представляет из себя четырехступенчатый метод. На первом шаге происходит генерация синтетических картинок, максимально похожих на реальные капчи. На втором шаге реализуется предобработка картинки – удаление шумов и чистка фона. На третьем шаге происходит обучение классификатора на этих искусственно созданных и очищенных капчах. Четвертый шаг – шаг улучшения качества, посредством частичного дообучения на небольшом количестве реальных размеченных изображений (производится тонкая настройка модели).

Данный метод является многоступенчатым, что усложняет обучение и отладку: может быть непонятно, какой из шагов вносит наибольшее ухудшение в работу общего алгоритма. Также стоит отметить один ощутимый недостаток – длина текста на изображении строго фиксирована. Авторы сами указывают на этот недостаток и предлагают сначала распознавать длину капчи, а уже потом использовать описанный выше метод. Однако это затрудняет обучение генератора, т.к. реальные изображения вначале придётся разметить и разделить по длине, что усложняет обучение и ведет к дополнительной работе экспертов по разметке.

2.4 Обучение представлениями

Относительно недавно была опубликована статья [16], которая демонстрирует вариант обучения представлениями CPC. Обучение представлениями – это техника, которая позволяет модели обнаруживать и выявлять внутренние признаки в неразмеченных данных. Иными словами, мы выявляем признаки в режиме самообучения, то есть используя для обучения сами целевые данные без их меток. Конкретно в данной модели CPC

использовалась идея выявления признаков путем обучения модели предсказывать следующие элементы последовательности, зная предыдущие. Таким образом и происходит отбор целевых признаков.

Авторы статьи [8] спроецировали данный метод на задачу выявления целевых признаков из реальных картинок. Их модель, названная ``extractor", училась находить определенные сущности из входного изображения.

Далее к модели «extractor» добавлялся классификатор, и получившаяся модель с замороженными весами модели «extractor» тренировалась разгадывать обычные общедоступные размеченные синтетические капчи. Но, так как «extractor» изначально обучался выявлять признаки из целевых картинок, то модель добилась высокой точности классификации и на реальных изображениях.

Рассмотрены современные методы распознавания шумных картинок. Стоит отметить, что во многих способах существует ограничение в необходимости априорного знания длины текста на картинке, что в реальном мире не всегда так. Также многие методы являются многоступенчатыми, что привносит дополнительное неудобство в реализацию и применение методов. В следующем разделе мы попытаемся отойти от идеи многоступенчатости, а также устраним ограничение, касающееся фиксированной длины текста.

3. Построение решения задачи

3.1 Описание метода распознавания текста на изображении

В 2015 году вышла статья Баогуан Ши (Baoguang Shi), Сян Бай (Xiang Bai) и Конг Яо (Cong Yao) [17]. Основа их метода – нейронная сеть с архитектурой CRNN (Сверточная рекуррентная нейронная сеть). Архитектура такой нейронной сети состоит из трех частей - сверточных слоев, рекуррентных слоев и слоев транскрипции.

В начале сверточная часть извлекает последовательность признаков из каждого изображения. Далее эта последовательность признаков попадает в рекуррентные слои LSTM [18], после чего проходит слой softmax, где формируется предсказание для каждого элемента этой последовательности. После этого, эта предсказанная последовательность, образующаяся на выходе из рекуррентных слоев и слоя softmax, попадает в слой транскрипции, где преобразуется в последовательность готовых символов. Благодаря одноступенчатости этого метода, сеть тренируется, используя только одну общую функцию потерь. Для определения условной вероятности авторы используют CTC слой, описанный в [19]. Этот слой часто используется в задачах распознавания рукописного текста [20]. При этом для обучения необходимо знать только ответы к изображению, без уточнения, в какой части картинки находится тот или иной символ.

За основу нашего метода мы взяли рассмотренную архитектуру CRNN с параметрами, представленными в табл. 1.

Также при обучении в качестве оптимизирующего алгоритма использовался стохастический градиентный спуск [21] с изменяющимися в течение обучения параметрами. Параметр шага градиентного спуска (скорость обучения) μ изменялся на каждой эпохе по формулам:

$$\mu = \frac{0{,}01}{(1 + 10 \cdot p)^{0.75}}, \text{где } p = \frac{\text{номер эпохи}}{\text{общее количество эпох}}. \quad (1)$$

Табл. 1. Слои и их параметры, используемые в модели CRNN
Table 1. Layers and their parameters used in the CRNN model

Слои	**Параметры**
Input	size = (64, 256, 1)
Convolutional	64 kernels, size = (3,3)

MaxPooling	Window – (2,2), strides = (2,2)
Convolutional	256 kernels, size = (3,3)
Convolutional	256 kernels, size = (3,3)
MaxPooling	Window – (2,1), strides = (2,1)
Convolutional	512 kernels, size = (3,3)
Batch_norm	–
MaxPooling	Window – (2,1), strides = (2,1)
Lambda	squeeze
B-LSTM cell	128
B-LSTM cell	128
Dropout	0.5
Dense	128
Dense	64
Dropout	0.5
Softmax	Количество символов
CTC	–

3.2 Алгоритм DA-CRNN

Собирать большое количество реальных изображений и размечать вручную для обучения модели CRNN – очень дорогой и долгий процесс.

Заметим, что нашу задачу можно переформулировать как задачу адаптации к предметной области: мы хотим, имея модель, показывающую хорошие результаты на искусственно сгенерированных данных, адаптировать ее для распознавания реальных изображений, используя только картинки этих реальных изображений и не зная ответов. Поэтому логично исследовать применимость методов адаптации к предметной области к нашей задаче.

Ярослав Ганин и соавторы в 2016 году опубликовали статью [9], в которой описывают метод адаптации к предметной области с помощью доменно-состязательной нейронной сети.

Для того, чтобы классификатор мог предсказывать реальные изображения также хорошо, как и искусственные, необходимо, чтобы внутреннее представление сети не имело никакой дискриминационной информации о происхождении картинки (реальная или искусственная картинка) при этом сохраняя низкую ошибку классификации на размеченных искусственных примерах.

Суть метода представляет из себя добавление к классификатору дискриминатора, который будет различать реальные картинки от искусственных. Архитектура состоит из 3 частей – часть отбора признаков, классификатор, дискриминатор; пусть $\theta_f, \theta_y, \theta_d$, соответственно, весовые коэффициенты этих частей. L_y, L_d – ошибки классификатора и дискриминатора соответственно. Общая ошибка задается взвешенной суммой этих ошибок, она минимизируется.

Идея состоит в том, чтобы дискриминатор работал таким образом, что принимая на вход результат части отбора признаков, он не мог различить реальную картинку от синтетической, то есть полученные признаки были свойственны как реальным так и не реальным изображениям. Грубо говоря, реализуется правильный и нужный отбор общих признаков, не заточенный под особенности искусственной картинки. Этого можно добиться, если обновлять коэффициенты θ_f, руководствуясь не уменьшением ошибки L_d (ход по антиградиенту), а руководствуясь увеличением ошибки L_d (ход по градиенту). То есть слой отбора признаков препятствует дискриминатору определять истинную природу картинки.

Но, в свою очередь, веса θ_d обновляются для уменьшения L_d. Тем самым поддерживается адекватный уровень работы дискриминатора после выхода признаков из первой части, но также поддерживается стремление удалить из выхода первой части какие-либо признаки, свойственные происхождению картинки (реальная или синтетическая).

Одновременно с этим классификатор обучается на синтетических картинках и дает определенные полезные признаки на выходе части отбора признаков.

Веса обновляются таким образом:

$$\theta_f \leftarrow \theta_f - \mu\left(\frac{\partial L_y}{\partial \theta_f} - \lambda\frac{\partial L_d}{\partial \theta_f}\right), \tag{2}$$

$$\theta_y \leftarrow \theta_y - \mu\frac{\partial L_y}{\partial \theta_y}, \tag{3}$$

$$\theta_y \leftarrow \theta_y - \mu\lambda\frac{\partial L_d}{\partial \theta_d}, \tag{4}$$

μ – скорость обучения, λ – вес ошибки дискриминатора.

$L_{common} = L_y + \lambda L_d$ – это общая функция потерь, которую мы минимизируем.

Оптимизация функции потерь происходила с помощью метода стохастического градиентного спуска. Параметр скорости обучения μ изменялся на каждой эпохе в соответствии с формулой (1).

Гиперпараметр веса ошибки дискриминатора λ фиксируется в пределах 5-20 и не изменяется в процессе обучения. Точное значение параметра λ мы подбираем с помощью валидации, то есть обучаем несколько моделей с разными значениями λ и выбираем ту, которая показывает наивысшее качество на валидационной выборке.

Для реализации метода, используется дополнительный слой – слой обращения градиента. Этот слой умножает на отрицательное число градиент функции потерь дискриминатора по θ_f при обратном распространении ошибки, но не изменяет веса при прямом проходе. Это и обеспечивает ход по градиенту, а не по антиградиенту для весов слоя отбора признаков θ_f. Математически этот слой описывается так:

$$R(x) = x, \frac{dR(x)}{dx} = -\lambda_p I, \tag{6}$$

где I – единичная матрица, а λ_p изменяется от 0 до 1 в течении обучения по следующей формуле:

$$\lambda_p = \frac{2}{(1 + e^{-10\cdot p})} - 1, \text{где } p = \frac{\text{номер эпохи}}{\text{общее количество эпох}}. \tag{7}$$

Эта стратегия постепенного увеличения числа λ_p, во-первых, позволяет классификатору быть менее чувствительным к шумным данным на ранних этапах обучения, а во-вторых, не дает дискриминатору обучиться быстрее классификатора, то есть сохраняется нужный баланс.

Архитектура GAN, о которой мы говорили в подразделе 2.3 многоступенчатая; чтобы этого избежать, мы попробовали интегрировать архитектуру CRNN как основную модель в задачу адаптации к домену DANN (в дальнейшем будем называть получившуюся архитектуру DA-CRNN). Схема новой архитектуры приведена на рис. 1.

Параметры начальных сверточных слоев и верхней части новой модели DA-CRNN приведены в табл. 1, а отдельно параметры архитектуры дискриминатора приведены в табл. 2.

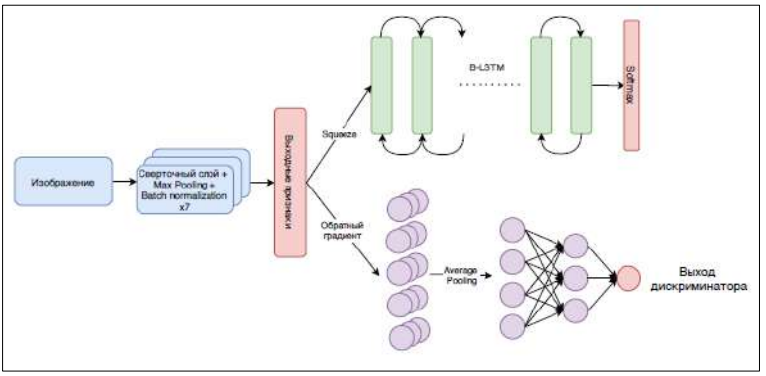


Рис. 1. Архитектура DA-CRNN
Fig. 1. DA-CRNN Architecture

Параметры начальных сверточных слоев и верхней части новой модели DA-CRNN приведены в табл. 1, а отдельно параметры архитектуры дискриминатора приведены в таблице табл. 2.

Табл. 2. Слои и их параметры, используемые в архитектуре дискриминатора
Table 2. Layers and their parameters used in the architecture of the discriminator

Слои	**Параметры**
Input	size = (32, 512)
Dense	64
Average Pooling	32
Lambda	squeeze
Dropout	0.5
Dense	16
Softmax	Ответ дискриминатора

4. Эксперименты

4.1 Набор данных

Датасет для обучения модели представляет из себя набор искусственно-сгенерированных размеченных капч в количестве 19 тысяч штук, а также набор реальных неразмеченных капч, собранных с сайтов Яндекс, Wikipedia и eBay, каждый набор в количестве 9000 штук, с помощью сборщика, написанного на языке Python с использованием библиотеки для автоматизации работы браузера Selenium WebDriver[1].

Длина текста на каждом наборе реальных данных разная, но всегда варьируется в районе 5-10 символов. Под символами могут пониматься английские и русские буквы, а также арабские цифры.

Примеры капч из Wikipedia, Яндекса, eBay изображены на рис. 2.

Рис. 2. Примеры реальных капч
Fig. 2. Examples of real captchas

[1] https://www.selenium.dev/

1500 тысячи капч из каждого набора реальных примеров были размечены вручную для обучения и тестирования модели. Для обучения использовалось 100-500 примеров, для тестирования всегда 1 тысяча.

Для генерации искусственных размеченных изображений в количестве 20 тысяч используется библиотека для генерации размеченных картинок Captcha[2]. Для каждого обучения на определенном наборе реальных данных использовались по возможности похожие по шрифту искусственно-сгенерированные размеченные изображения. Примеры сгенерированных капч изображены на рис. 3.

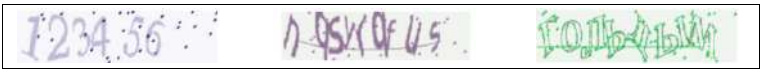


Рис. 3. Примеры синтетических капч
Fig,. 3. Examples of synthetic captchas

Валидационная выборка, использующаяся для тонкой настройки весов в методах CRNN и DA-CRNN, применяется для определения некоторых оптимальных гиперпараметров и для сохранения наилучшей модели с этими параметрами. Для модели CRNN это количество эпох при пост-обучении, а для DA-CRNN гиперпараметрами являются: количество эпох при непосредственном обучении, параметр λ – вес ошибки дискриминатора (см.подраздел 3.2), а также количество эпох пост-обучения.

4.2 Результаты модели CRNN на искусственных изображениях

Цель эксперимента состоит в том, чтобы выяснить применимость CRNN для разгадывания капчи, поэтому этот эксперимент проводился на синтетических капчах для упрощения получения размеченных примеров.

Также при обучении на синтетических данных была получена предобученная модель с полезными признаками для дальнейшего переноса знаний.

Для обучения моделей мы используем 19 тысяч сгенерированных размеченных шумных картинок, используя библиотеку для генерации капч Captcha (подробнее см. подраздел 4.1), то есть используем обучение с учителем. Далее модель тестируется на одной тысяче также искусственно сгенерированных изображений.

Результаты теста на искусственных сгенерированных данных после обучения модели приведено в табл. 3. Обучение и подсчеты производились с помощью видеокарты GeForce GTX 1080.

В роли метрики качества мы используем метрику «Достоверность», определенную как:

$$\text{Достоверность} = \frac{\text{множество правильно отгаданных тестовых изображений}}{\text{множество всех тестовых изображений}}. \quad (8)$$

Табл. 3. Результаты модели CRNN на синтетических данных
Table 3. Results of the CRNN model on synthetic data

	Время обучения, м	**Достоверность, %**	**Среднее время разгадывания 1 капчи, с**
CRNN	85	97	9,0016

Стоит отметить, что время отгадывания таких изображений человеком примерно 10-15 секунд, тем самым существует большой смысл использовать нейронные сети для решения задачи сбора информации из Интернета для уменьшения вносимой латентности.

[2] https://github.com/lepture/captcha

4.3 Результаты модели DA-CRNN на реальных данных

Проводится обучение новой построенной модели DA-CRNN на 19 тысячах сгенерированных капчах, при этом для адаптации мы используем 9 тысяч неразмеченных реальных картинок с сайтов. После этого, веса тонко настраиваются на 100-500 размеченных реальных изображениях. Полученная модель тестируется на 1000 реальных картинках (подробно о датасете можно прочитать в подразделе 4.1). Для чистоты эксперимента мы проводили по 5 независимых процессов обучения, в каждом из которых случайно выбирался поднабор тренировочных, валидационных и тестовых выборок. Валидационный набор считался используемым в обучении, то есть если используется n размеченных примеров для тонкой настройки, то 0.2\% от них используется для валидации (более подробно про валидацию см. в разделе 4.1). Результаты тестирования качества распознавания приведены на рис. 4.

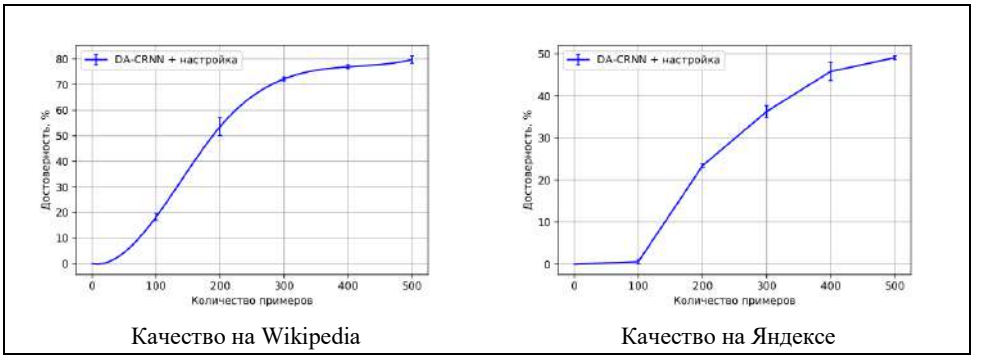


Рис. 4. Результаты модели DA-CRNN на капчах из Wikipedia, Яндекс и eBay
Fig. 4. Results of the DA-CRNN model on captchas from Wikipedia, Yandex and eBay

Заметим, что для достижения точности в 50% для капч из Wikipedia и eBay требуется всего 200 размеченных реальных примеров для обучения. А качество в 50\% показывают многие сторонние методы из литературы только на 500 размеченных реальных данных.

4.4 Сравнение с современными методами

Сравним теперь результаты распознавания нашей модели и самых эффективных на данный момент методов. Показатели достоверности, скорости работы и обучения взяты из соответствующих статей.

В методах [7-8] модели DA-CRNN тонко настраивались на 500 размеченных изображениях и тестировались на 1000 размеченных изображениях. Сравнение методов представлено в таблице 4. Стоит также отметить, что один из самых эффективных на данный момент методов на основе GAN [7] работает лишь с изображениями фиксированной длины, поэтому для теста на представленных сайтах авторами брался поднабор с фиксированной длиной, т.е. задача упрощалась. А вот метод [8] предполагает распознавание капч с произвольной длиной.

Табл. 4. Сравнение качества модели DA-CRNN с современными методами
Table 4. Comparison of the quality of the DA-RNN model with modern methods

Метод	**Wikipedia**	**Яндекс**	**eBay**
[22]	23.8%	–	58.8%
[23]	25%	–	43%
[3]	28%	–	51.39%
[24]	–	56.0%	–
[8]	66.5%	63.2%	**91.5%**
[7]	78%	–	85.6%
DA-CRNN	**79%**	**72.26%**	79.3%

Как мы видим, наша модель DA-CRNN превосходит множество ранее известных методов на некоторых наборах реальных данных и имеет качество, сравнимое с наилучшими на данный момент на остальных. Реализация метода DA-CRNN не требует априорного знания длины капчи и дополнительную предобработку изображения. Процесс обучения использует одну функцию потерь и имеет одноступенчатую архитектуру, тем самым снижается время обучения. В табл. 5 представлено сравнение времени обучения и тестирования методов, показавших наибольшую достоверность.

Табл. 5. Время обучения и время тестирования современных методов
Tab. 5. Training time and testing time of modern methods

Метод	**Ресурсы**	**Время на обучение, ч**	**Время разгадывания 1 капчи, мс**
[7]	4 x NVIDIA Tesla P40 GPU, 256GB of RAM	53	50
[8]	2 x NVIDIA Tesla P40 GPU	–	6
DA-CRNN	NVIDIA GeForce GTX 1080, 64GB of RAM	**12**[3]	**1,6**

Из табл. 5 видно, что наш метод DA-CRNN, имея одну видеокарту, затрачивает на обучение и тестирование гораздо меньше времени, чем остальные методы с более сильными ресурсами. Это происходит благодаря простоте модели и относительно небольшому числу примеров для обучения. Например, в статье [7]} используется 200000 сгенерированных картинок, а в методе [8] используется 800000, в нашем же методе всего 19000. Исходя из этого, можно предположить, что метод [8] обучается на порядок дольше метода DA-CRNN.

Беря во внимание быстроту нашей модели, можно объяснить более низкий процент качества DA-CRNN на наборе капч из сайта eBay. Модель показывает сравнимое качество распознавания на всех рассматриваемых наборах данных, при этом затрачивая наименьшее время на обучение и распознавание.

В итоге, мы можем наблюдать, что для достижения точности в 50\% (что дает нам математическое ожидание числа попыток до успешного обхода капчи, равное двум) нам требуется всего 200-500 размеченных реальных примеров, что делает нашу модель релевантной и практичной.

5. Заключение

В данной работе был представлен метод распознавания текста на изображениях, содержащих текстовую капчу с текстом произвольной длины. Было экспериментально показано, что метод, основанный на архитектуре нейронной сети CRNN и использующий подход к

[3] 12 часов требуется для полного цикла обучения подбором всех оптимальных гиперпараметров. Для быстрого получения не самой точной модели за одну итерацию обучения потребуется около 30 минут.

адаптации к предметной области DANN достигает качества распознавания, не уступающее другим эффективным методам.

Главные преимущества построенного метода в том, что он не требует априорного знания длины капчи и использует для обучения 200-500 реальных размеченных примеров. Также к достоинствам данного метода можно отнести то, что он состоит из одной нейросетевой модели и не требует серьезной предобработки изображения. Процесс настройки модели включает в себя два этапа -- обучения начального приближения и тонкой настройки на небольшом количестве реальных примеров, что позволяет уменьшить время обучения примерно в 5 раз по сравнению с современными методами.

Мы добились высокого качества распознавания новой модели DA-CRNN на нескольких наборах реальных изображений с разных сайтов. В будущем хотелось бы улучшить это качество, использовать еще меньше примеров для обучения для снижения времени обучения, а также провести дополнительный ряд экспериментов с наборами данных других поставщиков реальных капч с веб-ресурсов, таких как Вконтакте, Qihu360, cadocs.nsd.ru, Alipay, Baidu, Sina, Google. каждой строки можно определять её уровень вложенности по отношению к документу.

Список литературы / References

[1] А.К. Яцков, М.И. Варламов, Д.Ю. Турдаков. Сбор и извлечение данных с веб-сайтов СМИ. Программирование, том 44, №5, 2018 г., стр. 68-80 / A. K. Yatskov, M. I. Varlamov и D. Y. Turdakov. Extraction of data from mass media web sites. Programming and Computer Software, vol. 44, № 5, 2018, pp. 344–352.

[2] M. Kopp, M. Nikl, and M. Holena. Breaking captchas with convolutional neural networks. In Proc. of the 17th Conference on Information Technologies – Applications and Theory (ITAT 2017), 2017, pp. 93-99.

[3] E. Bursztein, J. Aigrain, and A. Moscicki. The end is nigh: generic solving of text-based captchas. In Proc. of the 8th USENIX Workshop on Offensive Technologies, 2014, 15 p.

[4] T. A. Le, A. G. Baydin, R. Zinkov и F. D. Wood. Using synthetic data to train neural networks is model-based reasoning. arXiv: 1703.00868, 2017.

[5] X. Wang, M. You, and C. Shen. Adversarial generation of training examples for vehicle license plate recognition. arXiv: 1707.03124, 2017.

[6] F. Zhan, C. Xue, and S. Lu. GA-DAN: geometry-aware domain adaptation network for scene text detection and recognition. arXiv: 1907.09653, 2019.

[7] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang. Yet another text captcha solver: A generative adversarial network based approach. In Proc. of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 332-348

[8] S. Tian and T. Xiong. A generic solver combining unsupervised learning and representation learning for breaking text-based captchas. In Proc. of the Web Conference, 2020, pp. 860-871.

[9] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V.S. Lempitsky. Domain-adversarial training of neural networks. Journal of Machine Learning Research, volo. 17, 2016, pp. 1-35.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv: 1512.03385, 2015.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, vol. 86, no. 11, 1998, pp. 2278—2324.

[13] Botdetect captcha generator [online]. URL: www.captcha.com, accessed 25.08.2020.

[14] F. Stark, C. Hazırba ， R. Triebel, and D. Cremers. Captcha recognition with active deep learning. In Proceedings of the German Conference on Pattern Recognition, Workshop on new challenges in neural computatio, 2015.

[15] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial networks. ArXiv: 1406.2661, 2014.

[16] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. arXiv: 1807.03748, 2018.

[17] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. arXiv: 1507.05717, 2015.
[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, vol. 9, no. 8, 1997, pp. 1735—1780.
[19] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proc. of the Twenty-Third International Conference on Machine Learning, 2006, pp. 369-376.
[20] T. Bluche, J. Louradour, and R. O. Messina. Scan, attend and read: end-to-end handwritten paragraph recognition with MDLSTM attention. arXiv: 1604.03286, 2016.
[21] S. Ruder. An overview of gradient descent optimization algorithms. arXiv: 1609.04747, 2016.
[22] H. Gao, J. Yan, F. Cao, Z. Zhang, L. Lei, M. Tang, P. Zhang, X. Zhou, X.Wang, and J. Li. A simple generic attack on text captchas. In Proc. of the Network and Distributed System Security Symposium, 2016, 14 p.
[23] E. Bursztein, M. Martin, and J. Mitchell. Text-based captcha strengths and weaknesses. In Proc. of the 18th ACM Conference on Computer and Communications Security, 2011, pp. 125-138.
[24] M. Tang, H. Gao, Y. Zhang, Y. Liu, P. Zhang, and P. Wang. Research on deep learning techniques in breaking text-based captchas and designing image-based captcha. IEEE Transactions on Information Forensics and Security, vol. 13, no.10, 2018, pp. 2522—2537.

Информация об авторах / Information about authors

Денис Олегович КУЩУК – студент магистратуры Физтех школы прикладной математики и информатики. Научные интересы: машинное обучение, состязательное обучение, оптическое распознавание символов.

Denis Olegovitch KUSHCHUK – Master's student at the Phystech School of Applied Mathematics and Informatics. Research Interests: Machine Learning, Adversarial Learning, Optical Character Recognition.

Максим Алексеевич РЫНДИН – аспирант. Научные интересы: методы адаптации к домену и переноса знаний, онлайн обучение, обработка текстов на естественном языке, векторное представление слов/предложений/текстов, генеративные модели, активное и проактивное обучение, анализ социальных сетей.

Maxim Alexeevitch RYNDIN – PhD Student. Research interests: domain adaptation, transfer learning, online learning, natural language processing, embeddings, generative models, active and proactive learning, social media analysis.

Александр Константинович ЯЦКОВ – аспирант. Научные интересы: сбор данных из веба, автоматизация процесса сбора данных, фокусированный сбор данных извлечение информации, машинное обучение.

Alexander Konstantinovitch YATSKOV – PhD Student. web crawling, web data extraction); focused crawling, information extraction, machine learning.

Максим Игоревич ВАРЛАМОВ – младший научный сотрудник. Научные интересы: автоматизация сбора данных из веб-ресурсов, автоматическое реферирование, семантическая близость понятий.

Maksim Igerevitch VARLAMOV – junior researcher. Research interests: automation of data collection from web resources, automatic summarization, semantic proximity of concepts.

DOI: 10.15514/ISPRAS–2020–32(4)–16

Разработка решателя iceFoam для моделирования процесса обледенения

[1] *К.Б. Кошелев, ORCID: 0000-0002-7124-3945 <koshelevkb@mail.ru>*
[2,3] *В.Г. Мельникова, ORCID: 0000-0002-1280-2647 <vg-melnikova@yandex.ru>*
[2] *С.В. Стрижак, ORCID: 0000-0001-5525-5180 <s.strijhak@ispras.ru>*

[1] *Институт водных и экологических проблем СО РАН,*
656038, Алтайский край, г. Барнаул, ул. Молодежная, д.1
[2] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[3] *МГТУ им. Н.Э. Баумана,*
105005, Россия, г. Москва, 2-ая Бауманская ул., д.5., стр.1

Аннотация. В настоящее время в РФ активно ведется освоение северных территорий. Вопросы изучения физических процессов обледенения являются актуальными, так как климатические условия оказывают влияние на поверхность исследуемых объектов (линии электропередач, жилые строения, энергетические установки, летательные аппараты), на безопасность людей и экологию. В облаках возможно появление и движение жидких капель. При исследовании двухфазных потоков, содержащих взвесь аэрозольных частиц (дисперсная фаза) в несущей среде (дисперсионная среда) в атмосфере важно правильно оценивать основные параметры, определяющие систему, и адекватно описывать реальный процесс при помощи сформулированной математической модели. Данная статья посвящена разработке нового решателя iceFoam в составе открытого пакета OpenFOAM v1912 для моделирования процесса обледенения при характерном размере частиц порядка 40 мкм, что соответствует Приложению С Авиационных правил АП-25. Для описания динамики жидких капель используется Эйлер-Лагранжев подход. В качестве термодинамической модели реализована модифицированная модель жидкой пленки по теории мелкой воды. В расчете используется две сетки: одна для моделирования внешнего газокапельного потока, другая, толщиной в одну ячейку, для расчета нарастания льда. При разработке исходного кода на языке программирования C++ использовалась технология наследования, т.е. создания базовых и производных классов. В результате был разработан параллельный решатель iceFoam для моделирования движения динамики жидких частиц и образования льда на поверхности исследуемого тела. Представлены результаты расчета для случая обтекания цилиндра и профиля крыла NACA 0012 с помощью метода URANS и и высокорейнольдсовой модели турбулентности Spalart-Allmaras. Приведены картины распределения толщины льда. Для расчета одного тестового примера было использовано от 8 до 32 вычислительных ядер на вычислительном кластере ИСП РАН.

Ключевые слова: обледенение**;** течение; моделирование; решатель; библиотека; разработка; класс; частицы; пленка; цилиндр; профиль; скорость; вязкость; турбулентность.

Для цитирования: Кошелев К.Б., Мельникова В.Г., Стрижак С.В. Разработка решателя iceFoam для моделирования процесса обледенения. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 217–234. DOI: 10.15514/ISPRAS–2020–32(4)–16

Благодарности: Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-29-13016.

Development of iceFoam solver for modeling ice accretion

[1] *K.B. Koshelev, ORCID: 0000-0002-7124-3945 <koshelevkb@mail.ru>*
[2,3] *V.G. Melnikova, ORCID: 0000-0002-1280-2647 <vg-melnikova@yandex.ru>*
[2] *S.V. Strijhak, ORCID: 0000-0001-5525-5180 <s.strijhak@ispras.ru>*

[1] *Institute for Water and Environmental Problems SB RAS*
1,Molodyoznaya St., Barnaul, 656038, Altai Krai, Russia
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[3] *Bauman Moscow State Technical University*
5, Baumanskaya 2nd St., Moscow, 105005, Russia

Abstract. Currently, RF is actively developing the Northern territories. Questions of studying the physical processes of icing are relevant, since climate conditions affect the surface of the objects under study (power lines, residential buildings, power plants, aircraft), human safety and ecology. In clouds, the appearance and movement of liquid droplets-particles is possible. When studying two-phase flows containing a suspension of aerosol particles (dispersed phase) in the carrier medium (dispersion medium) in the atmosphere, it is important to correctly evaluate the main parameters that define the system, and adequately describe the real process using a formulated mathematical model. This article is devoted to the development of a new iceFoam solver as part of the OpenFOAM v1912 package for modeling the icing process at a typical particle size of about 40 microns, which corresponds to Annex C of the AP-25 Aviation rules. The Euler-Lagrangian approach and finite volume method are used to describe the dynamics of liquid droplets. A modified liquid film model based on the shallow water theory is used as a thermodynamic model. The results of calculation for the case of flow around the cylinder and airfoil NACA 0012 using the URANS method and Spalart-Allmaras turbulence model are presented. In the calculation domain, two variants of grids are constructed: for an external gas-drop flow and for a liquid thin film with a thickness of one cell in height. Patterns of ice thickness distribution are given. When developing the source code using C++ language, the technology of inheritance was used, i.e. creating base and derived classes. As a result, a parallel iceFoam solver was developed for simulating the motion of liquid particles and the formation of ice on the bodies' surface. For the calculation of one test case 8-32 computing cores were used on the ISP RAS HPC.

Keywords: ice accretion; flow; simulation; solver; library; development; class, particles; film; cylinder; airfoil; velocity; viscosity; turbulence

For citation: Koshelev K.B., Melnikova V.G. Strijhak S.V. Development of iceFom solver for modeling ice accretion. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 217–234 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–16

Acknowledgements. The reported study was funded by RFBR, project number № 19-29-13016.

1. Введение

В ИСП РАН ведется разработка различных тематических решателей на базе открытой библиотеки OpenFOAM для моделирования процессов в ветропарках, моделирования сжимаемых течений в широком диапазоне чисел Маха [1-3].

Изучение процессов обледенения на элементах самолета, вертолета, беспилотных летательных аппаратов, ветроэлектрической установки, проводов линий электропередач является актуальной задачей в связи с безопасностью и надежностью эксплуатации данной техники. Авиационные правила АП-25 (МАК 2015) регламентирует метеорологические режимы обледенения в условиях полета летательных аппаратов (ЛА) и задают основные параметры (водность облачной массы, среднемедианный диаметр капель облака - MVD, температуру окружающего воздуха) для различных Приложений C, O, P, D [27].

Для Приложения С характерно изучении движения капель с параметром MVD до 40 микрометров. Известно, что при движении ЛА в облаках могут образовываться различные формы и типы льда (рыхлый (rime ice), прозрачный (glaze ice), смешанный (mixed ice),

барьерный (spanwise-ridge ice), рогообразный (horn ice)), что связано с изменением безразмерных чисел Рейнольдса *Re*, Вебера *We* и температуры окружающей среды [4, 7].

Ранее проводились экспериментальные исследования в аэрохолодильных установках, климатических трубах [4-6] и численные расчеты для модельных тел простейшей формы (2D цилиндр, 2D симметричный и несимметричный профиль, пластина, 3D стреловидное крыло, 3D входная часть двигателя, 3D клин и другие) [4-7]. Как правило, для моделирования процессов обледенения используются различные математические модели, которые основаны на Эйлер-Эйлер, Эйлер-Лагранж моделях сплошной среды. Также используются панельный метод для расчета потенциального течения и решение уравнений пограничного слоя интегральным методом, подход на базе метода дискретных частиц. Данные подходы реализованы в известных расчетных кодах LEWICE [8], CANICE, Ansys FENSAP-ICE [9], STAR-CCM+, NSCODE-ICE [10, 11], IGLOO2D/3D [12], PoliMiCE [13, 14], ICESIM-NSDG [15].

2. Требования к решателю

Решатель iceFoam должен обладать следующим функционалом:

- исходный код написан с использованием ООП языка программирования;
- наличие ядра программы и отдельных модулей;
- возможность проверки работоспособности решателя по модулям;
- возможность выбора способа перестроения геометрии тела, например, с помощью методов нормали или биссектрис;
- наличие термодинамических моделей по теории пленки SWIM;
- возможность добавления новых термодинамических моделей (Messinger, Myers, Onera);
- проверка на типовых примерах;
- возможность расчета различных режимов образования льда (rime, glaze, mixed);
- линейная масштабируемость решателя до 256 вычислительных ядер.

Требования к гидродинамическому решателю:

- Эйлер-Лагранжева модель для описания газокапельного потока;
- моделирование несжимаемых, сжимаемых течений с диапазоном изменения чисел Маха от 0.1 до 0.9;
- моделирование вязких турбулентных течений с помощью уравнений Рейнольдса и двухпараметрических моделей турбулентности k-e, Spalart-Allmaras (SA);
- моделирование вязких турбулентных течений с помощью метода крупных вихрей и различных замыкающих моделей для подсеточной вязкости (модель Смагоринского, модель одного дифференциального уравнения для подсеточной кинетической энергии турбулентности);
- моделирование вязких турбулентных течений с помощью моделей отсоединенных вихрей: Spalart-Allmaras DES, Spalart-Allmaras DDES, Spalart-Allmaras IDDES;
- учет влияния двухфазности в двухпараметрических моделях турбулентности;
- использование алгоритмов SIMPLE/PISO/PIMPLE для связи уравнений давление-скорость;
- учет влияния шероховатости поверхности;
- расчет коэффициентов улавливания и теплоотдачи.

Требования к сеточному модулю:

- работа с неструктурированными сетками;
- работа с перекрывающимися сетками – Overset grids;
- работа с вращающимися сетками.

Требования к пре- и постпроцессингу:

- возможность работы с открытым пакетом Paraview для визуализации результатов расчета;
- визуализация жидкой плёнки и жидких частиц;
- визуализация формы льда;
- визуализация скорости ледообразования.

Требования к производительности:

- продолжительность двумерного расчёта – менее 72 часа;
- продолжительность трёхмерного расчёта – менее 168 часов.

Требования к точности решателя:

Основные параметры для случая образования льда на поверхности тела (масса льда, толщина льда, коэффициент улавливания частиц β и другие) должны быть рассчитаны с инженерной точностью в 5 % в сравнении с данными экспериментов.

3. Архитектура решателя

В ИСП РАН разрабатывается собственный тематический решатель-библиотека iceFoam для моделирования различных режимов и типов нарастания льда, движения снежинок и кристаллов льда. Разработка ведется на базе открытой библиотеки OpenFOAM v1912, открытых пакетов Yade, LIGGGHTS с использованием метода дискретных элементов и частиц (Discrete Element Method), гибридного метода на базе методов контрольного объема и дискретных частиц.

В настоящий момент iceFoam включает в себя базовый решатель и две библиотеки: одна библиотека реализует термодинамическую модель жидкой пленки по теории мелкой воды («surfaceFilmModels» library –libsurfaceFilmModelsSWIM), вторая библиотека предназначена для расчета коэффициента сопротивления движения капель и коэффициента теплообмена капель с окружающей средой (libIceFoamParticles). При разработке используется технология наследования языка программирования C++, т.е. использование родительского и производного классов. Архитектура решателя iceFoam является модульной, т.е. существует возможность добавления новых библиотек с физико-математическими моделями среды. Общая структура решателя iceFoam показана на рис. 1-3. Существует возможность расширения функционала решателя для различных Приложений C, O, P, D (рис. 2).

Модуль расчета аэродинамики состоит из исполняемого файла программы iceFoam, реализующего процедуру интегрирования уравнений аэродинамики, законов сохранения массы, количества движения и энергии, в соответствии с описанной ниже математической моделью.

Модуль расчета газокапельного потока состоит из того же исполняемого файла программы iceFoam, и разработанной авторами дополнительной библиотеки расчета силы сопротивления, действующей на сферическую частицу в потоке, с выбором модели расчета коэффициента сопротивления капли. Доступны модели: Путнема, Приходько и Шиллера-Неймана. Также имеется библиотека для расчета безразмерного числа Нуссельта методами Клифта, Фенга и Уитакера. Число Нуссельта необходимо рассчитывать для определения коэффициента теплоотдачи между частицей и окружающей средой.

Все модули разработаны на универсальном языке программирования C++. Информация для проведения расчета задается в соответствующих файлах, определяющих начальные

параметры расчета, свойства окружающей среды и капель, параметры интегрирования, расчетную сетку и т.п.

На Рис.1 представлена типовая блок-схема для решателя, который производит расчет образования льда. На первой этапе производится расчет обтекания модельного тела (крыла). Определяются газодинамические поля для плотности, скорости, давления воздуха, а также определяется величина касательного напряжения и величина теплового потока. Начальные газодинамические поля передаются в решатель для газокапельного потока, где проводится расчет движения капель, определяется скорость капель и коэффициент улавливания. Далее значения скорости капель и коэффициент улавливания используется в термодинамическом модуле для расчета толщины льда. После определения толщины льда на теле происходит перестроение начальной геометрии и расчетной сетки. Далее процедура расчета повторяется до тех пор, пока толщина льда на исследуемом теле больше не изменяется.

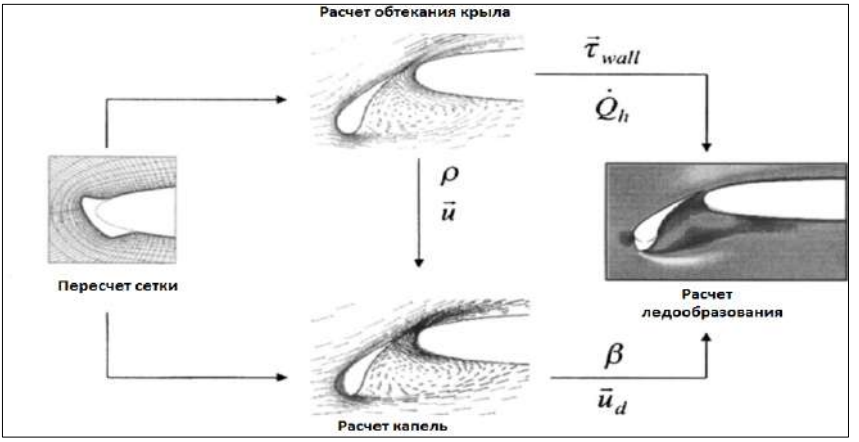


Рис. 1. Блок-схема решателя
Fig. 1. Solver block diagram

Рис. 2. Блок-схема развития решателя
Fig. 2. Solver development block diagram

На рис. 1 имеются следующие обозначения: ρ– плотность, $\vec{u}$ – скорость воздуха, $\vec{u}_d$ – скорость капель, β – коэффициент улавливания, $\vec{\tau}$ - напряжение трения на стенке, Q_h – тепловой поток.

На рис. 2, 3 имеются следующие обозначения: ПОС – протиобледенительная система, FVM-Finite Volume Method, FEM – Finite Element Method, *k-e* и *SA* – название моделей турбулентности, URANS – Unsteady Reynolds Averaged Navier-Stokes, ROM – Reduced Order

Modeling, POD – Proper Ortogonal Decomposition, IBM – Immersed Boundary Method, LES – Large Eddy Simulation.

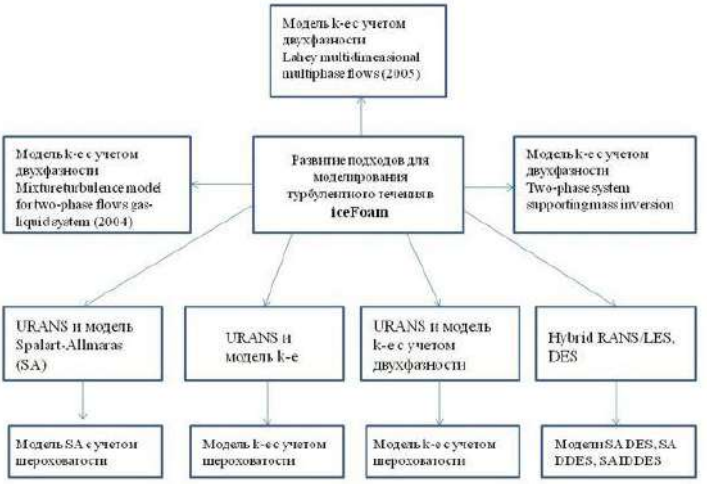


Рис. 3. Различные модели турбулентности в решателе iceFoam
Fig. 3. Different turbulence models in the iceFoam solver

На рис. 2 показана возможность расширения решателя для Приложений C, O, P, D. На рис. 3 показана возможность подключения различных моделей турбулентности, в том числе с учетом влияния двухфазности, для моделирования турбулентных течений.

4. Математическая модель

Моделирование многофазных потоков с континуальной и дисперсной фазами проводится с использованием Эйлер-Лагранжев подхода, в котором непрерывная фаза рассматривается в Эйлеровой постановке, в то время как частицы в дисперсной фазе отслеживаются в Лагранжевой постановке. Из-за малых размеров капель, много меньших размера ячеек, внутренняя динамика капли не рассматривается, и они полагаются движущимися частицами. При взаимодействии газокапельного потока с неровностями и шероховатостями твердой поверхности тела возможно появление и нарастание пленки льда и жидкой пленки воды. Для описания процессов в пленке использовалась модель на базе теории мелкой воды, в основе которой используются осредненные уравнения по толщине пленки, отражающие законы сохранения количества массы, количества движения и энергии для жидкости. Такой подход требует наличия отдельной сетки для области тонкой пленки. Что касается частиц, в правой части уравнений для массы и энергии присутствуют источниковые слагаемые, которые характеризуют процессы плавления, испарения, излучения, разбрызгивания частиц. Нарастание льда приводит к изменению начальной формы тела. Граница тела перемещается в пространстве по нормали. При этом при пересчете положения узлов сетки необходимо обеспечить одновременное перемещение границ для двух разных сеток. Одним из эффективных способов такого пересчета является использование решения уравнения Лапласа. Для того, чтобы охарактеризовать способность искривленной поверхности тела улавливать жидкие капли, используется параметр – коэффициент аккумулирования капель β, для описания тепловых процессов – коэффициент теплоотдачи.

4.1 Эйлеров континуальный подход

Рассматриваемая среда представляет собой нереагирующую равновесную смесь газов с общей температурой T, плотностью ρ и парциальными давлениями p_i для различных компонент смеси. В рамках выбранного приближения смеси предполагается, что импульс,

масса и энергия всего потока переносятся со среднемассовой скоростью $\vec{U}$, массовая доля компонентов смеси набегающего на исследуемое тело потока не меняется со временем. Взаимное движение компонент смеси учитывается в диффузионном приближении. Влияние дисперсной фазы на континуальную вводится в качестве дополнительных членов в уравнениях.

Уравнение сохранения массы смеси:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\vec{U}\rho\right) = \dot{\rho_v}. \quad (1)$$

Уравнение баланса импульса смеси:

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot \left(\vec{U}\rho\vec{U}\right) + \sum_i \rho_i^0 \vec{W_i}\vec{W_i} = \dot{\rho_v}\overrightarrow{U_v} + \nabla \cdot \hat{\sigma} - \nabla p + \rho\overrightarrow{g}. \quad (2)$$

Уравнение баланса энергии смеси:

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot \left(\vec{U}\rho e\right) + \sum_i \nabla \cdot \vec{W_i}\rho_i^0 e_i + \nabla \cdot \left(p\,\vec{U}\right) = -\nabla \cdot \left(\hat{\sigma} \cdot \vec{U}\right) - \nabla \cdot \vec{q} + \dot{\rho_v} e_v, \quad (3)$$

где ρ – плотность смеси, $\vec{W_i}$ – относительная скорость, $\dot{\rho_v}$ – обмен массой между окружающей средой и каплями, $\dot{\rho_v}\overrightarrow{U_v}$ – обмен импульсом между окружающей средой и каплями, p – давление окружающей среды, $\vec{g}$ – ускорение свободного падения, $\hat{\sigma} = \mu_e\left(\nabla\vec{U} + (\nabla\vec{U})^T\right) - \frac{2}{3}\mu_e I \nabla \cdot \vec{U}$ – тензор вязких напряжений; $\mu_e = \mu + \mu_t$ – коэффициент эффективной вязкости смеси, μ и μ_t – коэффициенты ламинарной и турбулентной вязкости соответственно, I – единичный тензор, e – внутренняя энергия смеси, $\dot{\rho_v} e_v$ – обмен энергией между окружающей средой и каплями.

Вектор теплового потока вычисляется в соответствии с законом Фурье: $\vec{q} = -\lambda \nabla T$, где λ – коэффициент теплопроводности смеси: $\lambda = \frac{\mu C_p}{Pr} + \frac{\mu_t C_p}{Pr_t}$, $C_p = \left(\frac{\partial h}{\partial T}\right)_p$, $C_V = \left(\frac{\partial h}{\partial T}\right)_V$, $\nabla T = \frac{\nabla h}{C_p}$, $h = e + \frac{\mathrm{p}}{\rho}$ – удельная энтальпия смеси, Pr и Pr_t – ламинарный и турбулентный числа Прандтля.

Удельная энтальпия смеси является взвешенной суммой энтальпий ее составляющих:

$h = \sum_i Y_i h_i$, где $Y_i = \frac{\rho_i^0}{\rho}$ – массовая доля i-ой компоненты.

Уравнение баланса массы i-ой компоненты:

$$\frac{\partial \rho Y_i}{\partial t} + \nabla \cdot \left(\vec{U}\rho Y_i\right) + \nabla \cdot \vec{W_i}\rho_i^0 = 0. \quad (4)$$

Замыкающее соотношение для массовых долей смеси: $\sum_i Y_i = 1$.

Среднемассовая скорость $\vec{\mathrm{U}}$ и относительные скорости $\vec{W_i}$ введены так, чтобы:

$$\vec{U} = \frac{\sum_i \rho_i^0 \vec{U}_i}{\rho}, \vec{W_i} = \vec{U} - \vec{U_i}, \sum_i Y_i \vec{W_i} = 0.$$

Для вычисления относительных скоростей движения компонент газа используется диффузионное приближение: $\rho_i^0 \vec{W_i} = -D_i \nabla \rho_i^0$, где D_i – коэффициент диффузии i-ой компоненты: $D_i = \frac{\nu_i}{Sc}$ (значения числа Sc рассчитывается по таблицам свойств среды в зависимости от температуры и состава смеси).

Все компоненты газообразной смеси представляют собой совершенный газ с постоянной молярной массой: $p_i = \rho_i R_i T$, где R_i – газовая постоянная.

Решатель iceFoam использует алгоритм PIMPLE, который представляет собой комбинацию алгоритмов SIMPLE и PISO для решения уравнений скорости и давления. Уравнение неразрывности (1) (в коде *rhoEqn*) решается один раз вне основного цикла, в то время как уравнения для импульса (2) (*UEqn.H*), примеси (4) (*YEqn.H*) и энергии (3) (*EEqn.H*) решаются

внутри основного цикла. Уравнение коррекции давления (*pEqn.H*) решается в цикле корректора давления внутри основного цикла вместе с уравнением неразрывности. Файл *createFields.H* отвечает за инициализацию начальных аэродинамических полей. Блок-схема последовательности решения уравнений – алгоритма PIMPLE представлена на рис. 4.

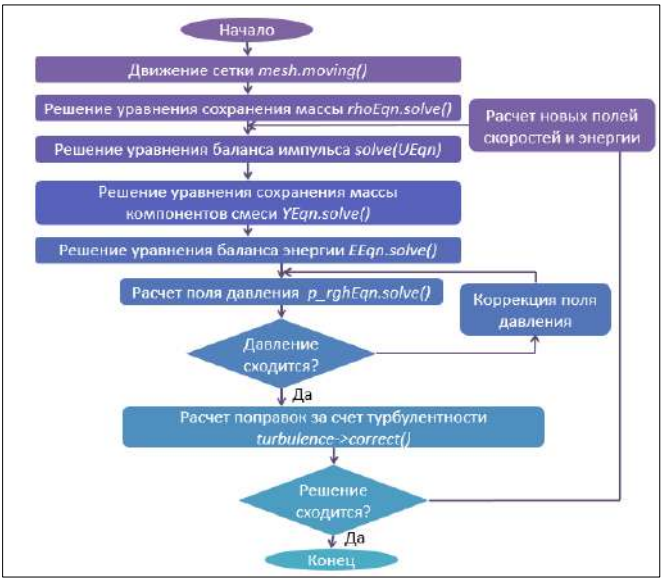


Рис. 4. Последовательность решения уравнений аэродинамики алгоритмом PIMPLE
Fig. 4. The sequence of solving the aerodynamics equations by the PIMPLE algorithm

4.2 Дискретный Лагранжев подход

Дискретный Лагранжев метод учитывает динамику капельной смеси в потоке и взаимодействие капель воды с набегающим потоком и твердой поверхностью.

В качестве базовой модели используется модель облака частиц *sprayCloud* пакета OpenFOAM. Облако сферических частиц определяется положением его центра масс $\bar{x}_p$, диаметром входящих капель D_p, скоростью капель $\overline{U}_p$ и плотностью вещества ρ_p. Тогда масса одной частицы $m_p = \frac{1}{6}\rho_p \pi D_p^3$. Частицы со схожими параметрами представлены облаком, так как имитирование всех реальных капель по отдельности затратно по вычислительным ресурсам. Объемная концентрация частиц, характерная для изучаемых процессов обледенения, невелика (порядка 10^{-6}). В этом случае взаимодействием облаков частиц друг с другом пренебрегаем.

Траектория движения облака частиц определяется путем интегрирования уравнения кинематики:

$$\frac{d\vec{x}_p}{dt} = \vec{U}_p. \tag{5}$$

Скорость частиц определяется из решения уравнения баланса сил. Сила, действующая на частицу, представляет собой сумму всех действующих сил. Примерами таких сил являются сила сопротивления окружающей среды, сила тяжести, плавучесть, сила давления:

$$m_p \frac{d\vec{U}_p}{dt} = \sum \vec{F}_i = \vec{F}_D + \vec{F}_G = \frac{3}{4}\frac{m_p \mu C_D Re_p}{\rho_p {D_p}^2}\left(\vec{U} - \vec{U}_p\right) + m_p \vec{g}\left(1 - \frac{\rho}{\rho_p}\right), \tag{6}$$

где $\vec{F}_D$ – сила давления, $\vec{F}_G$ – сила тяжести. Силой поверхностного натяжения пренебрегаем.

Комплекс $C_D Re_p$ вычисляется в зависимости от выбранной модели расчета коэффициента сопротивления капли с помощью функции, зависящей от числа Рейнольдса частицы (Re_p). Доступны модели: Путнэма; Хабаши; Приходько; Гента; Очкова; Шиллера-Неймана.

Число Рейнольдса частицы: $Re_p = \frac{\rho|U-U_p|D_p}{\mu}$.

Также капельная модель включает в себя уравнение баланса массы капли:

$$\frac{dm_p}{dt} = \dot{m}_p = 0 \tag{7}$$

и уравнение теплового баланса капли:

$$m_p C_{p\,p} \frac{dT_p}{dt} = Q_T + Q_{vap} + Q_{rad}, \tag{8}$$

где C_{pp} – удельная теплоемкость при постоянном давлении капли; T_p – среднеобъемная температура капли.

Так как в решаемых задачах температура окружающей среды значительно меньше температуры кипения капель, то процессы кипения и испарения не рассматриваем $Q_{vap} = 0$. Теплообмена с окружающей средой излучением нет $Q_{rad} = 0$.

В результате конвективного взаимодействия с основным потоком капельные струи забирают или отдают часть внутренней энергии потока газа. Тепловой поток от окружающей среды:

$$Q_T = htc \cdot S_p \cdot (T_{sp} - T), \tag{9}$$

где $S_p = \pi D_p^2$ – площадь поверхности капли; T – температура окружающей среды.

Температура поверхности капли T_{sp}:

$$T_{sp} = \frac{2}{3} T_p + \frac{1}{3} T. \tag{10}$$

Коэффициент теплоотдачи htc:

$$htc = \frac{Nu \cdot \lambda}{D_p}, \tag{11}$$

где λ – коэффициент теплопроводности окружающей среды.

В качестве модели передачи тепла между каплей и окружающем газом можно выбрать из четырех моделей: Клифта; Фенга; Ранца-Маршалла и Уитакера. Наиболее используемая модель – Ранца-Маршалла, в которой используются коэффициенты для расчета числа Нуссельта Nu для капли сферической формы:

$$Nu = 2 + 0.6\sqrt{Re_p} \cdot \sqrt[3]{Pr}. \tag{12}$$

Число Прандтля газа Pr:

$$Pr = \frac{c_p \mu}{\lambda},$$

где c_p – удельная теплоемкость окружающей среды при постоянном давлении; μ – динамическая вязкость окружающего газа.

Дополнительно реализованные модели расчета силы лобового сопротивления капли с различным коэффициентом сопротивления находятся в папке subModels/particlesLib/SphereChooseCdDrad.

Дополнительно реализованные модели расчета передачи тепла между каплями-частицами и окружающим воздухом находятся в папке subModels/particlesLib/HeatTransferModel. Блок-схема последовательности решения уравнений для частиц представлена на рис. 5.

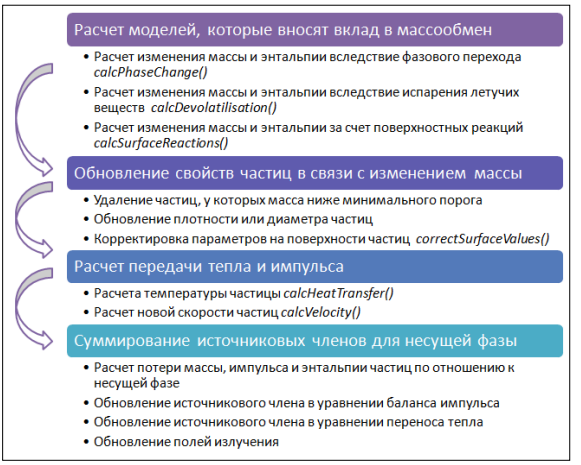


Рис. 5. Алгоритм расчета частиц
Fig. 5. Algorithm for calculating droplet particles

4.3 Источниковые члены, связывающие Эйлеров и Лагранжев подходы

Обмен импульсов между каплями и континуальной фазой:

$$\dot{\rho}_v \boldsymbol{V}_v == -\frac{1}{V_{cell}} \sum N_p m_p \frac{d\boldsymbol{U}_p}{dt}. \quad (13)$$

Обмен энергией между каплями и континуальной фазой :

$$\dot{\rho}_v e_v = -\frac{1}{V_{cell}} \sum N_p m_p \frac{dh_p}{dt}, \quad (14)$$

h_p- удельная энтальпия капли.

Подводя итог, можно отметить, что данная модель является более корректной с физической точки зрения, а именно с точки зрения описания движения облака одинаковых частиц, по сравнению с Эйлер-Эйлер подходом.

4.4 Модель турбулентности Spalart-Allmaras

Для расчета величины турбулентной вязкости используется модель турбулентности *Spalart-Allmaras (SA).* Турбулентная модель *SA* представляет собой одно уравнение переноса для величины $\tilde{\nu}$, которая эквивалентна вихревой вязкости ν_t вдали от границ. Уравнение переноса было получено эмпирически. Это уравнение, пренебрегая условиями перехода, имеет вид [16]:

$$\frac{\partial \tilde{\nu}}{\partial t} + u_i \frac{\partial \tilde{\nu}}{\partial x_i} = C_{b1} \tilde{S} \tilde{\nu} - C_{\omega 1} f_\omega \left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{1}{\sigma}\left[\nabla \cdot \left((\nu + \tilde{\nu})\nabla \tilde{\nu}\right) + \mathrm{C_{b2}}\left((\nabla \tilde{\nu}) \cdot (\nabla \tilde{\nu})\right)\right] \quad (15)$$

Здесь d - расстояние до ближайшей стенки. Турбулентная кинематическая вязкость ν_t связана с рассчитываемой величиной $\tilde{\nu}$ выражением

$$\nu_t = f_{v1} \tilde{\nu} \quad (16)$$

где $f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}$, $\chi = \frac{\tilde{\nu}}{\nu}$, ν – ламинарная кинематическая вязкость. $\tilde{S}$ связана с завихренностью S формулой $\tilde{S} = S + \frac{\tilde{\nu}}{\kappa d^2} f_{v2}$, $f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}$, κ – постоянная фон Кармана.

$f_\omega = g\left(\frac{1 + C_{\omega 3}^6}{g^6 + C_{\omega 3}^6}\right)^{1/6}$, $g = r + C_{\omega 2}(r^6 - r)$, $r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}$, $\sigma = \frac{2}{3}$, $\kappa = 0.41$, $C_{b1} = 0.1355$,

$C_{b2} = 0.622$, $C_{\omega 1} = \frac{C_{b1}}{\kappa^2} + \frac{1 + C_{b2}}{\sigma}$, $C_{\omega 2} = 0.3$, $C_{\omega 3} = 2$, $C_{v1} = 7.1$.

Модель SA получила дальнейшее свое развитие [17-19]. В модели «Дискретный элементный подход» шероховатость учитывается с помощью дополнительных членов в уравнениях потока, которые представляют блокировку потока из-за шероховатостей, сопротивления и теплового потока на элементах с шероховатостями [17]. Однако этот подход требует радикальных изменений в уравнениях потока и не используется на практике.

Модель «Эквивалентный частичный подход» связывает реальную шероховатость с идеализированной шероховатостью со ссылкой на данные из экспериментов [18, 19], где высота отдельной части выводится из реальной формы шероховатости с помощью эмпирических корреляций. Эффект шероховатости имитируется повышением турбулентной вихревой вязкости в области стенки для достижения высоких уровней трения и теплового потока.

4.5 Модель жидкой пленки

Для моделирования процесса образования льда на поверхности тела используются различные термодинамические модели. Среди них можно выделить модель Messinger [20], расширенную модель Messenger (модель Myers) [21], модель на основе теории мелкой воды (Shallow Water Ice Model - SWIM) [22], модель гидротермодинамики жидкой пленки с кристаллами [23].

В библиотеке OpenFOAM существуют две модели тонкой пленки *surfaceFilmModel*: *kinematicSingleLayer* и *thermoSingleLayer* [24, 25]. Вторая модель базируется на первой, только с учетом термодинамических эффектов, таких как испарение. Параметры и определения подмоделей задаются пользователем в словаре *constant/surfaceFilmProperties*.

В табл. 1 указаны основные модели для пленки.

Класс *thermoSingleLayer* можно применять для расчета поверхностных пленок, состоящих из одного компонента с постоянными или изменяющимися термодинамическими свойствами.

Он также может учитывать фазовый переход, впрыск капель из пленки в облако, передачу тепла, как со стенкой, так и с жидкостью, излучение, а также взаимодействие с гидрофильными и гидрофобными поверхностями.

Табл. 1. Подмодели для thermoSingleLayer
Table 1. Submodels for thermoSingleLayer

Модель	Название	Возможные варианты	Описание
Термическая модель	thermoModel	constant	Постоянные термодинамические характеристики
		singleComponent	Термодинамические характеристики вычисляются
Силы	forces	surfaceShear	Сила трения по поверхности
		thermocapillary	Термокапилярная сила (Марангони)
		contactAngle	Сила краевого угла смачивания
Модели впрыска	injectionModels	curvatureSeparation	Разрыв пленки из-за кривизны стенки
		drippingInjection	Ввод капель вследствие капания
Модель изменения фазы	phaseChangeModel	None	Не учитывается
		standardPhaseChange	Модель испарения, включая кипение

Для моделирования пленочных слоев в OpenFOAM необходимо выделить из аэродинамической области дополнительную область сетки около обтекаемого тела. В этой выделенной области сетки рассчитываются все параметры тонкой пленки. Эта область сетки может быть создана с использованием двух утилит OpenFOAM. Сначала утилита *topoSet* используется для извлечения всех граней ячеек участка профиля из существующей

аэродинамической сетки, а извлеченный набор граней ячеек используется для выдавливания новой области сетки с помощью утилиты *extrudeToRegionMesh*. В словаре *extrudeToRegionMeshDict* задаются такие параметры, как набор граней ячейки для использования, количество слоев и толщина экструзии.

Для моделирования пленочного слоя используется так называемая тонкопленочная аппроксимация, которая означает, что скорость, нормальная к сетке на стенке, принимается равной нулю.

Поток в пристеночной пленке рассчитывается с помощью уравнения неразрывности:

$$\frac{\partial \rho\delta}{\partial t} + \nabla \cdot \left(\rho\delta\vec{U}\right) = S_{imp} + S_{splash} + S_{evap} + S_{sep}, \quad (17)$$

где δ – толщина слоя пленки, S_{imp} – масса, добавленная в пленочный слой из-за столкновения капель, S_{splash} – масса, покидающая пленочный слой из-за разбрызгивания капель, S_{evap} – испарившаяся масса и S_{sep} – изменение массы за счет потенциального разделения пленочного слоя.

Уравнения импульса:

$$\frac{\partial \rho\delta\vec{U}}{\partial t} + \nabla \cdot \left(\rho\delta\vec{U}\vec{U}\right) = -\delta\nabla p + S_{\rho\delta U} + \tau. \quad (18)$$

Здесь p – давление, $S_{\rho\delta U}$ - вклад от падающих капель; τ - напряжение от сил, действующих на пленку.

Вклад от давления состоит из 3-х частей:

- капиллярный эффект: $p_\sigma = -\sigma \frac{\partial \delta}{\partial x_i \partial x_i}$;
- гидростатический вклад: $p_\delta = -\rho x_i g_i \delta$;
- давление окружающего газа: p_g.

σ - тензор поверхностного натяжения.

Уравнение энергии:

$$\frac{\partial (\rho\delta h)}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho U_j \delta h\right) = S_{imp} + S_{evap}. \quad (19)$$

Был разработан новый класс *iceSingleLayer*, производный от *thermoSingleLayer*. Данный класс реализует наиболее общие особенности, характерные для моделей пленки, предназначенных для расчета процесса нарастания льда. Одной из таких моделей является модель SWIM [22], в которой уравнение энергии используется не для нахождения температуры, а для вычисления толщины льда в предположении. что внутри двухфазной пленки температуры воды и льда близки к температуре замерзания. Разработанный класс *SWIMIILayer*, производный от *iceSingleLayer*, позволяет выполнять вычисления по обтеканию профилей с расчетом толщины льда по модели SWIM.

Все модели (газовой фазы, динамики капель, пленки) в конечном итоге были объединены в решателе iceFoam без учета движения границы льда и в решателе iceDyMFoam с поддержкой перестройки сетки в соответствии с фактическим положением границы льда.

5. Результаты расчета

Ранее проводились экспериментальные исследования в аэрохолодильных установках, климатических трубах и численные расчеты для модельных тел простейшей формы (2D цилиндр, 2D симметричный и несимметричный профиль, пластина, 3D стреловидное крыло, 3D входная часть двигателя, 3D клин, крыльчатка вентилятора и другие).

Для проверки работоспособности решателей iceFoam и iceDyMFoam были сформулированы краевые задачи для случая обтекания 2D цилиндра и 2D NACA 0012 профиля для случая «rime ice». Для обеих задач, на входе в расчетную область задавалось начальное положение сферических частиц, частота ввода частицы в расчетную область и суммарная масса частиц. Температура жидких частиц задавалась равной температуре окружающей среды. Таким образом, можно было рассчитать величину водности LWC. Для описания газокапельной среды использовались уравнения Рейнольдса и высорейнольдсовая модель турбулентности $K-\omega$ SST с пристеночными функциями. Шаг по времени выбирался равным $\Delta t = 10^{-5}$ с.

5.1 Результаты расчета для цилиндра

Многоблочная расчетная сетка для 2D цилиндра была построена с использованием открытой программы *gmsh* (рис. 6). Для цилиндра скорость набегающего потока задавалась равной $U = 94$ м/с, диаметр цилиндра $D = 15.2$ мм, размер капель $D_p = 30$ мкм, $T = 247$ К, $Re = 1.4 \cdot 10^6$. Также был реализован более универсальный метод биссектрис, позволяющий достаточно корректно перемещать узлы сетки для широкого набора обтекаемых профилей [26]. На рис. 7 показана схема перестроения геометрии тела при нароcте льда по алгебраическому методу биссектрис, где $\vec{b}_1$ и $\vec{b}_2$ это вектора биссектрис углов между соседними расчетными ячейками. Рост льда в узлах происходит вдоль вектора биссектрисы $\vec{b}$.

Важным недостатком метода биссектрис является то, что при таком подходе не выполняется закон сохранения массы. Чтобы сохранить массу, смещение красных узлов должно быть уменьшено (Рис. 7а). Помимо этого, при использовании алгебраического метода может возникнуть другая проблема – образование самопересечений границ нароста льда. Разделив процесс роста льда на итерации (Рис. 7б), можно применить поправку для учета кривизны поверхности (выпуклой или вогнутой), которая вызывает растяжение или сжатие граней новой геометрии. Фактически этот процесс эквивалентен вычислению объема через интеграл по n трапециям вместо интеграла по прямоугольнику. Поскольку промежуточные этапы выполняются заранее, это также позволяет обнаруживать и корректировать самопересечения льда.

При моделировании нарастания льда расчетные узлы сетки перемещаются с использованием обычного метода смещения узлов, следуя направлению начальных биссектрис. Затем находится новое криволинейное расстояние, после чего процесс переходит на следующую итерацию и повторяется вновь.

На рис. 8 представлена картина результатов моделирования намерзания рыхлого льда (rime ice) на цилиндре в момент времени t=80 секунд. Данный расчетный случай соответствует случаю 4а работы [5]. Синяя кривая показывает форму льда, рассчитанную в iceFoam с помощью модели SWIM в сравнении с экспериментальными данными (красная кривая).

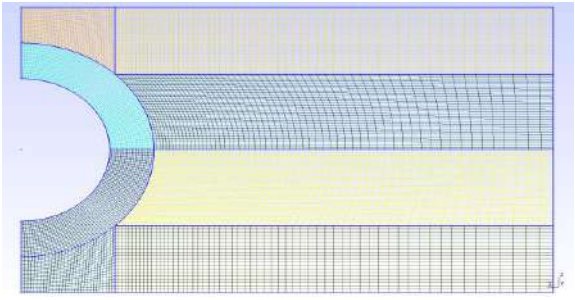


Рис. 6. Расчетная область и расчетная сетка
Fig. 6. Computational domain and computational grid

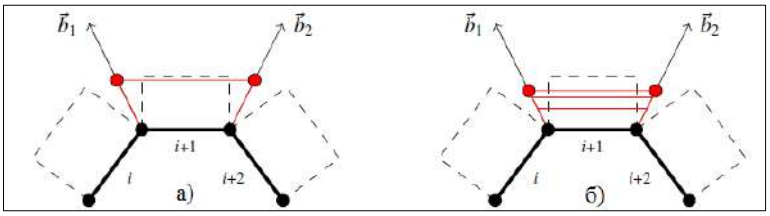


Рис. 7. Метод биссектрис а) одна итерация, б) несколько итераций
Fig. 7. Bisector method a) one iteration, b) several iterations

На рис. 9 представлена картина результатов моделирования намерзания прозрачного льда (glaze ice) на цилиндре. Движение перестроение геометрии тела рассчитывается методом биссектрис, а расчет движения узлов сетки рассчитывается с помощью решения уравнения Лапласа.

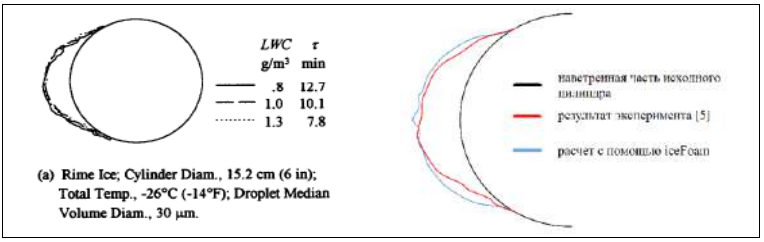


Рис.8. Расчет обтекания цилиндра с «rime ice»
Fig. 8. Flow around cylinder with rime ice simulation

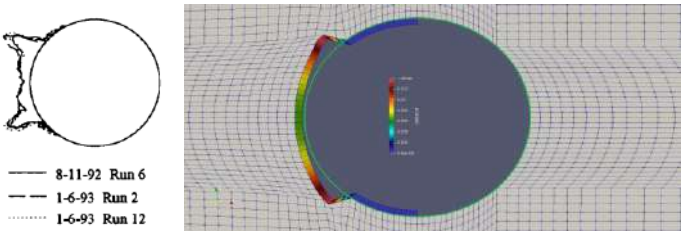


Рис. 9. Картина результатов моделирования намерзания льда с «glaze ice» для t= 125 с
Fig. 9. Results of ice freezing modeling

Видно, что наплыв льда приплюснут, что соответствует экспериментальным данным. Максимальная толщина льда получилась близкой к эксперименту, а точную форму, высоту наростов в форме рожков не удалось правильно воспроизвести.

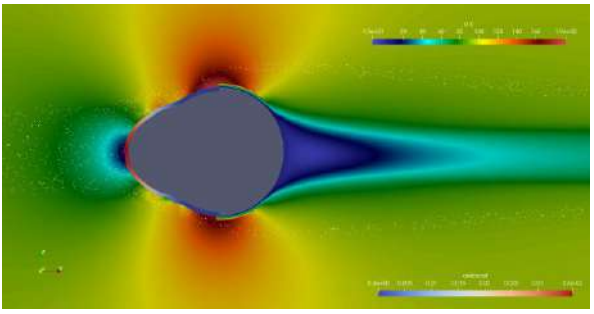


Рис. 10. Картина результатов моделирования движения частиц и намерзания льда
Fig. 10. Picture of the results of modeling ice freezing

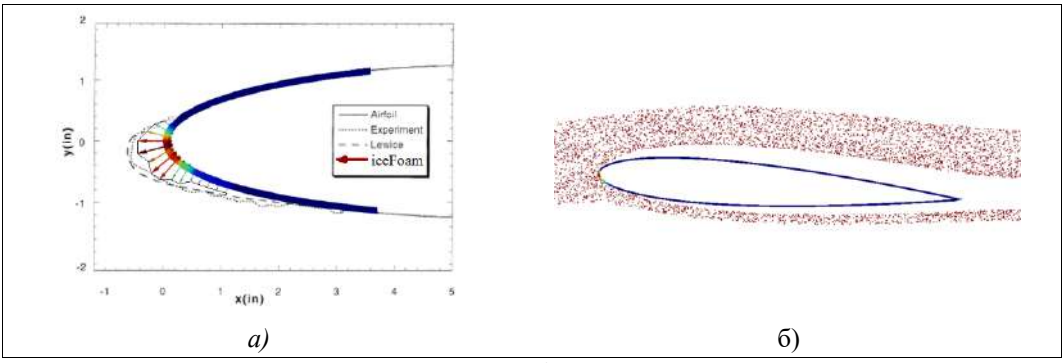
Картина результатов моделирования движения частиц представлены на рис. 10. Показаны полученные значения поле скорости для компоненты U_x для случая «rime ice», траектории движении частиц и распределение толщины пленки льда в момент времени t= 680 с.

5.2 Результаты расчета для профиля NACA 0012

Для 2D профиля NACA 0012 при обтекании потоком под углом атаки 4 градуса расчетная сетка была построена с помощью утилит *blockMesh, extrudeMesh* и *extrudeToRegionMesh*. На первом этапе исследований расчетная сетка включала в себя 16 000 ячеек. Дополнительная сетка для моделирования динамики жидкой пленки и замерзания льда вокруг профиля состоит из 120 ячеек толщиной в одну ортогональную ячейку по высоте.

Решалась задача обтекания профиля под углом 4° потоком воздуха со скорость $U = 102.8$ м/с, размер капель $D_p = 20$ мкм, температура T = 250.37 К, водность LWC = 0.55 г/м^3 . На выходе расчетной области задавалось фиксированное значение для давления и температуры. Значения величины турбулентной вязкости nut в начальный момент времени во всей расчетной области и на границах входа задавались исходя из значения степени турбулентности набегающего потока *Tu*=5% соответственно, на стенке профиля крыла задавалась пристеночная функция *nutWallFunction*.

В решателе iceFoam возможно использование дополнительной пристеночной функции *nutURoughWallFunction* для учета влияния шероховатости поверхности. При этом необходимо задать среднюю величину шероховатости *roughnessHeight,* 2 дополнительных параметра *roughnessConstant, roughnessFactor.*



а) б)

Рис. 11. Результаты моделирования профиля NACA 0012 на неподвижной сетке
Fig. 11. Modeling results of the NACA 0012 airfoil on non-dynamic mesh

На рис. 11 показана траектория частиц (а) и распределение толщины пленки льда (б) по поверхности для расчета с неподвижной сеткой. Результаты расчетов сравнивались с результатами экспериментов NASA, случай № 405 [4]. Серой сплошной линией показаны результаты эксперимента, пунктирной линией – результаты кода LEWICE, цветными векторами – расчет с помощью iceFoam. Шаг по времени составил около 10^{-6} секунд, количество пакетов частиц около 4 000. Скорость счета составила 0.75 ч/с на 8 ядрах.

Расчеты также проводились на сетках 64 000, 144 000 и 256 000 ячеек. Размер и форма льда изменялись незначительно при изменении количества ячеек в расчетной области. Среднее значение *Yplus*, безразмерное расстояние от стенки до первого узла, на сетке с 256 000 ячеек было равно 49. Данное значение удовлетворяет требованиям модели пристеночных функций для высокорейнольдсовой модели турбулентности [28].

Также для этого расчетного случая было проведено моделирование толщины льда с подвижной сеткой по методу биссектрис. Для лучшего разрешения была использована сетка на 144 000 ячеек. Результаты представлены на рис. 12 и 13.

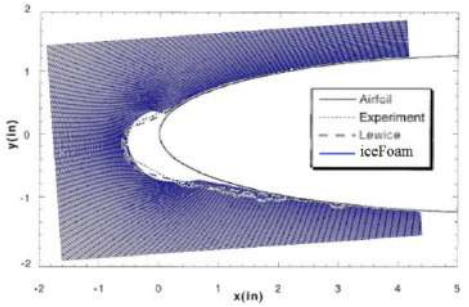
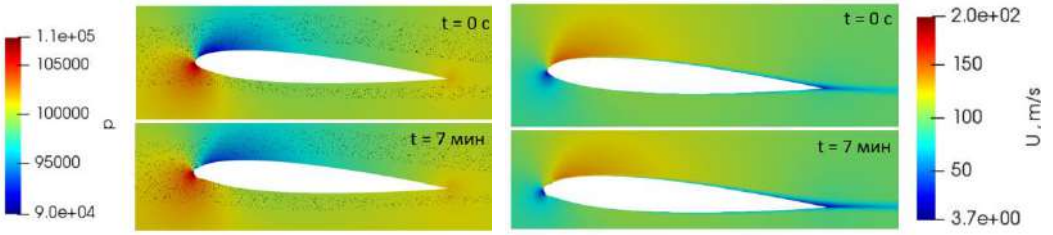


Рис.12. Результаты моделирования с подвижной сеткой нароста льда для профиля NACA 0012
Fig.12 Ice airfoil simulation results for NACA 0012 with dynamic mesh



а) *б)*

Рис.13. Результаты моделирования с подвижной сеткой нароста льда для профиля NACA 0012
а) поле давления; б) поле скорости
Fig.13. Ice airfoil simulation results for NACA 0012 with dynamic mesh
a) pressure field; b) velocity field

Наличие подвижной сетки позволяет более корректно определять границу льда, которая и является искомой величиной. Тем более нарастание льда значительно изменяет течение и положение точки полного торможения потока (рис. 13), что в свою очередь влияет на аэродинамические характеристики крыла.

6. Заключение

В статье рассмотрены вопросы разработки и возможности решателя iceFoam для моделирования обтекания модельных тел газокапельным потоком и образования льда. Дальнейшее развитие решателя состоит в следующем:

- валидация и верификация решателя по отдельным модулям;
- добавление новых термодинамических моделей;
- добавление новых моделей для расчета границы между воздухом и льдом;
- сравнение расчетов с результатами эксперимента на известных тестовых задачах;
- улучшение масштабируемости решателя при запуске в параллельном режиме.

Вычисления были проведены с использованием ресурсов вычислительного кластера лаборатории СПО ЦМТС ИСП РАН. Для расчета одного примера было использовано от 8 до 32 вычислительных ядер.

Список литературы / References

[1]. Крапошин М.В., Стрижак С.В. Проблемно-ориентированная библиотека SOWFA для решения прикладных задач ветроэнергетики. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 259-274. DOI: 10.15514/ISPRAS-2018-30(6)-14 / Kraposhin M.V., StrijhakS.V. The problem-oriented library SOWFA for solving the applied tasks of wind energy. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 259-274 (in Russian).

[2]. Кошелев К.Б., Стрижак С.В. Моделирование динамики частиц в планетарном пограничном слое и в модельном ветропарке. Труды ИСП РАН, том 31, вып. 6, 2019 г., стр. 177-186. DOI: 10.15514/ISPRAS–2019–31(6)–10. / Koshelev K.B., Sttrijhak S.V. Simulation of particle dynamics in planetary boundary layer and in a model wind farm. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 6, 2019. pp. 177-186 (in Russian).

[3]. Kraposhin M, Bovtrikova A, Strijhak S. Adaptation of Kurganov-Tadmor numerical scheme for applying in combination with the PISO method in numerical simulation of flows in a wide range of Mach numbers. Procedia Computer Science, vol. 66, 2015, pp. 43-52.

[4]. Shin J., Bond T.H. Result of an Icing Test on a NACA 0012 Airfoil in the NASA Lewis Icing Research Tunnel. In Proc. of the 30th Aerospace Sciences Meeting & Exhibit, 1992. 20 p.

[5]. Andreson D.N. Rime-, Mixed-, and Glaze-Ice Evaluations on Three Scaling Laws. NASA Technical Memorandum 106461. AIAA-94-0718, 1994. 14 p.

[6]. Papadakis M. et al. Experimental Investigation of Water Droplet Impingement on Airfoils, Finite Wings, and an S-duct Engine Inlet. NASA/TM-2002-211700, 2002. 435 p.

[7]. Алексеенко С.В., Приходько А.А. Численное моделирование обледенения цилиндра и профиля. Обзор моделей и результаты расчетов. Ученые записки ЦАГИ, том XLIV, № 6, 2013 г., стр. 25-57 / Alekseenko S.V., Prikhodko A.A. Numerical modeling of cylinder and profile icing. Review of models and calculation results. TsAGI Scientific Notes, vol. XLIV, № 6, 2013, pp. 25-57 (in Russian).

[8]. Wright W.B. User's manual for the improved NASA Lewis ice accretion code LEWICE 1.6. Technical report, NASA TR-198355, 1995. 95 p.

[9]. Bourgault Y., Boutanios Z., Habashi W.G. Three-dimensional Eulerian approach to droplet impingement simulation using FENSAP-ICE, Part 1: Model, Algorithm, and Validation. Journal of Aircraft, vol. 37, no. 1, 2000, pp. 95-103.

[10]. Pena D, Hoarau Y, Laurendeau E. Development of a three-dimensional icing simulation code in the NSMB flow solver. International Journal of Engineering Systems Modelling and Simulation, vol. 8, № 2, 2016, pp. 86-98.

[11]. Pena D., Hoarau Y., Laurendeau E. A single step ice accretion model using Level-Set method. Journal of Fluids and Structures, vol. 65, 2016, pp. 278–294.

[12]. Gori G., Zocca M., Garabelli M., Guardone A., Quaranta G. PoliMIce: A simulation framework for three-dimensional ice accretion. Applied Mathematics and Computation, vol. 15, 2015, pp. 96-107.

[13]. Zocca M., Gori G., and Guardone A. Blockage and Three-Dimensional Effects in Wind-Tunnel Testing of Ice Accretion over Wings. Journal of Aircraft, vol. 54, no. 2, 2017, pp. 759-767.

[14]. Trontin P., Blanchard G., Kontogiannis A., Villedieu P. Description, assessment of the new ONERA 2D icing suite IGLOO2D. In Proc. of the 9th AIAA Atmospheric and Space Environments Conference, 2017. 28 p.

[15]. Волков А.В., Зыонг Д.Т. Применение метода Галеркина с разрывными функциями к решению системы уравнений динамики водяной взвеси в воздушном потоке. Ученые записки ЦАГИ, том XLVIII, № 5, 2017 г., стр. 1-18 / Volkov A.V., Zyong D.T. Application of the Galerkin method with discontinuous functions to the solution of the system of equations for the dynamics of a water suspension in an air flow. TsAGI Scientific Notes, vol. XLVIII, № 5, 2017, pp. 1-18 (in Russian).

[16]. Spalart P. R., Allmaras S. R. A One-Equation Turbulence Model for Aerodynamic Flows. In Proc. of the 30th Aerospace Sciences Meeting and Exhibit, 1992, AIAA Paper 1992-0439.

[17]. Aupoix B. Modelling of boundary layers over rough surfaces. Fluid Mechanics and Its Applications, vol. 24, 1994, pp. 16-20.

[18]. Spalart P. Trends in turbulence treatments. In Proc, of the Fluids 2000 Conference and Exhibit, 2000, AIAA Paper 2000-2306.

[19]. Aupoix B., Spalart P.R. Extensions of the Spalart–Allmaras turbulence model to account for wall roughness // International Journal of Heat and Fluid Flow, vol. 24, issue 4, 2003, pp. 454-462.

[20]. Messinger B. Equilibrium temperature of an unheated icing surface as a function of airspeed. Journal of the Aeronautical Sciences, vol. 1, no. 20, 1953, pp. 29-42.

[21]. Myers T.G. Extension to the Messinger model for aircraft icing. AIAA Journal, vol. 39, no. 2, 2001, pp. 211–218.

[22]. Bourgault Y., Beaugendre H., Habashi W. Development of a shallow-water icing model in FENSAP-ICE. Journal of Aircraft, vol. 37, no. 4, 2000, pp. 640–646.

[23]. Кашеваров А.В., Стасенко А.Л. Гидротермодинамика жидкой пленки с кристаллами на поверхности тела в потоке воздуха, содержащем частицы льда. Прикладная математика и техническая физика, № 2, 2017 г., стр. 103–114 / Kashevarov A.V., Stasenko A.L. Hydro-

Thermodynamics of a Liquid Film with CRystals on the Body Surface in an Air Flow Containing Ice Particles. Journal of Applied Mechanics and Technical Physics, № 2, 2017, pp. 103–114 (in Russian).

[24]. Jungskog E. Description of reactingParcelFilmFoam. CFD with OpenSource software. A course at Chalmers University of Technology Taught by Hakan Nilsson. 2014. 31 p.

[25]. Beld E. J. Droplet impingement and film layer modeling as a basis for aircraft icing simulations in OpenFOAM. Master thesis. TU Twente, 2013, 49 p.

[26]. Bourgault-Côté S., Hasanzadeh K., Lavoie P., Laurendeau E. Multi-Layer Icing Methodologies for Conservative Ice Growth. In Proc. of the 7th European Conference for Aeronautics and Aerospace Sciences (EUCASS), 2017, 15 p.

[27]. Ципенко В.Г., Шевяков В.И. Обеспечение безопасности полета транспортных воздушных судов с учетом новых сертификационных требований к условиям обледенения. Научный вестник МГТУ ГА, том 22, no. 3, 2019 г., стр. 45–56 / Tsipenko V.G., Shevyakov V.I. Promotion of transport aircraft flight safety taking into account updated certification requirements for icing conditions. Civil Aviation High Technologies (Nauchnyi Vestnik MGTU GA), vol. 22, no. 3, 2019, pp. 45–56 (in Russian).

[28]. Смирнов Е.М., Гарбарук А.В. Конспекты лекций дисциплины. Течения вязкой жидкости и модели турбулентности: методы расчета турбулентных течений. СПбГПУ, 2010 г., 127 стр. / Smirnov E.M., Garbaruk A.V. Discipline lecture notes. Viscous fluid flows and turbulence models: methods for calculating turbulent flows. SPbPU, 2010, 127 p. (in Russian).

Информация об авторах / Information about authors

Константин Борисович КОШЕЛЕВ – кандидат физико-математических наук, доцент, старший научный сотрудник. Сфера научных интересов: вычислительная гидродинамика, гидрология, геоинформатика.

Konstantin Borisovich KOSHELEV is candidate of physical and mathematical sciences, associate professor, senior researcher. Research interests: computational fluid dynamics.

Валерия Геннадиевна МЕЛЬНИКОВА – аспирант МГТУ, кафедра «Аэрокосмические системы», научный сотрудник ИСП РАН. Сфера научных интересов: вычислительная гидродинамика, метод контрольного объема, подвижные сетки, лагранжев подход.

Valeriia Gennadievna MELNIKOVA – PhD student of Bauman Moscow State Technical University, «Aerospace systems» department, researcher at ISP RAS. Research interests: computational fluid dynamics, finite volume method, dynamic meshes, particles.

Сергей Владимирович СТРИЖАК – кандидат технических наук, ведущий инженер. Сфера научных интересов: вычислительная гидродинамика, многофазные течения, турбулетность.

Sergei Vladimirovich STRIJHAK – candidate of technical sciences, leading engineer. Research interests: computational fluid dynamics.

DOI: 10.15514/ISPRAS–2020–32(4)–17

Модели процессов, сопровождающих кристаллизацию переохлажденных капель

[1,2] *И.А. Амелюшкин, ORCID: 0000-0002-4281-3531 <Amelyushkin_Ivan@mail.ru>*
[3] *М.А. Кудров, ORCID: 0000-0003-2056-1932 <MKudrov@mail.ru>*
[3] *А.О. Морозов, ORCID: 0000-0003-4620-9662<MorozovAO@ phystech.edu>*
[1,3] *А.Л. Стасенко, ORCID: 0000-0001-9608-3186 <Stasenko@serpantin.ru>*
[1,3] *А.С. Щеглов, ORCID: 0000-0002-0669-3392 <Shcheglov@phystech.edu>*

[1] *Центральный аэрогидродинамический институт им. проф. Н.Е. Жуковского, 140180, Россия, г. Жуковский, ул. Жуковского, д. 1*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*
[3] *Московский физико-технический институт, 141701, Россия, Долгопрудный, Институтский пер., д. 9*

Аннотация. Развит метод расчета взаимодействия переохлаждённых капель с твердым телом, покрытие которого имеет рельеф и обладает различной степенью гидрофобности. Сформулированы основные критерии соответствия результатов молекулярного моделирования физической реальности. Получены численные оценки параметров рельефа гидрофобной поверхности твердого тела в зависимости от безразмерных динамических параметров удара переохлажденных капель. На основании проведенных ранее экспериментальных исследований, теоретических оценок, аналитических и экспериментальных данных других исследователей в настоящей работе развиты математические модели особенностей кристаллизации переохлажденной метастабильной жидкости. Получены оценки параметров процессов, сопровождающих движение фронта кристаллизации в переохлажденных метастабильных каплях воды в приложении к проблеме обледенения летательных аппаратов.

Ключевые слова: метастабильные капли; молекулярное моделирование; фронт кристаллизации; поток энергии на межфазной границе

Для цитирования: Амелюшкин И.А., Кудров М.А., Морозов А.О., Стасенко А.Л., Щеглов А.С. Модели процессов, сопровождающих кристаллизацию переохлажденных капель. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 235–244. DOI: 10.15514/ISPRAS–2020–32(4)–17

Благодарности: работа выполнена при поддержке РФФИ (проект № 19-29-13016)

Models of processes accompanying crystallization of supercooled droplets

[1,2] *I.A. Amelyushkin, ORCID: 0000-0002-4281-3531 <Amelyushkin_Ivan@mail.ru>*
[3] *M.A. Kudrov, ORCID: 0000-0003-2056-1932 <MKudrov@mail.ru>*
[3] *A.O. Morozov, ORCID: 0000-0003-4620-9662 <MorozovAO@ phystech.edu>*
[1,3] *A.L. Stasenko, ORCID: 0000-0001-9608-3186 < Stasenko@serpantin.ru >*
[1,3] *A.S. Sheglov, ORCID: 0000-0002-0669-3392 <Shcheglov@phystech.edu >*

[1] *Central Aerohydrodynamic Institute,*
1, Zhukovskogo st., Zhukovsky, 140180, Russia
[2] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[3] *Moscow Institute of Physics and Technology,*
9, Insitutsky per., Dolgoprudny, 141701, Russia

Abstract. The creation of high-performance methods for calculating the interaction of aerosol flows with a solid is of great practical interest in the problems of preventing surfaces from icing, predicting climatic phenomena, metallurgy and astronomical processes. One of the methods of icing diminishing is the use of hydrophobic coatings, which, as a rule, work effectively with insignificant ratios of inertial forces to the forces of surface tension of a liquid near the relief of the streamlined body. However, when the surface density of the kinetic energy of the supercooled drop exceeds a certain critical value, the ice-phobic properties lead to negative effects due to the penetration of the supercooled liquid into the depressions and solidification in them. A method is developed for calculating the interaction of supercooled drops with a relief solids, which have various degrees of hydrophobicity. Basic criteria for corresponding the results of molecular modeling to physical reality are formulated. The need to develop algorithms for numerical simulation is due to the fact that significant computational resources are required even for calculating small droplets which are several tens of nanometers in size. Numerical estimates of the parameters of the relief of a hydrophobic surface of a solid are obtained depending on the dimensionless dynamic parameters of the impact of supercooled drops. Moving interface – the crystallization front in supercooled metastable liquid droplets has specific properties. On the basis of previously carried out experimental studies, theoretical estimates, analytical and experimental data of other researchers, in present work mathematical models of the crystallization features of a supercooled metastable liquid are developed. Estimates of the parameters of the processes accompanying the movement of the crystallization front in supercooled metastable water droplets are obtained with application to the problem of icing of aircraft.

Keywords: metastable droplets; molecular modeling; crystallization front; energy flow at the interface boundary

For citation: Amelyushkin I.A., Kudrov M.A., Morozov A.O., Stasenko A.L., Shcheglov A.S. Models of processes accompanying crystallization of supercooled metastable droplets. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 235–244 (in Russian). DOI: 10.15514/ISPRAS-2020-32(4)-17

Acknowledgements. The work is supported by Russian Foundation of Fundamental Research (project No 19-29-13016).

1. Введение. Описание актуальности темы и краткий обзор опыта предшественников

Создание высокопроизводительных методов расчета взаимодействия аэрозольных течений с твердым телом представляет большой практический интерес в задачах противодействия обледенению, предсказания климатических явлений, металлургии и астрономических процессах. Один из методов препятствия обледенению представляет собой использование гидрофобных покрытий, которые, как правило, эффективно работают при незначительных отношениях сил инерции к силам поверхностного натяжения жидкости вблизи рельефа обтекаемого тела. Однако, при превышении поверхностной плотности кинетической энергии

переохлаждённой капли некоторого критического значения, льдофобные свойства приводят к отрицательным эффектам ввиду проникновения переохлажденной жидкости в углубления и застывания в них. Для расчета режимов взаимодействия капель с гидрофобными поверхностями ранее [1, 2] применялся метод молекулярной динамики, который позволяет рассчитывать поведение молекулярных кластеров (объединений) и нанокапель размером до нескольких десятков нанометров. Под наночастицами понимают частицы размером от 1 до 100 нанометров. В то же время под термином молекулярные кластеры принимают молекулярные объединения от 2 до 1000 молекул. Таким образом, существует диапазон размеров капель и соответствующих им чисел молекул, при котором термины наночастицы и молекулярные кластеры в равной степени применимы для описания молекулярной системы. Кристаллизация растворенных в переохлажденной воде кристаллов описывается уравнением Кана-Хиллиарда [3] $\partial n_c/\partial t = \overline{\nabla} \cdot M(n_c)\overline{\nabla}[f(n_c) - \varepsilon^2 \Delta n_c]$ (здесь n_c – концентрация кристаллов, ε – коэффициент градиента энергии, $M = M(n_c)$ – коэффициент подвижности, $f(n_c)$ – гомогенная свободная энергия) и уравнением Гинсбурга-Ландау; при исследовании процессов кристаллизации используется теория неустойчивости Муллинс-Сикерки [4]. Данные об особенностях кристаллизации переохлажденной воды получены в предшествующих исследованиях [5–8]. Стремление понять внутреннюю структуру воды возрастает тем более, чем больше накапливается фактов о ее разнообразных проявлениях [9–12]. В настоящей работе развит предложенный ранее метод моделирования взаимодействия микро- и макроскопических капель с твердым телом методами молекулярного моделирования [1, 2], эффективных используют при исследовании нанообъектов.

2. Метод моделирования гидрофобных свойств

Необходимость развития алгоритмов численного моделирования вызвана тем, что даже для расчета молекулярных соединений размером в несколько десятков нанометров требуются значительные вычислительные затраты. По аналогии с описанием эволюции формы капли в безграничном несущем потоке, для случая столкновения с твердым телом можно также ввести следующие характерные времена и безразмерные критерии: τ_D – время прохождения расстояния порядка диаметра D=2R капли со скоростью ее соударения, τ_σ – период колебаний формы капли, τ_μ –время их вязкого затухания. Из определений чисел Вебера и Лапласа и характерных времен можно получить следующие соотношения: добротность колебаний формы $Q \sim \tau_\mu/\tau_\sigma \sim Lp^{-2}, We \sim (\tau_\sigma/\tau_D)^2, We/Lp \sim (\tau_\mu/\tau_D)^2$. В частности, из последнего соотношения видно, что значение $We/Lp \sim 1$ осуществляется, когда движение жидкости в деформируемой капле должно затухнуть за время прохождения размера со скоростью столкновения. Другой физический аспект проблемы связан с существованием нестабильных конфигураций молекулярных кластеров.

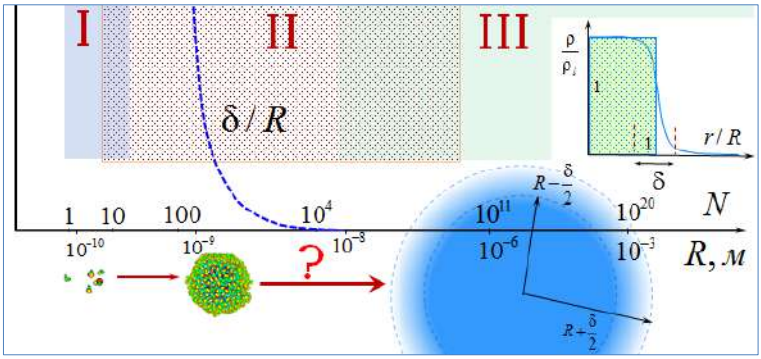


Рис. 1. Особенности моделирования микро- и макро капель молекулярно-динамическими методами
Fig. 1. Features of modeling micro and macro drops by molecular dynamics methods

На рис. 1 схематически изображена суть проблемы, отмеченная вопросительным знаком. По горизонтальной оси отложены размеры рассматриваемых «капель» воды – от отдельной молекулы до миллиметрового масштаба. Подобно тому, как трудно определить, с какого числа зерен можно говорить об их «куче» (апория Зенона), так и трудно сказать, с какого количества молекул начинается капля. Некоторой подсказкой здесь может служить характерная толщина $\delta = (a_{vw} < r_m^2 >/\sigma_l)\left(\rho_l - \rho_{sv}(T)\right)^2 \cong 5$ нм поверхностного слоя, в пределах которого действуют поверхностные силы (здесь a_{vw}, r_m, ρ_l, ρ_{sv} – параметр уравнения состояния газа Ван-дер-Ваальса, средний радиус молекулы, плотность жидкости и плотность настыщенных паров при температуре T, соответственно) поверхностного слоя макрокапли [13] (справа внизу на рис. 1), который заменяет привычную ступеньку радиального распределения плотности капли в паре той же жидкости. Кроме того, на рис.1 изображены области нанокапель, в которых эффективные методы молекулярной динамики, основанные на «первых принципах» и макроскопической гидродинамике. Результаты исследований эволюции наночастиц в первой из них почти недоступны для экспериментальной проверки; во второй – накоплен большой опытный материал по исследованию быстро протекающих процессов столкновения макрокапли с поверхностью твердого тела, сухой или покрытой пленкой жидкости.

На рис. 2 приведен пример гидрофобной поверхности, оценка расстояния между углублениями в зависимости от числа Вебера и пример численного моделирования методом молекулярной динамики удара капли о гидрофобное тело.

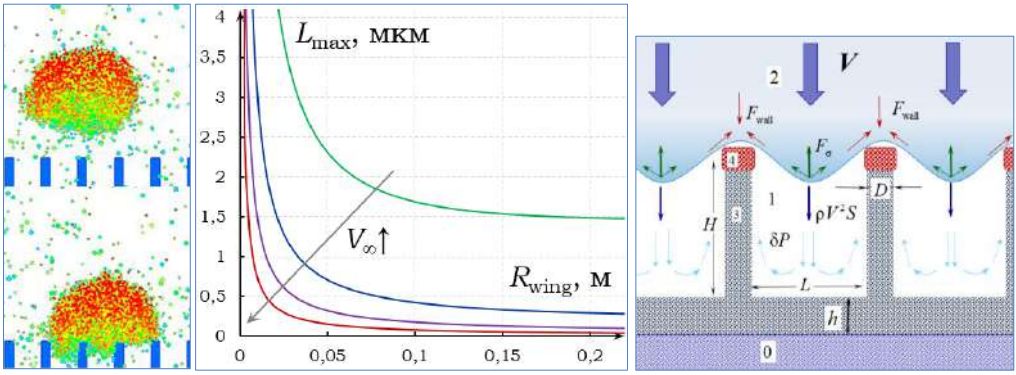


Рис. 2. Слева – пример удара капли о гидрофобное рельефное тело;
в центре – оценка максимального расстояния между выступами гидрофобного тела при котором капли размером 20 мкм не будут проникать в углубления гидрофобного тела;
справа – структура гидрофобного покрытия и схема взаимодействия с ним жидкости: 0 – материал конструкции обтекаемого тела, 1 – газ (воздух), 2 – жидкость, 3 – покрытие из гидрофобного материала, 4 – гидрофильные «шляпки»; в центре – оценка зависимости максимального значения расстояния между выступами рельефа гидрофобного тела L_{max} от радиуса кривизны обтекаемого тела R_{wing} и скорости полета V_∞. Кривые соответствуют значениям скорости 10, 25, 50 и 100 м/с.
Fig. 2. Left – an example of a drop hitting a hydrophobic relief body;
in the center - an estimate of the maximum distance between the protrusions of the hydrophobic body at which 20 μm droplets will not penetrate into the depressions of the hydrophobic body;
on the right - the structure of the hydrophobic coating and the scheme of liquid interaction with it: 0 - material of the structure of the streamlined body, 1 - gas (air), 2 - liquid, 3 - coating made of hydrophobic material, 4 - hydrophilic «caps»; in the center - an estimate of the dependence of the maximum value of the distance between the ridges of the relief of the hydrophobic body L_{max} on the radius of curvature of the streamlined body Rwing and the flight speed V∞. Curves correspond to speeds of 10, 25, 50 and 100 m/s

В настоящей работе описаны условия соответствия результатов молекулярного моделирования малых частиц более крупным микро- и макроскопическим объектам: 1. число молекул в частице много больше 10. 2. Характерный размер частицы значительно превышает толщину поверхностного слоя δ. 3. При выполнении первых двух условий необходимо

равенство основных критериев подобия (числа Вебера $We = D\rho_1 V^2/\sigma_l$ и капиллярности $Ca = \mu V/\sigma_l$ при значительной роли вязких сил); результаты, полученные для наночастиц, справедливы для микро- и макрочастиц. Здесь σ_l – коэффициент поверхностного натяжения, μ – коэффициент динамической вязкости жидкости, V – скорость удара капли. 4. Взаимодействие молекул предполагается парным, потенциал взаимодействия симметричный, его параметры подобраны для учета гидродинамических, термодинамических и упругих свойств жидкости. 5. При фазовых переходах изменение свойств материала приводит к изменению свойств потенциала взаимодействия молекул, между которыми образуются водородные и другие связи. Параметры потенциала подбираются на основании сравнения результатов расчета с данными экспериментальных исследований. Заметим, что отношение числа Вебера к числу Лапласа $LP = \sigma_l \rho D/2\mu^2$ не зависит от диаметра частицы, $We/Lp = \mu_l V/\sigma_l$. В качестве второго безразмерного параметра подобия в настоящей работе принято отношение диаметра частицы к размеру h рельефа шероховатости D/h.

Для эффектной работы гидрофобного покрытия необходимо, чтобы элементы деформируемой при ударе капли не проникали в углубления или поры гидрофобного покрытия. Для этого необходимо превышение давления сил поверхностного натяжения над характерным значением скоростного напора жидкости в капле: $\sigma/(L/2) > \rho V^2$, если расстояние между углублениями существенно меньше диаметра капли: $L << R_d$. Таким образом можно предъявить следующее необходимое условие работы гидрофобного покрытия: $L < 2\sigma/\rho V^2 << R_d$. Скорость удара капель о поверхность обтекаемого тела как правило значительно меньше скорости набегающего потока в силу ограниченности значений числа Стокса $Stk = \frac{\tau_{Rel}}{R/V_\infty} = \frac{2}{9}\frac{\rho V_\infty R_d^2}{\mu R}$. Здесь V_∞ – скорость потока, обтекающего тело с характерным размером R, μ – коэффициент динамической вязкости газа. Скорость V удара переохлажденных капель связана со скоростью обтекающего потока V_∞ соотношением $V/V_\infty \cong 1 - exp(-1/Stk)$. Имеем $L_{max}/2D = ([1 - exp(-Stk^{-1})]We_\infty)^{-1}$. Так, например, при радиусе капель 20 микрометров (типичный размер капель при обледенении [14]), скорости обтекания $V_\infty = 100$ м/c и радиусе передней кромки крыла $R_{wing} = 0.1$ м, имеем $Stk \cong 5 \cdot 10^{-3}$, $We_\infty = D\rho_l V_\infty/\sigma_l \cong 55$, получим $L_{max} \cong 0.72$ мкм. На рис. 2 показана зависимость максимального значения расстояния между неровностями рельефа поверхности от радиуса передней кромки крыла

3. Особенности кристаллизации переохлажденной метастабильной жидкости, математические модели и численные оценки характерных физических величин

Десятки лет назад была подмечена общность процессов, сопровождающихся преодолением барьера между двумя состояниями вещества, например, при химических реакциях, термоядерных превращениях, электрических и оптических разрядах, образовании твердых растворов, кристаллизации метастабильной жидкости, спекании порошков металлов. Перемещающаяся поверхность раздела фаз или реагирующих компонентов обладает специфическими свойствами [15–17]. Скорость распространения фронта химической реакции или фазовых превращений для всех перечисленных процессов зависит от значения энергетического барьера L_b [Дж/кг], значения которого для случая кристаллизации воды определены в предшествующих работах (напр., [1]). Скорость перемещения фронта кристаллизации в переохлажденной жидкости (температурные зависимости которой получены в [7]), описывается следующим (достаточно общим для упомянутых выше процессов) соотношением: $u^2 = \chi\left(T/T_f\right)^m Zexp(-L_b/RT)\left(\left(T_f - T\right)/T_f\right)^{m+1}$. Здесь $\chi = \lambda/\rho_l C_p$ – температуропроводность вещества перед фронтом (λ – коэффициент теплопроводности, C – удельная теплоемкость), индекс b означает *barrier* – барьер при

фазовом переходе. В качестве характерной температуры принято значение температуры фазового перехода (для воды $T_f = 273\,K$). Второй, третий и четвертый сомножители в выражении соответствуют формуле Аррениуса; R – удельная газовая постоянная. Последний сомножитель учитывает, что при $T = T_f$, имеем $u = 0$; показатель степени m в случае химической реакции представляет ее порядок. Отметим, что в условиях больших ускорений (например, при ударной встряске) физико-механические характеристики жидкости и твердого тела могут существенно отличаться от справочных значений, полученных в квазистатических измерениях. Фазовые переходы приводят к появлению потока энергии на межфазной границе [15–17].

На основании полученных ранее результатов [1, 5–7] в настоящей работе получены количественные оценки энергетического вклада основных физических механизмов в суммарную плотность потока энергии на межфазной границе при движении фронта кристаллизации. Оценим характерную длину волны излучения, которое сопровождает кристаллизацию: приравняем энергию фазового перехода, приходящуюся на одну молекулу, к энергии излучения: $Lm_{H_2O} = L\,\mu_{H_2O}/N_A = 2\pi\hbar c/\lambda^*$, откуда $\lambda^* = 2\pi\hbar N_A c/\mu_{H_2O}L \cong 20$ мкм ($\hbar$ – постоянная Планка, c – скорость света, μ_{H_2O} – молярная масса воды).

После механического воздействия (направленного удара) часть молекул приобретает кинетическую энергию, достаточную для преодоления потенциального барьера между локальным и глобальным минимумами энергии межмолекулярного взаимодействия. При попадании молекулы в глобальный минимум энергии взаимодействия, потенциальная энергия молекул переходит в кинетическую энергию, которая передается соседним молекулам, выводя их из локального минимума энергии межмолекулярного взаимодействия. Плотность потока энергии на межфазной границе q при кристаллизации переохлажденной воды можно найти из уравнения баланса энергии:

$$q\Delta S\Delta t = C\Delta m\big(T_f - T\big) - L\Delta m\alpha_m = \big(C\big(T_f - T\big) - L\alpha_m\big)\rho\Delta S u\Delta t, \qquad (1)$$

$$\text{откуда } q = \rho u\big(C\big(T_f - T\big) - L\alpha_m\big).$$

Здесь Δm – элемент массы, ΔS – элемент площади, Δt – малый промежуток времени, C – удельная теплоемкость воды.

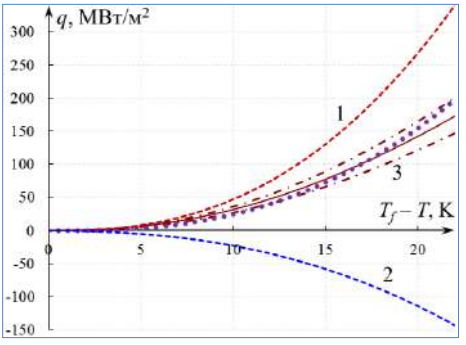


Рис. 3. Оценка зависимости энергии фазового перехода на фронте кристаллизации. 1 – первое слагаемое в правой части выражения (3), 2 – второе слагаемое в правой части выражения (3), 3 – сплошная линия – оценка плотности потока энергии на межфазной границе по формуле (1), штрих-пунктирные линии показывают отклонение от среднего значения (3); кружками обозначена кривая зависимсости плотности потока энергии на межфазной границе по формуле (3) при толщине межфазной границы ΔX = 2.5 мкм

Fig. 3. Estimation of the dependence of the phase transition energy at the crystallization front. 1 - the first term on the right side of expression (3), 2 - the second term on the right side of expression (3), 3 - solid line - estimate of the energy flux density at the interface according to formula (1), dash-dotted lines show deviation from the mean values (3); the circles indicate the curve of the dependence of the energy flux density at the interface according to formula (3) with the thickness of the interface ΔX = 2.5 μm

Другой подход к оценке потока излучения при фазовом переходе дает следующее выражение типа граничного условия Стефана [18]:

$$\left(\oiint_S qdSn\right) \Big/ \oiint_S dS + \lambda_l \partial T_l/\partial n - \lambda_{ls}\partial T_{ls}/\partial n = L_{ls}\rho_l \alpha_m u. \quad (2)$$

Здесь $\lambda_{ls}/\lambda_l = (1+\alpha_m\rho_l/\rho_s)^m$ – отношение коэффициентов теплопроводности водно-кристаллической смеси и воды. Индексы l и s означают *liquid* и *solid* соответственно. Первое слагаемое в выражении (2) представляет собой поток энергии на межфазной границе при движении фронта кристаллизации. Если рассматривать задачу о распространении фронта кристаллизации в одномерном приближении, то выражение (2) примет следующий вид:

$$q(T) \cong L_{ls}\rho_l\alpha_m u - \lambda_l \lfloor (1+\alpha_m\rho_l/\rho_s)^m - 1 \rfloor \left(T_f - T\right)/\delta. \quad (3)$$

Здесь $\delta_c \cong \Delta X = a_s - a_l = \sqrt[3]{\mu_m/N_A}\left(\sqrt[3]{2/\rho_s} - \sqrt[3]{2/\rho_l}\right) \cong 0.92$ Å; $n = 4$;
$\tau \cong \Delta X/u(T) \cong 1.3$ нс – характерное время поворота молекулы при фазовом переходе. На рис. 3 показаны оценки температурных зависимостей потока энергии на межфазной границе при кристаллизации переохлажденной метастабильной жидкости.

4. Заключение

Развит численный алгоритм, позволяющий проводить расчеты взаимодействия капель с твердым телом в диапазоне размеров от 10^{-9} до 10^{-3} м. Получены оценки максимальных значений характерных размеров рельефа гидрофобного покрытия твердого тела в зависимости от радиуса передней кромки обтекаемого тела, который определяет скорость удара капель о поверхность тела при заданной скорости обтекающего потока. На основании измеренных ранее параметров физических процессов, сопровождающих кристаллизацию переохлажденной метастабильной жидкости, получены численные оценки плотности потока энергии на межфазной границе – фронте кристаллизации переохлажденных метастабильных капель.

Список литературы / References

[1]. Amelyushkin I.A., Stasenko A.L. Interaction of supercooled droplets and nonspherical ice crystals with a solid body in a mixed cloud. CEAS Aeronautics Journal, vol. 9, no. 4, 2018, pp. 711-720.
[2]. Amelyushkin I.A., Stasenko A.L. Simulation of gas-dispersed flow particles' interaction with a solid body. Journal of Physics: Conference Series, vol. 1560, 2020, article no, (012064).
[3]. Cahn J.W., Hilliard J.E. Free energy of a nonuniform system. I. Interfacial free energy. Journal of Chemical Physics, vol. 28, issue 2, 1958, pp. 258-267.
[4]. Li S., Lowengrub J.S., Leo P.H., Cristini V. Nonlinear stability analysis of self-similar crystal growth: control of the Mullins-Sekerka instability. Journal of Crystal Growth, vol. 277, issues 1-4, 2005, pp. 578-592.
[5]. Аверков В.А., Желтов М.А., Скворцов В.В., Шибков А.А. Морфологическая неустойчивость межфазной границы лед-вода. Труды Второго международного симпозиума «Плавление и кристаллизация металлов и оксидов, 2009, стр. 179-182 / Averkov V.A., Zheltov M.A., Skvortsov V.V., Shibkov A.A. Morphological instability of the ice-water interface. In Proc. of the Second International Meeting on Melting and Crystallization of Metals and Oxides, 2009, pp. 179-182 (in Russian).
[6]. Amelyushkin I.A. Mathematical models of two-phase flows' interaction with a solid body. Journal of Journal of Physics: Conference Series, vol. 1129, 2018, article no. 012003.
[7]. Amelyushkin I.A., Stasenko A.L., Zhbanov V.A. Experimental investigation, mathematical and numerical simulation of aircraft icing phenomena. In Proc. from the 31th Congress of the International Council of the Aeronautical Sciences, 2018, paper no. ICAS2018_0741.
[8]. Aboud D.G.K., Kietzig A.M. On the oblique impact dynamics of drops on superhydrophobic surfaces. Part II: Restitution coefficient and contact time. Langmuir, vol. 34, issue 2018, pp. 9889-9896.
[9]. Jung S., Tiwari M.K., Doan N.V., Poulikakos D. Mechanism of supercooled droplet freezing on surfaces.

Nature Communications, vol. 3, 2012, article no. 615.
[10]. Хргиан А.Х. Физика атмосферы. Физматлит. 1958 г., 476 стр. / Khrgian A.Kh. Physics of the atmosphere. Fizmatlit, 1958, 476 p. (in Russian).
[11]. Скрипов В.П. Метастабильная жидкость. Наука. 1972 г., 312 стр. / Skripov V.P. Metastable liquids. Nauka, 1972, 312 p. (in Russian).
[12]. Вигасин А.А., Юхневич Г.В. Использование данных спектроскопии в структурных моделях жидкой воды. Теплофизические свойства веществ и материалов, вып. 21, 1984 г., стр.13-31 / Vigasin A.A., Yukhnevich G.V. Using spectroscopic data in structural models of liquid water. Thermophysical properties of substances and materials, issue. 21, 1984, pp. 13-31 (in Russian).
[13]. Радченко И.В. Молекулярная физика. Наука, 1965 г., 480 стр. / Radchenko I.V. Molecular physics. Nauka, 1965, 480 p. (in Russian).
[14]. Авиационные правила. Часть 25. Нормы летной годности самолетов транспортной категории (утверждено Постановлением 28-й сессии Совета по авиации и использованию воздушного пространства от 11.12.2008). Приложение С. URL: https://legalacts.ru/doc/aviatsionnye-pravila-chast-25-normy-letnoi-godnosti-samoletov-transportnoi, 01.09.2020 / Aviation regulations. Part 25. Standards of airworthiness of aircraft of the transport category (approved by the Resolution of the 28th session of the Council for Aviation and Airspace Use, dated 11.12.2008) Appendix C (in Russian).
[15]. Gimelshein N., Lyons R., Reuster J., Gimelshein S. Numerical prediction of UV radiation from two-phase plumes at high altitudes. AIAA journal, vol. 46, no. 7, 2008, pp. 1764-1772.
[16]. Татарченко В.А. Инфракрасное характеристическое излучение фазовых переходов первого рода и его связь с оптикой атмосферы. Оптика атмосферы и океана, том 23, no. 3, 2010 г., стр. 169-175 / Tatarchenko V.A. Infrared characteristic radiation of first-order phase transitions and its relationship with atmospheric optics. Optics of Atmosphere and Ocean, vol. 23, no. 3, 2010, pp. 169-175 (in Russian).
[17]. Шавлов А.В. Электрический потенциал на фронте кристаллизации воды и растворов. Роль протонов и ориентационных дефектов. Журнал физической химии, том 79, no. 9, 2005 г., стр. 1626-1630 / Shavlov A.V. Electric Potential at the Front of Crystallization of Water and Solutions: The Role of Protons and Orientational Defects. Russian Journal of Physical Chemistry A, vol. 79, no. 9, 2005, pp. 1438-1442.
[18]. Stefan J. Über die Theory der Eisbildung, insbesondere über die Eisbildung in Polarmeere. Annalen der Physik, vol. 278, issue 2, pp.269-286 (in German).

Information about authors / Информация об авторах

Иван Алексеевич АМЕЛЮШКИН – кандидат физико-математических наук, старший научный сотрудник ЦАГИ, преподаватель кафедры молекулярных процессов и экстремальных состояний вещества физического факультета МГУ. Научные интересы: динамика многофазных сред, высокопроизводительные вычисления, обратные задачи и их приложения, молекулярное моделирование и многомасштабный анализ.

Ivan Alekseevich AMELYUSHKIN – Candidate of Physical and Mathematical Sciences, Senior Researcher at TsAGI, Lecturer at the Department of Molecular Processes and Extreme States of Matter at the Physics Faculty of Moscow State University. Research interests: dynamics of multiphase environments, high performance computing, inverse problems and their applications, molecular modeling and multiscale analysis.

Максим Александрович КУДРОВ – кандидат технических наук, доцент, ведущий научный сотрудник, заведующий лабораторией информационных технологий и прикладной математики. Научные интересы: разработка комплекса программ для компьютерного моделирования физических процессов.

Maksim Aleksandrovich KUDROV – Candidate of Technical Sciences, Associate Professor, Leading Researcher, Head of the Laboratory of Information Technologies and Applied Mathematics. Research interests: development of software for computer modeling of physical processes.

Алексей Олегович МОРОЗОВ – младший научный сотрудник. Научные интересы: компьютерное моделирование физических процессов, численные методы в задачах аэрогидродинамики,

Alexey Olegovich MOROZOV – Junior Researcher. Research interests: computer modeling of physical processes, numerical methods in problems of aerohydrodynamics,

Альберт Леонидович СТАСЕНКО – доктор технических наук, профессор МФТИ, главный научный сотрудник ЦАГИ. Научные интересы: физическая механика многофазных потоков, математическое моделирование физических явлений, компьютерное моделирование.

Albert Leonidivich STASENKO - Doctor of Technical Sciences, Professor of the Moscow Institute of Physics and Technology, Chief Researcher at TsAGI. Research interests: physical mechanics of multiphase flows, mathematical modeling of physical phenomena, computer modeling.

Андрей Сергеевич ЩЕГЛОВ – инженер ЦАГИ, аспирант МФТИ. Научные интересы: численные методы, динамика полета, вихревые методы расчета аэродинамических характеристик летательных аппаратов.

Andrey Sergeevich SHCHEGLOV –engineer at TsAGI, post-graduate student at MIPT. Research interests: numerical methods, flight dynamics, vortex methods for calculating the aerodynamic characteristics of aircraft.

DOI: 10.15514/ISPRAS–2020–32(4)–18

Совершенные множества путей в полном графе коммутаторов SDN-сети

[1] *И.Б. Бурдонов, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>*
[2] *Е.М. Винарский, ORCID: 0000-0002-7328-0942 <vinevg2015@gmail.com>*
[1,3] *Н.В. Евтушенко, ORCID: 0000-0002-4006-1161 <evtushenko@ispras.ru>*
[1] *А.С. Косачев, ORCID: 0000-0001-5316-3813 <kos@ispras.ru>*

[1] *Институт системного программирования РАН им. В.П. Иванникова, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский государственный университет им. М.В. Ломоносова, 119234, Россия, г. Москва, Ленинские Горы, 1, д. 1*
[3] *Национальный исследовательский университет «Высшая школа экономики», 101000, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В статье исследуется задача виртуализации сети на плоскости данных программно-конфигурируемой сети, моделируемой графом физических связей между узлами сети. Виртуальная сеть задается как множество упорядоченных пар хостов (отправитель, получатель), а реализуется множеством путей хост-хост, однозначно определяющим настройки коммутаторов. Множество путей совершенное, если любое подмножество связываемых им пар хостов связывается соответствующим подмножеством путей без возникновения бесконечного движения пакетов по циклу, без дублирующих путей, когда хост получает один и тот же пакет несколько раз, и без непредусмотренных путей, когда хост получает пакет, ему не предназначенный. Для случая, когда подграф, порождённый коммутаторами, является полным графом, устанавливаются достаточные условия существования наибольшего совершенного множества путей, связывающего все пары различных хостов. Предлагаются алгоритмы построения такого наибольшего совершенного множества и даются оценки их сложности. Приводятся результаты компьютерных экспериментов.

Ключевые слова: программно-конфигурируемые сети; виртуализация сети; совершенные множества путей; полный граф коммутаторов

Для цитирования: Бурдонов И.Б., Винарский Е.М., Евтушенко Н.В., Косачев А.С. Совершенные множества путей в полном графе коммутаторов SDN-сети. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 245–260. DOI: 10.15514/ISPRAS–2020–32(4)–18

Благодарности. Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект 17-07-00682-а.

Perfect sets of paths in the full graph of SDN network switches

[1] *I.B. Burdonov, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>*
[2] *E.M. Binarskii, ORCID: 0000-0002-7328-0942 <vinevg2015@gmail.com>*
[1,3] *N.V. Yevtushenko, ORCID: 0000-0002-4006-1161 <evtushenko@ispras.ru>*
[1] *A.S. Kossatchev, ORCID: 0000-0001-5316-3813 <kos@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, 1, Lenin mountains, Moscow, 119234, Russia*
[3] *National Research University Higher School of Economics, 20, Myasnitskaya st., Moscow, 101000, Russia*

Abstract. The paper investigates the implementation of virtual networks on the SDN data plane which is modeled by a graph of physical connections between network nodes. A virtual network is defined as a set of ordered host pairs (sender, receiver), and it is implemented by a set of host-host paths that uniquely determine the switch settings. A set of paths is perfect if any subset of its paths can be loop-free implemented, i.e., can be implemented without the occurrence of an endless movement of packets in a loop, without duplicate paths, when the host receives the same packet several times, and without unintended paths when the host receives the packet that was directed to another host. For the case when the switchgraph is a complete graph, sufficient conditions for the existence of the largest perfect set of paths connecting all pairs of different hosts are established. Algorithms for constructing such a largest perfect set are proposed with the estimates of their complexity. The paper also has the preliminary results of computer experiments which show that proposed sufficient conditions are not necessary conditions.

Keywords: software defined networking; network virtualization; perfect sets of paths; complete graph of switches

For citation: Burdonov I.B., Binarskii E.M., Yevtushenko N.V., Kossatchev A.S. Perfect sets of paths in the full graph of SDN network switches. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 245–260 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–18

Acknowledgments. This work was supported by the Russian Foundation for Basic Research, project 17-07-00682-a.

1. Введение

Одной из основных технологий виртуализации [1-5] в настоящее время являются программно-конфигурируемые сети (SDN) с разделенными плоскостями данных и управления. Пакеты между хостами пересылаются на плоскости данных через промежуточные коммутаторы, система правил которых (настройка) осуществляется специальными SDN-контроллерами. Правило определяет, каким соседним узлам пересылается принятый коммутатором пакет в зависимости от того, откуда пришел пакет, и от вектора параметров в заголовке пакета [6]. Иными словами, настройка коммутаторов определяет множество путей от хоста к хосту, по которым и будут пересылаться пакеты. Ситуация может быть промоделирована с использованием графа физических связей, вершинами которого являются хосты и коммутаторы, а ребра соответствуют физическим связям между ними, при этом каждый хост может быть соединен только с одним коммутатором.

Соответственно в таком графе возникают задачи двух уровней. 1) Возможно ли и если да, то каким образом, реализовать заданное множество путей хост-хост через подходящие настройки коммутаторов? 2) Возможно ли и если да, то каким образом, реализовать заданное множество пар (хост, хост) через подходящие пути хост-хост в графе физических связей?

Известно, что при решении первой задачи возникают три эффекта, которые не позволяют реализовать произвольное множество путей хост-хост через подходящие настройки

коммутаторов. Во-первых, как показано в [4-5], при реализации множества реберно-простых путей хост-хост на панели данных могут появляться пути хост-хост, которых нет в заданном множестве, в том числе, могут появиться циклы, по которым пакеты будут передаваться бесконечно и, соответственно, бесконечно размножаться. Кроме того, могут появиться дублирующие пути, из-за чего хост-адресат получает один и тот же пакет не один, а несколько раз, и этот вопрос с точки зрения его полезности и нежелательных эффектов исследуется в [7].

Вторая задача сводится к первой задаче выбором подходящего множества путей, по возможности избегая описанных выше ситуаций, и таким образом, возникает вопрос (2a), можно ли заданное множество пар хостов реализовать на графе, т.е. выбрать подходящее множество путей, по возможности без нежелательных эффектов? На следующем шаге возникает вопрос (2b), можно ли любое множество пар хостов реализовать без указанных выше эффектов на данном графе физических связей?

В [7] показано, что любое множество пар хостов можно реализовать на связном графе без возникновения циклов и дублирования (ответ на вопрос 2a), однако могут возникать пути, соединяющие непредусмотренные пары хостов. Также показано, что в некоторых случаях реализация заданного множества пар хостов без непредусмотренных путей неизбежно приводит к возникновению дублирования или циклов.

В [7] также устанавливается достаточное условие положительного ответа на вопрос (2b). Это достаточное условие опирается на понятие наибольшего совершенного множества путей. Множество путей хост-хост называется совершенным, если никакие два пути, ведущие из разных начальных хостов в разные конечные хосты, не проходят по одному ребру графа физических связей в одном и том же направлении. Любое подмножество множества пар хостов, связываемых совершенным множеством путей, реализуется соответствующим подмножеством путей без «зацикливания», дублирования и непредусмотренных путей. Наибольшее совершенное множество путей связывает все пары разных хостов и, тем самым, любое множество пар разных хостов реализуется без «зацикливания», дублирования и непредусмотренных путей.

В данной статье исследуется вопрос о существовании наибольшего совершенного множества путей для случая, когда подграф, порождённый коммутаторами, является полным графом, т.е. для любой пары коммутаторов существует физическая связь. Устанавливаются достаточные условия существования наибольшего совершенного множества путей, предлагаются два алгоритма построения такого множества и даются их оценки (раздел 4). Эти алгоритмы опираются на два способа построения совершенного множества путей для случая, когда подграф, порождённый коммутаторами, является графом-звездой (раздел 3). Показано, что максимальное (по числу связываемых пар хостов) совершенное множество путей для графа-звезда строится одним из этих двух алгоритмов, выбираемых в зависимости от соотношения числа коммутаторов и числа хостов, подсоединенных к каждому коммутатору. Приводятся результаты компьютерных экспериментов (раздел 5). В частности, получен следующий результат. Как известно [7], наличие совершенного множества путей является достаточным условием для того, чтобы каждое множество пар различных хостов имело строгую реализацию без циклов и дублирования. Необходимость этого условия принималась как гипотеза, которая должна быть доказана или опровергнута. Эксперименты показали, что эта гипотеза не верна: для SDN-сети максимальное совершенное множество может не существовать, однако каждое множество пар различных хостов имеет строгую реализацию без циклов и дублирования.

2. Основные понятия

Графом физических связей (далее просто графом) будем называть связный неориентированный граф $G = \{V, E\}$ без кратных ребер и петель, где V – множество

коммутаторов и хостов, $E \subseteq V \times V$ – множество ребер, моделирующих физические связи между коммутаторами и между коммутаторами и хостами. Если не оговорено противное, под графом будет пониматься именно такой граф. Поскольку ребро, соединяющее вершины a и b, неориентированное и нет кратных ребер, его можно обозначать как ab, так и ba. Поскольку нет петель, ребер вида aa в E нет. Поскольку нет кратных ребер, путь как последовательность смежных ребер однозначно задается последовательностью вершин $a_1 \dots a_n$, через которые он проходит. Путь, начинающийся в вершине a и заканчивающийся в вершине b, называют ab-путем. Если путь проходит по ребру ab из a в b, то будем говорить, что он проходит дугу ab. Если a и b хосты, ab -путь, в котором все вершины, кроме первой вершины a и последней вершины b, коммутаторы, будем называть *полным*. Путь, в котором вершины (дуги) не повторяются, называется *вершинно-простым* (*реберно-простым*). Вершины графа будем обозначать строчными буквами $a, b, c, \dots, x, y, z$, пути – жирными строчными буквами $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}$, …, а множества путей – прописными буквами – P, Q, R, …

Мы будем предполагать, что каждый хост x подсоединен ровно к одному коммутатору [3]. Поэтому хост – это терминальная вершина графа, т.е. вершина степени 1. Коммутатор, к которому не подсоединены хосты, будем называть *пустым* коммутатором. Если коммутатор a имеет степень 1 и соединен с вершиной b, то любой полный путь, проходящий через a, имеет вид $\dots bab \dots$; удаляя из него все циклы bab, получаем путь, не проходящий через a. Это значит, что такой коммутатор «лишний», и достаточно рассматривать графы, в которых терминальные вершины – это только хосты. Множества хостов и коммутаторов обозначим через H и S, соответственно; $H \cup S = V, H \cap S = \varnothing$.

В общем случае правило коммутатора b имеет вид σabc, где a и c соседи b, а σ вектор параметров заголовка пакета, которые можно использовать в правилах. Такое правило означает, что коммутатор b, получив пакет с вектором σ от соседа a, пересылает его соседу c. Предполагается, что коммутатор не меняет σ. Тем самым, для вектора σ порождаются полные пути вида $a_1 \dots a_n$, где для $i = 2..n - 1$ в коммутаторе a_i есть правило $\sigma a_{i-1} a_i a_{i+1}$. Если есть два правила σabc и $\sigma abc`$, где $c \neq c`$, говорят, что пакет *клонируется*, т.е. пересылается обоим соседям c и $c`$.

Заданное множество P полных путей однозначно определяет минимальный набор правил коммутаторов, порождающий все пути из P [5]. Однако это не значит, что порождаются только пути из P. Будем говорить, что два пути *сливаются* на дуге ab в вершине a, если у них дуга ab общая и не первая, а непосредственно предшествующие ей дуги ca и $c`a$ разные (т.е. $c \neq c`$), и *разделяются* после дуги de в вершине e, если у них дуга de общая и не последняя, а непосредственно следующие дуги ef и $ef`$ разные (т.е. $f \neq f`$).

Цикл порождается, если полный xy-путь проходит через некоторую дугу дважды, т.е. путь имеет вид $\boldsymbol{paqer}(\boldsymbol{aqer})^{*}\boldsymbol{aqes}$, где отрезок $\boldsymbol{p}$ начинается в хосте $x \neq a$, отрезки $\boldsymbol{p}$ и $\boldsymbol{r}$ не заканчиваются в одной вершине, после этих отрезков в пути следует коммутатор a, отрезок $\boldsymbol{aqer}$ проходится один или несколько раз, после коммутатора e отрезки $\boldsymbol{r}$ и $\boldsymbol{s}$ не начинаются в одной вершине, и отрезок $\boldsymbol{s}$ заканчивается в хосте $y \neq e$. Двигаясь вдоль пути, мы видим, что в вершине a путь сливается сам с собой, потом в вершине e разделяется сам с собой, а затем это слияние и разделение происходит ещё раз (если путь проходит цикл k раз, то будет $k + 1$ раз как разделение после слияния, так и слияние после разделения). Пакеты будут не только бесконечно ходить по циклу $\boldsymbol{aqer}$, но и бесконечно клонироваться в вершине e, так что хост y будет получать бесконечное число клонов пакета.

Путь, который не сливается сам с собой, это реберно-простой путь. Для отсутствия циклов необходимо, чтобы все пути множества P были реберно-простыми. Но этого недостаточно. Если два реберно-простых полных пути из P после слияния на дуге ab разделяются (после этой же или другой дуги), т.е. имеют вид $x\boldsymbol{p}ab\boldsymbol{q}y$ и $x`\boldsymbol{p}`ab\boldsymbol{q}`y`$ с разными начальными и конечными хостами $x \neq x`$ и $y \neq y`$, то порождаются и новые пути $x\boldsymbol{p}ab\boldsymbol{q}`y`$ и $x`\boldsymbol{p}`ab\boldsymbol{q}y$. Эта операция порождения новых путей называется замыканием по дугам, а результат замыкания

по дугам всех пар путей из P обозначается $P{\downarrow}{\uparrow}$[4-5]. Очевидно, $P \subseteq P{\downarrow}{\uparrow}$. Если $P \neq P{\downarrow}{\uparrow}$, т.е. P не замкнуто по дугам, то возникают непредусмотренные пути. В частности, могут возникнуть не реберно-простые пути и, следовательно, циклы. Появление циклов в замыкании по дугам множества полных путей всегда свидетельствует о бесконечности этого замыкания и, тем самым, наличии дублирования. В конечном замкнутом по дугам множестве полных реберно-простых путей циклов нет.

Для множества полных путей P через $H(P) \subseteq H{\times}H$ обозначим множество пар xy, для которых в P есть xy-путь. Множество пар хостов $D \subseteq H{\times}H$, не содержащее пар вида xx, будем называть *нормальным*. Будем говорить, что нормальное множество D (*нестрого*) *реализуется* замкнутым по дугам множеством полных путей P, если $D \subseteq H(P)$, и, кроме того, *реализуется без циклов*, если P конечно, *строго реализуется*, если $D = H(P)$, *реализуется без дублирования*, если P содержит ровно один xy-путь для каждой пары $xy \in D$.

Если адрес хоста-отправителя входит в вектор параметров σ, то правила для векторов параметров с разными адресами хоста-отправителя работают независимо друг от друга. Для каждого хоста-отправителя x в графе можно выбрать исходящее из x дерево I_x кратчайших путей, ведущих во все остальные хосты. Для любого нормального множества D пар хостов и любого хоста x выбирается подмножество D_x пар, где первый элемент пары – это хост x, а в дереве I_x – поддерево $I_x(D)$, в котором листовые вершины – это вершины y такие, что $xy \in D_x$. В исходящем дереве все пути реберно-простые (даже вершинно-простые), и нет слияния, тем самым, нет и разделения после слияния. Поэтому $I_x(D)$ замкнуто по дугам и, очевидно, строго реализует D_x без циклов и дублирования, причём используются кратчайшие полные пути. Таким образом, в этом случае нет проблем со строгой реализацией без циклов и дублирования любого нормального множества пар хостов. Более того, реализация любого такого множества оказывается подмножеством одного и того же множества путей – объединения деревьев I_x по всем хостам x. Аналогичная процедура с аналогичным результатом применима тогда, когда в вектор параметров σ входит адрес хоста-получателя. Только здесь строится входящее дерево O_x для каждого хоста x.

Ниже мы рассматриваем случай, когда адрес хоста-отправителя и адрес хоста-получателя не входят в вектор параметров σ. Остальные параметры никак не влияют на перемещение пакетов с данным вектором σ, поэтому мы будем опускать σ в обозначении правила и писать вместо σ*abc* просто *abc*.[1] Вектор σ определяет идентификатор потока, в котором передаются векторы с такими параметрами. Иными словами, правила коммутатора (для данного вектора σ) определяют, кому должен быть послан пакет, только в зависимости от соседа, от которого пакет принят. В этом случае число правил, по которым работает коммутатор, зависит только от числа его соседей и не зависит от числа хостов в сети.

В [7] доказаны следующие утверждения.

1) На связном графе G любое нормальное множество D пар хостов нестрого реализуется без циклов и дублирования.
2) Любое множество D пар хостов, строго реализуемое без циклов, может быть строго реализовано множеством вершинно-простых путей.
3) Строгая реализация не всегда возможна без дублирования: существует граф, на котором некоторое нормальное множество пар хостов строго реализуется только с дублированием.
4) Строгая реализация не всегда возможна без циклов: существует граф, на котором некоторое нормальное множество пар хостов строго реализуемо, но только бесконечными замкнутыми по дугам множествами путей.

[1] В настоящей работе рассматриваются пакеты потока с одним идентификатором, и соответственно, при описании правил и путей опускается вектор параметров σ.

5) Если множество полных путей конечно, то отсутствие разделения после слияния достаточно, но не необходимо для замкнутости по дугам и отсутствия циклов.
6) Для замкнутого по дугам множества P полных путей условие отсутствия слияния после разделения необходимо и достаточно для отсутствия дублирования.

Также в [7] дано определение совершенного множества путей и доказано соответствующее утверждение.

Множество P путей называется *совершенным*, если оно конечно, содержит только полные пути, и в нем нет как разделения после слияния путей, т.е. нет двух путей вида $\boldsymbol{p}ab\boldsymbol{q}$ и $\boldsymbol{p}`ab\boldsymbol{q}`$, где $\boldsymbol{p} \neq \boldsymbol{p}`$и $\boldsymbol{q} \neq \boldsymbol{q}`$,так и слияния после разделения путей, т.е. нет двух путей вида $\boldsymbol{p}ab\boldsymbol{q}cd\boldsymbol{r}$ и $\boldsymbol{p}`ab\boldsymbol{q}`cd\boldsymbol{r}`$, где $\boldsymbol{q} \neq \boldsymbol{q}`$. Если при этом множество $H(P)$ содержит все пары разных хостов (т.е. является наибольшим нормальным множество пар хостов), то множество P наибольшее совершенное множество. Заметим, что отсутствие разделения после слияния эквивалентно отсутствию дублирующих путей.

7) Совершенное множество путей для каждого нормального множества пар хостов содержит его строгую реализацию без циклов и без дублирования как своё подмножество.

3. Совершенное множество путей в графе-звезда

Граф-звезда – это граф, в котором выделена одна вершина – центр звезды, которая соединена ребрами со всеми остальными вершинами графа, и других ребер в графе нет. В этом разделе мы рассматриваем графы, в которых подграф, порожденный коммутаторами, является графом-звездой, и центр – пустой коммутатор (к нему не подсоединены хосты). Центр звезды обозначим через s_0, а остальные коммутаторы – через s_1, s_2, …, s_k,где k – число коммутаторов, не считая центра звезды. Хосты, подсоединенные к одному коммутатору, будет обозначать 1, 2, … Исследуем вопрос о максимальном совершенном множестве P путей в таком графе, где максимальность понимается как максимальное число связываемых им пар хостов, т.е. максимум $|H(P)|$.

Замечание 1. Пусть множество P путей совершенное. Удалим из P все пути, соединяющие хосты, подсоединенные к одному коммутатору, а потом добавим все пути вида $hs_ih`$, где $h$ и $h`$ разные хосты, подсоединенные к одному коммутатору s_i. Множество P останется совершенным.

С учетом этого замечания, а также утверждения 2 из [7], достаточно рассматривать совершенные множества вершинно-простых путей, соединяющих хосты, подсоединенные к разным коммутаторам. Хосты, подсоединенные к одному коммутатору, всегда можно дополнительно соединить путями, не проходящими по ребрам, соединяющим коммутаторы. Число таких дополнительных путей для k непустых коммутаторов, к каждому из которых подсоединено $n > 1$ хостов, равно $kn(n - 1)$.

Рассмотрим два способа соединения хостов в графе.

Способ 1: в одном непустом коммутаторе выделяем один хост, для определенности в коммутаторе s_1 выделяем хост 1, и соединяем его со всеми хостами h, подсоединенными к другим коммутаторам, путями вида $1s_1s_0s_ih$ и $hs_is_0s_11$. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число пар соединяемых хостов $N_1 = 2n(k - 1)$.

Способ 2: в каждом непустом коммутатореs_i выделяем один хост 1 и соединяем путями вида $1s_is_0s_j1$ все выделенные хосты каждый с каждым. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число пар соединяемых хостов $N_2 = k(k - 1)$.

Утверждение 1. Для графа, в котором подграф, порожденный коммутаторами, является графом-звездой, центр звезды – пустой коммутатор, а к каждому из остальных k коммутаторов подсоединено по n хостов, максимальное совершенное множество путей

строится способом 1 или способом 2 в зависимости от того, какая величина больше: $2n(k-1)$ или $k(k-1)$.

Доказательство. Сначала нужно показать, что множество путей, которое строится по способу 1 или 2 совершенное. Но это непосредственно следует из построения: число путей конечно, все пути полные, любые два пути либо только сливаются, либо только разделяются.

Теперь докажем остальную часть утверждения.

В совершенном множестве путей пути, проходящие через одну дугу, либо разделяются, либо сливаются. Поэтому для данных i и j, $i \neq j$, все пути вида $hs_is_0s_jh`$ либо начинаются в одном хосте $h$ (разделение), либо заканчиваются в одном хосте $h`$ (слияние). Пусть число таких путей равно m. Если $m = 0$, $m = 1$ или $m > 1$ и имеет место разделение, то обозначим $\boldsymbol{m}(i, j) = m$, если $m > 1$ и имеет место слияние, то обозначим $\boldsymbol{m}(i, j) = -m$. В случае разделения $\boldsymbol{m}(i, j) > 1$ должно быть $\boldsymbol{m}(i`, j) = 0$ для $i` \neq i$, а в случае слияния $\boldsymbol{m}(i, j) < -1$ должно быть $\boldsymbol{m}(i, j`) = 0$ для $j` \neq j$. Кроме того, в случае разделения $\boldsymbol{m}(i, j) > 1$ мы можем, сохраняя совершенность множества путей, для каждого хоста $h`$, подсоединенного к коммутатору $s_j$, добавить путь $hs_is_0s_jh`$, если такого пути не было. Соответственно, в случае слияния $\boldsymbol{m}(i, j) < -1$ мы можем, сохраняя совершенность множества путей, для каждого хоста h, подсоединенного к коммутатору s_i, добавить путь $hs_is_0s_jh`$, если такого пути не было. Это значит, что в максимальном совершенном множестве путей $\boldsymbol{m}(i, j) > 1 \Rightarrow \boldsymbol{m}(i, j) = n$ и $\boldsymbol{m}(i, j) < -1 \Rightarrow \boldsymbol{m}(i, j) = -n$.

Рассмотрим полученную матрицу $\boldsymbol{m}(i, j)$, где $i = 1, ..., k$ и $j = 1, ..., k$. Она обладает следующими свойствами:

1) На главной диагонали находятся нули, т.е. $\boldsymbol{m}(i, i) = 0$ для $i = 1, ..., k$. Это объясняется тем, что мы не рассматриваем пути, соединяющие хосты, подсоединенные к одному коммутатору.
2) Если $\boldsymbol{m}(i, j) = n$, то в остальных ячейках столбца j находятся нули. Это разделение.
3) Если $\boldsymbol{m}(i, j) = -n$, то в остальных ячейках строки i находятся нули. Это слияние.
4) Если множество совершенных путей максимально, то в остальных ячейках матрицы находятся 1.

Матрицу порядка k с этими свойствами будем называть *правильной*. Число путей в совершенном множестве равно сумме абсолютных значений во всех ячейках матрицы, т.е. $\Sigma|\boldsymbol{m}(i, j)|$. Максимум $\Sigma|\boldsymbol{m}(i, j)|$ на классе всех правильных матриц обозначим M_k. Покажем, что $M_k = max\{2n(k-1), k(k-1)\}$. Для этого сначала докажем несколько лемм. Утверждение теоремы будет непосредственно следовать из лемм 4, 5, 6.

Лемма 1. Всегда существует правильная матрица $\boldsymbol{m}$, в которой $\Sigma|\boldsymbol{m}(i, j)| = 2n(k-1)$.

Доказательство. Рассмотрим матрицу, в которой в первой строке во всех ячейках, кроме первой, размещается "n", а в первом столбце во всех ячейках, кроме первой, размещается "$-n$". Очевидно, $\Sigma\,|\,\boldsymbol{m}(i, j)\,| = 2n(k-1)$.

□

Лемма 2. Если $n > 1$ и $2 < k \leq 2n$, то в правильной матрице, для которой достигается M_k, n и $(-n)$ либо оба присутствуют, либо оба отсутствуют.

Доказательство. По лемме 1, $M_k \geq 2n(k-1)$.

Рассмотрим матрицу, для которой имеет место один из двух случаев: 1) в матрице есть n и нет $(-n)$, 2) в матрице нет n и есть $(-n)$.

Случай 1. Пусть число ячеек с n, равно $x > 0$. Поскольку в столбце, где есть n, в остальных ячейках находится 0, число столбцов, где есть n, равно x. В каждом из остальных столбцов не более $(k-1)$ ячеек с 1, а остальные ячейки с 0. Тогда $\Sigma\,|\,\boldsymbol{m}(i, j)\,| \leq xn + (k-x)(k-1) = k(k-1) + x(n-k+1)$. Если $k \leq n+1$, то $(n-k+1) \geq 0$, и $x \leq k$ влечет $k(k-1) + x(n-k+1) \leq k(k-1) + k(n-k+1) = kn$; $k > 2$ влечет $kn < 2n(k-1)$. Тем самым,

$\Sigma \mid \boldsymbol{m}(i, j) \mid < 2n\,(k - 1)$. Если $k > n + 1$, то $(n - k + 1) < 0$, и, учитывая, что $x > 0$, имеем $k\,(k - 1) + x\,(n - k + 1) < k\,(k - 1)$; $k \le 2n$ влечет $k\,(k - 1) \le 2n\,(k - 1)$. Тем самым, $\Sigma \mid \boldsymbol{m}(i, j) \mid < 2n\,(k - 1)$.

В случае 2 аналогично случаю 1 доказывается, что $\Sigma \mid \boldsymbol{m}(i, j) \mid < 2n\,(k - 1)$.

Таким образом, в обоих случаях для рассматриваемой матрицы не достигается M_k. Тем самым, лемма доказана.

□

Лемма 3. Если $n > 1$ и $2 < k < 2n$, то в матрице, для которой достигается M_k, существует столбец, в которой есть $(-n)$, и строка, в которой есть n.

Доказательство. По лемме 1, $M_k \ge 2n\,(k - 1)$, а по лемме 2 в матрице, для которой достигается M_k, n и $(-n)$ либо 1) оба присутствуют, либо 2) оба отсутствуют. Покажем, что второго случая быть не может. Действительно, если n и $(-n)$ оба отсутствуют, то на главной диагонали стоят 0, в остальных ячейках 0 или 1. Поэтому $\Sigma \mid \boldsymbol{m}(i, j) \mid \le k\,(k - 1)$. Если $k < 2n$, то $k\,(k - 1) < 2n\,(k - 1)$, поэтому $\Sigma \mid \boldsymbol{m}(i, j) \mid < 2n\,(k - 1)$. Лемма доказана.

□

Лемма 4. Если $n > 1$и $2 \le k < 2n$, то $M_k = 2n\,(k - 1)$.

Доказательство. По лемме 1, $M_k \ge 2n\,(k - 1)$. Покажем по индукции, что M_k не может быть больше $2n\,(k - 1)$.

При $k = 2$ утверждение верно, поскольку сумма чисел в ячейках равна 0, 1, 2, n, $n + 1$ или $2n$, а $n > 1$ влечет $2n(k - 1) = 2n = max\{\ 0, 1, 2, n, n + 1, 2n\ \}$. Предположим, что утверждение верно при $k = m - 1$, где $2 < m < 2n$ и покажем, что оно верно при $k = m$. По лемме 3, в матрице, для которой достигается M_m, существует столбец, в которой есть $(-n)$, и строка, в которой есть n. Без ограничения общности будем считать, что это последний столбец и последняя строка, и рассмотрим минор в левом верхнем углу порядка $(m – 1)$. Если $M_m > 2n\,(m - 1)$, то для этого минора $M_{m-1} > 2n\,(m - 1) - 2n = 2n\,(m - 2)$, что невозможно по предположению шага индукции.

□

Лемма 5. Если $n > 1$и $k = 2n$, то $M_k = 2n\,(k - 1) = k\,(k - 1)$.

Доказательство. По лемме 1, $M_k \ge 2n\,(k - 1)$. Условия $n > 1$ и $k = 2n$ влекут $k > 2$. Согласно лемме 2, в матрице, для которой достигается M_k, n и $(-n)$ либо 1) оба присутствуют, либо 2) оба отсутствуют. В первом случае существуют столбец, в котором есть $(-n)$, и строка, в которой есть n. Без ограничения общности будем считать, что это последний столбец и последняя строка, и рассмотрим минор в левом верхнем углу порядка $(m – 1)$. Заметим, что $2 < k = 2n$ влечет $2 \le k - 1 < 2n$. Поэтому по лемме 4, $M_{k-1} = 2n\,(k - 2)$, и соответственно, $\Sigma \mid \boldsymbol{m}(i, j) \mid \le M_{k-1} + 2n = 2n\,(k - 1)$. Во втором случае $\Sigma \mid \boldsymbol{m}(i, j) \mid = k\,(k - 1)$. Поскольку $k = 2n$, имеем $M_k = 2n\,(k - 1) = k\,(k - 1)$. Лемма доказана.

□

Лемма 6. Если $n > 1$, $k \ge 2n$, то $M_k = k\,(k - 1)$.

Доказательство. Известно, как построить матрицу с $M_k = k\,(k - 1)$. Эта матрица содержит все 1, кроме 0 на главной диагонали. Покажем по индукции, что M_k не может быть больше $k\,(k – 1)$. По лемме 5 это верно для $k = 2n$. Предположим, что это так при $2n \le k < m$ и покажем, что это верно при $k = m$.

Рассмотрим матрицу порядка m, в которой достигается M_m, и минор в левом верхнем углу порядка $(m - 1)$. Определим, какие возможности есть для последней строки и столбца матрицы, чтобы получить максимальное значение. Возможны четыре случая: 1) последний столбец содержит n и остальные 0, последняя строка содержит $(-n)$ и остальные 0; 2) последний столбец содержит n и остальные 0, последняя строка содержит $(m - 1)$ единиц и 0 в последнем столбце; 3) последняя строка содержит $(-n)$ и остальные 0, последний столбец

содержит (m - 1) единиц и 0 в последней строке; 4) последний столбец содержит (m - 1) единиц и 0 в последней строке, последняя строка содержит (m - 1) единиц и 0 в последнем столбце.

По предположению шага индукции в рассматриваемом миноре сумма абсолютных значений ячеек не превосходит (m - 1)(m - 2).

В случае 1 это верно, поскольку $m \geq 2n$ и $\Sigma \mid \boldsymbol{m}(i, j) \mid \leq (m - 1)(m - 2) + 2n \leq$

$\leq (m - 1)(m - 2) + m < m\,(m - 2) + m = m\,(m - 1)$.

В случаях 2 и 3, условие $m \geq 2n$ влечет $\Sigma \mid \boldsymbol{m}(i, j) \mid \leq (m - 1)(m - 2) + (m - 1) + n \leq$

$\leq (m - 1)(m - 2) + (m - 1) + m / 2 = m\,(m - 1) - (m - 1) + m / 2$. Поскольку $m > 2$, имеем $- (m - 1) + m / 2 < 0$, что влечет $m\,(m - 1) - (m - 1) + m / 2 < m\,(m - 1)$.

В случае 4, поскольку $m \geq 2n$, $\Sigma \mid \boldsymbol{m}(i, j) \mid \leq (m - 1)(m - 2) + 2(m - 1) = m\,(m - 1)$.

4. Совершенное множество путей в полном графе

В этом разделе мы исследуем вопрос о существовании наибольшего совершенного множества путей для графа, в котором подграф, порожденный коммутаторами, является полным графом. Пусть имеется k непустых коммутаторов $s_1, s_2, \ldots, s_k$, к которым подсоединено $n_1, n_2, \ldots, n_k$ хостов, соответственно, остальные коммутаторы пустые. Упорядочим непустые коммутаторы по неубыванию числа подсоединенных к ним хостов, т.е. будем считать, что $n_1 \leq n_2 \leq \ldots \leq n_k$. Как отмечено в разделе 3, достаточно рассматривать совершенные множества вершинно-простых путей, соединяющих хосты, подсоединенные к разным коммутаторам, поскольку хосты, подсоединенные к одному коммутатору, всегда можно дополнительно соединить путями, не проходящими по ребрам, соединяющим коммутаторы.

Для хоста x через $s(x)$ будем обозначать коммутатор, к которому подсоединен хост x.

Утверждение 2. Если подграф, порожденный коммутаторами, является полным графом, и существует не более одного коммутатора, к которому подсоединено более одного хоста, то существует наибольшее совершенное множество путей.

Доказательство. Рассмотрим множество P путей, состоящее из путей вида $xs(x)s(y)y$, где x и y хосты, $s(x) \neq s(y)$. Пути из P, очевидно, полные и вершинно-простые. Также очевидно, что для каждой пары хостов x иy, подсоединенных к разным коммутаторам, в P существует ровно один путь из x в y. Разъединение после слияния возможно только для двух путей вида $xs(x)s(y)y$ и $x`s(x`)s(y`)y`$, где $x \neq x`$, $y \neq y`$, $s(x) = s(x`)$, $s(x) \neq s(y)$, $s(y) = s(y`)$. Из этих условий следует, что $s(x)$ и $s(y)$ два разных коммутатора, к каждому из которых подсоединено более одного хоста, что противоречит условию. Таким образом, множество P после добавления в него путей, соединяющих хосты, подсоединенные к одному коммутатору, является совершенным.

□

Далее рассматривается случай, когда имеется более одного коммутатора, к которому подсоединено более одного хоста. Мы определим два условия на число пустых коммутаторов, которые позволяют построить наибольшее совершенное множество путей, и, соответственно, предложим два алгоритма построения таких множеств путей. Эти алгоритмы основаны на алгоритмах для графа-звезда, рассмотренных в предыдущем разделе.

4.1 Алгоритм 1

Этот алгоритм основан на способе 1 построения максимального совершенного множества путей для графа-звезда.

Алгоритм 1. Для каждого непустого коммутатораs_i, $i = 1, \ldots, k$, выделяем один хост 1, подсоединенный к этому коммутатору. Алгоритм состоит из двух этапов.

Этап 1: Соединяем каждый выделенный хост 1, подсоединенный к коммутатору s_i, со всеми хостами h (не только выделенными), подсоединенными ко всем последующим коммутаторам s_j, где $j>i$, путями вида $1s_is_jh$ и hs_js_i1.

Этап 2: Далее для каждого невыделенного хоста $h \neq 1$, подсоединенного к коммутатору s_i, где $i < k$, выделяем пустой хост $s(i, h)$ и соединяем хост h со всеми хостами $h`$ (не только выделенными), подсоединенными ко всем последующим коммутаторам $s_j$, где $j > i$, путями вида $hs_is(i, h)s_jh`$ и $h`s_js(i, h)s_ih$.

Утверждение 3. Алгоритм 1 строит множество вершинно-простых путей, которое становится наибольшим совершенным множеством путей после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору, при условии, что число пустых коммутаторов не меньше $\Sigma\{ n_i | i = 1, \ldots, k \} - max\{ n_i | i = 1, \ldots, k \} - (k - 1)$. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов $K_1 = (n - 1)(k - 1)$.

Доказательство. На этапе 2 для каждого невыделенного хоста h, подсоединенного к каждому коммутатору s_i, кроме последнего коммутатора s_k, выделяется пустой хост $s(i, h)$. Поскольку для каждого непустого коммутатора выделяется ровно один хост, подсоединенный к нему, число требуемых пустых коммутаторов равно (напомним, что мы упорядочили непустые коммутаторы по неубыванию числа подсоединенных к ним хостов) $\Sigma\{ n_i - 1 \mid i = 1, \ldots, k \} = \Sigma\{ n_i | i = 1, \ldots, k \} - max\{ n_i | i = 1, \ldots, k \} - (k - 1)$. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов равно $(n - 1)(k - 1)$.

Покажем, что, если число пустых коммутаторов достаточно велико, построенное множество путей является наибольшим совершенным множеством вершинно-простых путей (после добавления в него путей, соединяющих хосты, подсоединенные к одному коммутатору). Действительно, по построению все эти пути являются полными и вершинно-простыми, а их число конечно. Для каждых i и j, $i \neq j$, $h \neq 1$, $h` \neq 1$ строится следующее множество P путей:

	в хост 1		в хост $h` \neq 1$	
	$i < j$	$i > j$	$i < j$	$i > j$
из хоста 1	$1s_is_j1$	$1s_is_j1$	$1s_is_jh$	$1s_is(j, h)s_jh$
из хоста $h \neq 1$	$hs_is(i, h)s_j1$	hs_is_j1	$hs_is(i, h)s_jh`$	$hs_is(j, h`)s_jh`$

Мы видим, что множество P содержит единственный путь из каждого хоста в каждый хост, подсоединенный к другому коммутатору, т.е. $H(P)$ состоит из всех пар хостов, подсоединенных к разным коммутаторам. Пути, проходящие по дуге s_is_j, либо только разделяются, если $i < j$, либо только сливаются, если $i > j$. Пути, проходящие по дугам $s_is(i, h)$ и $s(i, h)s_j$, ведут из хоста h, подсоединенного к коммутатору s_i, и только разделяются, а пути, проходящие по дугам $s_is(j, h`)$ и $s(j, h`)s_j$, ведут в хост $h`$, подсоединенный к коммутатору s_j, и только сливаются. Тем самым, нет разделения после слияния и, поскольку нет дублирующих путей, нет слияния после разделения. Мы доказали, что построенное множество путей (после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору) является наибольшим совершенным множеством путей.

4.2 Алгоритм 2

Этот алгоритм основан на способе 2 построения максимального совершенного множества путей для графа-звезда. В алгоритме используется понятие минимального покрывающего набора глубиной 2 – это такое минимальное семейство $\mathcal{K}$ множеств хостов, что 1) в каждом множестве для каждого непустого коммутатора содержится ровно один хост, подсоединенный к этому коммутатору, 2) для любых двух хостов, подсоединенных к разным коммутаторам, в семействе есть множество, содержащее эти два хоста. Мощность минимального покрывающего набора глубиной 2 обозначается как $CAN(2, k, n_1, \ldots, n_k)$, где

n_i число хостов, подключенных к коммутатору s_i. Если $n_1 = \ldots = n_k = n$, то пишут сокращенно CAN(2, k, n).

Алгоритм 2. Пусть $\mathcal{K}$ минимальный покрывающий набор множества хостов глубиной 2. Выделим один элемент $\mathcal{K}$ и обозначим его L. Пусть число пустых коммутаторов не меньше $|\mathcal{K}| - 1$. Выделим $|\mathcal{K}| - 1$ пустых коммутаторов и поставим множество этих коммутаторов во взаимно-однозначное соответствие с множеством $\mathcal{K} \setminus \{L\}$. Коммутатор, соответствующий множеству $K \in \mathcal{K} \setminus \{L\}$ обозначим $t(K)$. Линейно упорядочим семейство $\mathcal{K}$: $K_0 = L$, K_1, …, $K_{|\mathcal{K}|-1}$. Для каждой пары хостов x и y, подсоединенных к разным коммутаторам, обозначим через $K(x, y)$ первое множество из $\mathcal{K}$, содержащее оба этих хоста. Очевидно, для каждой пары разных хостов x и y, принадлежащих L, будет $K(x, y) = L$. Строится множество $P = P_L \cup P_{\mathcal{K}}$, объединяемые множества определены ниже.

Множество P_L – это множество путей, соединяющих все пары разных хостов из L: для каждых $x \in L$, $y \in L$, $x \neq y$ выбирается путь $xs(x)s(y)y$ длиной 3.

Множество $P_{\mathcal{K}}$ состоит из путей, соединяющих все остальные пары хостов, подсоединенных к разным коммутаторам. Для каждых хостов x и y таких, что $s(x) \neq s(y)$ и $x \notin L$ или $y \notin L$, выбирается путь $xs(x)t(K(x, y))s(y)y$ длиной 4.

Утверждение 4. Алгоритм 2 строит множество вершинно-простых путей, которое становится наибольшим совершенным множеством путей после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору, при условии, что число пустых коммутаторов не меньше CAN(2, k, n_1, …, n_k) - 1. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов K_2 = CAN(2, k, n) - 1.

Доказательство. По построению для успешного завершения алгоритма требуется число пустых коммутаторов не меньшее чем CAN(2, k, n_1, …, n_k) - 1.

Покажем, что, если число пустых коммутаторов достаточно велико, построенное множество путей является наибольшим совершенным множеством вершинно-простых путей (после добавления в него путей, соединяющих хосты, подсоединенные к одному коммутатору).

1. Каждый путь из P вершинно-простой.

Вершины этого пути являются хостами тогда и только тогда, когда это начало и конец пути. Поэтому достаточно показать, что эти хосты разные, а все коммутаторы на пути тоже попарно разные. Путь из P_L имеет вид $xs(x)s(y)y$, где $x \in L$, $y \in L$, $x \neq y$. Эти условия влекут $s(x) \neq s(y)$, поэтому этот путь вершинно-простой. Путь из $P_{\mathcal{K}}$ имеет вид $xs(x)t(K(x, y))s(y)y$, где $s(x) \neq s(y)$ и $x \notin L$ или $y \notin L$. Условие $s(x) \neq s(y)$ влечет $x \neq y$. Поскольку $t(K(x, y)) \in H_0$, то $s(x) \neq t(K(x, y))$, $s(y) \neq t(K(x, y))$. Поэтому этот путь вершинно-простой.

2. P содержит ровно один путь от каждого хоста x до каждого другого хоста y.

Пусть $x \in L$, $y \in L$, $x \neq y$. Тогда в P_L есть xy-путь и только один, а в $P_{\mathcal{K}}$ нет xy-путей. Пусть $x \notin L$ или $y \notin L$, и $s(x) \neq s(y)$. Тогда в $\mathcal{K} \setminus \{L\}$ есть множество $K(x, y)$, которому принадлежат x и y, и ровно один xy-путь, а в P_L нет xy-путей.

3. Разъединение после слияния.

Для разъединения после слияния двух путей каждый из них должен быть длиной не меньше 3. Рассмотрим 6 теоретически возможных случаев, и покажем, что ни один из этих случаев невозможен для путей из P.

1) Два пути длиной 4 из $P_{\mathcal{K}}$: $xs(x)t(K(x, y))s(y)y$ и $x`s(x`)t(K(x`, y`))s(y`)y`$, где $x \neq x`$, $y \neq y`$, $s(x) = s(x`)$, $t(K(x, y)) = t(K(x`, y`))$. В этом случае в одном множестве $K(x, y) = K(x`, y`)$ есть два разных хоста $x$ и $x`$, подсоединенных к одному коммутатору $s(x) = s(x`)$, что невозможно по построению.
2) Два пути длиной 4 из $P_{\mathcal{K}}$: $xs(x)t(K(x, y))s(y)y$ и $x`s(x`)t(K(x`, y`))s(y`)y`$, где $x \neq x`$, $y \neq y`$, $s(x) = t(K(x`, y`))$, $t(K(x, y)) = s(y`)$. В этом случае к каждому из коммутаторов $s(x)$ и $s(y`)$

подсоединено более одного хоста, а $K(x, y)$ и $K(x`, y`)$ должны быть пустыми коммутаторами, что противоречит равенствам $s(x) = t(K(x`, y`))$, $t(K(x, y)) = s(y)$.

3) Два пути длиной 4 из P_K: $xs(x)\underline{t(K(x, y))s(y)}y$ и $x`s(x`)\underline{t(K(x`, y`))s(y`)}y`$, где $x \neq x`$, $y \neq y`$, $t(K(x, y)) = t(K(x`, y`))$, $s(y) = s(y`)$. В этом случае в одном множестве $K(x, y) = K(x`, y`)$ есть два разных хоста $y$ и $y`$, подсоединенных к одному коммутатору $s(y) = s(y`)$, что невозможно по построению.
4) Два пути длиной 3 из P_L: $x\underline{s(x)s(y)}y$ и $x`\underline{s(x`)s(y`)}y`$, где $x \neq x`$, $y \neq y`$, $s(x) = s(x`)$, $s(y) = s(y`)$. В этом случае два разных хоста x и $x`$подсоединены к одному коммутатору $s(x) = s(x`)$, и два разных хоста y и $y`$подсоединены к одному коммутатору $s(y) = s(y`)$, что невозможно для P_L по построению.
5) Один путь длиной 4 из P_Kи другой путь длиной 3 из P_L: $xs(x)\underline{t(K(x, y))s(y)}y$ и $x`\underline{s(x`)s(y`)}y`$, где $x \neq x`$, $y \neq y`$, $t(K(x, y)) = s(x`)$ и $s(y) = s(y`)$. В этом случае к коммутатору $s(x`)$ должно быть подсоединено более одного хоста, а коммутатор $t(K(x, y))$ должен быть пустым коммутатором, что противоречит $t(K(x, y)) = s(x`)$.
6) Один путь длиной 4 из P_Kи другой путь длиной 3 из P_L: $xs(x)\underline{t(K(x,))}s(y)y$ и $x`\underline{s(x`)s(y`)}y`$, где $x \neq x`$, $y \neq y`$, $s(x) = s(x`)$ и $t(K(x, y)) = s(y`)$. В этом случае к коммутатору $s(y`)$ должно быть подсоединено более одного хоста, а коммутатор $t(K(x, y))$ должен быть пустым коммутатором, что противоречит $t(K(x, y)) = s(y`)$.

Таким образом, мы доказали, что построенное множество путей (после добавления путей, соединяющих хосты, подсоединенные к одному коммутатору) является наибольшим совершенным множеством путей.

4.3 Сравнение двух алгоритмов

Выбор алгоритма 1 или алгоритма 2 для построения наибольшего совершенного множества путей определяется соотношением числа путей. Для случая, когда ко всем непустым коммутаторам подсоединено одно и то же число n хостов, число требуемых пустых коммутаторов в этих алгоритмах равно $K_1 = (n - 1)(k - 1)$ и $K_2 = \mathrm{CAN}(2, k, n) - 1$.

Для $\mathrm{CAN}(2, k, n)$ существуют только асимптотическая оценка сверху $\mathrm{CAN}(2, k, n) \leq n^2\log(k)(1 + o(1))$ при $n \to \infty$ и очевидная нижняя граница $\mathrm{CAN}(2,k,n) \geq n^2$, основанная на том, что всевозможные комбинации из 2 значений, каждое из которых можно выбрать n способами, в покрывающем наборе глубины 2 должны присутствовать [8].

Если k фиксировано, а $n \to \infty$, то асимптотически $(n - 1)(k - 1) < n^2 - 1$ (меньше нижней границы). Если, наоборот, n фиксировано, а $k \to \infty$, то асимптотически $(n - 1)(k - 1) > n^2 log k - 1$ (больше верхней границы). Поэтому общая оценка для этих двух алгоритмов $min\{(n - 1)(k - 1), \mathrm{CAN}(2,k, n) - 1\}$.

К сожалению, более точные оценки $\mathrm{CAN}(2,k, n)$ неизвестны. Только для $n = 2$ доказано, что $k = (x - 1)! / (\lfloor x/2 \rfloor - 1)! (x - \lfloor x/2 \rfloor)!$, где $x = \mathrm{CAN}(2, k, 2)$ [9]. Уже для n = 3 $\mathrm{CAN}(2, k, 3)$ известно только для $k \leq 7$ [10].

5. Экспериментальные результаты

5.1 Формула логики первого порядка для нахождения максимального совершенного множества

Согласно утверждению 2, для любого $k > 0$, если подграф, порожденный коммутаторами $s_1, \dots, s_k$, является полным графом, и существует не более одного коммутатора, к которому подсоединено более одного хоста, то существует наибольшее совершенное множество путей. Возникает следующий вопрос: остаётся ли утверждение верным, если существует более одного коммутатора, к которому подсоединено более одного хоста. Проведенные

компьютерные эксперименты показывают, что это не так. Чтобы доказать этот факт, мы свели задачу поиска совершенного множества путей к задаче проверки выполнимости формулы логики первого порядка [11], и для проверки выполнимости такой формулы использовали солвер Z3 [12] для языка программирования python 2.7. Все исходные коды экспериментов доступны по ссылке [13].

Без потери общности рассуждений можно рассматривать SDN-сеть с коммутаторами $s_1, \dots, s_k$ для $k \geq 2$, в которой

- коммутаторы s_1 и s_2 соединены с двумя хостами;
- для любого $i \in \{3, \dots, k\}$ коммутатор s_i соединенён с одним хостом.

Запишем условие существования совершенного множества путей в виде формулы логики первого порядка. Если такая формула выполнима, то существует максимальное совершенное множество путей, иначе такого множества не существует. Введем следующие обозначения:

- к коммутатору s_1 присоединены хосты h_1 и h_2;
- к коммутатору s_2 присоединены хосты h_3 и h_4;
- для любого $i \in \{3, \dots, k\}$ к коммутатору s_i присоединён коммутатор h_{i+2}.

Количество хостов в такой SDN-сети равно $k + 2$. Напомним, что любой вершинно-простой полный путь $path = hss_{i_1} \dots s_{i_m} s'h'$, где $m \leq k$. Обозначим ребро hs как *начальный отрезок* пути $path$, и ребро $s'h'$ - как *конечный отрезок* пути $path$. Полный путь между хостами h_i и h_j обозначим $path_{i,j}$. Определим следующие предикаты:

- $diffSwitch(h, h')$ принимает значение «истина», если хосты h и h' присоединены к различным коммутаторам;
- $diffBeg(path, path')$ принимает значение «истина», если полные пути $path$ и $path'$ имеют различные начальные отрезки;
- $diffEnd(path, path')$ принимает значение «истина», если полные пути $path$ и $path'$ имеют различные конечные отрезки.

Обозначим через $len(path)$ длину полного пути $path = hss_{i_1} \dots s_{i_m} s'h'$. Для $x \in \{1, len(path)\}$ элемент, стоящий на позиции x, обозначим $path[x]$. Тогда $h = path[1]$, и $h' = path[len(path)]$. Запишем условие отсутствия пересечения ($Ind(path, path')$) по ребру двух полных путей $path$ и $path'$:

$$Ind(path, path') = \forall x \forall y \big((x \in \{2, len(path) - 1\}) \& (y \in \{2, len(path') - 1\})\big) \\ \rightarrow \big((path[x] != path[y]) \cup (path[x+1] != path[y+1])\big).$$

Истинность формулы ниже устанавливает наличие максимального совершенного множества путей. Для любых $i, j, k, l \in \{1, \dots, k+2\}$ справедливо: если h_i и h_j присоединены к различным коммутаторам, и h_k и h_l присоединены к различным коммутаторам, то должна быть истинна следующая формула

$$\big(diffBeg(path_{ij}, path_{kl}) \& diffEnd(path_{ij}, path_{kl})\big) \rightarrow \\ Ind(path_{ij}, path_{kl}). \quad (1)$$

Интуитивно, формула означает, что если любые два полных пути не сливаются и не разделяются, то они не имеют общих рёбер. Далее эта формула проверяется при помощи солвера Z3.

5.2 Случай с 4-мя коммутаторами и 6-ю хостами

Рассмотрим применение формулы (1) для нахождения наибольшего совершенного множества путей для SDN-сети, изображённой на рис. 1.

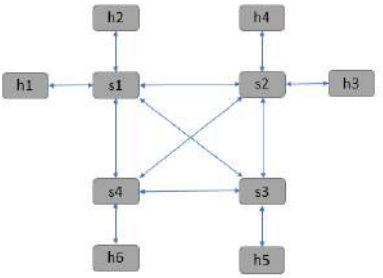


Рис.1. SDN-сеть с 4-мя коммутаторами и 6-ю хостами
Fig. 1. SDN network with 4 switches and 6 hosts

Формула (1) не верна для топологии, изображённой на рис. 1. Соответствующий код на языке python, реализующий данную топологию и использующий солвер Z3, доступен по ссылке [13].

В [7] было показано, что наличие совершенного множества путей является достаточным условием для того, чтобы каждое нормальное множество пар хостов имело строгую реализацию без циклов и дублирования. Необходимость этого условия принималась как гипотеза, которая должна быть доказана или опровергнута. Эксперименты показали, что эта гипотеза не верна: для SDN-сети, изображённой на рис. 1, максимального совершенного множества не существует, но каждое нормальное множество пар хостов имеет строгую реализацию без циклов и дублирования. Отчёт выполнения солвера Z3 доступен по ссылке [13].

Тот факт, что не существует наибольшего совершенного множества для SDN-сети, изображённой на рис. 1, даёт основание для выдвижения следующей гипотезы: для любого $k > 0$, если подграф, порожденный коммутаторами $s_1, \dots, s_k$, является полным графом, и существует более одного коммутатора, к которому подсоединено более одного хоста, то не существует наибольшего совершенного множества путей. Однако это не так, и в следующем разделе мы приводим соответствующий контрпример.

5.3. Случай с 5-мя коммутаторами и 7-ю хостами

Рассмотрим SDN-сеть с 5-ю коммутаторами и 7-ю хостами, изображённую на рис. 2.

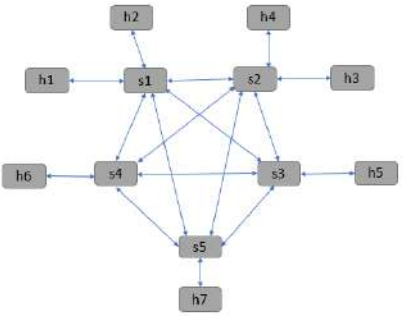


Рис.2. SDN-сеть с 5-ю коммутаторами и 7-ю хостами
Fig. 2. SDN network with 5 switches and 7 hosts

Формула (1) верна для топологии, изображённой на рис. 2. Соответствующий код на языке python, реализующий данную топологию и использующий солвер Z3, доступен по ссылке [13]. Наибольшее совершенное множество путей изображено на рис. 3.

	h_1	h_2	h_3	h_4	h_5	h_6	h_7
h_1	$h_1s_1h_1$	$h_1s_1h_2$	$h_1s_1s_2h_3$	$h_1s_1s_2h_4$	$h_1s_1s_3h_4$	$h_1s_1s_2s_5s_3s_4h_5$	$h_1s_1s_2s_5h_7$
h_2	$h_2s_1h_1$	$h_2s_1h_2$	$h_2s_1s_5s_4s_2h_3$	$h_2s_1s_5s_2h_4$	$h_2s_1s_3h_5$	$h_2s_1s_5s_3\ s_4h_6$	$h_2s_1s_5h_7$
h_3	$h_3s_2s_1h_1$	$h_3s_2s_3s_1h_2$	$h_3s_2h_3$	$h_3s_2h_4$	$h_3s_2s_3h_5$	$h_3s_2s_3s_4h_6$	$h_3s_2s_3s_5h_7$
h_4	$h_4s_2s_1h_1$	$h_4s_2s_4s_3s_1h_2$	$h_4s_2h_3$	$h_4s_2h_4$	$h_4s_2s_4s_3h_5$	$h_4s_2s_4h_6$	$h_4s_2s_4s_3s_5h_7$
h_5	$h_5s_3s_2s_1h_1$	$h_5s_3s_1h_2$	$h_5s_3s_2h_3$	$h_5s_3s_2h_4$	$h_5s_3h_5$	$h_5s_3s_4h_6$	$h_5s_3s_5h_7$
h_6	$h_6s_4s_1h_1$	$h_6s_4s_1h_2$	$h_6s_4s_2h_3$	$h_6s_4s_5s_2h_4$	$h_6s_4s_1s_3h_5$	$h_6s_4h_6$	$h_6s_4s_5h_7$
h_7	$h_7s_5s_1h_1$	$h_7s_5s_1h_2$	$h_7s_5s_1s_4s_2h_3$	$h_7s_5s_2h_4$	$h_7s_5s_1s_3h_5$	$h_7s_5s_3s_4h_6$	$h_7s_5h_7$

Рис.3. Наибольшее совершенное множество путей для SDN-сети с 5-ю коммутаторами и 7-ю хостами
Fig. 3. Largest perfect set of paths for an SDN network with 5 switches and 7 hosts

6. Заключение

Таким образом, в результате проведенных авторами исследований установлены достаточные условия существования наибольшего совершенного множества путей для плоскости данных программно-конфигурируемых сетей (SDN), предложены алгоритмы построения таких множеств и получены оценки их сложности. Компьютерные эксперименты показали, что для SDN-сети наибольшее совершенное множество может не существовать, однако каждое множество пар различных хостов имеет строгую реализацию без циклов и дублирования.

Проведённые компьютерные эксперименты показывают, что условия существования наибольшего совершенного и замкнутого множества для произвольной SDN-сети должны быть изучены отдельно. Эти вопросы авторы планируют исследовать в будущей работе.

Список литературы / References

[1] Sezer S., Scott-Hayward S., Chouhan P.K., Fraser B., Lake D., Finnegan J., Viljoen N., Miller M. and Rao N. Are we ready for sdn? Implementation challenges for software-defined networks. IEEE Communications Magazine, vol. 51, no. 7, 2013, pp. 36-43.

[2] López J., Kushik N., Yevtushenko N. and Zeghlache D. Analyzing and Validating Virtual Network Requests. In Proc. of the 12th International Conference on Software Technologie, 2017, pp 441-446.

[3] Yevtushenko N, Kossatchev A, Lopez J, Kushik N and Zeghlache D. Test Derivation for the Software Defined Networking Platforms: Novel Fault Models and Test Completeness. In Proc. of the IEEE East-West Design and Test Symposium, 2018, pp 1-6.

[4] Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Тестирование правил настройки сетевого коммутатора программно конфигурируемой сети. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 69-88. DOI: 10.15514/ISPRAS-2018-30(6)-4 / Burdonov I.B., Yevtushenko N.V. and Kossatchev.A.S. Testing switch rules in software defined networks. Trudy ISP RAN/Proc. ISP RAS, vol 30, issue 6, 2018, pp 69-88 (in Russian).

[5] Burdonov I., Kossatchev A., Yevtushenko N., López J., Kushik N., and Zeghlache D. Verifying SDN Data Path Requests. arXiv:1906.03101, 2019.

[6] Boufkhad Y., De La Paz R., Linguaglossa L., Mathieu F., Perino D., and Viennot L, Forwarding tables verification through representative header sets. arXiv:1601.07002, 2016.

[7] Burdonov I., Yevtushenko N., and Kossatchev A. Implementing a virtual network on the SDN data plane. Accepted at the 2020 IEEE East-West Design and Test Symposium.

[8] Кулямин В.В., Петухов А.А. Обзор методов построения покрывающих наборов. Программирование, том 37, № 3, стр. 3-41 / V.V. Kuliamin, A.A. Petukhov. A survey of methods for constructing covering arrays. Programming and Computer Softwareб vol. 37, № 3, 2011, article no. 121. https://doi.org/10.1134/S0361768811030029.

[9] Kleitman Daniel J. and Spencer Joe. Families of k-independent sets. Discrete mathematics, vol. 6, 1973, pp. 255-262.

[10] Choi Soohak, Kim Hyun Kwang, Oh Dong Yeol. Structures and lower bounds for binary covering arrays. Discrete mathematics, vol. 312, 2012, pp. 2958-2968.

[11] Верещагин Н К, Шень А. Лекции по математической логике и теории алгоритмов. Часть 2. Языки и исчисления. МЦНМО, 2012, 240 стр. / Vereshchagin N.K., Shen A. Lectures on mathematical logic and theory of algorithms. Part 2. Languages and Calculus. Moscow center for continuous mathematical education, 2012, 240 p. (in Russian).
[12] Yurichev D. SAT/SMT by Example. URL; https://yurichev.com/SAT_SMT.html, accessed 20,07.2020.
[13] Vinarskii E. perfect_set. URL: https://github.com/vinevg1996/perfect_set, accessed 20,07.2020.

Информация об авторах / Information about authors

Игорь Борисович БУРДОНОВ – доктор физико-математических наук, ведущий научный сотрудник. Научные интересы: формальные спецификации, генерация тестов, технология компиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработки программного обеспечения.

Igor Borisovich BURDONOV – Doctor of Physical and Mathematical Sciences, Leading Researcher. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.

Евгений Максимович ВИНАРСКИЙ – студент магистратуры, кафедра математической кибернетики факультета ВМК. Научные интересы: конечные автоматы, верификация.

Evgeny Maksimovich VINARSKY – Master's student, Department of Mathematical Cybernetics, Faculty of CMC. Research interests: finite state machines, verification.

Нина Владимировна ЕВТУШЕНКО, доктор технических наук, профессор, г.н.с. ИСП РАН, до 1991 года работала научным сотрудником в Сибирском физико-техническом институте. С 1991 г. работала в ТГУ профессором, зав. кафедрой, зав. лабораторией по компьютерным наукам. Её исследовательские интересы включают формальные методы, теорию автоматов, распределенные системы, протоколы и тестирование программного обеспечения.

Nina Vladimirovna YEVTUSHENKO, Doctor of Technical Sciences, Professor, a Leading Researcher of ISP RAS, worked at the Siberian Scientific Institute of Physics and Technology as a researcher up to 1991. In 1991, she joined Tomsk State University as a professor and then worked as the chair head and the head of Computer Science laboratory. Her research interests include formal methods, automata theory, distributed systems, protocol and software testing.

Александр Сергеевич КОСАЧЕВ – кандидат физико-математических наук, ведущий научный сотрудник. Научные интересы: формальные спецификации, генерация тестов, технология компиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработки программного обеспечения.

Alexander Sergeevitch KOSSATCHEV – Candidate of Physical and Mathematical Sciences, Leading Researcher. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.

DOI: 10.15514/ISPRAS–2020–32(4)–19

Time Causal Processes in Time Petri Nets with Weak Semantics

[1,2]I.B. Virbitskaite, ORCID: 0000-0002-4475-3480 <virb@iis.nsk.su>
[1]A.Yu. Zubarev, ORCID: 0000-0002-3499-3253 <auzubarev@gmail.com>
[1] A.P. Ershov Institute of Informatics Systems, SB RAS
6, Acad. Lavrentiev avenue, 630090, Novosibirsk, Russia
[2] Novosibirsk State University,
2, Pirogova st., Novosibirsk, 630090, Russia

Abstract. In this paper, we present a method for state space reduction of dense-time Petri nets (TPNs) – an extension of Petri nets by adding a time interval to every transition for its firing. The time elapsing and memory operating policies define different semantics for TPNs. The decidability of many standard problems in the context of TPNs depends on the choice of their semantics. The state space of the TPN is infinite and non-discrete, in general, and, therefore, the analysis of its behavior is rather complicated. To cope with the problem, we elaborate a state space discretization technique and develop a partial order semantics for TPNs equipped with weak time elapsing and intermediate memory policies.

Keywords: dense-time Petri nets; weak time elapsing semantics; intermediate memory policy; state space discretization; partial order semantics; time causal processes

For citation: Virbitskaite I.B., Zubarev A.Yu. Time Causal Processes in Time Petri Nets with Weak Semantics. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020, pp. 261-284. DOI: 10.15514/ISPRAS–2020–32(4)–19

Временные причинно-упорядоченные процессы временных сетей Петри со «слабой» семантикой

[1,2] И.Б. Вирбицкайте, ORCID: 0000-0002-4475-3480 <virb@iis.nsk.su>
[1] А.Ю. Зубарев, ORCID: 0000-0002-3499-3253 <auzubarev@gmail.com>
[1] Институт систем информатики им. А.П. Ершова СО РАН,
630090, Новосибирск, проспект Лаврентьева, 6
[2] Новосибирский государственный университет,
630090, Новосибирск, ул. Пирогова, 2

Abstract. В данной статье предлагается метод редукции пространства состояний непрерывно-временных сетей Петри (НВСП) – расширения сетей Петри, где каждому переходу ставится в соответствие временной интервал его срабатывания. Техники контроля времени и памяти определяют различные семантики для НВСП, которые влияют на разрешимость многих стандартных проблем анализа поведения НВСП. В общем случае, пространство состояний НВСП бесконечно и несчетно, и, следовательно, анализ их поведения довольно сложен. С целью разрешения данной проблемы выполняется дискретизация пространства состояний и определяется семантика частичного порядка для НВСП со «слабой» техникой продвижения времени (продвижение времени неограничено) и «промежуточной» техникой контроля памяти (с учетом промежуточных разметок при срабатывании сетевых переходов).

Ключевые слова: непрерывно-временные сети Петри; «слабая» семантика продвижения времени; «промежуточная» техника сброса часов; дискретизация пространства состояний; семантика частичного порядка, временные причинные процессы

Для цитирования: Вирбицкайте И.Б., Зубарев А.Ю. Временные причинно-упорядоченные процессы временных сетей Петри со «слабой» семантикой. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 261-284 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(4)–19

1 Introduction

Dense-Time Petri Nets (TPNs) are now a well-established model to describe and study safety-critical systems that often require verification of real time (quantitative) characteristics, in addition to functional (qualitative) properties. In the TPN, each transition is associated with a time interval. With that, each transition is assumed to have its own local clock. A state of the TPN contains a current marking and readings of the local clocks of enabled transitions (i.e. transitions whose all input places have enough tokens at the marking). A transition can fire from a state only if the transition is enabled at the corresponding marking and its clock reaches a moment in time that is within the interval associated. So, the firing of an enabled transition can be suspended for a certain time. Along with that, the firing itself takes no time. State changes are divided in two types: either *time elapses*, i.e. the clocks of enabled transitions go forward, or *a transition fires*, i.e. a current marking is changed to a new one (in which tokens are consumed from the input places and tokens are produced to the output places of the transition that fires) and the clocks of the transitions that become enabled at the new marking (newly enabled transitions) are reset to zero.

There are two policies of time elapsing in TPNs, which define strong and weak semantics. In the former semantics, time elapsing cannot exceed the upper bounds of enabled transitions and, therefore, an enabled transition must fire no later than the upper bound of its time interval is reached. On the contrary, any time elapsing is allowed in the latter semantics and, therefore, enabled transitions are not forced to fire. In [1], the authors have proven that the two semantics are incomparable w.r.t. timed weak bisimulation.

Memory policies in TPNs determine when the local clocks of enabled transitions are reset. Intermediate and atomic memory policies are put forward in the literature. The former treats intermediary marking, i.e. the marking after consumption of tokens from the input places and before production of tokens to the output places of a transition t that fires. A transition t' is regarded as newly enabled and its clock is reset to zero after the firing of t whenever t' is disabled at the intermediary marking and becomes enabled at the new marking, i.e. after production of tokens to the output places of t. Instead, the latter policy considers a firing as one-step. The clock of t' is reset to zero only if it is disabled at the marking before t fires and becomes enabled at the new marking after t fires. The memory policies were studied in [2] for strong semantic and in [3] for weak semantics. It was shown that the marking reachability/coverability and boundedness problems are undecidable for time Petri nets with strong semantics and any memory policy, whereas the problems are decidable in the case of TPNs with weak intermediate semantics but not with weak atomic semantics.

The state space of the TPN is infinite and non-discrete, in general, that increases the difficulty of the model analysis. In the work [4], a transformation to the behavior with only integer time elapsings has been suggested for TPNs with strong semantics, while the discretization of the state space for weak semantics has hitherto not be treated in the literature, to the best of our knowledge.

The classical interleaving behavior of the TPN is described by runs – sequences of changes in states by time elapsings or transition firings. Interleaving semantics allows for analyzing some safety and liveness properties of systems. However, using partial order semantics seems preferable because it captures in a natural way «true concurrency». Partial order semantics of Petri nets is most often represented by means of the so-called causal net processes, which include events and conditions related by causal dependence and concurrency. This information can be useful for formal

verification of the system behavior or for reducing the number of analyzed system states, without taking into account all interleaving sequences. Partial order semantics is put forward for safe TPNs with strong and clocks-on-transitions semantics in [5]. The presented in [6] approach to construct a partial order and non-deterministic representation of the behavior of safe TPNs with strong and clocks-on-tokens semantics consists in transforming time characteristics into net structure, i.e. representing them by additional places, transitions, and arcs. This allows for removing the restrictions of diverging time and of finite upper time bounds for transitions. In [7], the authors inspect free choice TPNs (i.e. net transitions sharing an input place do have exactly the same input places), develop and compare partial order representations of runs, based on various clocks-on-tokens semantics.

In this paper, we deal with dense-time Petri nets with weak and intermediate policies. Our intention here is twofold. First, we develop a discrete representation of the interleaving behavior (runs) of the TPN by transforming its runs with real-number time elapsings to parametric sequences with time variables that are then assigned natural-number values. Second, partial order clocks-on-tokens semantics in terms of time causal processes of the TPN, by converting time elapsings into net structure, is elaborated. Also, for the TPN, a bijective mapping between its runs and computations (called linearizations) of its time causal processes is constructed, in order to demonstrate the correctness of the partial order semantics w.r.t. interleaving one. Partial order semantics allow for taking into account the processes' timing behavior in addition to their degrees of relative concurrency.

The paper is organized as follows. In Section 2, we consider some definitions for TPNs and their interleaving semantics in terms of runs – sequences of changes in states by time elapsings and transition firings. In the following section, it is established that the discretization of TPN's state space is possible by demonstrating that in the TPN for any run with transition firings and real-number time elapsings there exists a run having the same transition firings and only natural-number (even unit) time elapsings. In Section 4, we introduce and examine properties of a casual net, its linearization, and a time causal process of the TPN, consisting of a casual net and its homomorphism into the TPN. In the next section, a bijective mapping from a set of linearizations of causal nets of time processes of the TPN to its set of runs is developed and studied. Section 6 concludes the paper.

2 Time Petri Nets

In this section, some terminology concerning the model of Petri nets with timing constraints (time intervals on the firings of transitions) are defined. We start with recalling the definitions of the structure and behavior of Petri nets.

The Petri net (PN) consists of two different sets of elements – places and transitions; a flow relation representing arrows between the elements; an initial marking – a subset of places initially containing tokens; and a labeling function mapping each transition to an action from the alphabet Act of actions. A state of the PN is called a marking – a subset of places that receive tokens when the net functions. A transition is enabled at a marking if the input places of the transition contain tokens. The firing of a transition enabled at a marking results in the new marking in which tokens are consumed from the input places and tokens are produced to the output places of the transition. A sequence of changes in markings is called a run of the PN.

Definition 1. A (*labeled over Act*) *Petri net* (PN) is a tuple $\mathcal{N} = (P, T, F, M_0, L)$, where P is a finite *set of places* and T is a finite *set of transitions* such that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*; $\emptyset \neq M_0 \subseteq P$ is an *initial marking*; $L : T \to Act$ is a *labeling function*. For $x \in P \cup T$, let $\bullet x = \{y | (y, x) \in F\}$ and $x\bullet = \{y | (x, y) \in F\}$ be the *preset* and *postset* of x, respectively. For $X \subseteq P \cup T$, define $\bullet X = \bigcup_{x \in X} \bullet x$ and $X\bullet = \bigcup_{x \in X} x\bullet$.

A *marking* M of a Petri net $\mathcal{N}$ is any subset of P. A transition $t \in T$ is *enabled at a marking* M if $\bullet t \subseteq M$. Let $En(M)$ be the set of transitions enabled at M.

The *firing of a transition* t enabled at a marking M leads to the new marking M' (denoted $M \xrightarrow{t} M'$) iff $M' = (M \setminus {\bullet t}) \cup t{\bullet}$. We write $M \xrightarrow{\vartheta} M'$ iff $\vartheta = t_1 \dots t_k$ and $M = M^0 \xrightarrow{t_1} M^1 \dots M^{k-1} \xrightarrow{t_k} M^k = M'$ $(k \geq 0)$. In this case, ϑ is a *run* of $\mathcal{N}$ from M (to M'), and M' is a *reachable marking of* $\mathcal{N}$ *from* M. Let $\mathcal{RM}(\mathcal{N})$ be the set of all reachable markings of $\mathcal{N}$ from M_0.

The time Petri net (TPN) consists of an underlying PN and a static timing function mapping each transition to a time interval with non-negative rational boundaries. With that, each transition is assumed to have its own local clock. A marking alone is not enough to describe a state of the TPN, so a dynamic timing function is added to indicate the clock values of the transition enabled at a current marking. In fact, the clocks of enabled transitions show the times passed since then as the transitions become enabled. The initial state consists of the initial marking and the dynamic timing function with zero clock values for all enabled transitions. When the TPN is running, there are two ways to change states: either by time elapsings or by transition firings. Following the approach of [3], we consider TPNs with weak semantics. This means that any time elapsing is allowed, i.e. any time can be added to the clock values of enabled transitions. A transition can fire from a current state only if the transition is enabled at the current marking and its clock value belongs to its time interval. The firing of a transition that can fire from a state results in a new state, i.e. a new marking and new dynamic timing function with the clock values reset to zero for the newly enabled transitions and with the old clock values for the transitions which continue to be enabled. We deal with TPNs with intermediate memory policy, i.e. the predicate $\uparrow enabled(t', M, t)$ determining a newly enabled transition t' after the firing of a transition t at a marking M has a true value if and only if t' is disabled at intermediary marking (i.e. the marking between consumption and production of tokens by the firing of *t*) and becomes enabled at the new marking (i.e. the marking after production of tokens by the firing of *t*). A sequence of changes in states is called a run of the TPN. The runs from the initial state represent interleaving semantics of the TPN.

Definition 2. A (*labeled over* Act) *time Petri net* (TPN) is a pair $\mathcal{TN} = (\mathcal{N}, D)$, where $\mathcal{N} = (P, T, F, M_0, L)$ is the underlying Petri net and $D: T \to \mathbb{Q}_{\geq 0} \times (\mathbb{Q}_{\geq 0} \cup \{\infty\})$ is a *static timing function* mapping each transition to a closed non-empty interval with non-negative rational boundaries; right open infinite boundaries are allowed. For a transition $t \in T$, the boundaries of the interval $D(t)$ are called the *earliest firing time* (Eft) and *latest firing time* (Lft) *of* t.

A *state of* $\mathcal{TN}$ is a pair $S = (M, I)$, where M is a marking and $I: En(M) \to \mathbb{R}_{\geq 0}$ is a *dynamic timing function*. The *initial state* of $\mathcal{TN}$ is a pair $S_0 = (M_0, I_0)$, where M_0 is the initial marking and $I_0(t) = 0$, for all $t \in En(M_0)$.

A transition t *can fire from a state* $S = (M, I)$ if $t \in En(M)$ and $Eft(t) \leq I(t) \leq Lft(t)$.

In the TPN, two types of state changes are possible by:

a) the *elapsing of time* $\tau \in \mathbb{R}_{\geq 0}$, defined as follows:

$$(M, I) \xrightarrow{\tau} (M, I') \text{ iff } \forall t' \in En(M): I'(t') = I(t') + \tau;$$

b) the *firing of a transition* $t \in T$, defined as follows:

$$(M, I) \xrightarrow{t} (M', I') \text{ iff } \begin{cases} t \text{ can fire from } (M, I), \text{and} \\ M' = (M \setminus {\bullet t}) \cup t{\bullet}, \text{and} \\ \forall t' \in En(M'): I'(t') = \begin{cases} 0, & \text{if } \uparrow enabled(t', M, t), \\ I(t'), & \text{otherwise,} \end{cases} \end{cases}$$

where the predicate

$$\uparrow enabled(t', M, t) = t' \in En((M \setminus {\bullet t}) \cup t{\bullet}) \wedge (t' \notin En(M \setminus {\bullet t}) \vee t = t')$$

indicates whether we need to reset the clock of t' after the firing of t at M.

We use the notation $S \xrightarrow{\sigma} S'$ iff $\sigma = \overline{t_1} \dots \overline{t_k} \in (T \cup \mathbb{R}_{\geq 0})^k$ and $S = S^0 \xrightarrow{\overline{t_1}} S^1 \dots S^{k-1} \xrightarrow{\overline{t_k}} S^k = S'$ $(k \geq 0)$. In this case, σ is a *run* of $\mathcal{TN}$ *from* S (*to* S'), and S' is a *reachable state of* $\mathcal{TN}$ *from* S. Let $\mathcal{FS}(\mathcal{TN})$ be the set of all runs of $\mathcal{TN}$ from S_0, $\mathcal{RS}(\mathcal{TN})$ be the set of all reachable states of $\mathcal{TN}$ from S_0. We write $Untimed(\sigma)$ to denote the projection σ on T, i.e. the untimed part of σ.

Consider some properties of TPNs. We call $\mathcal{TN}$ *safe*, iff $M(p) \leq 1$, for all $S = (M, I) \in RS(\mathcal{TN})$ and $p \in P$; *contact-free* iff whenever t can fire from a state $S = (M, I)$, then $(M \backslash {\bullet t}) \cap t^{\bullet} = \emptyset$ for all $S \in \mathcal{RS}(\mathcal{TN})$; T*-restricted* iff ${\bullet t} \neq \emptyset$ and $t^{\bullet} \neq \emptyset$ for all transitions in the underlying Petri net.

Notice that the definition of the marking of the underlying PN as a subset, rather than a multiset, of the net places (see Definition 1) ensures that each place has at most one token when the TPN is functioning, i.e. it is safe. This leads to the fact that any transition can be enabled at most once at any marking M and can fire at most once from a corresponding state (M, I). As a consequence, $En(M)$ is a set, rather than a multiset, of transitions, and the dynamic timing function I is really a function, rather than a relation. The contact-freeness property says that a transition cannot fire from a state, if at least one output place (which is not the input place) of the transition already contains a token at the corresponding marking. In the case when the TPN is not contact-free, after the firing of an enabled transition from a state, two or more tokens can accumulate in the output places of the transition. However, some of the tokens may be lost, as the marking is defined as a subset, rather than a multiset, of places. Due to the *T*-restrictedness property, each net transition has at least one input place and at least one output place. This allows us to avoid livelock (useless work) situations as the transitions without input and output places can fire (work) infinitely many times without consuming and producing any tokens (results). So, the above properties facilitate the correct definitions and results concerning TPNs. In what follows, we will consider only safe, contact-free and *T*-restricted TPNs.

Example 1. A (labeled over $Act = \{a, b\}$) time Petri net $\widetilde{\mathcal{TN}}$ is shown in Fig. 1. Here, the places are represented by circles and transitions by squares; the names are depicted near the elements. The elements included in the flow relation are connected by arrows, and each place contained in the initial marking is that with a token (bold point). The values of the labeling and static timing functions are printed next to the transitions. It is easy to see that $\widetilde{\mathcal{TN}}$ is really safe, contact-free and T-restricted. Show that $\sigma = t_1 t_3 (2.3) t_2 (1.5) t_3$ is a run of $\mathcal{TN}$ from S_0.

- $S_0 = (M_0, I_0)$, where $M_0 = \{p_1, p_2\}, En(M_0) = \{t_1, t_3\}$ and $\forall t \in En(M_0) : I_0(t) = 0$.
- Due to $t_1 \in En(M_0)$ and $Eft(t_1) = 0 \leq I_0(t_1) = 0 \leq Lft(t_1) = 1$, we have that t_1 can fire from (M_0, I_0). Then, $S_0 \xrightarrow{t_1} S_1 = (M_1, I_1)$, where $M_1 = (M_0 \backslash {\bullet t_1}) \cup t_1^{\bullet} = \{p_2, p_3\}$, $En(M_1) = \{t_3\}$ and $I_1(t_3) = I_0(t_3) = 0$, because $\uparrow enabled(t_3, M_0, t_1) = false$.
- Due to $t_3 \in En(M_1)$ and $Eft(t_3) = 0 \leq I_1(t_3) = 0 \leq Lft(t_3) = 2$, we have that t_3 can fire from (M_1, I_1). Then, $S_0 \xrightarrow{t_1 t_3} S_2 = (M_2, I_2)$, where $M_2 = (M_1 \backslash {\bullet t_3}) \cup t_3^{\bullet} = \{p_3, p_4\}$, $En(M_2) = \{t_2\}$ and $I_2(t_2) = 0$, because $\uparrow enabled(t_2, M_1, t_3) = true$.
- $S_0 \xrightarrow{t_1 t_3 (2.3)} S_3 = (M_3, I_3)$, where $M_3 = M_2 = \{p_3, p_4\}$, $En(M_3) = \{t_2\}$ and $I_3(t_2) = I_2(t_3) + 2.3 = 2.3$.
- Due to $t_2 \in En(M_3)$ and $Eft(t_2) = 1 \leq I_3(t_2) = 2.3 \leq Lft(t_2) = 3$, we have that t_2 can fire from (M_3, I_3). Then, $S_0 \xrightarrow{t_1 t_3 (2.3) t_2} S_4 = (M_4, I_4)$, where $M_4 = (M_3 \backslash {\bullet t_2}) \cup t_2^{\bullet} = \{p_1, p_2\}$, $En(M_4) = \{t_1, t_3\}$ and $I_4(t_1) = I_4(t_3) = 0$, because $\uparrow enabled(t_1, M_3, t_2) = \uparrow enabled(t_3, M_3, t_2) = true$.

– $S_0 \xrightarrow{t_1 t_3(2.3) t_2(1.5)} S_5 = (M_5, I_5)$, where $M_5 = M_4 = \{p_1, p_2\}$, $En(M_5) = \{t_1, t_3\}$ and $I_5(t_1) = I_4(t_1) + 1.5 = 1.5$, $I_5(t_3) = I_4(t_3) + 1.5 = 1.5$.

– Due to $t_3 \in En(M_5)$ and $Eft(t_3) = 0 \leq I_5(t_3) = 1.5 \leq Lft(t_3) = 2$, we have that t_3 can fire from (M_5, I_5). Then, $S_0 \xrightarrow{t_1 t_3(2.3) t_2(1.5) t_3} S_6 = (M_6, I_6)$, where $M_6 = (M_5 \backslash \bullet t_3) \cup t_3 \bullet = \{p_1, p_4\}$, $En(M_6) = \{t_1\}$ and $I_6(t_1) = I_5(t_1) = 1.5$, because $\uparrow enabled(t_1, M_5, t_3) = false$.

Therefore, $\sigma = t_1 t_3(2.3) t_2(1.5) t_3$ is a run of $\mathcal{TN}$ from S_0. □

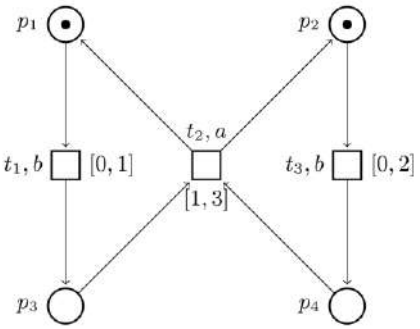


Fig. 1. *A time Petri net* $\widetilde{\mathcal{TN}} = (\widetilde{\mathcal{N}}, D)$

In order to display that every TPN can be transform into that with natural-valued boundaries of the intervals associated with its transitions, we need a notion of time equivalence. Two TPNs are considered time equivalent if they have the same underlying Petri net, and for each transition, its earliest and latest firing times in the TPNs are either proportional to a non-zero constant or its latest firing times are together equal to infinity.

Definition 3. Two time Petri nets $\mathcal{TN}_1 = (\mathcal{N}, D_1)$ and $\mathcal{TN}_2 = (\mathcal{N}, D_2)$ are *time equivalent* iff there exists a non-negative constant $c \neq 0$ such that for any transition t in $\mathcal{N}$ it holds:

– $Eft_2(t) = Eft_1(t) \cdot c$,

– $Lft_2(t) = \begin{cases} \infty, & \text{if } Lft_1(t) = \infty, \\ Lft_1(t) \cdot c, & \text{otherwise.} \end{cases}$

We next establish that for any TPN there is a time equivalent TPN with time intervals having natural-valued boundaries.

Theorem 1. Given a TPN $\mathcal{TN}_1 = (\mathcal{N} = (P, T, F, M_0, L), D_1)$, there exists a TPN $\mathcal{TN}_2 = (\mathcal{N}, D_2)$, with $D_2 \colon T \to \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$, such that $\mathcal{TN}_1$ and $\mathcal{TN}_2$ are time equivalent. Moreover, for any $\sigma_1 \in \mathcal{FS}(\mathcal{TN}_1)$, there is $\sigma_2 \in \mathcal{FS}(\mathcal{TN}_2)$ with the same transition firings and time elapsings multiplied by a constant c, and vice versa.

Proof. Construct the set $\mathcal{D}$ of the denominators of the boundaries from D_1 as follows: $\mathcal{D} = \left\{ n \,\middle|\, Eft_1(t) = \frac{m}{n};\ t \in T; m, n \in \mathbb{N}_{>0} \right\} \cup \left\{ n \,\middle|\, Lft_1(t) = \frac{m}{n};\ t \in T;\ m, n \in \mathbb{N}_{>0} \right\}$. Calculate the least common multiple of the denominators: $c = \begin{cases} 1, & \text{if } \mathcal{D} = \emptyset \\ LCM(\mathcal{D}), & \text{otherwise} \end{cases}$. Due to c being the least common multiple, we have non-negative constant $c \neq 0$. Define $D_2 \colon T \to \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ as follows:

– $Eft_2(t) = Eft_1(t) \cdot c$,

– $Lft_2(t) = \begin{cases} \infty, & \text{if } Lft_1(t) = \infty, \\ Lft_1(t) \cdot c, & \text{otherwise,} \end{cases}$
– $D_2(t) = (Eft_2(t),\ Lft_2(t))$.

So, $\mathcal{TN}_1$ and $\mathcal{TN}_2$ are time equivalent.

Take an arbitrary $\sigma_1 = \overline{t_1} \dots \overline{t_k} \in \mathcal{FS}(\mathcal{TN}_1)$, with $(M_0, I_0) \xrightarrow{\overline{t_1}} (M_1, I_1) \dots (M_{k-1}, I_{k-1}) \xrightarrow{\overline{t_k}} (M_k, I_k)$. Construct $\sigma_2 = \overline{t_1'} \dots \overline{t_k'}$ such that $\overline{t_i} \in T \Rightarrow \overline{t_i'} = \overline{t_i}$ and $\overline{t_i} \in \mathbb{R}_{\geq 0} \Rightarrow \overline{t_i'} = \overline{t_i} \cdot c$, for all $1 \leq i \leq k$. We shall prove that $\sigma_2 \in \mathcal{FS}(\mathcal{TN}_2)$, with $(M_0', I_0') \xrightarrow{\overline{t_1'}} (M_1', I_1') \dots (M_{k-1}', I_{k-1}') \xrightarrow{\overline{t_k'}} (M_k', I_k')$, by induction on $1 \leq i \leq k$.

$i = 0$. Then, $M_0' = M_0$, due to $\mathcal{TN}_1$ and $\mathcal{TN}_2$ having the same underlying Petri net; and $I_0'(t) = 0$, for all $t \in En(M_0')$, due to Definition 2.

$i > 0$. By the induction hypothesis, we have that $(M_0', I_0') \xrightarrow{\overline{t_1'} \dots \overline{t_{i-1}'}} (M_{i-1}', I_{i-1}')$. Thanks to the construction of σ_2, we obtain that $M_j' = M_j$, and, hence, $En(M_j') = En(M_j)$, for all $0 \leq j \leq i-1$, due to Definition 2. Then, it holds that $\uparrow enabled(t, M_{j-1}', \overline{t_j'}) = \uparrow enabled(t, M_{j-1}, \overline{t_j})$, for all $t \in En(M_{j-1}') = En(M_{j-1})$ and for all $1 \leq j \leq i$, i.e. the prefixes of σ_1 and σ_2 have the same clock resets for enabled transitions. This implies that $I_j'(t) = I_j(t) \cdot c$, for all $t \in En(M_j')$ and $0 \leq j \leq i-1$, due to Definition 2 and the construction of σ_2. Show that $(M_{i-1}', I_{i-1}') \xrightarrow{\overline{t_i'}} (M_i', I_i')$. Two cases are admissible.

1. $\overline{t_i'} \in \mathbb{R}_{\geq 0}$. Then, $\overline{t_i'} = \overline{t_i} \cdot c$, by the construction of σ_2. As $\sigma_1 \in \mathcal{FS}(\mathcal{TN}_1)$, we have that $M_i = M_{i-1}$ and $\forall t \in En(M_{i-1}) : I_i(t) = I_{i-1}(t) + \overline{t_i}$, due to Definition 2. Then, it holds that $I_i(t) \cdot c = I_{i-1}(t) \cdot c + \overline{t_i} \cdot c = I_{i-1}'(t) + \overline{t_i'} = I_i'(t)$, for all $t \in En(M_{i-1}' = M_i')$, Therefore, $(M_{i-1}', I_{i-1}') \xrightarrow{\overline{t_i'}} (M_i', I_i')$, according to Definition 2.
2. $\overline{t_i'} \in T$. Then, $\overline{t_i'} = \overline{t_i}$, by the construction of σ_2. As $\sigma_1 \in \mathcal{FS}(\mathcal{TN}_1)$, we have that $\overline{t_i} \in En(M_{i-1})$ and $Eft_1(\overline{t_i}) \leq I_{i-1}(\overline{t_i}) \leq Lft_1(\overline{t_i})$, because $\overline{t_i}$ can fire from (M_{i-1}, I_{i-1}). Hence, we obtain that $\overline{t_i'} \in En(M_{i-1}')$ and $Eft_2(\overline{t_i'}) = Eft_1(\overline{t_i}) \cdot c \leq I_{i-1}(\overline{t_i}) \cdot c = I_{i-1}'(\overline{t_i'}) = I_{i-1}(\overline{t_i}) \cdot c \leq \left\{ \begin{cases} \infty, & \text{if } Lft_1(\overline{t_i}) = \infty, \\ Lft_1(\overline{t_i}) \cdot c, & \text{otherwise.} \end{cases} \right\} = Lft_2(\overline{t_i'})$, due to $\mathcal{TN}_1$ and $\mathcal{TN}_2$ being time equivalent with a proportionality constant c. So, $\overline{t_i'}$ can fire from (M_{i-1}', I_{i-1}'). Since $\sigma_1 \in \mathcal{FS}(\mathcal{TN}_1)$, we have that $M_i = (M_{i-1} \backslash \bullet \overline{t_i}) \cup \overline{t_i} \bullet$ and $\forall t \in En(M_i) : I_i(t) = \begin{cases} 0, & \text{if } \uparrow enabled(t, M_{i-1}, \overline{t_i}), \\ I_{i-1}(t), & \text{otherwise} \end{cases}$, by Definition 2. Then, it holds that $(M_{i-1}' \backslash \bullet \overline{t_i'}) \cup \overline{t_i'} \bullet = M_i' = M_i$ and $\forall t \in En(M_i') : I_i(t) \cdot c = \begin{cases} 0, & \text{if } \uparrow enabled(t, M_{i-1}', \overline{t_i'}), \\ I_{i-1}(t) \cdot c = I_{i-1}'(t), & \text{otherwise} \end{cases} = I_i'(t)$. Therefore, $(M_{i-1}', I_{i-1}') \xrightarrow{\overline{t_i'}} (M_i', I_i')$, thanks to Definition 2.

Thus, $(M_0', I_0') \xrightarrow{\sigma_2} (M_k', I_k')$, i.e. $\sigma_2 \in \mathcal{FS}(\mathcal{TN}_2)$.
Reasoning analogously, we can show that for each $\sigma_2 \in \mathcal{FS}(\mathcal{TN}_2)$, there exists $\sigma_1 \in \mathcal{FS}(\mathcal{TN}_1)$ such that σ_1 and σ_2 have the same untimed part. □

Example 2. Consider the PN $\widetilde{\mathcal{N}}$ whose structure is shown in Fig. 1. Define the TPN $\widetilde{\mathcal{TN}_1} = (\widetilde{\mathcal{N}}, D_1)$, with $D_1(t_1) = [0, \frac{1}{4}]$, $D_1(t_2) = [\frac{1}{4}, \frac{3}{4}]$, $D_1(t_3) = [0, \frac{1}{2}]$. Construct a TPN $\widetilde{\mathcal{TN}_2} = (\widetilde{\mathcal{N}}, D_2)$, with $D_2 : T \to \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ such that $\widetilde{\mathcal{TN}_1}$ and $\widetilde{\mathcal{TN}_2}$ are time equivalent. Define the set $\mathcal{D}$ as follow: $\mathcal{D} = \left\{ n \,\middle|\, Eft_1(t) = \frac{m}{n},\ t \in T, m, n \in \mathbb{N}_{>0} \right\} \cup \left\{ n \,\middle|\, Lft_1(t) = \frac{m}{n},\ t \in T, m, n \in \mathbb{N}_{>0} \right\} = \{2, 4\}$. Let $c = LCM(\mathcal{D}) = LCM(2, 4) = 4$, $D_2(t_1) = \left[Eft_1(t_1) * c = 0 * 4, Lft_1(t_1) * c = \frac{1}{4} * 4 \right] =$

$[0,1], D_2(t_2) = \left[Eft_1(t_2) * c = \frac{1}{4} * 4, Lft_1(t_2) * c = \frac{3}{4} * 4\right] = [1,3]$, and $D_2(t_3) = \left[Eft_1(t_3) * c = 0 * 4, Lft_1(t_3) * c = \frac{1}{2} * 4\right] = [0,2]$. Then, the time intervals of $\widetilde{\mathcal{TN}_2}$ have natural-valued boundaries. Moreover, $\widetilde{\mathcal{TN}_1}$ and $\widetilde{\mathcal{TN}_2}$ are time equivalent, because the static functions in the TPNs are proportional to the non-zero constant 4. As a consequence, for example, for the run $\sigma_1 = t_1 t_3(0.25) t_2(0.125) t_3$ of $\widetilde{\mathcal{TN}_1}$, there exists the run $\sigma_2 = t_1 t_3(1) t_2(0.5) t_3$ of $\widetilde{\mathcal{TN}_2}$, with time elapsings that are four longer times. Conversely, for the run σ_2 of $\widetilde{\mathcal{TN}_2}$, there is the run σ_1 of $\widetilde{\mathcal{TN}_1}$ with time elapsings that are four times shorter. □

So, if two time Petri nets are time equivalent, then each run of the one TPN can be obtained from some run of the other TPN with the same untimed part and with time elapsings multiplied by the proportionality constant. In other words, time in the one TPN goes slower (or faster) than in the other TPN. With that, it is worth stressing that the TPNs have the same behavioral properties (e.g., safeness, liveness, marking reachability, etc.).

In the sequel, we will consider only TPNs with time intervals having natural-valued boundaries.

3. State Space Discretization for $\mathcal{TN}$

In this section, we demonstrate that in the TPN for any run there exists a run with the same untimed part and with natural-value time (and even unit time) elapsings.

For the TPN $\mathcal{TN}$, let $\widehat{\mathcal{FS}}(\mathcal{TN})$ be the set of all runs from $\mathcal{FS}(\mathcal{TN})$ of the form: $\hat{\sigma} = \tau_0\, t_1\, \tau_1 \dots \tau_{k-1}\, t_k\, \tau_k$, i.e. time elapsings and transition firings alternate in the runs. So, in $\mathcal{TN}$ for the run $\hat{\sigma} \in \widehat{\mathcal{FS}}(\mathcal{TN})$, we have the following changes in states: $(M_0, I_0) \xrightarrow{\tau_0} (M_0, I_0') \xrightarrow{t_1} (M_1, I_1) \xrightarrow{\tau_1} (M_1, I_1') \dots (M_{k-1}, I_{k-1}) \xrightarrow{\tau_{k-1}} (M_{k-1}, I'_{k-1}) \xrightarrow{t_k} (M_k, I_k) \xrightarrow{\tau_k} (M_k, I_k')$. As we will see later (in the proof of Corollary 2), any run from $\mathcal{FS}(\mathcal{TN})$ can be represented in the above form. Define the set $\widetilde{\mathcal{UFS}}(\mathcal{TN}) = \{Untimed(\hat{\sigma}) | \hat{\sigma} \in \widehat{\mathcal{FS}}(\mathcal{TN})\}$.

Next, for an untimed sequence from $\widetilde{\mathcal{UFS}}(\mathcal{TN})$, we construct the parametric run (which is, in fact, a modification of a run, with variables instead of the time elapsing values) and a set of conditions on the values of these variables, by induction of the number of the variables. At each induction step, we define a prefix of the parametric run and the conditions for the values of its variables, increasing the run's length by one variable.

Definition 4. Let $\mathcal{TN} = ((P, T, F, M_0, L), D)$ be a time Petri net, $t_1 \dots t_k \in \widetilde{\mathcal{UFS}}(\mathcal{TN})$ and $X = \{x_0, \dots, x_k\}$ be a set of variables. We construct a finite sequence of the tuples of the form $(\omega_i, B_{\omega_i}, M_{\omega_i}, I'_{\omega_i})$, by induction on $0 \le i \le k$.

$i = 0$. Then,
- $\omega_0 = x_0$;
- $B_{\omega_0} = \emptyset$;
- $M_{\omega_0} = M_0$;
- $I'_{\omega_0}(t) = x_0$, for all $t \in En(M_{\omega_0})$.

$i > 0$. Assume that $(\omega_{i-1}, B_{\omega_{i-1}}, M_{\omega_{i-1}}, I'_{\omega_{i-1}})$ is already constructed. Then,
- $\omega_i = \omega_{i-1} t_i x_i$;
- $B_{\omega_i} = B_{\omega_{i-1}} \cup \{Eft(t_i) \le I'_{\omega_{i-1}}(t_i) \le Lft(t_i)\}$;
- $M_{\omega_i} = (M_{\omega_{i-1}} \setminus {\bullet t_i}) \cup t_i{\bullet}$;
- $I'_{\omega_i}(t) = \begin{cases} x_i, & \text{if } \uparrow enabled(t, M_{\omega_{i-1}}, t_i), \\ I'_{\omega_{i-1}}(t) + x_i, & otherwise, \end{cases}$ for all $t \in En(M_{\omega_i})$.

Then, $\omega = \omega_k = x_0 t_1 x_1 \dots t_k x_k$ is a *parametric run of* $\mathcal{TN}$, and $B_\omega = B_{\omega_k} \cup \{0 \le x_k < \infty\}$ is the *set of conditions on the values of the variables from* X.

We use $\mathcal{B}_\omega = \{I'_{\omega_{i-1}}(t_i) | (Eft(t_i) \le I'_{\omega_{i-1}}(t_i) \le Lft(t_i)) \in B_\omega, 1 \le i \le k\} \cup \{x_k\}$ to denote the *set of the variable parts of the inequalities from* B_ω.

Example 3. Contemplate the TPN $\widetilde{\mathcal{TN}} = \big((P, T, F, M_0, L), D\big)$, shown in Fig. 1. For the transition sequence $Untimed(\hat{\sigma}) = t_1\ t_3\ t_2$, obtained from the run $\hat{\sigma} = (0.5)t_1(0.5)t_3(2.3)t_2(1.7)$ of $\widetilde{\mathcal{TN}}$, we construct the sequence of the following tuples $(\omega_i, B_{\omega_i}, M_{\omega_i}, I'_{\omega_i})$, with $0 \le i \le 3$.

– $i = 0$. Set $\omega_0 = x_0$; $B_{\omega_0} = \emptyset$; $M_{\omega_0} = M_0 = \{p_1, p_2\}$; $En(M_{\omega_0}) = \{t_1, t_3\}$; and $I'_{\omega_0}(t_1) = I'_{\omega_0}(t_3) = x_0$.

– $i = 1$. Set $\omega_1 = \omega_0 t_1 x_1 = x_0 t_1 x_1$;
$B_{\omega_1} = B_{\omega_0} \cup \{Eft(t_1) = 0 \le I'_{\omega_0}(t_1) \le Lft(t_1) = 1\} = B_{\omega_0} \cup \{0 \le x_0 \le 1\}$;
$M_{\omega_1} = (M_{\omega_0} \backslash {\bullet t_1}) \cup t_1 \bullet = \{p_3, p_2\}$ and $En(M_{\omega_1}) = \{t_3\}$;
$I'_{\omega_1}(t_3) = I'_{\omega_0}(t_3) + x_1 = x_0 + x_1$, as $\uparrow enabled(t_3, M_{\omega_1}, t_1)$ is false.

– $i = 2$. Set $\omega_2 = \omega_1 t_3 x_2 = x_0 t_1 x_1 t_3 x_2$;
$B_{\omega_2} = B_{\omega_1} \cup \{Eft(t_3) = 0 \le I'_{\omega_1}(t_3) \le Lft(t_3) = 2\} = B_{\omega_1} \cup \{0 \le x_0 + x_1 \le 2\}$;
$M_{\omega_2} = (M_{\omega_1} \backslash {\bullet t_3}) \cup t_3 \bullet = \{p_3, p_4\}$ and $En(M_{\omega_2}) = \{t_2\}$;
$I'_{\omega_2}(t_2) = x_2$, as $\uparrow enabled(t_2, M_{\omega_2}, t_3)$ is true.

– $i = 3$. *Set* $\omega_3 = \omega_2 t_2 x_3 = x_0 t_1 x_1 t_3 x_2 t_2 x_3$;
$B_{\omega_3} = B_{\omega_2} \cup \{Eft(t_2) = 1 \le I'_{\omega_2}(t_2) \le Lft(t_2) = 3\} = B_{\omega_2} \cup \{1 \le x_2 \le 3\}$;
$M_{\omega_3} = (M_{\omega_2} \backslash {\bullet t_2}) \cup t_2 \bullet = \{p_1, p_2\}$ and $En(M_{\omega_3}) = \{t_1, t_3\}$;
$I'_{\omega_3}(t_1) = I'_{\omega_3}(t_3) = x_3$, as $\uparrow enabled(t_1, M_{\omega_3}, t_2)$ and $\uparrow enabled(t_3, M_{\omega_3}, t_2)$ are true.

Then, the parametric run has the form $\omega = \omega_3 = x_0 t_1 x_1 t_3 x_2 t_2 x_3$, where $X = \{x_0, \dots, x_k\}$ is the set of the real variables. Moreover, it holds that

$$B_\omega = B_{\omega_3} = \begin{Bmatrix} 0 \le x_0 \le 1, \\ 0 \le x_0 + x_1 \le 2, \\ 1 \le x_2 \le 3, \\ 0 \le x_3 < \infty \end{Bmatrix}, \text{ and}$$

$$\mathcal{B}_\omega = \{x_0, x_0 + x_1, x_2, x_3\}.$$

□

A function $\beta: X = \{x_0, \dots, x_k\} \to \mathbb{R}_{\ge 0}$ is called *assignment of* ω. We write $[\omega]_\beta$ $([I'_{\omega_{i-1}}(t_i)]_\beta)$ for a parametric run ω (for the value of a linear function $I'_{\omega_{i-1}}(t_i) \in \mathcal{B}_\omega$) under the assignment β. The mapping β is a *solution* of B_ω iff $Eft(t_i) \le [I'_{\omega_{i-1}}(t_i)]_\beta \le Lft(t_i)$, for all $I'_{\omega_{i-1}}(t_i) \in \mathcal{B}_\omega$.

Example 4. Consider the parametric run $\omega = x_0 t_1 x_1 t_3 x_2 t_2 x_3$, the set $\mathcal{B}_\omega = \{x_0, x_0 + x_1, x_2, x_3\}$, from Example 3, and an assignment $\beta: \{x_0, \dots, x_3\} \to \mathbb{R}_{\ge 0}$ such that $\beta(x_0) = 0.7, \beta(x_1) = 0.3, \beta(x_2) = 1.4, \beta(x_3) = 2$. Then, we obtain that $[\omega]_\beta = (0.7)t_1(0.3)t_3(1.4)t_2(2)$. Moreover, we get: $Eft(t_1) = 0 \le [x_0]_\beta = 0.7 \le Lft(t_1) = 1$, $Eft(t_3) = 0 \le [x_0 + x_1]_\beta = 1 \le Lft(t_3) = 2$, $Eft(t_2) = 1 \le [x_2]_\beta = 1.4 \le Lft(t_2) = 3$. Therefore, β is a solution of B_ω. □

We next establish that any solution of B_ω maps a parametric run ω of the TPN to its run.

Lemma 1. Let $\mathcal{TN}$ be a TPN, $\omega = x_0 t_1 x_1 \dots t_k x_k$ be a parametric run of $\mathcal{TN}$, and B_ω be the set of conditions on the values of the variables $x_0, x_1, \dots, x_k$. If β is a solution of B_ω, then $[\omega]_\beta \in$

$\widehat{\mathcal{FS}}(\mathcal{TN})$. Moreover, for any $I'_{\omega_{i-1}}(t_i) \in \mathcal{B}_\omega$, $[I'_{\omega_{i-1}}(t_i)]_\beta$ is the value of $I'_{i-1}(t_i)$ in the run $[\omega]_\beta$, when $\mathcal{TN}$ functions along the run.

Proof. See Appendix. □

Next, for an arbitrary run $\hat{\sigma}$ from $\widehat{\mathcal{FS}}(\mathcal{TN})$ with real-value time elapsings, we construct a natural-value assignment β_ω to the variables in the corresponding parametric run ω of $\mathcal{TN}$, by induction on the number of the variables in ω (i.e. the number of time elapsings in the run $\hat{\sigma}$). Starting from the end of the run $\hat{\sigma}$, at each induction step, we round, the value of the corresponding time elapsing of $\hat{\sigma}$, down or up to a natural number nearest to the value and agreed with the values of the other time elapsings in $\hat{\sigma}$.

Definition 5. Let $\mathcal{TN}$ be a TPN, $\tau_0 t_1 \tau_1 \dots t_k \tau_k \in \widehat{\mathcal{FS}}(\mathcal{TN})$, $\omega = x_0 t_1 x_1 \dots t_k x_k$ be the parametric run of $\mathcal{TN}$, B_ω be the set of conditions on the values of the variables from $X = \{x_0, \dots, x_k\}$, and $\mathcal{B}_\omega$ be the set of the variable parts of the inequalities from B_ω. We construct a sequence of functions $\beta_i: X \to \mathbb{R}_{\geq 0}$ by induction on $0 \leq i \leq k$.

$i = 0$. Then, for all $0 \leq j \leq k$,

$$\beta_0(x_j) = \begin{cases} \lfloor \tau_j \rfloor & \text{if } j = k, \\ \tau_j & \text{otherwise} \end{cases}.$$

$i > 0$. In the construction of β_i, we use auxiliary functions defined for all $0 \leq j \leq k$ as follows:

$$\underline{\beta_i}(x_j) = \begin{cases} \lfloor \beta_{i-1}(x_j) \rfloor & \text{if } j = k - i, \\ \beta_{i-1}(x_j) & \text{otherwise;} \end{cases} \text{ and } \overline{\beta_i}(x_j) = \begin{cases} \lceil \beta_{i-1}(x_j) \rceil & \text{if } j = k - i, \\ \beta_{i-1}(x_j) & \text{otherwise.} \end{cases}$$

If $\exists\, I'_{\omega_{l-1}}(t_l) \in \mathcal{B}_\omega$ $(1 \leq l \leq k)$ such that $\lceil [I'_{\omega_{l-1}}(t_l)]_{\underline{\beta_i}} \rceil < \lfloor [I'_{\omega_{l-1}}(t_l)]_{\beta_0} \rfloor$,
then $\beta_i = \overline{\beta_i}$, else $\beta_i = \underline{\beta_i}$.

Define a natural-value assignment $\beta_\omega: X \to \mathbb{N}$ as follows: $\beta_\omega = \beta_k$.

Example 5. Consider the TPN $\widetilde{\mathcal{TN}}$ from Example 1, the run $\hat{\sigma} = (0.5)t_1(0.5)t_3(2.3)t_2(1.7) \in \widehat{\mathcal{FS}}(\widetilde{\mathcal{TN}})$, the parametric run $\omega = x_0 t_1 x_1 t_3 x_2 t_2 x_3$ and the set $\mathcal{B}_\omega = \{x_0, x_0 + x_1, x_2, x_3\}$ from Example 3. Using Definition 5, construct the sequence of the assignments β_i by induction on $0 \leq i \leq 3$.

– $i = 0$. We set $\beta_0(x_j)$, for all $0 \leq j \leq k = 3$, as follows: $\beta_0(x_0) = \tau_0 = 0.5$, $\beta_0(x_1) = \tau_1 = 0.5$, $\beta_0(x_2) = \tau_2 = 2.3$, and $\beta_0(x_3) = \lfloor \tau_3 = 1.7 \rfloor = 1$, because $j = k = 3$.

– $i = 1$. Construct auxiliary functions $\underline{\beta_1}(x_j)$ and $\overline{\beta_1}(x_j)$, for all $0 \leq j \leq k = 3$, as follows: $\underline{\beta_1}(x_0) = \overline{\beta_1}(x_0) = \beta_0(x_0) = 0.5$, $\underline{\beta_1}(x_1) = \overline{\beta_1}(x_1) = \beta_0(x_1) = 0.5$, $\underline{\beta_1}(x_3) = \overline{\beta_1}(x_3) = \beta_0(x_3) = 1$, and $\underline{\beta_1}(x_2) = \lfloor \beta_0(x_2) \rfloor = 2$, $\overline{\beta_1}(x_2) = \lceil \beta_0(x_2) \rceil = 3$, because $j = k - i = 2$.

Moreover, we get: $\lceil [x_0]_{\underline{\beta_1}} \rceil = 1 \geq \lfloor [x_0]_{\beta_0} \rfloor = 0$, $\lceil [x_0 + x_1]_{\underline{\beta_1}} \rceil = 1 \geq \lfloor [x_0 + x_1]_{\beta_0} \rfloor = 1$, $\lceil [x_2]_{\underline{\beta_1}} \rceil = 2 \geq \lfloor [x_2]_{\beta_0} \rfloor = 2$. Then, $\beta_1 = \underline{\beta_1}$.

– $i = 2$. Construct $\underline{\beta_2}(x_j)$ and $\overline{\beta_2}(x_j)$, with $0 \leq j \leq 3$, as follows: $\underline{\beta_2}(x_0) = \overline{\beta_2}(x_0) = \beta_1(x_0) = 0.5$, $\underline{\beta_2}(x_2) = \overline{\beta_2}(x_2) = \beta_1(x_2) = 2$, $\underline{\beta_2}(x_3) = \overline{\beta_2}(x_3) = \beta_1(x_3) = 1$ and $\underline{\beta_2}(x_1) = \lfloor \beta_1(x_1) \rfloor = 0$, $\overline{\beta_2}(x_1) = \lceil \beta_1(x_1) \rceil = 1$, because $j = k - i = 1$.

Moreover, we get: $\lceil [x_0]_{\underline{\beta_2}} \rceil = 1 \geq \lfloor [x_0]_{\beta_0} \rfloor = 0$, $\lceil [x_0 + x_1]_{\underline{\beta_2}} \rceil = 1 \geq \lfloor [x_0 + x_1]_{\beta_0} \rfloor = 1$,

$\lceil [x_2]_{\underline{\beta_2}} \rceil = 2 \geq \lfloor [x_2]_{\beta_0} \rfloor = 2$. Then, $\beta_2 = \underline{\beta_2}$.

– $i = 3$. Construct $\underline{\beta_3}(x_j)$ and $\overline{\beta_3}(x_j)$, with $0 \leq j \leq 3$, as follows: $\underline{\beta_3}(x_1) = \overline{\beta_3}(x_1) = \beta_2(x_1) = 0$, $\underline{\beta_3}(x_2) = \overline{\beta_3}(x_2) = \beta_2(x_2) = 2$, $\underline{\beta_3}(x_3) = \overline{\beta_3}(x_3) = \beta_2(x_3) = 1$ and $\underline{\beta_3}(x_0) = \lfloor \beta_2(x_0) \rfloor = 0$, $\overline{\beta_3}(x_0) = \lceil \beta_2(x_0) \rceil = 1$, because $j = k - i = 0$.

Moreover, we have $\lceil [x_0]_{\underline{\beta_3}} \rceil = 0 \geq \lfloor [x_0]_{\beta_0} \rfloor = 0$, $\lceil [x_0 + x_1]_{\underline{\beta_3}} \rceil = 0 < \lfloor [x_0 + x_1]_{\beta_0} \rfloor = 1$, $\lceil [x_2]_{\underline{\beta_3}} \rceil = 2 \geq \lfloor [x_2]_{\beta_0} \rfloor = 2$. Then, there exists $I'_{\omega_1}(t_3) \in \mathcal{B}_\omega$ such that $\lceil \left[I'_{\omega_1}(t_3)\right]_{\underline{\beta_3}} \rceil < \lfloor \left[I'_{\omega_1}(t_3)\right]_{\beta_0} \rfloor$. Therefore, $\beta_3 = \overline{\beta_3}$.

Then, $\beta_\omega = \beta_3$, and we obtain the sequence $[\omega]_{\beta_\omega} = (1)t_1(0)t_3(2)t_2(1)$. □

Next, we show that the assignment β_ω is a solution of ω, i.e. β_ω satisfies the inequalities from B_ω.

Proposition 1. β_ω is a solution of B_ω.

Proof. See Appendix. □

Thanks to Lemma 1 and Proposition 1, the theorem below follows immediately.

Theorem 2. Let $\mathcal{TN}$ be a time Petri net and $\omega = x_0 t_1 x_1 \dots t_k x_k$ be a parametric run of $\mathcal{TN}$. Then, there exists a mapping $\beta_\omega : X = \{x_0, \dots, x_k\} \to \mathbb{N}$ such that $[\omega]_{\beta_\omega} \in \widehat{\mathcal{FS}}(\mathcal{TN})$.

Proof. Consider the mapping $\beta_\omega : X = \{x_0, \dots, x_k\} \to \mathbb{N}$ from Definition 5. By Proposition 1, β_ω is a solution of B_ω, and, moreover, $[\omega]_{\beta_\omega} \in \widehat{\mathcal{FS}}(\mathcal{TN})$, due to Lemma 1. □

We are now ready to show that in the TPN for any run, there exists a run with the same untimed part and with unit time elapsings.

Corollary 2. Let $\mathcal{TN}$ be a time Petri net and $\sigma \in \mathcal{FS}(\mathcal{TN})$. Then, there is $\sigma' \in \mathcal{FS}(\mathcal{TN})$ with unit-value time elapsings such that $Untimed(\sigma) = Untimed(\sigma')$.

Proof. Due to Definition 2, we obtain the following properties for time elapsings:

a) $S \xrightarrow{0} S$;

b) if $S \xrightarrow{\tau} S'$ and $S' \xrightarrow{\tau'} S''$, with $\tau, \tau' \in \mathbb{R}_{\geq 0}$, then $S \xrightarrow{\tau+\tau'} S''$;

c) if $S \xrightarrow{\tau} S'$, then for every $\tau', \tau'' \in \mathbb{R}_{\geq 0}$ such that $\tau = \tau' + \tau''$, $S \xrightarrow{\tau'} S'' \xrightarrow{\tau''} S'$ for some S''.

By items a) and b), there exists $\hat{\sigma} \in \widehat{\mathcal{FS}}(\mathcal{TN})$ such that $Untimed(\sigma) = Untimed(\hat{\sigma})$, thanks to σ being a finite sequence. Due to Theorem 2, there is a run $\widehat{\sigma'} \in \widehat{\mathcal{FS}}(\mathcal{TN})$ with natural-value time elapsings such that $Untimed(\hat{\sigma}) = Untimed(\widehat{\sigma'})$. Thanks to c), we can construct $\sigma' \in \mathcal{FS}(\mathcal{TN})$ with unit time elapsings such that $Untimed(\widehat{\sigma'}) = Untimed(\sigma')$. Therefore, we obtain that $Untimed(\sigma) = Untimed(\sigma')$. □

Thanks to Corollary 2, in the sequel, we will consider time Petri nets with unit time elapsings (denoted by √).

4. Time Processes of $\mathcal{TN}$

In this section, the concept of causality-based net process is presented and studied in the context of TPNs with weak semantics. We start with definitions related to casual nets that contain events and conditions connected by causal dependence and concurrency (absence of causality).

Definition 6. A (*labeled over* $Act \cup \{tick\}$) *casual net* is a finite, acyclic net $TN = (B, E, G, l)$ with a set B of conditions; a set E of events; a flow relation $G \subseteq (B \times E) \cup (E \times B)$ such that $|b\bullet| \leq 1 \wedge |\bullet b| \leq 1$, for all $b \in B$; a labeling function $l : E \to Act \cup \{tick\}$.

Informally speaking, the «tick» label means a clock ticking.

Casual nets $TN = (B, E, G, l)$ and $TN' = (B', E', G', l')$ are called *isomorphic* (denoted $TN \simeq TN'$) iff there exists a bijective mapping $\gamma : B \cup E \to B' \cup E'$ such that:

- $\gamma(B) = B'$ and $\gamma(E) = E'$;
- $x\ G\ y \Leftrightarrow \gamma(x)\ G'\ \gamma(y)$, for all $x, y \in B \cup E$;
- $l(e) = l'(\gamma(e))$, for all $e \in E$.

Introduce auxiliary notions and notations for the casual net $TN = (B, E, G, l)$.

The set $\bullet b$ ($b\bullet$) is associated with a single event, for any $b \in B$. Let $\bullet TN = \{b \in B \mid \bullet b = \emptyset\}$.

Define $\prec = G^+, \preccurlyeq = G^*$ (*causality*). A subset $E' \subseteq E$ is a *downward closed set of events* iff $e \in E'$ implies $e' \in E'$, for all $e' \prec e$. In this case, $Cut(E') = (\bullet TN \cup E'\bullet) \setminus \bullet E'$.

A subset $B' \subseteq B$ is a *co-set* (*a subset of concurrent conditions*) iff $\neg(b \prec b')$ and $\neg(b' \prec b)$, for all $b, b' \in B'$. A *cut* is a maximal (w.r.t. set inclusion) co-set.

A sequence $\rho = e_1 \ldots e_n$ ($n \geq 0$) of events is a *linearization* of TN [5] if each event of TN appears in the sequence exactly once, and the following holds: $e_i \prec e_j \Rightarrow i < j$, for all $1 \leq i, j \leq n$. For a linearization $\rho = e_1 \ldots e_n$ ($n \geq 0$) of TN, define the following:

- ρ_0 is the empty sequence and $\rho_k = e_1 \ldots e_k$ ($1 \leq k \leq n$);
- $E_0 = \emptyset$ and $E_k = \bigcup_{1 \leq i \leq k} e_i$ ($1 \leq k \leq n$).

By the construction of the linearization, E_k is a downward closed set of events, for all $0 \leq k \leq n$. As we will see later (in Lemma 2), $Cut(E_k)$ is a cut, for all $1 \leq k \leq n$.

Informally speaking, a linearization is an interleaving representation of a "computation" of TN and the value of the function Cut of any prefix of the linearization is a "marking" of TN, reachable after occurring the events from the prefix.

Example 6. Fig. 2 shows a causal net $\widetilde{TN} = (B, E, G, l)$, with $B = \{b_1, \ldots, b_{10}\}$; $E = \{e_1, \ldots, e_5\}$; $G = \{(b_1, e_1), (e_1, b_3), (b_3, e_2), (b_2, e_2), \ldots, (e_5, b_9), (e_5, b_{10})\}$; $l(e_1) = l(e_3) = b$, $l(e_2) = l(e_4) = tick$, $l(e_5) = a$. We see that $|\bullet b| \leq 1 \wedge |b\bullet| \leq 1$, for all $b \in B$, and $\bullet\widetilde{TN} = \{b_1, b_2\}$. Clearly, $e_1 \prec e_2 \prec e_3 \prec e_4 \prec e_5$. Moreover, $\{b_1, b_2\}$, $\{b_2, b_3\}$, ..., $\{b_9, b_{10}\}$ are cuts in $\widetilde{TN}$. It is easy to check that $\tilde{\rho} = e_1, e_2, e_3, e_4, e_5$ is a linearization of $\widetilde{TN}$, because each event of $\widetilde{TN}$ appears in the sequence exactly once, and if $e_i \prec e_j$, then $i < j$, for all $1 \leq i, j \leq 5$. Define the downward closed sets $E_i = \bigcup_{1 \leq j \leq k} e_j$ and the sets $Cut(E_i) = (\bullet TN \cup E_i\bullet) \setminus \bullet E_i = (\{b_1, b_2\} \cup E_i\bullet) \setminus \bullet E_i$, for all $0 \leq i \leq 5$, as follows:

- $E_0 = \emptyset$, $Cut(E_0) = (\{b_1, b_2\} \cup \emptyset) \setminus \emptyset = \{b_1, b_2\}$;
- $E_1 = \{e_1\}$, $Cut(E_1) = (\{b_1, b_2\} \cup \{b_3\}) \setminus \{b_1\} = \{b_2, b_3\}$;
- $E_2 = \{e_1, e_2\}$, $Cut(E_2) = (\{b_1, b_2\} \cup \{b_3, b_4, b_5\}) \setminus \{b_1, b_2, b_3\} = \{b_4, b_5\}$;
- $E_3 = \{e_1, e_2, e_3\}$, $Cut(E_3) = (\{b_1, b_2\} \cup \{b_3, b_4, b_5, b_6\}) \setminus \{b_1, b_2, b_3, b_5\} = \{b_4, b_6\}$;
- $E_4 = \{e_1, e_2, e_3, e_4\}$, $Cut(E_4) = (\{b_1, b_2\} \cup \{b_3, b_4, b_5, b_6, b_7, b_8\}) \setminus \{b_1, b_2, b_3, b_4, b_5, b_6\} = \{b_7, b_8\}$;
- $E_5 = \{e_1, e_2, e_3, e_4, e_5\}$, $Cut(E_5) = B \setminus \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8\} = \{b_9, b_{10}\}$. □

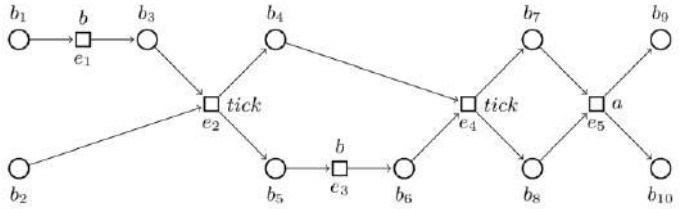


Fig. 2. *A casual net* $\widetilde{TN}$

Proposition 2. Any casual net TN has a linearization $\rho = e_1 \dots e_n$.

Proof. Take the maximal (w.r.t. set inclusion) set $min(E) \subseteq E$ such that if $e \in min(E)$, then $\neg(e' \prec e)$, for all $e' \in E$. After removing an event e_{min} from E, we get that $min(E') = min(E) \setminus e_{min}$ or $min(E') = (min(E) \setminus e_{min}) \cup (e_{min}\bullet)\bullet$, where $E' = E \setminus e_{min}$. Construct a sequence $\rho = e_1 \dots e_n$ of events, by selecting a minimal event and removing it from E, at each step. By the construction, any event of TN appears in the sequence exactly once and $e_i \prec e_j \Rightarrow i < j$, for all $1 \leq i, j \leq n$, due to TN being a causal net. □

The results of the below lemma will be useful to establish the relationships between "markings" (values of the function Cut) of a causal net TN and markings of a time Petri net $\mathcal{TN}$, and between linearizations of TN and runs of $\mathcal{TN}$, when we construct partial order semantic for time Petri nets.

Lemma 2. Let TN be a casual net and $\rho = e_1 \dots e_n$ its linearization. Then, it holds:

a) $Cut(E_k) = (Cut(E_{k-1}) \setminus \bullet e_k) \cup e_k\bullet$, for all $1 \leq k \leq n$;

b) $\bullet e_k \subseteq Cut(E_{k-1})$, for all $1 \leq k \leq n$;

c) if $Cut(E_k) \neq \emptyset$, then $Cut(E_k)$ is a cut of TN, for all $0 \leq k \leq n$.

Proof. See Appendix. □

Next, we introduce a notion of a homomorphism from a causal net to a time Petri net, in order to define the concept of time processes of the time Petri net.

Definition 7. Let $\mathcal{TN} = ((P, T, F, M_0, L), D)$ be a TPN and $TN = (B, E, G, l)$ be a casual net. A *homomorphism*[1] from TN to $\mathcal{TN}$ is a mapping $\varphi : (B \cup E) \to (P \cup T \cup \{\surd\})$ such that it holds the following:

- $\varphi(B) \subseteq P$ and $\varphi(E) \subseteq (T \cup \{\surd\})$;
- for all $e \in E$ such that $\varphi(e) \in T$,
  - the restriction of φ to $\bullet e$ is a bijection between $\bullet e$ and $\bullet\varphi(e)$,
  - the restriction of φ to $e\bullet$ is a bijection between $e\bullet$ and $\varphi(e)\bullet$;
- for all $e \in E$ such that $\varphi(e) = \surd$,
  - the restriction of φ to $\bullet e$ and the restriction of φ to $e\bullet$ are injections,
  - $\bullet e$ and $e\bullet$ are cuts of TN and $\varphi(\bullet e) = \varphi(e\bullet)$;
- the restriction of φ to $\bullet TN$ is a bijection between $\bullet TN$ and M_0;

[1] In fact, φ is a homomorphism from TN to $\mathcal{TN}' = \big((P, T, F', M_0, L'), D\big)$, where $F' \subseteq (P \times T \cup \{\surd\}) \cup (T \cup \{\surd\} \times P)$ such that $F' = F \cup \{(p, \surd), (\surd, p) \mid p \in P\}$, and $L' : T \cup \{\surd\} \to Act \cup \{tick\}$, such that $L'(x) = \begin{cases} L(x), \text{if } x \in T \\ tick, \text{otherwise} \end{cases}$. In this case, we can see that φ is a structure-preserving mapping. However, following the traditions of terms and definitions in the literature on TPNs, we omit this construction of $\mathcal{TN}'$.

– $l(e) = \begin{cases} L(\varphi(e)), & \text{if } \varphi(e) \in T, \\ tick, & \text{otherwise} \end{cases}$, for all $e \in E$.

Then, the pair $\pi = (TN, \varphi)$ is called *a time process of* $\mathcal{TN}$.

Time processes $\pi = (TN, \varphi) = \big((B, E, G, l), \varphi\big)$ and $\pi' = (TN', \varphi') = \big((B', E', G', l'), \varphi'\big)$ of $\mathcal{TN}$ are *isomorphic* (denoted $\pi \simeq \pi'$) if there is an isomorphism $\gamma : TN \simeq TN'$ such that $\varphi(x) = \varphi'(\gamma(x))$, for all $x \in B \cup E$.

Example 7. Consider the TPN $\widetilde{\mathcal{TN}}$ depicted in Fig. 1, the casual net $\widetilde{TN}$ shown in Fig. 2, and a mapping φ defined as follows: $\varphi(b_1) = \varphi(b_9) = p_1$, $\varphi(b_2) = \varphi(b_5) = \varphi(b_{10}) = p_2$, $\varphi(b_3) = \varphi(b_4) = \varphi(b_7) = p_3, \varphi(b_6) = \varphi(b_8) = p_4, \varphi(e_1) = t_1$, $\varphi(e_5) = t_2, \varphi(e_3) = t_3$, $\varphi(e_2) = \varphi(e_4) = \surd$; Then, we have that $\varphi(B) \subseteq P$, $\varphi(E) \subseteq (T \cup \{\surd\})$, and the restriction of φ *to* $\bullet TN = \{b_1, b_2\}$ is a bijection between $\{b_1, b_2\}$ and $M_0 = \{p_1, p_2\}$. Moreover, it holds the following: for all $e \in E$ such that $\varphi(e) \in T$, the restriction of φ to •e ($e\bullet$) is a bijection between •e ($e\bullet$) and •φ(e) ($\varphi(e)\bullet$); and for all $e \in E$ such that $\varphi(e) = \surd$, the restriction of φ to $\bullet e$ ($e\bullet$) is an injection, $\bullet e$ and $e\bullet$ are cuts of TN and $\varphi(\bullet e) = \varphi(e\bullet)$. For example, consider the events e_1 and e_2. We know that $\varphi(e_1) = t_1$ and $\varphi(e_2) = \surd$. The restriction of φ to $\bullet e_1 = \{b_1\}$ ($e_1\bullet = \{b_3\}$) is a bijection between $\{b_1\}$ ($\{b_3\}$) and $\varphi(b_1) = \{p_1\}$ ($\varphi(b_3) = \{p_3\}$). Furthermore, the restriction of φ to $\bullet e_2 = \{b_2, b_3\}$ ($e_2\bullet = \{b_4, b_5\}$) is an injection, $\{b_2, b_3\}$ and $\{b_4, b_5\}$ are cuts of TN and $\varphi(\{b_2, b_3\}) = \varphi(\{b_5, b_4\}) = \{p_2, p_3\}$. We see that $l(e) = \begin{cases} L(\varphi(e)), \text{if } \varphi(e) \in T \\ \text{tick, otherwise} \end{cases}$, for all $e \in E$. Therefore, φ is the homomorphism from $\widetilde{TN}$ to $\widetilde{\mathcal{TN}}$, and, hence, $\pi = (\widetilde{TN}, \varphi)$ is a time process of $\widetilde{\mathcal{TN}}$. □

For a time process $\pi = (TN, \varphi)$ of $\mathcal{TN}$, we introduce the function Age that defines "age" of each condition b of TN. More specifically, if $b \in \bullet TN$ is an input condition in TN, i.e. the place $\varphi(b)$ contains a token at the initial marking of $\mathcal{TN}$, then the "age" of b is equal to 0. Also, if b is an output condition of an event e that corresponds to the firing of the transition $\varphi(e)$ of $\mathcal{TN}$, i.e. the place $\varphi(b)$ of $\mathcal{TN}$ got a token immediately after the firing of $\varphi(e)$, then the "age" of b is equal to 0. Otherwise, i.e. if b is an output condition of an event e that corresponds to time elapsing, then the "age" of b is the increased by 1 "age" of the input condition b' of e, such that $\varphi(b) = \varphi(b')$, i.e. b and b' match in the same place of $\mathcal{TN}$.

$$Age(b) = \begin{cases} 0, & if\ b \in \bullet TN \vee (b \in e\bullet, \varphi(e) \in T), \\ Age(b') + 1, & if\ b \in e\bullet, \varphi(e) = \surd, b' \in \bullet e, \varphi(b') = \varphi(b). \end{cases}$$

Notice that if $b \in e\bullet$ and $\varphi(e) = \surd$, then there exists the only one condition $b' \in \bullet e$ such that $\varphi(b') = \varphi(b)$, due to Definition 7. In this case, the definition of the function Age is correct.
Informally speaking, the function Age matches each condition b of TN to the amount of time that has elapsed since the corresponding place $\varphi(b)$ of $\mathcal{TN}$ got a token, when $\mathcal{TN}$ progresses.

For a co-set B' of conditions of TN and a transition t of $\mathcal{TN}$ such that t is enabled at the marking $\varphi(B')$, determine the function **Clock** whose value is equal to the minimum "age" of the conditions from B', that correspond the input places of t.

$$\mathbf{Clock}(B', t) = \begin{cases} \bot, & if\ \bullet t = \emptyset, \\ \min\{Age(b) \mid \varphi(b) \in \bullet t, b \in B'\}, & otherwise. \end{cases}$$

Informally speaking, the function **Clock** matches B' and t to the amount of time that has elapsed since a token from the marking $\varphi(B')$ appeared in the last input place of the transition t, i.e. since the transition t became enabled at $\varphi(B')$. Later, we will establish a correspondence between the function **Clock** and the dynamic timing function I, when the TPN progresses.

We are now ready to introduce the concept of an admissible (correct) time process of $\mathcal{TN}$.

Definition 8. Let $\mathcal{TN}$ be a time Petri net. A time process $\pi = (TN, \varphi)$ of $\mathcal{TN}$ is *admissible* iff for all $e \in E$ it holds:

$$\varphi(e) \in T \Rightarrow Eft(\varphi(e)) \leq \mathbf{Clock}(\bullet e, \varphi(e)) \leq Lft(\varphi(e)).$$

Example 8. Verify that the time process π from Example 7 of $\widetilde{\mathcal{TN}}$ shown in Fig. 1 is admissible. By definitions, we have:

- $\mathbf{Clock}(\bullet e_1, \varphi(e_1)) = Age(b_1) = 0$;
- $\mathbf{Clock}(\bullet e_3, \varphi(e_3)) = Age(b_5) = Age(b_2) = 0 + 1 = 1$;
- $\mathbf{Clock}(\bullet e_5, \varphi(e_5)) = \min\{Age(b_7), Age(b_8)\} = \min\{Age(b_4) + 1, Age(b_6) = 0 + 1\} = \min\{(Age(b_3) = 0 + 1) + 1, 1\} = 1$.

Then, we obtain:

- $Eft(t_1) = 0 \leq \mathbf{Clock}(\bullet e_1, \varphi(e_1) = t_1) \leq 1 = Lft(\varphi(e_1))$,
- $Eft(t_3) = 0 \leq \mathbf{Clock}(\bullet e_3, \varphi(e_3) = t_3) \leq 2 = Lft(t_3)$,
- $Eft(t_2) = 1 \leq \mathbf{Clock}(\bullet e_5, \varphi(e_5) = t_2) \leq 3 = Lft(t_2)$.

So, π is an admissible time process of $\widetilde{\mathcal{TN}}$. □

5. Relating Runs and Time Processes of $\mathcal{TN}$

In this section, relationships between runs and linearizations (computations) of admissible time processes are investigated, in the context of time Petri nets. For this purpose, we define a mapping FS from a linearization $\rho = e_1 \ldots e_n$ of a time process $\pi = (TN, \varphi)$ of the TPN $\mathcal{TN}$ to the sequence of the form: $FS(\rho) = \varphi(e_1) \ldots \varphi(e_n)$. Here, TN is a causal net and φ is a homomorphism from TN to $\mathcal{TN}$.

First, we prove that if FS maps a prefix ρ_i $(0 \leq i \leq n)$ of the linearization ρ to the run of $\mathcal{TN}$ and (M_i, I_i) is the state reachable by the run, then φ maps the value of the function Cut of this prefix to the marking M_i. Moreover, for any transition t enabled at M_i, the value of the dynamic timing function $I_i(t)$ is equal to $\mathbf{Clock}(Cut(E_i), t)$, where E_i is the set of events from ρ_i.

Lemma 3. Let $\pi = (TN, \varphi)$ be a time process of the TPN $\mathcal{TN}$ and $\rho = e_1 \ldots e_n$ be a linearization of TN. If $FS(\rho_i) = \varphi(e_1) \ldots \varphi(e_i)$ is the run of $\mathcal{TN}$ from (M_0, I_0) to (M_i, I_i) for some $0 \leq i \leq n$, then it holds:

a) the restriction of φ to $Cut(E_i)$ is a bijection between $Cut(E_i)$ and M_i;
b) $\mathbf{Clock}(Cut(E_i), t) = I_i(t)$, for all $t \in En(M_i)$;
c) if $i < n$ and $\varphi(e_{i+1}) \in En(M_i)$, then $\mathbf{Clock}(\bullet e_{i+1}, \varphi(e_{i+1}) = I_i(\varphi(e_{i+1}))$.

Proof. See Appendix. □

We are now ready to establish an important property of the FS mapping – any linearization of a time process of the TPN is mapped to its runs.

Theorem 3. Given an admissible time process $\pi = (TN, \varphi)$ of $\mathcal{TN}$ and a linearization $\rho = e_1 \ldots e_n$ of TN, $FS(\rho)$ is a run of $\mathcal{TN}$.

Proof. We shall prove by induction on $0 \leq i \leq n$ that $FS(\rho_i)$ is a run of $\mathcal{TN}$.

$i = 0$. Then, $FS(\rho_0)$ is the empty run.

$i > 0$. By the induction hypothesis, $FS(\rho_{i-1})$ is a run of $\mathcal{TN}$. If $\varphi(e_i) = \sqrt{}$, then $FS(\rho_i)$ is a run of $\mathcal{TN}$, due to Definition 2. Assume $\varphi(e_i) \in T$. By Lemma 2(b), we have $\varphi(\bullet e_i) \subseteq \varphi(Cut(E_{i-1}))$.

As the restriction of φ to $\bullet e$ is a bijection between $\bullet e$ and $\bullet\varphi(e)$, we obtain $\bullet\varphi(e_i) \subseteq \varphi(Cut(E_{i-1}))$. Thus, we get $\varphi(e_i) \in En(M_{i-1})$, due to Lemma 3(a). Thanks to Lemma 3(c), we have that $I_{i-1}(\varphi(e_i)) = \mathbf{Clock}(\bullet e_i, \varphi(e_i))$. Then, it holds that $Eft(\varphi(e_i)) \le I_{i-1}(\varphi(e_i)) \le Lft(\varphi(e_i))$, by Definition 8. Therefore, $\varphi(e_i)$ can fire from (M_{i-1}, I_{i-1}), i.e. $FS(\rho_i)$ is a run of $\mathcal{TN}$. □

Next, we show that the mapping FS is a surjection, i.e. for an arbitrary run σ of $\mathcal{TN}$, there is exists an admissible time process $\pi^* = (TN^*, \varphi^*)$ of $\mathcal{TN}$ and a linearization ρ^* of TN^* such that $FS(\rho^*) = \sigma$. The following definition provides constructions of π^* and ρ^*.

Definition 9. Let $\mathcal{TN} = ((P, T, F, M_0, L), D)$ be a time Petri net and $\sigma = \overline{t_1} \dots \overline{t_n} \in (T \cup \{\surd\})^n$ be a run of $\mathcal{TN}$.

We construct a finite sequence of tuples (E_i, B_i, G_i, C_i) by induction on $0 \le i \le n$.

$i = 0$. Then, set:
- $E_0 = \emptyset$;
- $B_0 = \{b_{0,p} | \, p \in M_0\}$;
- $G_0 = \emptyset$;
- $C_0 = B_0$;

$i > 0$. Assume that $(E_{i-1}, B_{i-1}, G_{i-1}, C_{i-1})$ is already constructed. Then, set:
- $E_i = E_{i-1} \cup \{e_i\}$;
- $B_i = B_{i-1} \cup \begin{cases} \{b_{i,p} \, | b_{j,p} \in C_{i-1}\}, & if \ \overline{t_i} = \surd, \\ \{b_{i,p} \, | p \in \overline{t_i}\bullet\}, & otherwise \end{cases}$;
- $G_i = G_{i-1} \cup \{(e_i, b_{i,p}) | \, b_{i,p} \in B_i\} \cup \begin{cases} \{(b_{j,p}, e_i) | b_{j,p} \in C_{i-1}\}, & if \ \overline{t_i} = \surd, \\ \{(b_{j,p}, e_i) | b_{j,p} \in C_{i-1}, \ p \in \bullet\overline{t_i}\}, & otherwise \end{cases}$;
- $C_i = (C_{i-1} \setminus \bullet e_i) \cup e_i\bullet$.

Define $\pi^* = (TN^*, \varphi^*) = ((B, E, G, l^*), \varphi^*)$ as follows:
- $B = B_n, E = E_n, G = G_n$;
- $\varphi^*(e_i) = \overline{t_i}$, for all $e_i \in E$, and $\varphi^*(b_{i,p}) = p$, for all $b_{i,p} \in B$;
- $l^*(e) = \begin{cases} tick, & if \varphi^*(e) = \surd, \\ L(\varphi^*(e)), & otherwise \end{cases}, for \ all \ e \in E$.

Determine $\rho^* = e_1 \dots e_n$.

Example 9. Consider the time Petri net $\widetilde{\mathcal{TN}}$ depicted in Fig. 1 and its run $\sigma = t_1\surd t_3\surd t_2$. We construct the sequence of the following tuples (E_i, B_i, G_i, C_i), with $0 \le i \le 5$.
- $i = 0$. Set $E_0 = \emptyset$; $C_0 = B_0 = \{b_{0,p} | p \in M_0 = \{p_1, p_2\}\} = \{b_{0,p_1}, b_{0,p_2}\}$; $G_0 = \emptyset$.
- $i = 1$. Set $E_1 = E_0 \cup \{e_1\}$; $B_1 = B_0 \cup \{b_{1,p} \mid p \in t_1\bullet = \{p_3\}\} = \{b_{1,p_3}\}$; $G_1 = G_0 \cup \{(e_1, b_{1,p}) | \, b_{1,p} \in B_1\} \cup \{(b_{j,p}, e_1) \mid b_{j,p} \in C_0, \ p \in \bullet t_1 = \{p_1\}\} = G_0 \cup \{(e_1, b_{1,p_3}), (b_{0,p_1}, e_1)\}$; $C_1 = (C_0 \setminus \bullet e_1) \cup e_1\bullet = \{b_{0,p_2}, b_{1,p_3}\}$.
- $i = 2$. Set $E_2 = E_1 \cup \{e_2\}$; $B_2 = B_1 \cup \{b_{2,p} \mid b_{j,p} \in C_1\} = \{b_{2,p_2}, b_{2,p_3}\}$; $G_2 = G_1 \cup \{(e_2, b_{2,p}) | \, b_{2,p} \in B_2\} \cup \{(b_{j,p}, e_2) \mid b_{j,p} \in C_1\} = G_1 \cup \{(e_2, b_{2,p_3}), (e_2, b_{2,p_2}), (b_{1,p_3}, e_2), (b_{0,p_2}, e_2)\}$; $C_2 = (C_1 \setminus \bullet e_2) \cup e_2\bullet = \{b_{2,p_3}, b_{2,p_2}\}$.
- $i = 3$. Set $E_3 = E_2 \cup \{e_3\}$; $B_3 = B_2 \cup \{b_{3,p} \mid p \in t_3\bullet = \{p_4\}\} = \{b_{3,p_4}\}$; $G_3 = G_2 \cup \{(e_3, b_{3,p}) \mid b_{3,p} \in B_3\} \cup \{(b_{j,p}, e_3) \mid b_{j,p} \in C_2, \ p \in \bullet t_3 = \{p_2\}\} = G_3 \cup \{(e_3, b_{3,p_4}), (b_{2,p_2}, e_3)\}$; $C_3 = (C_2 \setminus \bullet e_3) \cup e_3\bullet = \{b_{2,p_3}, b_{3,p_4}\}$.
- $i = 4$. Set $E_4 = E_3 \cup \{e_4\}$; $B_4 = B_3 \cup \{b_{4,p} \mid b_{j,p} \in C_3\} = \{b_{4,p_3}, b_{4,p_4}\}$; $G_4 = G_3 \cup \{(e_4, b_{4,p}) | \, b_{4,p} \in B_4\} \cup \{(b_{j,p}, e_4) \mid b_{j,p} \in C_3\} = G_3 \cup$

$\{(e_4, b_{4,p_3}), (e_4, b_{4,p_4}), (b_{2,p_3}, e_4), (b_{3,p_4}, e_4)\}$; $C_4 = (C_3 \setminus {\bullet e_4}) \cup e_4{\bullet} = \{b_{4,p_3}, b_{4,p_4}\}$.

– $i = 5$. Set $E_5 = E_4 \cup \{e_5\}$; $B_5 = B_4 \cup \{b_{5,p} \mid p \in t_2{\bullet} = \{p_1, p_2\}\} = \{b_{5,p_1}, b_{5,p_2}\}$; $G_5 = G_4 \cup \{(e_5, b_{5,p}) \mid b_{5,p} \in B_5\} \cup \{(b_{j,p}, e_5) \mid b_{j,p} \in C_4,\ p \in {\bullet t_2} = \{p_3, p_4\}\} = G_4 \cup \{(e_5, b_{5,p_1}), (e_5, b_{5,p_2}), (b_{4,p_3}, e_5), (b_{4,p_4}, e_5)\}$; $C_5 = (C_4 \setminus {\bullet e_5}) \cup e_5{\bullet} = \{b_{5,p_1}, b_{5,p_2}\}$.

Determine the following: $\varphi^*(e_i) = \overline{t_i}$, for all $e_i \in E_5$, $\varphi^*(b_{j,p}) = p$, for all $b_{j,p} \in B_5$, $l^*(e_i) = \begin{cases} tick, if\ \varphi^*(e_i) = \surd, \\ L(\varphi^*(e_i)), otherwise \end{cases}$, for all $e_i \in E_5$. Identify $\pi^* = (\widetilde{TN}^* = (E_5, B_5, G_5, C_5, l^*), \varphi^*)$ and $\rho^* = e_1 \dots e_n$. Notice that $\widetilde{TN}^*$ and $\widetilde{TN}$ from Example 6 are equal up to renaming their conditions. □

The following lemmas demonstrate important properties of the constructions from Definition 9.

Lemma 4.

a) TN^* is a casual net;

b) $\rho^* = e_1 \dots e_n$ is a linearization of TN^*;

c) $C_i = Cut(E_i)$, for all $0 \le i \le n$.

Proof. See Appendix. □

Lemma 5. The mapping φ^* is a homomorphism from TN^* to $\mathcal{TN}$.

Proof. See Appendix. □

We are now ready to establish that FS is a surjective mapping.

Theorem 4. Given a run σ of a time Petri net $\mathcal{TN}$, there exists an admissible time process $\pi^* = (TN^*, \varphi^*)$ of $\mathcal{TN}$ and a linearization $\rho^* = e_1 \dots e_n$ of TN^* such that $\sigma = FS(\rho^*) = \varphi^*(e_1) \dots \varphi^*(e_n)$.

Proof. Consider the construction of π^* from Definition 9. According to Lemma 4(a) and Lemma 5, π^* is a time process of $\mathcal{TN}$. Take an arbitrary $1 \le i \le n$ such that $\varphi^*(e_i) \in T$. Then, $\mathbf{Clock}({\bullet e_i}, \varphi^*(e_i)) = I_{i-1}(\varphi^*(e_i))$, by Lemma 3(c). Due to σ being a run of $\mathcal{TN}$, we obtain that $Eft(\varphi^*(e_i)) \le \mathbf{Clock}({\bullet e_i}, \varphi^*(e_i)) \le Lft(\varphi^*(e_i))$. Hence, π^* is an admissible time process of $\mathcal{TN}$. Thanks to Lemma 4(b), $\rho^* = e_1 \dots e_n$ is a linearization of TN^*. By the construction of φ^*, we get that $FS(\rho^*) = \sigma$. □

The following theorem shows that FS is an injection, i.e. the constructed in Definition 9 time process $\pi^* = (TN^*, \varphi^*)$ is unique up to isomorphism.

Theorem 5. Let σ be a run of $\mathcal{TN}$. The time process $\pi = (TN, \varphi)$ of $\mathcal{TN}$ with linearization $\rho = e_1 \dots e_n$ of TN, such that $\sigma = FS(\rho) = \varphi(e_1) \dots \varphi(e_n)$ is unique up to isomorphism.

Proof. Take arbitrary time process $\pi' = (TN' = (B', E', G', l'), \varphi')$ of $\mathcal{TN}$ and linearization $\rho' = e'_1 \dots e'_n$ $(n \ge 0)$ of TN' such that $FS(\rho') = \sigma$. Then, $E' = \{e'_1, \dots, e'_n\}$, by the definition of the linearization.

Moreover, $B' = {\bullet TN'} \oplus e'_1{\bullet} \oplus \dots \oplus e'_n{\bullet}$, due to TN' being an acyclic net. Set $B'_0 = {\bullet TN'}$ and $B'_i = B'_{i-1} \cup e_i'{\bullet}$, for $1 \le i \le n$. By Definition 7, the restriction of φ' to ${\bullet TN'}$ is a bijection between ${\bullet TN'}$and M_0. Then, w.l.o.g. assume $B'_0 = \{b'_{0,p} | p \in M_0\}$, with $\varphi'(b'_{0,p}) = p$. Take an arbitrary $1 \le i \le n$. Suppose $\varphi'(e'_i) \in T$. Then, due to the restriction of φ' to $e'_i{\bullet}$ being a bijection between $e'_i{\bullet}$ and $\varphi'(e'_i){\bullet}$, w.l.o.g. assume $e'_i{\bullet} = \{b'_{i,p} | p \in \varphi'(e'_i){\bullet}\}$. If $\varphi'(e'_i) = \surd$, then $e'_i{\bullet}$ is a cut, i.e. $e'_i{\bullet} \ne \emptyset$, thanks to Definition 7. In addition, we have that $e'_i{\bullet} \subseteq Cut(E'_i)$, by Lemma 2(a), and $Cut(E'_i)$ is cut, by Lemma 2(c), i.e. $e_i'{\bullet} = Cut(E'_i)$. Thanks to Definition 2, it holds that $M_i = M_{i-1}$. Hence, the restriction of φ' to $e_i'{\bullet}$ is a bijection between $e_i'{\bullet}$ and $\varphi'(Cut(E'_{i-1}))$, according

to Lemma 3(a). W.l.o.g. suppose $e_i'\bullet = \{b'_{i,p} \mid p \in \varphi'(Cut(E'_{i-1}))\}$. Thus, for all $1 \le i \le n$, we obtain the following:

– $B'_i = B'_{i-1} \cup \begin{cases} e_i'\bullet = \{b'_{i,p} \mid p \in \varphi'(Cut(E'_{i-1}))\}, & if\ \varphi'(e_i') = \surd, \\ e_i'\bullet = \{b'_{i,p} \mid p \in \varphi'(e_i')\bullet\}, & otherwise \end{cases}$;
– $\varphi'(b'_{i,p}) = p$, for all $b'_{i,p} \in B'_i$.

Compare the time process π' of $\mathcal{TN}$ and the time process $\pi^* = ((B, E, G, l^*), \varphi^*)$ of $\mathcal{TN}$ of TN^* (from Definition 9). Clearly, E' and E have the same cardinality. Due to $FS(\rho') = FS(\rho^*)$, we obtain that $\varphi'(e_i') = \varphi^*(e_i)$, for all $1 \le i \le n$. According Lemma 3(a) and Lemma 4(c), it holds that $\varphi'(Cut(E_{i-1})) = M_{i-1} = \varphi^*(Cut(E_{i-1}) = C_{i-1})$, for all $1 \le i \le n$. Hence, B_i and B_i' have the same cardinality, for all $0 \le i \le n$.

Thanks to the definitions of E' (E^*) and B' (B^*), we can construct a bijective mapping $\gamma : (E' \cup B') \to (E \cup B)$, with $\gamma(e_i') = e_i$, for all $e_i' \in E'$, and $\gamma(b'_{j,p}) = b_{j,p}$, for all $b'_{j,p} \in B'$ such that $\gamma(B') = B$ and $\gamma(E') = E$. Clearly, $\varphi'(x) = \varphi^*(\gamma(x))$, for all $x \in B' \cup E'$, and hence, $l'(e_i') = l^*(\gamma(e_i'))$, for all $e_i' \in E'$. It remains to show that G' is isomorphic to G. Take an arbitrary $1 \le i \le n$. Due to the definitions of B_i and B'_i, we have that $(e_i', b'_{j,p}) \in G' \Leftrightarrow b'_{j,p} \in B'_i \Leftrightarrow b_{j,p} \in B_i \Leftrightarrow (e_i, b_{j,p}) \in G$. Check that $(b'_{j,p}, e_i') \in G' \Leftrightarrow (b_{j,p}, e_i) \in G$.

Claim. $b_{j,p} \in Cut(E_{i-1}) \Leftrightarrow b'_{j,p} \in Cut(E'_{i-1})$.

Proof. We prove the case with $b_{j,p} \in Cut(E_{i-1})$ (the case with $b'_{j,p} \in Cut(E'_{i-1})$ is symmetric). Then, there exists $b'_{j',p} \in Cut(E'_{i-1})$, according to Lemma 3(a). Suppose a contrary, i.e. $j' \ne j$. W.l.o.g. assume $j < j' < i$. As $\gamma : B' \to B$ is bijection, there exists $b_{j',p} = \gamma(b'_{j',p})$. Due to Definition 9, we have that $b_{j,p} \in \bullet TN$, if $j = 0$, or $b_{j,p} \in e_j\bullet \subseteq E_j\bullet$, if $j > 0$, and $b_{j',p} \in e_{j'}\bullet$, because $j' > j$. Then, $b_{j',p} \in Cut(E_{j'})$, by Lemma 2(a). However, $b_{j,p} \notin Cut(E_{j'})$, thanks to Lemma 3(a). Since $E_j\bullet \subseteq E_{j'}\bullet$, we get that $b_{j,p} \in \bullet TN \cup E_{j'}\bullet$. Hence, $b_{j,p} \in \bullet E_{j'}$, due to the definition of $Cut(E_{j'})$. This implies that $b_{j,p} \in \bullet E_{i-1}$, contradicting $b_{j,p} \in Cut(E_{i-1})$. □

According to Lemma 2(b), we have $\bullet e_i' \subseteq Cut(E'_{i-1})$. Assume that $\varphi'(e_i') = \surd$. Then, $\bullet e_i'$ is a cut, i.e $\bullet e_i' \ne \emptyset$, due to Definition 7. Moreover, we have that $Cut(E'_{i-1})$ is cut, by Lemma 2(c), i.e. $\bullet e_i' = Cut(E'_{i-1})$. Therefore, $(b'_{j,p}, e_i') \in G' \Leftrightarrow b'_{j,p} \in Cut(E'_{i-1}) \Leftrightarrow b_{j,p} \in Cut(E_{i-1}) \Leftrightarrow (b_{j,p}, e_i) \in G$, thanks to Claim. Assume $\varphi'(e_i') \in T$. Then, it holds that $\bullet e_i' = \{b'_{j,p} \mid p \in \bullet\varphi'(e_i') \wedge b'_{j,p} \in Cut(E'_{i-1})\}$, due to the restriction of φ' to $\bullet e_i'$ being a bijection between $\bullet e_i'$ and $\bullet\varphi'(e_i')$. By virtue of Claim, we get that $(b'_{j,p}, e_i') \in G' \Leftrightarrow p \in \bullet\varphi'(e_i') \wedge b'_{j,p} \in Cut(E'_{i-1}) \Leftrightarrow p \in \bullet\varphi^*(e_i) \wedge b_{j,p} \in Cut(E_{i-1}) \Leftrightarrow (b_{j,p}, e_i) \in G$.

Therefore, we obtain that $\gamma: \pi' \simeq \pi^*$. □

Thus, we have demonstrated that FS is a bijective mapping between linearizations of time processes and runs from the initial state, in the context of the TPN $\mathcal{TN}$.

6. Conclusion

In this paper, we have introduced and studied partial order semantics for TPNs with weak time elapsing and intermediate memory policies. First, we have developed a state space discretization technique for the TPN, i.e. we have shown that any of its run with real-value time elapsings can be represented as that with the same untimed part and with only unit time elapsings. This allows us to transform time elapsings into the structure of a causal net with tick-events. Second, partial order semantics of the TPN has been proposed in the terms of time causal processes which consist of causal nets and their homomorphism into the TPN. Partial order semantics is useful for taking into account the processes' timing behavior in addition to their degrees of relative concurrency. Also, in

the context of the TPN, a bijective mapping has been proved to exist between interleaving runs and computations (linearizations) of time causal processes, demonstrating that the partial order semantics is correct w.r.t. the interleaving that.

As for future work, we plan to extend the results obtained to atomic memory and back in time policies. As well, we believe that partial order semantics developed here allows us to elaborate and investigate behavioral equivalences of TPNs with weak semantics, in interleaving – partial order dichotomy.

References

1. M. Boyer and O. H. Roux. On the compared expressiveness of arc, place and transition time Petri nets. Fundamenta Informaticae, vol. 88, no. 3, 2008, pp. 225-249.
2. B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux. Comparison of different semantics for time Petri nets. Lecture Notes in Computer Science, vol. 3707, 2005, pp. 293-307.
3. P.-A. Reynier and A. Sangnier. Weak time Petri nets strike back! Lecture Notes in Computer Science, vol. 5710, 2009, pp. 557-571.
4. L. Popova-Zeugmann. Time Petri nets. Springer, 2013, pp. 31-137.
5. T. Aura and J. Lilius. A causal semantics for time Petri nets. Theoretical Computer Science, vol. 243, no. 1-2, 2000, pp. 409-447.
6. H. Fleischhack and C. Stehno. Computing a finite prefix of a time Petri net. Lecture Notes in Computer Science, vol. 2360, 2002, pp. 163-181
7. T. Chatain and C. Jard. Back in time Petri nets. Lecture Notes in Computer Science, vol. 8053, 2013, pp. 91-105.

Appendix

Proof of Lemma 1. Let β is a solution of B_ω. We shall prove, that $(M_0, I_0) \overset{[\omega_i]_\beta}{\longrightarrow} (M_{\omega_i}, [I'_{\omega_i}]_\beta)$, for all $0 \le i \le k$, by induction on i.

$i = 0$. Due to Definition 4, we get: $\omega_0 = x_0$; $M_{\omega_0} = M_0$; $\forall t \in En(M_{\omega_0}) : I'_{\omega_0}(t) = x_0$. Hence, $(M_0, I_0) \overset{[\omega_0]_\beta}{\longrightarrow} (M_{\omega_0}, [I'_{\omega_0}]_\beta)$, thanks to Definition 2.

$i > 0$. By the induction hypothesis, we have that $(M_0, I_0) \overset{[\omega_{i-1}]_\beta}{\longrightarrow} (M_{\omega_{i-1}}, [I'_{\omega_{i-1}}]_\beta)$. Due to Definition 4, it holds:

- $\omega_i = \omega_{i-1} t_i x_i$;
- $M_{\omega_i} := (M_{\omega_{i-1}} \setminus {\bullet t_i}) \cup t_i{\bullet}$ (i.e. $t_i \in En(M_{\omega_{i-1}})$);
- $Eft(t_i) \le I'_{\omega_{i-1}}(t_i) \le Lft(t_i)$ in B_ω;
- $\forall t \in En(M_{\omega_i}), I'_{\omega_i}(t) - x_i = \begin{cases} 0, & \text{if } \uparrow enabled(t, M_{\omega_{i-1}}, t_i) \\ I'_{\omega_{i-1}}(t), & \text{otherwise} \end{cases}$.

Then, $Eft(t_i) \le [I'_{\omega_{i-1}}(t)]_\beta \le Lft(t_i)$, because β is a solution of B_ω. Therefore, t_i can fire from the state $(M_{\omega_{i-1}}, [I'_{\omega_{i-1}}]_\beta)$. By Definition 2, we have that $(M_0, I_0) \overset{[\omega_{i-1}]_\beta}{\longrightarrow} (M_{\omega_{i-1}}, [I'_{\omega_{i-1}}]_\beta) \overset{t_i}{\longrightarrow} (M_{\omega_i}, [I'_{\omega_i}]_\beta - \beta(x_i)) \overset{\beta(x_i)}{\longrightarrow} (M_{\omega_i}, [I'_{\omega_i}]_\beta)$. Hence, it is true that $(M_0, I_0) \overset{[\omega]_\beta}{\longrightarrow} (M_\omega, [I'_\omega]_\beta)$. Moreover, for all $1 \le i \le k$, we have $[I'_{\omega_{i-1}}(t_i)]_\beta = I'_{i-1}(t_i)$. □

Proof of Proposition 1.

Let $\omega = x_0 t_1 x_1 \dots t_k x_k$; β_i be the functions from Definition 5, with $0 \le i \le k$; $\beta_\omega = \beta_k$; and $\mathcal{B}_\omega$ be the set of the variable parts of the inequalities from B_ω. In order to show that the assignment β_ω is a solution of ω, we consider an important property of the mappings β_i, for all $0 \le i \le k$.

Claim. For all $g \in \mathcal{B}_\omega$ and $0 \le i \le k$, it holds that $\lceil [g]_{\beta_i} \rceil \ge \lfloor [g]_{\beta_0} \rfloor$ and $\lfloor [g]_{\beta_i} \rfloor \le \lceil [g]_{\beta_0} \rceil$.

Proof. We shall prove by induction on i.

$i = 0$. Obvious.

$i > 0$. Take an arbitrary $g \in \mathcal{B}_\omega$. By the induction hypothesis, we have that $\lceil [g]_{\beta_{i-1}} \rceil \ge \lfloor [g]_{\beta_0} \rfloor$ and $\lfloor [g]_{\beta_{i-1}} \rfloor \le \lceil [g]_{\beta_0} \rceil$. Let $\underline{\beta_i}$ be the function from Definition 5.

Assume that does not exist $h \in \mathcal{B}_\omega \; s.t. \lceil [h]_{\underline{\beta_i}} \rceil < \lfloor [h]_{\beta_0} \rfloor$. Then, $[g]_{\beta_i} = [g]_{\underline{\beta_i}}$ and $[g]_{\beta_i} \le [g]_{\beta_{i-1}}$, due to Definition 5. Then $\lceil [g]_{\beta_i} \rceil = \lceil [g]_{\underline{\beta_i}} \rceil \ge \lfloor [g]_{\beta_0} \rfloor$ and $\lfloor [g]_{\beta_i} \rfloor \le \lfloor [g]_{\beta_{i-1}} \rfloor \le \lceil [g]_{\beta_0} \rceil$.

Assume that there is exists $h \in \mathcal{B}_\omega \; s.t. \lceil [h]_{\underline{\beta_i}} \rceil < \lfloor [h]_{\beta_0} \rfloor$. Then, $[g]_{\beta_i} \ge [g]_{\beta_{i-1}}$, by Definition 5. Therefore, $\lceil [g]_{\beta_i} \rceil \ge \lceil [g]_{\beta_{i-1}} \rceil \ge \lfloor [g]_{\beta_0} \rfloor$. Suppose a contrary, i.e. $\lfloor [g]_{\beta_i} \rfloor > \lceil [g]_{\beta_0} \rceil$. Then, $[g]_{\beta_i} \ge \lfloor [g]_{\beta_i} \rfloor \ge \lceil [g]_{\beta_0} \rceil + 1$. According to Definition 5, x_{k-i} appears in g and h and, moreover, $[h]_{\beta_i} = [h]_{\beta_{i-1}} - \beta_{i-1}(x_{k-i}) + \beta_i(x_{k-i}) = [h]_{\beta_{i-1}} - \beta_{i-1}(x_{k-i}) + \lceil \beta_{i-1}(x_{k-i}) \rceil \le [h]_{\beta_{i-1}} - \beta_{i-1}(x_{k-i}) + \lfloor \beta_{i-1}(x_{k-i}) \rfloor + 1 = [h]_{\underline{\beta_i}} + 1$. As $[h]_{\underline{\beta_i}} + 1 \le \lceil [h]_{\underline{\beta_i}} \rceil + 1 \le \lfloor [h]_{\beta_0} \rfloor$, we have $[h]_{\beta_i} \le \lfloor [h]_{\beta_0} \rfloor$. Let $S : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_{\ge 0}$ be such that $S(a,b) = \sum_{j=a}^{b} x_j$, if $a < b$, and $S(a,b) = 0$, otherwise. Due to Definition 4, it holds that $h = S(m,n)$ and $g = S(m',n')$, where $0 \le m, m' \le k - i \le n, n' \le k$. By the construction of β_k, we have that $[S(a,b)]_{\beta_c} = [S(a,b)]_{\beta_0}$, if $0 \le b < k - c \le k$; and $[S(a,b)]_{\beta_c} = [S(a,b)]_{\beta_k}$, if $0 \le k - c \le a \le k$. Therefore, we obtain the following:

(1) $[S(k-i,n)]_{\beta_i} \le \lfloor [S(k-i,n)]_{\beta_0} \rfloor$, due to $[S(m,k-i-1)]_{\beta_i} = [S(m,k-i-1)]_{\beta_0}$ and $[h]_{\beta_i} \le \lceil [h]_{\underline{\beta_i}} \rceil < \lfloor [h]_{\beta_0} \rfloor$;

(2) $[S(k-i,n')]_{\beta_i} \ge \lceil [S(k-i,n')]_{\beta_0} \rceil + 1$, due to $[S(m',k-i-1)]_{\beta_i} = [S(m',k-i-1)]_{\beta_0}$ and $[g]_{\beta_i} \ge \lceil [g]_{\beta_0} \rceil + 1$;

(3) $[S(m',n)]_{\beta_{k-n-1}} = [S(m',n)]_{\beta_0}$;

(4) $[S(m,n')]_{\beta_{k-n'-1}} = [S(m,n')]_{\beta_0}$;

(5) $[S(n+1,n')]_{\beta_{k-n-1}} = [S(n+1,n')]_{\beta_i} = [S(n+1,n')]_{\beta_k}$;

(6) $[S(n'+1,n)]_{\beta_{k-n'-1}} = [S(n'+1,n)]_{\beta_i} = [S(n'+1,n)]_{\beta_k}$.

Three cases are admissible:

- $n = n'$. Then, (1) contradicts (2).
- $n < n'$. Then, it holds that $[g]_{\beta_{k-n-1}} = [S(m',n)]_{\beta_{k-n-1}} + [S(n+1,n')]_{\beta_{k-n-1}} =^{(3)} [S(m',k-i-1)]_{\beta_0} + [S(k-i,n)]_{\beta_0} + [S(n+1,n')]_{\beta_{k-n-1}} \ge^{(1),(5)} [S(m',k-i-1)]_{\beta_0} + [S(k-i,n)]_{\beta_i} + [S(n+1,n')]_{\beta_i} \ge^{(2)} \lceil [g]_{\beta_0} \rceil + 1$. This contradicts the induction hypothesis, because $k - n \le i$.
- $n > n'$. Then, it holds that $[h]_{\beta_{k-n'-1}} = [S(m,n')]_{\beta_{k-n'-1}} + [S(n'+1,n)]_{\beta_{k-n'-1}} =^{(4)} [S(m,k-i-1)]_{\beta_0} + [S(k-i,n')]_{\beta_0} + [S(n'+1,n)]_{\beta_{k-n'-1}} \le^{(2),(6)} [S(m,k-i-1)]_{\beta_0} + [S(k-i,n')]_{\beta_i} - 1 + [S(n'+1,n)]_{\beta_i} \le^{(1)} \lfloor [h]_{\beta_0} \rfloor - 1$. We get a contradiction the induction hypothesis, because $k - n' \le i$. □

By Definition 4, x_k does not appear in B_ω. Then, β_0 is a solution of B_ω, according to Definition 5. Take an arbitrary g from $\mathcal{B}_\omega$. Then, it holds that $(a \le g \le b) \in B_\omega$, where $a, b \in \mathbb{Z}$. By Claim, we have $\lfloor [g]_{\beta_0} \rfloor \le [g]_{\beta_\omega} \le \lceil [g]_{\beta_0} \rceil$. Due to β_0 being a solution, we obtain $a \le [g]_{\beta_0} \le b$ and, moreover $a \le \lfloor [g]_{\beta_0} \rfloor, \lceil [g]_{\beta_0} \rceil \le b$, because $a, b \in \mathbb{N} \cup \infty$. Thus, it holds that $a \le \lfloor [g]_{\beta_0} \rfloor \le [g]_{\beta_\omega} \le \lceil [g]_{\beta_0} \rceil \le b$. Therefore, β_ω is a solution of B_ω. □

Proof of Lemma 2.

a) Take an arbitrary $1 \le k \le n$. By definitions, we have that $Cut(E_k) = (\bullet TN \cup E_k\bullet) \setminus \bullet E_k = (\bullet TN \cup E_{k-1}\bullet \cup e_k\bullet) \setminus (\bullet E_{k-1} \cup \bullet e_k)$. As TN is an acyclic net, we obtain that $\bullet e_k \cap e_k\bullet = \emptyset$ and $\bullet E_{k-1} \cap e_k\bullet = \emptyset$. Then, it holds that $Cut(E_k) = (((\bullet TN \cup E_{k-1}\bullet) \setminus \bullet E_{k-1}) \setminus \bullet e_k) \cup e_k\bullet = (Cut(E_{k-1}) \setminus \bullet e_k) \cup e_k\bullet$.

b) Take arbitrary $1 \le k \le n$. The case with $\bullet e_k = \emptyset$ is trivial. Suppose $b \in \bullet e_k$. By definition, we have that $Cut(E_{k-1}) = (\bullet TN \cup E_{k-1}\bullet) \setminus \bullet E_{k-1}$. Due to TN being a causal net, $b \notin \bullet e_i$, for all $1 \le i < k$. Hence, $b \notin \bullet E_{k-1}$. If $b \in \bullet TN$, then $b \in Cut(E_{k-1})$. Consider the case when $b \notin \bullet TN$. Then, there is e_i such that $b \in e_i\bullet$. Clearly, $e_i \prec e_k$. This implies that $i < k$, in the linearization ρ. Hence, $b \in E_{k-1}\bullet$ and $b \in Cut(E_{k-1})$.

c) As $Cut(E_0) = \bullet TN = \{b \in B \mid \bullet b = \emptyset\}$, we have $Cut(E_0)$ is a co-set. Suppose a contrary, i.e. there are $b, b' \in Cut(E_k)$, for some $1 \le k \le n$, such that $b \prec b'$. As ρ is a linearization, we have $bGe_i \dots e_j Gb'$, with $i \le j$. Due to $Cut(E_k) = (\bullet TN \cup E_k\bullet) \setminus \bullet E_k$, we get $b, b' \notin \bullet E_k$ and $b' \in E_k\bullet$. Since $\bullet b' = e_j$, it holds that $j \le k$, i.e. $i \le k$. This means that $b \in \bullet e_i \subseteq \bullet E_k$, contradicting $b \notin \bullet E_k$. Thus, $\neg(b \prec b')$.

We shall show that $Cut(E_k)$ is a cut, for all $0 \le k \le n$. Suppose a contrary, i.e. there exists $b \notin Cut(E_k)$, for some $0 \le k \le n$, such that $\neg(b \prec b')$ and $\neg(b' \prec b)$, for all $b' \in Cut(E_k)$. W.l.o.g. assume $b \in Cut(E_i)$, for some $0 \le i \ne k \le n$. Thanks to item a), $Cut(E_j) = (Cut(E_{j-1}) \setminus \bullet e_j) \cup e_j\bullet$, for all $1 \le j \le n$. If $i < k$, then we get that $bGe_l \dots e_m Gb'$, for some $b' \in Cut(E_k)$ and $i < l \le m \le k$, i.e. $b \prec b'$, because TN is a causal net. If $i > k$, then we have that $b'Ge_l \dots e_m Gb$, for some $b' \in Cut(E_k)$ and $k < l \le m \le i$, i.e. $b' \prec b$, again because TN is a causal net. Thus, $Cut(E_k)$ is a cut, for all $0 \le k \le n$. □

Proof of Lemma 3.

a), b) We shall verify the items by induction on $0 \le i \le n$.

$i = 0$. By definitions, it holds that $Cut(E_0) = \bullet TN$.

a) The restriction of φ to $Cut(E_0)$ is a bijection between $Cut(E_0)$ and M_0, due to Definition 7.

b) As $Age(b) = 0$, for all $b \in \bullet TN$, $\mathbf{Clock}(Cut(E_0), t) = 0 = I_0(t)$, for all $t \in En(M_0)$, thanks to $\mathcal{TN}$ being T-restricted.

$i > 0$. By the induction hypothesis, the items hold for $i-1$. We now check them for i. Two cases are admissible.

Case 1: $\varphi(e_i) = \surd$. Then, it holds that $M_{i-1} = M_i$, by Definition 2. According to Definition 7, we have that $\varphi(e_i\bullet) = \varphi(\bullet e_i)$ and $\bullet e_i, e_i\bullet$ are cuts, i.e. $\bullet e_i \ne \emptyset, e_i\bullet \ne \emptyset$. In addition, we have that $\bullet e_i \subseteq Cut(E_{i-1})$, $e_i\bullet \subseteq Cut(E_i)$, and $Cut(E_{i-1})$, $Cut(E_i)$ are cuts, due to Lemma 2. Hence, we get that $\bullet e_i = Cut(E_{i-1})$ and $Cut(E_i) = e_i\bullet$, and, moreover, $\varphi(Cut(E_i)) = \varphi(Cut(E_{i-1})) = M_{i-1} = M_i$.

a) As the restriction of φ to $e_i\bullet$ is an injection, by Definition 7, the restriction of φ to $Cut(E_i)$ is a bijection between $Cut(E_i)$ and M_i.

b) Take an arbitrary $t \in En(M_i)$. According to Definition 2, we have that $t \in En(M_{i-1})$ and $I_{i-1}(t) + \varphi(e_i) = I_i(t)$. By definition, $Age(b) = Age(b') + \varphi(e_i)$, with $b' \in Cut(E_{i-1})$ and $\varphi(b) = \varphi(b')$, for all $b \in e_i\bullet \in Cut(E_i)$. Then, due to $\mathcal{TN}$ being T-restricted, we obtain that $\mathbf{Clock}(Cut(E_i), t) = min(\{Age(b) | \varphi(b) \in \bullet t, b \in Cut(E_i)\}) = min(\{Age(b) + \varphi(e_i) | \varphi(b) \in \bullet t, b \in Cut(E_{i-1})\}) = \mathbf{Clock}(Cut(E_{i-1}), t) + \varphi(e_i) = I_{i-1}(t) + \varphi(e_i) = I_i(t)$.

Case 2: $\varphi(e_i) \in T$. Then, $M_i = (M_{i-1} \setminus \bullet\varphi(e_i)) \cup \varphi(e_i)\bullet$, according to Definition 2. Due to Definition 7, the restrictions of φ to $\bullet e_i$ ($e_i\bullet$) are bijections between $\bullet e_i$ ($e_i\bullet$) and $\bullet\varphi(e_i)$ ($\varphi(e_i)\bullet$).

a) By the inductive hypothesis, the restriction of φ to $Cut(E_{i-1})$ is a bijection between $Cut(E_{i-1})$ and M_{i-1}. Then, due to Definition 2, we get that $M_i = (\varphi(Cut(E_{i-1})) \setminus \bullet\varphi(e_i)) \cup \varphi(e_i)\bullet = \varphi((Cut(E_{i-1}) \setminus \bullet e_i) \cup e_i\bullet) = \varphi(Cut(E_i))$, using Lemma 2(a). Since $\mathcal{TN}$ is contact-free, we obtain that $\varphi(Cut(E_{i-1}) \setminus \bullet e_i) \cap \varphi(e_i\bullet) = \emptyset$. Therefore, the restriction of φ to $Cut(E_i)$ is a bijection between $Cut(E_i)$ and M_i.

b) Take an arbitrary $t \in En(M_i)$. Assume that $\uparrow enabled(t, M_{i-1}, \varphi(e_i))$ is true. Then, we have that $t \notin En(M_{i-1} \setminus \bullet\varphi(e_i))$ or $t = \varphi(e_i)$, by Definition 2. If $t = \varphi(e_i)$, then $t \in En(M_{i-1})$ and $t \notin En(M_{i-1} \setminus \bullet t) = En(M_{i-1} \setminus \bullet\varphi(e_i))$. So, $t \notin En(M_{i-1} \setminus \bullet\varphi(e_i))$. Due to Definition 2, it is true that $En(M_i) = En((M_{i-1} \setminus \bullet\varphi(e_i)) \cup \varphi(e_i\bullet))$. Thanks to $\mathcal{TN}$ is T-restricted, we get $\bullet t \neq \emptyset$ and $\varphi(e_i\bullet) \neq \emptyset$. Since $t \in En(M_i)$ and $t \notin En(M_{i-1} \setminus \bullet\varphi(e_i))$, we have that $\bullet t \cap \varphi(e_i\bullet) \neq \emptyset$. According to Lemma 2(a), it holds that $e_i\bullet \subseteq Cut(E_i)$. Hence, there is $b \in Cut(E_i)$ such that $\varphi(b) \in \bullet t$ and $Age(b) = 0$. Therefore, due to $\mathcal{TN}$ being T-restricted, it is true that $\mathbf{Clock}(Cut(E_i), t) = (\min\{Age(b) | \varphi(b) \in \bullet t, b \in Cut(E_i)\}) = 0$.
Thus, $\mathbf{Clock}(Cut(E_i), t) = I_i(t)$, due to Definition 2.
Suppose that $\uparrow enabled(t, M_{i-1}, \varphi(e_i))$ is false. Then, we get that $t \in En(M_{i-1} \setminus \bullet\varphi(e_i))$ and $t \neq \varphi(e_i)$, by Definition 2. Hence, $\bullet\varphi(e_i) \cap \bullet t = \emptyset$, i.e. $\varphi(\bullet e_i) \cap \bullet t = \emptyset$. As $\mathcal{TN}$ is contact-free, it holds that $(M_{i-1} \setminus \bullet\varphi(e_i)) \cap \varphi(e_i)\bullet = \emptyset$. This means that, $\varphi(e_i)\bullet \cap \bullet t = \emptyset$, i.e. $\varphi(e_i\bullet) \cap \bullet t = \emptyset$,. Therefore, if $\varphi(b) \in \bullet t$, then $b \notin e_i\bullet$ and $b \notin \bullet e_i$. By the induction hypothesis, we have that $I_{i-1}(t) = \mathbf{Clock}(Cut(E_{i-1}), t) = min(\{Age(b) | \varphi(b) \in \bullet t, b \in Cut(E_{i-1})\}) = min(\{Age(b) | \varphi(b) \in \bullet t, b \in (Cut(E_{i-1}) \setminus \bullet e_i) \cup e_i\bullet\})$. Thanks to Lemma 2(a), we obtain that $I_{i-1}(t) = \mathbf{Clock}(Cut(E_i), t)$. Thus, it holds that $I_i(t) = I_{i-1}(t) = \mathbf{Clock}(Cut(E_i), t)$, due to Definition 2.

c) Assume that $i < n$ and $\varphi(e_{i+1}) \in En(M_i)$. Then, $\varphi(e_{i+1}) \in En(\varphi(Cut(E_i)))$, due to item a). By definition, due to $\mathcal{TN}$ being T-restricted, we have that $\mathbf{Clock}(Cut(E_i), \varphi(e_{i+1})) = min(\{Age(b) | \varphi(b) \in \bullet\varphi(e_{i+1}), b \in Cut(E_i)\})$. Take an arbitrary $b \in Cut(E_i)$ such that $\varphi(b) \in \bullet\varphi(e_{i+1})$. Thanks to the definition of a homomorphism, it holds that $\varphi(b) \in \varphi(\bullet e_{i+1})$. Hence, $b \in \bullet e_{i+1}$, due to item a). By virtue of Lemma 2(b), $\bullet e_{i+1} \subseteq Cut(E_i)$. This implies that $\mathbf{Clock}(Cut(E_i), \varphi(e_{i+1})) = \mathbf{Clock}(\bullet e_{i+1}, \varphi(e_{i+1}))$. Therefore, $\mathbf{Clock}(\bullet e_{i+1}, \varphi(e_{i+1})) = I_i(\varphi(e_{i+1}))$, due to item (b). □

Proof of Lemma 4.

a) By the construction of TN^*, we have the following. First, B and E are finite sets. Second, $G \subseteq (B \times E) \cup (E \times B)$ is a flow relation such that $e_j G_i b_{j,p} G_i e_i$, i.e. $j < i$. Hence, TN^* is acyclic. Third, $l^*: E \to Act \cup \{tick\}$ is a labeling function. By the construction of G, we obtain that $\bullet b_{0,p} = \emptyset$ and $\bullet b_{i,p} = \{e_i\}$, for all $1 \leq i \leq n$. Therefore, $|\bullet b_{i,p}| \leq 1$, for all $0 \leq i \leq n$. Suppose a contrary, i.e. $|b_{j,p}\bullet| > 1$, for some $b_{j,p} \in B$. Then, there exists $i \neq i'$ such that $\{b_{j,p}\} \in \bullet e_i$ and $\{b_{j,p}\} \in \bullet e_{i'}$. Hence, by the construction of G, we get that $j < i$, $b_{j,p} \in C_{i-1}$ and $j < i'$,

$b_{j,p} \in C_{i'-1}$. W.l.o.g. assume $i < i'$. As $C_l = (C_{l-1} \backslash {\bullet e_l}) \cup e_l{\bullet}$, for all $1 \le l \le n$, there exists $i \le k \le i' - 1$ such that $b_{j,p} \in e_k{\bullet}$. According to the construction of G_k, $j = k$, contradicting $j < k$.

b) Due to Definition 9, every event of TN^* appears in the sequence $\rho^* = e_1 \dots e_n$ exactly once. By the construction of G, it holds that $e_i \prec e_j$ implies $i < j$.

c) As $C_0 = {\bullet TN^*}$, we get $C_0 = Cut(E_0)$. Thanks to items a), b) and Lemma 2(a), we obtain $C_i = Cut(E_i)$, for $0 \le i \le n$. □

Proof of Lemma 5.

Claim. The restriction of φ^* to C_i is a bijection between C_i and M_i, for all $0 \le i \le n$.

Proof. We prove by induction on $0 \le i \le n$.

$i = 0$. Then, $C_0 = B_0 = \{b_{0,p} | \ p \in M_0\}$, i.e. the restriction of φ^* to C_0 is a bijection between C_0 and M_0.

$i > 0$. By the induction hypothesis, the restriction of φ^* to C_{i-1} is a bijection between C_{i-1} and M_{i-1}. Two cases are admissible.

Let $\varphi^*(e_i) = \overline{t_i} = \surd$. By Definition 2, we have $M_i = M_{i-1}$. Thanks to the construction of G_i and C_i, it holds that $C_i = (C_{i-1} \backslash {\bullet e_i}) \cup e_i{\bullet} = e_i{\bullet} = C_{i-1}$. Then, the restriction of φ^* to C_i is a bijection between C_i and M_i.

Let $\varphi^*(e_i) = \overline{t_i} \in T$. By Definition 2, we have that $M_i = (M_{i-1} \backslash {\bullet \varphi^*(e_i)}) \cup \varphi^*(e_i){\bullet}$. Take an arbitrary $p \in {\bullet \varphi^*(e_i)}$ (it exists because $\mathcal{TN}$ is T-restricted). Then, $p \in M_{i-1}$ and, moreover, there exists $b_{j,p} \in C_{i-1}$, due to the induction hypothesis. Thanks to the construction of G_i, we get that ${\bullet e_i} = \{b_{j,p} | \ p \in {\bullet \varphi^*(e_i)} \wedge b_{j,p} \in C_{i-1}\}$ and $e_i{\bullet} = \{b_{i,p} | \ p \in \varphi^*(e_i){\bullet}\}$. Hence, the restriction of φ^* to ${\bullet e_i}$ $(e_i{\bullet})$ is a bijection between ${\bullet e_i}$ $(e_i{\bullet})$ and ${\bullet \varphi^*(e_i)}$ $(\varphi^*(e_i){\bullet})$. Then, $\varphi^*(C_i) = \varphi^*((C_{i-1} \backslash {\bullet e_i}) \cup e_i{\bullet}) = (\varphi^*(C_{i-1}) \backslash \varphi^*({\bullet e_i})) \cup \varphi^*(e_i{\bullet}) = (M_{i-1} \backslash {\bullet \varphi^*(e_i)}) \cup \varphi^*(e_i){\bullet} = M_i$. Due to $\mathcal{TN}$ being contact-free, we obtain $\left(M_{i-1} \backslash {\bullet \varphi^*(e_i)}\right) \cup \varphi^*(e_i){\bullet} = \emptyset$.

Therefore, the restriction of φ^* to C_i is a bijection between C_i and M_i. □

By definition, we have that $\varphi^*(B) \subseteq P, \varphi^*(E) \subseteq (T \cup \{\surd\})$, and $l^*(e) = \begin{cases} tick, if \varphi^*(e) = \surd, \\ L\left(\varphi^*(e)\right), otherwise \end{cases}$, for all $e \in E$.

Take an arbitrary $1 \le i \le n$.

Assume $\varphi^*(e_i) \in T$. Due to Claim, the restriction of $\varphi^*$to ${\bullet e_i}$ $(e_i{\bullet})$ is a bijection between ${\bullet e_i}$ $(e_i{\bullet})$ and ${\bullet \varphi^*(e_i)}$ $(\varphi^*(e_i){\bullet})$.

Assume $\varphi^*(e_i) = \surd$. Thanks to the construction of G_i and C_i, we have that $C_{i-1} = {\bullet e_i} = e_i{\bullet} = C_i$. By Claim, the restriction of φ to ${\bullet e_i}$ and the restriction of φ to $e_i{\bullet}$ are injections. As $\mathcal{TN}$ is T-restricted, we obtain ${\bullet e_i} \ne \emptyset$ and $e_i{\bullet} \ne \emptyset$. According to Lemma 4(c), we have that ${\bullet e_i} = Cut(E_{i-1})$ and $e_i{\bullet} = Cut(E_i)$. Due to Lemma 2(c), ${\bullet e_i}$ and $e_i{\bullet}$ are cuts.

As ${\bullet TN^*} = C_0$, by construction, the restriction of φ^* to ${\bullet TN^*}$ is a bijection between ${\bullet TN^*}$ and M_0, due to Claim.

Thus, φ^* is a homomorphism from TN^* to $\mathcal{TN}$, by virtue of Lemma 4(a). □

Information about authors / Информация об авторах

Alexey Yurievich ZUBAREV, PhD student. Research interests: parallel computing, Petri nets.

Алексей Юрьевич ЗУБАРЕВ, аспирант. Научные интересы: параллельные вычисления, сети Петри.

Irina Bonaventurovna VIRBITSKAITE – Doctor of Physical and Mathematical Sciences, Professor, Head of the Laboratory of the Theory of Parallel Processes at IIS SB RAS, Professor at NSU. Research interests: theory of parallel processes; specification and verification of parallel real-time systems.

Ирина Бонавентуровна ВИРБИЦКАЙТЕ – доктор физико-математических наук, профессор, заведующая лабораторией теории параллельных процессов в ИСИ СО РАН, профессор НГУ. Научные интересы: теория параллельных процессов; спецификация и верификация параллельных систем реального времени.