ИСП Институт Системного Программирования Российской Академии наук

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

Труды Института Системного Программирования РАН

Proceedings of the Institute for System Programming of the RAS

Том 28, выпуск 6

Volume 28, issue 6

Москва 2016

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

Труды Института системного программирования РАН

Proceedings of the Institute for System Programming of the RAS

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

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

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

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

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

Труды ИСП РАН реферируются и/или индексируются в:

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

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

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

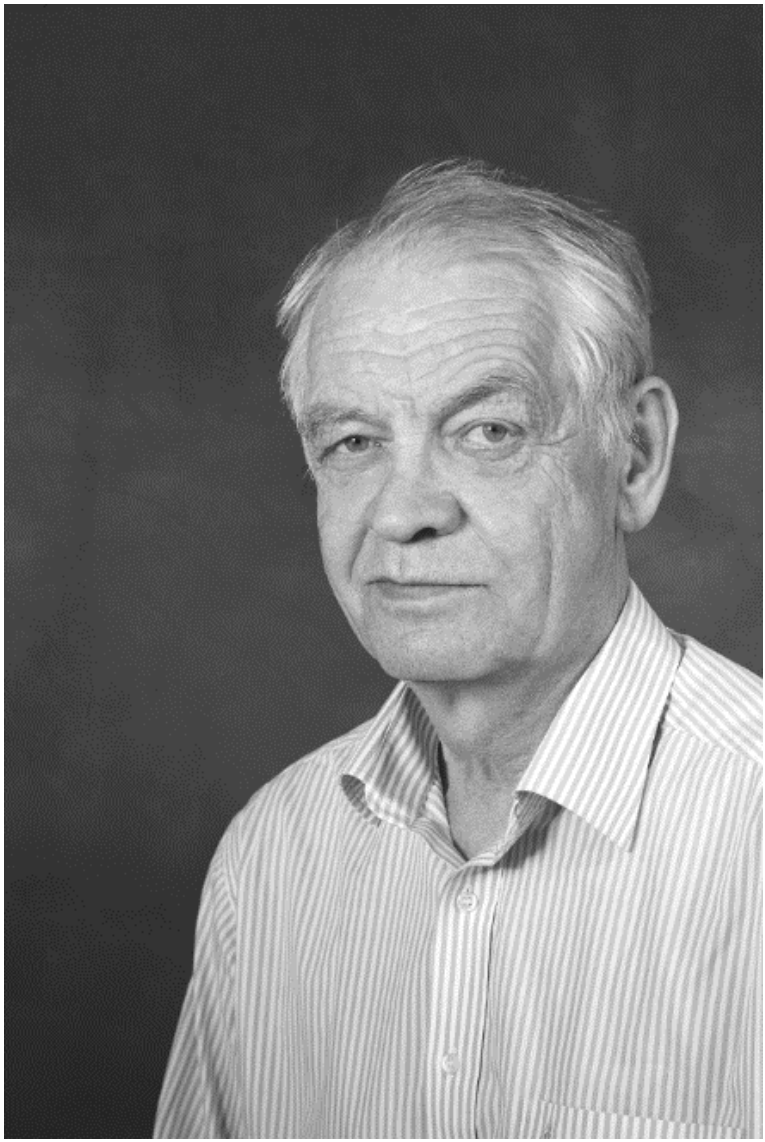
The journal is published one volume per year, six issues in each volume.

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

УДК004.45

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

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



Посвящается памяти академика

ВИКТОРА ПЕТРОВИЧА ИВАННИКОВА

(27.02.1940 - 27.11.2016)

Содержание

Table of Contents

Предисловие

Это последний выпуск Трудов ИСП РАН, выходящий при активном участии и под общей редакцией основателя ИСП РАН и его многолетнего директора Виктора Петровича Иванникова. Электронное издание Труды ИСП РАН было учреждено по инициативе Виктора Петровича. Благодаря его руководству международной редакционной коллегией издание смогло достичь зрелости и получить признание в России и за ее пределами. После кончины первого главного редактора Трудов ИСП РАН редакционная коллегия будет продолжать инициированную В.П. Иванниковым работу, чтобы сохранить и упрочить позиции издания.

В этом выпуске продолжается публикация статей, подготовленных по материалам докладов, которые были представлены на первой Открытой конференции ИСП РАН, проходившей в Москве в здании Президиума РАН 1-2 декабря 2016 г. Эта конференция также проводилась по замыслу академика В.П. Иванникова, который руководил подготовкой первой Открытой конференции и скоропостижно скончался за три дня до ее начала.

Открытая конференция ИСП РАН – это ежегодная конференция, где проводятся тематические мероприятия по тем направлениям IT-индустрии, в которых Институтом накоплен наиболее ценный опыт при выполнении фундаментальных исследований, разработок инновационных технологий, а также реализации совместно с партнерами конкретных проектов по внедрению разработанных технологий в промышленность. К участию в конференции приглашаются признанные эксперты отечественных и зарубежных научных и образовательных организаций, представители ведущих IT-компаний, а также исследователи, чьи работы прошли независимое профессиональное рецензирование.

В заключительном выпуске 28-го тома публикуются статьи, посвященные технологии анализа, моделирования и трансформации программ, технологии распределенных систем, анализу текстов на естественных языках и анализу социальных сетей.

Член-корреспондент РАН А.И. Аветисян

Foreword

This is the last issue of the Proceedings of ISP RAS released with active participation and general editorship of late Victor Petrovich Ivannikov, the founder and long-standing director of ISP RAS. The electronic edition of Proceedings of ISP RAS was established at the initiative of Victor Petrovich. Through his leadership of the international editorial board the edition has been able to reach maturity and to become recognized in Russia and abroad. After the death of the first chief editor of the Proceedings of ISP RAS Editorial Board will continue the work initiated by V.P. Ivannikov to preserve and strengthen the position of the edition.

In this issue, we continue to publish articles based on the reports, which were presented at the first Open conference of ISPRAS, held in Moscow in the building of the Presidium of the Russian Academy of Sciences 1-2 December 2016 This conference was also the intent of Academician V.P. Ivannikov who supervised the preparation of the first open conference, and died three days before its beginning.

Open Conference of ISPRAS is an annual conference, where themed events in the areas of IT-industry, in which the Institute has gained most valuable experience in carrying out fundamental research, development of innovative technologies, as well as the specific projects execution together with industrial partners on the business-implementation of the developed technologies. To participate in the conference are invited recognized experts of domestic and foreign scientific and educational institutions, representatives of the leading IT-companies, as well as researchers, whose works have been professionally reviewed.

In this final issue of the 28th volume, we publish articles on technologies for analysis, modeling and transformation of programs, distributed systems technology, analysis of texts in natural languages, and social network analysis.

Corresponding Member of RAS A.I. Avetisyan

О некоторых ограничениях полносистемного анализа помеченных данных ⋆

[1]*М.А. Климушенкова <maria.klimushenkova@ispras.ru>*
[2]*М.Г. Бакулин <bakulinm@ispras.ru>*
[2,3]*В.А. Падарян <vartan@ispras.ru>*
[1]*П.М. Довгалюк <pavel.dovgaluk@ispras.ru>*
[1]*Н.И. Фурсова <Natalia.Fursova@ispras.ru>*
[1]*И.А. Васильев <vasiliev@ispras.ru>*
[1]*Новгородский государственный университет имени Ярослава Мудрого*
173003, Россия, г. Великий Новгород, ул. Большая Санкт-Петербургская, д. 41
[2]*Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[3]*Московский государственный университет имени М.В. Ломоносова,*
119991 ГСП-1, Москва, Ленинские горы

Аннотация. Анализ помеченных данных неоднократно пытались применять для исследования безопасности бинарного кода, но все попытки наталкивались на ряд нерешенных вопросов. В данной работе рассматриваются ограничения анализа помеченных данных на уровне бинарного кода, когда он проводится в рамках всей системы. Предлагается подход, способный преодолеть такие ограничения, как высокие накладные расходы на анализ, разрыв в уровне абстракций бинарного и исходного кода и сложности переноса на другие процессорные архитектуры и ОС. Подход позволяет смягчить негативное влияние недостаточной и избыточной помеченности. В подходе используется полносистемный эмулятор, использующий бинарную трансляцию. Возможности анализа помеченных данных обеспечиваются тремя встроенными в эмулятор механизмами: детерминированным воспроизведением, плагинами инстроспекции ВМ и инструментированием промежуточного представления. Приводятся экспериментальные результаты, показывающие лучшую скорость работы в сравнении с аналогичными программными инструментами.

Ключевые слова: анализ помеченных данных; динамический анализ; QEMU.

DOI: 10.15514/ISPRAS-2016-28(6)-1

⋆ Работа поддержана грантом РФФИ № 16-29-09632

Для цитирования: Климушенкова М.А., Бакулин М.Г., Падарян В.А., Довгалюк П.М., Фурсова Н.И., Васильев И.А. О некоторых ограничениях полносистемного анализа помеченных данных. Труды ИСП РАН, том 28, вып. 6, стр. 11-26, 2016 г. DOI: 10.15514/ISPRAS-2016-28(6)-1

1. Введение

С развитием информационных технологий задача противодействия вредоносному ПО принимает все новые формы. Изначально она ограничивалась поиском и удалением вирусов и троянов, заражающих ПК посредством дискет. В настоящее время защита информационных систем обеспечивается сложными программными и аппаратными комплексами. Один из востребованных аспектов противодействия – выявление вредоносного ПО, распространяемого посредством магазинов приложений. Приложения, при подаче в Google Play, проходят обязательную проверку инструментом Bouncer с целью не допустить попадания на устройства пользователей вредоносного ПО. Устройство Bouncer не раскрывалось, известно, что в нем используется динамический анализ – приложение выполняется в контролируемом окружении. Один из известных критериев оценки поведения отслеживает потоки данных в работающей программе: недопустимы ситуации утечки конфиденциальных данных в открытые каналы. Для Android источники и каналы заранее известны в силу четкой спецификации архитектуры системы и многих интерфейсов.

Схожие механизмы выявления вредоносного ПО используются в honeypot-системах. Такие системы, как Argos (Portokalidis, Slowinska, & Bos, 2006) и Timescope (Srinivasan & Jiang, 2012) применяют анализ помеченных данных для выявления вредоносного кода, причем последняя система, Timescope, совмещает это с механизмом детерминированного воспроизведения. Практическое применение этого механизма двойственно. Первая решаемая задача заключается в пометке входных данных и отслеживании ситуаций в последующей работе контролируемой программы, когда помеченные данные начнут использоваться в недопустимых операциях, требующих «доверия». Наиболее наглядные примеры – выполнение помеченных данных или пометка счетчика команд. Обратная задача, когда помечаются конфиденциальные данные и отслеживаются ситуации попадания помеченных данных в «недоверенные», открытые каналы связи. Эта задача наиболее актуальна для мобильных приложений, которым доступны различные чувствительные данные: SMS-переписка, голосовые и видео звонки, почта, данные геолокации и многое другое.

Несмотря на то, что за последние десять лет было опубликовано множество работ, посвященных анализу помеченных данных на уровне исполняемого кода, исследования так и не перешли на промышленный уровень. Относительным исключением является Bouncer, но в силу закрытости системы невозможно провести анализ принятых в ней решений.

Предлагаемые системы столкнулись с рядом принципиальных проблем, не имеющих общего решения: учет зависимостей по управлению на уровне исполняемого кода, определение эквивалентности кода и т.д. Следствием этих причин обусловлены проблемы излишней и недостаточной помеченности, выражающиеся в ложных срабатываниях и ложных пропусках соответственно.

В данной статье исследуются причины, препятствующие широкой эксплуатации анализа помеченных данных для выявления вредоносного кода. Авторами статьи была разработана система анализа, основанная на полносистемном эмуляторе Qemu. Реализация системы поддерживает гостевую архитектору x86, позволяет выявлять срабатывание вредоносного кода и обеспечивает интроспекцию виртуальной машины, совмещенную с детерминированным воспроизведением. Главной целью разработки стало нахождение компромисса между несколькими различными критериями: высокой скоростью работы, гибкостью настройки политик анализа помеченных данных и расширяемости системы на другие процессорные архитектуры и гостевые ОС.

Дальнейший материал организован следующим образом. Во втором разделе дается обзор близких работ, рассматриваются проблемы, с которыми сталкивается анализ помеченных данных на уровне бинарного кода. В третьем разделе описываются детали разработанного подхода к выявлению вредоносного кода, в следующем разделе – некоторые аспекты его реализации и результаты экспериментов. В Заключении делаются итоговые выводы и предлагаются дальнейшие улучшения системы.

2. Обзор близких работ

Динамический анализ помеченных данных – крайне популярная техника, которой посвящено множество публикаций. Рассмотрение представительного набора работ является отдельной задачей, в данном обзоре приведены только некоторые, наиболее показательные работы.

Известные инструменты, работающие на уровне бинарного кода, делятся на две категории: полносистемные и поддерживающие только пользовательский код. Анализ в пределах одного процесса, как правило, выполняется гораздо быстрей. Среди таких инструментов стоит упомянуть libdft (Kemerlis, Portokalidis, Jee, & Keromytis, 2012), Dytan (Clause, Li, & Orso, 2007), Minemu (Bosman, Slowinska, & Bos, 2011). Libdft и Dytan реализованы как Pin-инструменты и не предлагают достаточно высокого семантического уровня, работая с отдельными командами. Наибольшую гибкость в настройке правил анализа помеченных данных предоставляет Dytan, но ценой значительной деградации производительности: на примере архиватора gzip было зафиксировано 50 кратное замедление. Противоположность ему – система Minemu, показывающая высокую скорость работы, но имеющая множество ограничений, в частности: поддерживается только x86, «цветные» пометки и самомодифицирующийся код не поддерживаются, а XMM-регистры

задействованы для хранения служебной информации. Libdtf пытается достичь компромисса в скорости и гибкости, показанное им замедление не превосходило шестикратного.

Полносистемные инструменты анализа помеченных данных базируются на бинарной трансляции кода и интроспекции виртуальных машин. Исключение – TaintBochs (J. Chow, B. Pfaff, T. Garfinkel, K. Christopher, and M. Rosenblum, “Understanding data lifetime via whole system simulation,” in Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, ser. SSYM’04. Berkeley, CA, USA: USENIX Associati), основанный на эмуляторе Bochs и программно интерпретирующий инструкции x86. TaintBosch был создан для отслеживания времени жизни чувствительных данных в памяти приложений. Интерпретация инструкций уступает в скорости бинарной трансляции, а отслеживание помеченных данных вносит дополнительное замедление в 2-10 раз.

Инструмент Timescope (Srinivasan & Jiang, 2012) предлагает использовать механизм детерминированного воспроизведения для детального анализа атак, происходящих по сети. Применяемый метод состоит в следующем: во время работы полносистемного эмулятора все события, приходящие извне, записываются, что позволяет в дальнейшем точно повторить состояние эмулятора в каждый момент воспроизведения. При воспроизведении используются различные методики для более точного анализа происходящего: сохраняется журнал системных вызовов и их параметров для, например, более детального определения создания вредоносного процесса из-за эксплуатации уязвимости; файлы, созданные или изменённые за время предполагаемой атаки, сохраняются на хозяйской машине; также все данные, полученные по сети, помечаются, пометки отслеживаются, все помеченные инструкции сохраняются, чтобы иметь возможность анализировать шелл-код. Благодаря детерминированному воспроизведению удаётся исполнять исследуемые приложения при незначительном замедлении (1-1,5 раз по сравнению с обычным эмулятором), а тяжеловесный анализ (в том числе анализ помеченных данных) производить без необходимости беспокоиться о вносимом замедлении. Следует заметить, однако, что при таком подходе анализ помеченных данных покажет наличие атаки уже только после её осуществления. Другой недостаток инструмента – механизм воспроизведения не поддерживает графический интерфейс гостевой системы.

DECAF (Henderson, et al., 2014) – полносистемный эмулятор с поддержкой битовой и байтовой точности пометок. Эмулятор предоставляет API к базовым средствам анализа: интроспекции, анализу помеченных данных и инструментированию. API используют плагины, нацеленные на решение практических задач: трассировку системных вызовов, выявление логгеров клавиатуры, сбора трассы. Авторами заявлена поддержка двух процессорных архитектур: x86 и ARM и современных ОС: Windows 7 и 8. Поскольку DECAF

основан на устаревшей версии Qemu 1.0 благополучно работают только 32-х разрядные версии ОС. На основе DECAF построена DroidScope – специализированная система динамического анализа для платформы Android. В ней обеспечивается интроспекция на уровне виртуальной машины Dalvik и сопоставление событий разных семантических уровней: уровня машинных команд, ОС Linux, Dalvik VM.

ARGOS (Portokalidis, Slowinska, & Bos, 2006) – ещё один инструмент, основанный на QEMU (версия 1.1). Эмулятор представляет собой honeypot, он предназначен для обнаружения атак и автоматической генерации сигнатур угроз. В системе автоматически помечаются входящие сетевые пакеты с последующим обнаружением попадания помеченных данных в счетчик команд или использования их в качестве аргументов функции execve. Поддерживаются только архитектуры i386 и x86-64. Основная особенность реализации – ручное инструментирование всех инструкций архитектуры x86, что, с одной стороны, крайне трудоёмко, с другой стороны теоретически должно повысить скорость анализа.

Следующий подход к отслеживанию помеченных данных применяется в другом эмуляторе на основе QEMU – PANDA (R. Whelan, 2013). Существенной проблемой для поддержки различных архитектур оказываются вспомогательные функции. Если простые команды успешно транслируются TCG, то для сложных, например, системных команд используется программная интерпретация во вспомогательных функциях, написанных на языке Си. Поддержка анализа помеченных данных для каждой новой архитектуры будет требовать модификации вспомогательных функций. Для решения этой проблемы в PANDA весь код гостевой системы транслируется в биткод LLVM (Lattner & Adve, 2004) и уже на полученном биткоде проводится анализ. Для внутреннего представления TCG такую трансляцию пришлось единожды реализовывать, а код вспомогательных функций различных процессорных архитектур транслируется компилятором clang (Официальный сайт «Clang»). Такой подход двухуровневой трансляции позволяет расширяться практически на любую платформу, поддерживаемую Qemu, но приводит к значительному замедлению, оценки которого авторы избегают. Несущественность этой характеристики, по их мнению, обусловлена механизмом детерминированного воспроизведения. Заявлено, что замедление при записи журнала ограничивается 20% и не оказывает влияния на работу гостевой системы. Выполнение тяжеловесного анализа помеченных данных перенесено на этап воспроизведения, когда работа гостя в принципе не чувствительна к замедлению эмулятора.

Идея использовать LLVM как промежуточный язык почерпнута авторами PANDA у системы выборочного символьного выполнения S2E, имеющей близкое архитектурное устройство. S2E реализована на основе двух машин: Qemu для конкретного выполнения и KLEE – для символьного. Поскольку KLEE работает с LLVM, TCG и код вспомогательных функций транслируется.

В S2E отслеживание символьных переменных аналогично задаче анализа помеченных данных и страдает от аналогичных проблем, а именно, от избыточной и недостаточной помеченности. Чтоб ограничить распространение пометок (т.е. число символьных переменных) авторы S2E предложили выборочно включать продвижение пометок для определенных программных модулей. Следует отметить, что S2E демонстрирует высокую производительность, замедляясь по сравнению с Qemu в 3 раза в режиме конкретного выполнения и в 78 раз в режиме символьного выполнения. Несмотря на высокий интерес к S2E со стороны исследовательского сообщества, продуктизации данной технологии за 5 лет не произошло, показанные результаты представляют по большей части академический интерес.

Перечислим основные затруднения, с которыми сталкиваются инструменты анализа, как рассмотренные выше, так и оставшиеся за рамками обзора:

- неприемлемое замедление;
- выявление исследуемой программой факта работы в контролируемом окружении;
- недостаточная помеченность;
- избыточная помеченность;
- семантический разрыв;
- нерасширяемость на другие процессорные архитектуры и гостевые ОС.

В дальнейшем материале делается попытка преодолеть или смягчить указанные затруднения, как на уровне подхода к применению анализа помеченных данных в задачах безопасности ПО, так и при его реализации в рамках полносистемного эмулятора.

3. Полносистемный анализ помеченных данных на уровне бинарного кода

Проблемы, с которыми сталкивается практическое применение анализа помеченных данных на уровне бинарного кода, обусловлены причинами разного уровня. Часть перечисленных проблем решается путем комбинирования различных механизмов, встроенных в эмулятор, и определением порядка их применения. Предлагается подход, нацеленный на достижение максимальной универсальности (Рис. 1) в рамках полносистемного анализа. Выбранная постановка означает, что в качестве среды контролируемого выполнения для исследуемого кода будет использован полносистемный программный эмулятор.

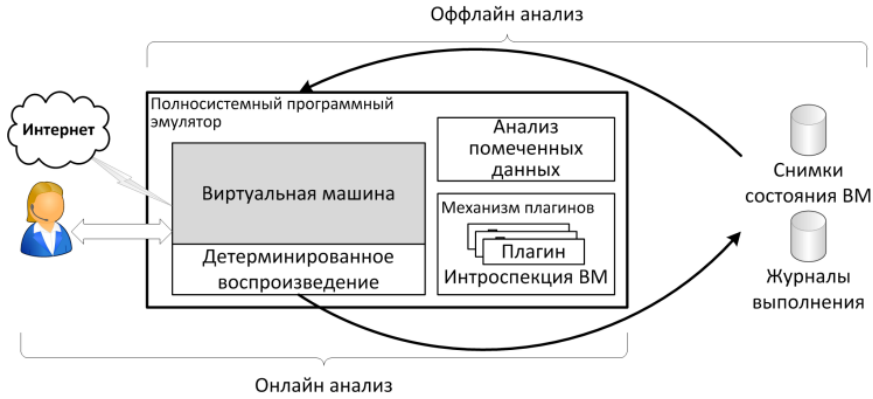


Рис. 1. Порядок применения эмулятора в онлайн и оффлайн анализе.

Fig. 1. Using full-system emulator for online and offline analysis.

Безальтернативной кандидатурой выступает эмулятор с открытым исходным кодом Qemu, поскольку он совмещает в себе целый ряд преимуществ: широкая практика промышленного использования, большое число поддерживаемых процессорных архитектур и периферийных устройств, встроенный, начиная с версии 2.5, механизм детерминированного воспроизведения. Последнее позволяет применять эмулятор как для онлайн, так и для оффлайн анализа, когда собранные журналы недетерминированных событий и снимки состояний используются для воспроизведения работы гостевой системы. Для применения эмулятора в онлайн анализе (фиксация и предотвращение недопустимых ситуаций) необходима эффективность анализа помеченных данных. Существует два подхода к реализации: системы PANDA, весь гостевой код транслируется в LLVM и анализ ведется на его уровне, и системы DECAF, когда инструментируется TCG. Второй подход имеет большую трудоемкость, т.к. для каждой новой архитектуры необходимо разрабатывать инструментирование новых вспомогательных функций, но не дает такого замедления, которое нарушает работу гостевой системы в онлайн режиме.

Другой важный аспект – обеспечение интроспекции произвольной ОС, работающей в гостевой системе для преодоления семантического разрыва. Перехват событий уровня ОС должен быть изолирован в пределах подключаемого модуля, что избавит от необходимости встраивать его в эмулятор на этапе сборки. Такого рода модули должны образовать расширяемую систему плагинов, способных взаимодействовать между собой и вести анализ, как во время реальной работы виртуальной машины, так и при ее воспроизведении. В настоящей версии Qemu должной инфраструктуры плагинов не имеется, ее разработка и реализация необходима для решения всей задачи в целом.

Механизм плагинов обеспечивает взаимодействие независимых модулей на основе API, предоставленный со стороны Qemu. Взаимодействия организованы согласно модели «сигнал-подписчик», схожей с моделью сигналов в Qt. Некоторый источник информации (Qemu или плагин) объявляет, что он предоставляет определенные данные с некоторым назначенным идентификатором, регистрируя новый сигнал в Qemu. Приемник информации (плагин), зная идентификатор интересующего его сигнала, «подписывается» на него, передавая указатель на функцию обработки сигнала. В момент готовности данных источник будет генерировать «сигнал», что является обращением к механизму плагинов. Передаваемые данные оформляются в виде параметров вызова и снабжаются должным идентификатором. В Qemu содержится список зарегистрированных сигналов; для каждого сигнала поддерживается список «подписчиков», т.е. список функций-обработчиков, которые Qemu вызывает с полученными из источника данными. Таким способом устроено оповещение плагинов как о служебных событиях (трансляция инструкции, трансляция базового блока, смена процесса), так и о событиях, возникающих в других плагинах.

Данный подход дает ряд преимуществ. От разработчика плагина скрываются особенности внутренней реализации Qemu, что значительно облегчает написание прикладного функционала. Такая организация взаимосвязи между плагинами позволяет установить многоуровневую систему повышения уровня абстракции событий без жесткой привязки к определенной версии ОС. С данными низкого уровня, получаемыми непосредственно от Qemu работают только т.н. плагины «первого уровня», плагины последующих уровней работают с данными, полученными от плагинов предыдущего уровня, что делает их независимыми от особенностей гостевой системы и процессорной архитектуры. Это существенно упрощает перенос уже существующих плагинов на новые версии Qemu или новые процессорные архитектуры, в идеале потребуется только доработка плагинов первого уровня.

Внутреннее устройство эмулятора, отвечающего требованиям, приведено на Рис. 2. В эмулятор встроено три независимо работающих механизма: детерминированное воспроизведение, анализ помеченных данных и механизм подключаемых плагинов анализа.

Механизм воспроизведения играет сугубо вспомогательную роль. Анализ помеченных данных совмещен с трансляцией гостевого кода в промежуточное представление TCG. Пометки хранятся в теневой памяти и регистрах. В код транслированных блоков добавляются инструментальные инструкции, продвигающие пометки.

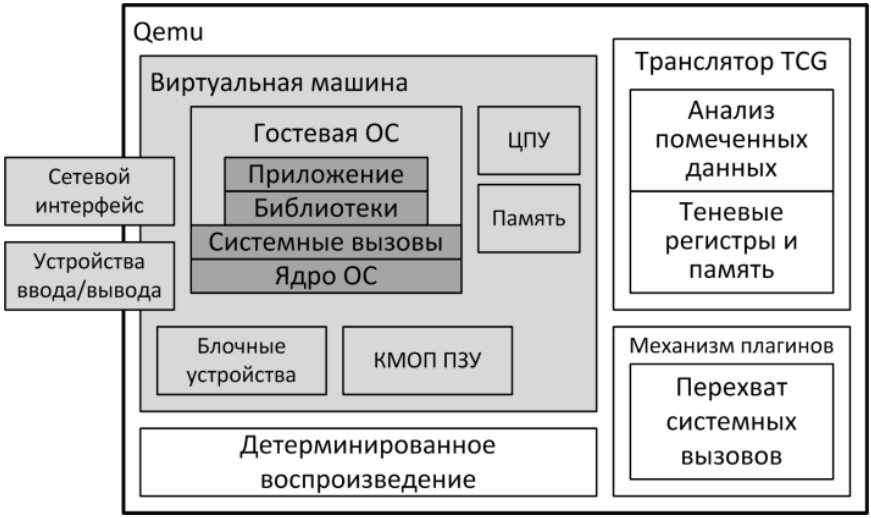


Рис. 2. Архитектура эмулятора с механизмом анализа помеченных данных.

Fig. 2. Architecture of the emulator with dynamic taint analysis support.

Пометки вносятся плагином, перехватывающим системные вызовы, такие как, открытие файла, отображение файла в память, работа с сетевым интерфейсом.

Недопустимые ситуации фиксируются двумя способами: инструментальными командами, встроенными в гостевой код, и плагинами. Первый способ работает для фиксирования низкоуровневых ситуаций, связанных с передачей управления на помеченную память или пометки счетчика команд. Второй способ – для фиксирования событий более высокого уровня, таких как передача помеченных данных в определенные системные вызовы и библиотеки.

Для исчерпывающего контроля потоков данных в системе необходимо отслеживать не только память и регистры процессора, но и потоки, проходящие через периферийные устройства. Блочные устройства, такие как жесткие диски или Flash-накопители хранят в себе данные, которые записываются и считываются программами. Более того, каждое устройство имеет служебные регистры, хранящие некоторое состояние, которое может быть записано, а затем считано. Вопрос отслеживания таких потоков сталкивается с необходимостью корректной интерпретации низкоуровневых обращений к устройствам, а это, в свою очередь, сопоставимо по сложности с разработкой всего комплекта драйверов системы. Тем не менее, должный контроль должен быть обеспечен для блочных устройств и КМОП ПЗУ, хранящей конфигурацию загрузчика.

При работе анализа помеченных данных неизбежно проявляются проблемы недостаточной и избыточной помеченности. Они обусловлены

принципиальными причинами: отсутствием на уровне бинарного кода информации о зависимостях по управлению, отсутствием универсальных правил продвижения пометок, когда помеченные данные влияют на адрес и т.п. Обобщить эти сложности можно в виде задачи определения, вырабатывает ли произвольный фрагмент кода константное значение или нет. В случае константного выражения пометки продвигаться на результат не должны. Известный и единственный способ борьбы с такими ситуациями – эвристическое улучшение правил продвижения пометок.

Стоит упомянуть, что известны работы, посвященные выявлению факта выполнения в эмуляторе Qemu (Raffetseder, Kruegel, & Kirda, 2007). Применяемая техника основывается на соотнесении неточностей эмуляции определенных машинных команд с эталонным поведением. Противодействие выявлению сводится к исправлению неточностей, открытый исходных код позволяет свободно это делать. Вопрос автоматического поиска неточностей и их диагностики в данной работе не рассматривается.

4. Особенности реализации подхода

Предложенный подход был реализован в виде прототипного инструмента, при разработке которого использовался инструмент DECAF. Одной из практических проблем стало то, что DECAF базируется на весьма старой версии Qemu 1.0. В процессе разработки было проведено перебазирование на актуальную версию Qemu 2.7. Основные усилия были потрачены на портирование вспомогательных функций на новую инфраструктуру эмулятора.

Поддержка битовой гранулярности не переносилась, оставшийся код байтовой гранулярности был специализирован для лучшей производительности. Поддерживается 8 независимых пометок. Пометка регистров сопроцессора с плавающей точкой не поддерживается. Т.к. не отслеживаются зависимости по управлению, регистр флагов тоже не помечается.

Устройство теневой памяти было сохранено, оно нацелено на экономию расхода памяти хостовой машины. Память представляет собой трехуровневое дерево, где корень указывает на промежуточные вершины, а промежуточные вершины содержат листья, отождествляющиеся с участками физической памяти гостевой системы. Листья и промежуточные вершины создаются только в том случае, если в соответствующем участке гостевой памяти содержится пометка.

В результате тестирования было выяснено, что битовые операции (такие как сдвиг и вращение) приводят к избыточной помеченности, поэтому результат таких операций считается помеченным только если исходное значение было помечено независимо от статуса пометки операнда, определяющего величину сдвига.

Для внесения пометок используется плагин, перехватывающий системные вызовы работы с файлами. Поддерживаются следующие функции Windows XP и Windows 7: NtCreateFile, NtOpenFile, NtReadFile, NtCreateSection, NtMapViewOfSection, NtOpenSection, NtWriteFile, NtUnmapViewOfSection, NtClose. Идентификация выбранного процесса делается на основе значения регистра CR3, т.к. в нем содержится база каталога таблиц виртуальной памяти. Это значение остается неизменным на всем протяжении жизни процесса.

5. Результаты экспериментов

Реализованный прототип был протестирован на нескольких примерах. В качестве гостя использовались операционные системы Windows XP SP2 и Windows 7, запуск производился в ОС Ubuntu 14.04LTS на компьютере с процессором Intel Core i5 2500 и 16 гигабайтами оперативной памяти. Примеры были скачаны с базы данных уязвимостей exploit-db.com. Результаты приведены в табл. 1.

Табл. 1. Результаты тестирования на известных уязвимостях.

Table 1. Evaluation of detection rate on known exploits.

Приложение	Поведение	Помеченный переход/код
Mediacoder 0.8.45.5852	Падение	+/+
Easy RM to MP3 Converter 2.7.3.700	Падение	+/-
Coolplayer 2.19.4	Падение	+/-
AllPlayer 5.8	Запуск calc.exe	-/-
MicroP 0.1.1.1600	Падение	-/-
Calavera Uploader! 3.5	Запуск calc.exe	+/+
AlReader 2.5	Запуск calc.exe	+/-
AudioCoder 0.8.22	Падение	+/+
BlazeDVD Pro 7.0	Падение	+/+

В первой колонке указано приложение, в котором проверялась известная уязвимость. Во второй колонке указано поведение системы при попытке эксплуатации уязвимости, например, аварийное завершение приложения при попытке открытия сформированного файла. В третьей колонке указано, выводилось ли предупреждение о подозрительной активности: первый знак (плюс или минус) сообщает о том, был ли совершён переход по помеченному адресу, второй – было ли исполнение помеченного кода. В трех случаях произошёл запуск стандартного приложения «Калькулятор», то есть

однозначно произошла эксплуатация уязвимости. В шести случаях произошло падение тестируемого приложения, что свидетельствует о неправильно обработанных входных данных; скорее всего, эксплуатации при этом не произошло из-за несоответствия версий операционных систем, в которых создавался эксплоит и проводилась проверка, что привело к неправильным адресам переходов в созданных файлах, или ошибок при формировании файла для эксплуатации уязвимости. Например, при попытке эксплуатации приложения Coolplayer был совершен переход на адрес 0x41414141, что соответствует последовательности символов “AAAA”, которые содержатся в файле как заполнитель для переполнения буфера на стеке; это позволяет предположить, что создателями эксплоита было неверно рассчитано расположение адреса возврата.

Также для некоторых приложений создавались «доброкачественные» файлы и проверялось, будет ли выводиться предупреждение при их открытии. В большинстве случаев предупреждение не выводилось, но, например, в приложении BlazeDVD Pro выводились сообщения о подозрительной активности и при открытии обычного файла.

В других инструментах (таких как Panda, Decaf) точность анализа не проверялась, так как в них не имеется механизмов, позволяющих осуществлять пометку файлов или выводить предупреждения, реализованные в разработанном прототипе.

Чтобы примерно оценить замедление, вносимое анализом помеченных данных, было проведено два теста: сжатие папки с различными файлами общим объемом 27 мегабайт при помощи архиватора 7-zip, шифрование папки объёмом 700 мегабайт по алгоритму AES-256, запуск операционной системы.

Табл. 2. Оценка замедления от анализа помеченных данных.

Table 2. Slowdown of different dynamic taint analysis engines.

Среда выполнения	Время, с	
	7-zip	AES-256
Argos	630	1118
Panda	646	730
Decaf	340	675
Разработанный прототип	***315***	***579***
QEMU	88	103
Native	6	12

Для сравнения использовались различные доступные эмуляторы с включенным анализом помеченных данных, эмулятор QEMU без внесенных

модификаций, а также исполнение на реальном оборудовании (обозначено в таблице как Native). Результаты приведены в табл. 2.

По результатам этих двух тестов видно, что предложенная реализация обладает наименьшим замедлением. Эмулятор DECAF, на котором базируется наш прототип, отстаёт, скорее всего, из-за двух своих особенностей реализации: использование битовой точности и старая версия базового эмулятора QEMU (1.0). Остальные эмуляторы показывают значительно большее замедление.

По результатам экспериментов можно сделать следующие выводы: разработанный прототип оказывается способен обнаруживать некоторые из угроз, а также показывает наибольшую скорость работы из доступных инструментов полносистемного анализа помеченных данных. В то же время требуется доработка в плане точности анализа (необнаруженные эксплуатации в двух приложениях) и расширение класса обнаруживаемых угроз.

6. Заключение

В статье исследованы ограничения, с которыми сталкивались попытки применить анализ помеченных данных на уровне бинарного кода. Среди них были указанны те, что представляют наибольшую актуальность при проведении анализа ПО в рамках всей вычислительной системы. Предложен расширяемый подход к организации анализа помеченных данных с целью выявления активности вредоносного ПО, как в виде недопустимого выполнения кода, так и в виде утечки чувствительных данных.

Дальнейшие работы предполагают поддержку таких распространенных архитектур, как ARM и MIPS. Целесообразно расширить поддержку источников помеченных данных на сетевые и USB-устройства, клавиатурный ввод.

Интересен вопрос выявления в исследуемом коде необратимых функций и их учет при продвижении пометок, а также расширение эвристик нахождения константных выражений. Оба улучшения нацелены на снижение доли ложных срабатываний.

На текущий момент не используется статико-динамический подход для отслеживания зависимостей по управлению на уровне бинарного кода. Потенциально он способен улучшить результаты в ситуациях, когда требуется изучить свойства фиксированной программы на большом числе различных входных данных. Потребуется исследовать и разработать эвристики выборочного учета зависимостей по управлению, схожие с теми, что были предложены в работе DTA++ (Kang, McCamant, Poosankam, & Song, 2011).

Список литературы

[1]. Portokalidis G., Slowinska A., Bos H. Argos: An emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. Proceedings of the

1st ACMSIGOPS/EuroSys European Conference on Computer Systems. New York. 2006. pp. 15-27.

[2]. Srinivasan D., Jiang X. Timescope: Time-Traveling Forensic Analysis of VM-Based High-Interaction Honeypots. In: Security and Privacy in Communication Networks: 7th International ICST Conference, SecureComm 2011, London, UK, September 7-9, 2011, Revised Selected Papers. Springer Berlin Heidelberg, 2012. pp. 209-226.

[3]. Kemerlis V.P., Portokalidis G., Jee K., Keromytis A.D. libdft: practical dynamic data flow tracking for commodity systems. VEE. 2012. pp. 121-132.

[4]. Clause J.A., Li W., Orso A. Dytan: a generic dynamic taint analysis framework. ISSTA. 2007. pp. 196-206.

[5]. Bosman E., Slowinska A., Bos H. Minemu: The World's Fastest Taint Tracker. RAID. 2011. Vol. 6961. pp. 1-20.

[6]. J. Chow, B. Pfaff, T. Garfinkel, K. Christopher, and M. Rosenblum, “Understanding data lifetime via whole system simulation,” in Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, ser. SSYM’04. Berkeley, CA, USA: USENIX Association [Электронный ресурс]

[7]. Henderson A., Prakash A., Yan L.K., Hu X., Wang X., Zhou R., Yin H. Make it work, make it right, make it fast: building a platform-neutral whole-system dynamic binary analysis platform. The International Symposium on Software Testing and Analysis (ISSTA'14). San Jose. 2014.

[8]. R. Whelan T.L.D.K. Architecture-independent dynamic information flow tracking. Proceedings of the 22Nd International Conference on Compiler Construction. Berlin. 2013.

[9]. Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis & transformation. Proceedings of the 2004 International Symposium on Code Generation and Optimization. 2004.

[10]. Официальный сайт «Clang» [Электронный ресурс]. http://clang.llvm.org/ (дата обращения: 29.09.2015).

[11]. Raffetseder T., Kruegel C., Kirda E. Detecting system emulators. Proceedings of the 10th International Conference on Information Security. 2007. pp. 1 - 18.

[12]. Kang M.G., McCamant S., Poosankam P., Song D. DTA++: dynamic taint analysis with targeted control-flow propagation. Proceedings of the Network and Distributed System Security Symposium. San Diego. 2011.

On Some Limitations of Information Flow Tracking in Full-system Emulators

[1]*M.A. Klimushenkova <maria.klimushenkova@ispras.ru>*
[2]*M.G. Bakulin <bakulinm@ispras.ru>*
[2,3]*V.A. Padaryan <vartan@ispras.ru>*
[1]*P.M. Dovgalyuk <pavel.dovgaluk@ispras.ru>*
[1]*N.I. Fursova <Natalia.Fursova@ispras.ru>*
[1]*I.A. Vasiliev <vasiliev@ispras.ru>*
[1]*Novgorod State University*
41 B. St. Petersburgskaya Str., Veliky Novgorod, 173003, Russian Federation
[2]*Institute for System Programming of the Russian Academy of Sciences,*
25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation
[3]*Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

Abstract. Tracking and verification of data flows includes set of techniques that can be applied to make applications more secure, to perform software analysis for debugging or reverse engineering, and so on. Taint analysis is one of the techniques used to control data flows. This paper presents an approach for system-wide lightweight platform-aware taint analysis. We implemented proof-of-concept tool based on our approach for i386 platform upon the multi-platform simulator QEMU. Our approach uses instrumentation of QEMU intermediate representation of binary code and processes up to 8 taints simultaneously. Most of the taint analysis code is target-independent and may be used with other target platforms. For some platforms (i386 with Windows XP and Windows 7) we present Virtual Machine Introspection plugins for automatic file tainting. We demonstrate how our taint analysis system may be used to detect exploitation of software vulnerabilities.

Keywords: taint analysis; dynamic analysis; QEMU.

DOI: 10.15514/ISPRAS-2016-28(6)-1

For citation: Klimushenkova M.A., Bakulin M.G., Padaryan V.A., Dovgalyuk P.M., Fursova N.I., Vasiliev I.A. On Some Limitations of Information Flow Tracking in Full-system Emulators. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 11-26 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-1

References

[1]. Portokalidis G., Slowinska A., Bos H. Argos: An emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. Proceedings of the 1st ACMSIGOPS/EuroSys European Conference on Computer Systems. New York. 2006. pp. 15-27.

[2]. Srinivasan D., Jiang X. Timescope: Time-Traveling Forensic Analysis of VM-Based High-Interaction Honeypots. In: Security and Privacy in Communication Networks: 7th

International ICST Conference, SecureComm 2011, London, UK, September 7-9, 2011, Revised Selected Papers. Springer Berlin Heidelberg, 2012. pp. 209-226.

[3]. Kemerlis V.P., Portokalidis G., Jee K., Keromytis A.D. libdft: practical dynamic data flow tracking for commodity systems. VEE. 2012. pp. 121-132.

[4]. Clause J.A., Li W., Orso A. Dytan: a generic dynamic taint analysis framework. ISSTA. 2007. pp. 196-206.

[5]. Bosman E., Slowinska A., Bos H. Minemu: The World's Fastest Taint Tracker. RAID. 2011. Vol. 6961. pp. 1-20.

[6]. J. Chow, B. Pfaff, T. Garfinkel, K. Christopher, and M. Rosenblum, “Understanding data lifetime via whole system simulation,” in Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, ser. SSYM’04. Berkeley, CA, USA: USENIX Associati [web-resource]

[7]. Henderson A., Prakash A., Yan L.K., Hu X., Wang X., Zhou R., Yin H. Make it work, make it right, make it fast: building a platform-neutral whole-system dynamic binary analysis platform. The International Symposium on Software Testing and Analysis (ISSTA'14). San Jose. 2014.

[8]. R. Whelan T.L.D.K. Architecture-independent dynamic information flow tracking. Proceedings of the 22Nd International Conference on Compiler Construction. Berlin. 2013.

[9]. Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis & transformation. Proceedings of the 2004 International Symposium on Code Generation and Optimization. 2004.

[10]. Clang [web-resource]. http://clang.llvm.org/ (last visit: 29.09.2015).

[11]. Raffetseder T., Kruegel C., Kirda E. Detecting system emulators. Proceedings of the 10th International Conference on Information Security. 2007. pp. 1 - 18.

[12]. Kang M.G., McCamant S., Poosankam P., Song D. DTA++: dynamic taint analysis with targeted control-flow propagation. Proceedings of the Network and Distributed System Security Symposium. San Diego. 2011.

Классификация ROP гаджетов

А.В. Вишняков <vishnya@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В данной работе предложен метод классификации ROP гаджетов, который позволяет аналитику сделать вывод о применимости техники ROP для эксплуатации уязвимостей в том или ином случае. Метод, реализованный в виде программного инструмента, применим к бинарным файлам программ. Работа инструмента продемонстрирована на 32-х и 64-х битных приложениях из дистрибутива Ubuntu 14.04, а возможность применения ROP на основе результатов классификации подтверждена на нескольких примерах.

Ключевые слова: уязвимость; ROP; классификация.

DOI: 10.15514/ISPRAS-2016-28(6)-2

Для цитирования: Вишняков А.В. Классификация ROP гаджетов. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 27-36. DOI: 10.15514/ISPRAS-2016-28(6)-2

1. Введение

Уязвимость переполнения буфера на стеке имеет широкую распространенность. Эксплуатация данной уязвимости может привести к перехвату потока управления [1]. Обычно перехват потока управления производится при помощи передачи управления на некоторую вредоносную нагрузку. Для формирования такой нагрузки может быть использована техника возвратно-ориентированного программирования (ROP), применимая в условиях работы современных защитных механизмов. Нагрузка формируется из последовательности ROP гаджетов, которые представляют собой набор инструкций, заканчивающийся инструкцией передачи управления.

Важной задачей является исследование программ на предмет применимости ROP. В данной работе предложен метод классификации ROP гаджетов, который позволяет аналитику сделать вывод о том, возможна ли эксплуатация в том или ином случае.

2. Предпосылки к появлению ROP

Получив контроль над значениями, записываемыми в буфер, можно перезаписать адрес возврата из функции указателем на размещенный на стеке вредоносный код. Чтобы избежать негативных последствий от переполнения буфера на стеке, появились различные защитные механизмы.

Предотвращение выполнения данных (DEP) – механизм защиты, запрещающий исполнение кода из областей памяти, помеченных как «данные». Механизм успешно применяется в операционных системах (Windows, Linux и др.) и предотвращает выполнение кода, размещенного на стеке. В ответ на DEP появилась атака возврата в библиотеку, позволяющая обойти DEP. Атака заключается в подмене адреса возврата адресом некоторой библиотечной функции, например, функции system из библиотеки libc.

Механизм рандомизации адресного пространства (ASLR) позволяет разместить важные структуры программы (стек, куча, образ программы, динамические библиотеки) по различным адресам во время каждого запуска программы. Данная защита противодействует проведению атаки возврата в библиотеку, т.к. адрес библиотечной функции неизвестен до выполнения программы. Однако для рандомизации адреса загрузки образа программы или библиотек они должны быть скомпилированы с соответствующими флагами компилятора, что не всегда выполняется. Так в Linux адрес загрузки большинства исполняемых файлов остается нерандомизированным, что оставляет пути для эксплуатации.

Возвратно-ориентированное программирование (ROP) [2] является развитием метода эксплуатации путем возврата в библиотеку. ROP так же, как и атака возврата в библиотеку позволяет перехватывать поток управления для эксплуатации в условиях работы DEP. Но метод представляет большую опасность: он может быть применен в условиях работы ASLR для нерандомизированных исполняемых файлов. Метод использует последовательности инструкций, заканчивающиеся инструкцией передачи управления, из нерандомизированных исполняемых секций программы. Такие последовательности инструкций называются гаджетами. Гаджеты собираются в цепочки, и их адреса размещаются от адреса возврата на стеке так, чтобы первый гаджет передавал управление второму, второй – третьему и т.д. Таким образом, с помощью цепочки гаджетов можно выполнить некоторые вредоносные действия. Например, ROP-цепочка на рис. 1 производит запись произвольного значения по произвольному адресу.

Следует отметить, что применение ROP для эксплуатации уязвимости возможно при условии наличия достаточного для осуществления эксплуатации набора гаджетов из нерандомизированных областей памяти. Выделение гаджетов, пригодных для составления ROP-цепочек, может быть автоматизировано, что достигается путем поиска и последующей классификации гаджетов.

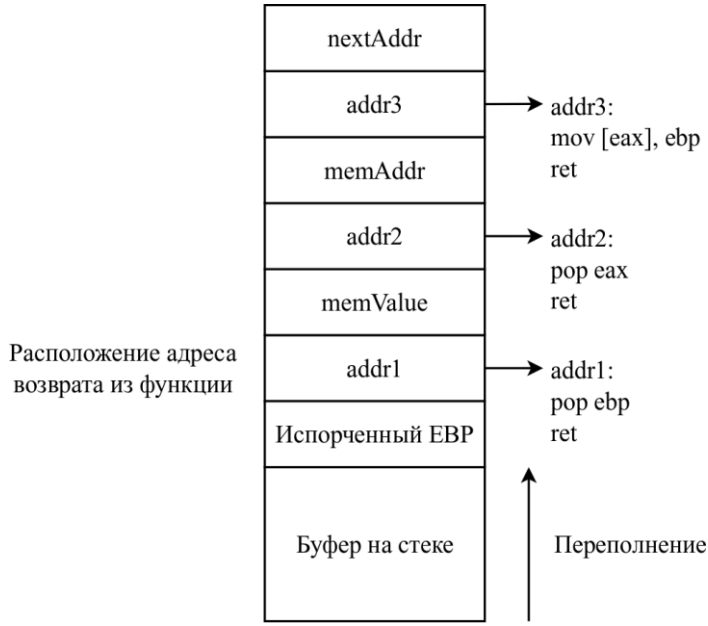


Рис. 1. Состояние стека после размещения ROP-цепочки, которая записывает значение memValue по адресу memAddr и передает управление на nextAddr

Fig. 1. The stack state after setting a ROP-chain which stores memValue to memAddr and transfers control to nextAddr

3. Поиск гаджетов

Для поиска гаджетов может быть использован алгоритм Галилео [2], который осуществляет поиск инструкций передачи управления в исполняемых секциях программы. Для каждой найденной инструкции производится дизассемблирование нескольких байт, предшествующих инструкции. Все корректно дизассемблированные последовательности инструкций добавляются в набор найденных гаджетов.

4. Классификация гаджетов

Классификация гаджетов позволяет повысить уровень представления результатов поиска гаджетов. Классификация производится на основе дополненного набора типов гаджетов, предложенного в Q [3] (табл. 1). Каждый тип гаджета определяется некоторой параметризованной семантикой. Задача классификации гаджетов заключается в сопоставлении каждого гаджета удовлетворяющим его семантике классам и их параметрам. Например, гаджет «mov ecx, edx ; mov edi, esi ; xor eax, eax ; ret» удовлетворяет следующим классам: «MoveRegG : ecx ← edx», «MoveRegG : edi ← esi», «InitConstG : eax ← 0».

Табл. 1. Типы гаджетов. [Addr] означает доступ к памяти по адресу Addr, ◦ – бинарную операцию. a ← b означает, что конечное значение a равно начальному значению b. X ◦← Y – сокращение для X ← X ◦ Y.

Table 1. Gadget types. [Addr] means accessing memory at address Addr, ◦ – binary operation. a ← b means that final value of a equals initial value of b. X ◦← Y is short for X ← X ◦ Y.

Тип	Параметры	Семантика
NoOpG	N/A	Не меняет ничего в памяти и на регистрах
JumpG	AddrReg	IP ← AddrReg
MoveRegG	InReg, OutReg	OutReg ← InReg
LoadConstG	OutReg, Offset	OutReg ← [SP + Offset]
ArithmeticG	InReg1, InReg2, OutReg, ◦	OutReg ← InReg1 ◦ InReg2
LoadMemG	AddrReg, OutReg, Offset	OutReg ← [AddrReg + Offset]
StoreMemG	AddrReg, InReg, Offset	[AddrReg + Offset] ← InReg
ArithmeticLoadG	AddrReg, OutReg, Offset, ◦	OutReg ◦← [AddrReg + Offset]
ArithmeticStoreG	AddrReg, InReg, Offset, ◦	[AddrReg + Offset] ◦← InReg
InitConstG	OutReg, Value	OutReg ← Value
ShiftStackG	Offset, ◦ (+ / –)	SP ◦← Offset
ArithmeticStackG	InReg, ◦ (+ / –)	SP ◦← InReg

Для использования гаджетов в целях эксплуатации необходимо, чтобы они удовлетворяли следующим требованиям:

- Гаджет может изменять указатель стека только на константное значение, исключение – ArithmeticStackG.
- Побочные эффекты выполнения гаджета не должны приводить к неконтролируемому поведению программы. Например, запись значения по произвольному адресу памяти может привести к аварийному завершению программы.

Результатами классификации гаджета являются:

- типы и параметры гаджета;
- список регистров, значения которых не сохраняются в результате выполнения гаджета;
- информация о фрейме гаджета (размер фрейма, смещение ячейки с адресом следующего гаджета относительно указателя стека после передачи управления текущему гаджету). Например, гаджет «pop rax ; ret 16» имеет размер фрейма 32, а адрес следующего гаджета размещается по смещению 8.

5. Метод классификации и детали реализации

Набор гаджетов для классификации получается при помощи инструмента с открытым исходным кодом ROPgadget [4], в котором реализован алгоритм Галилео поиска гаджетов.

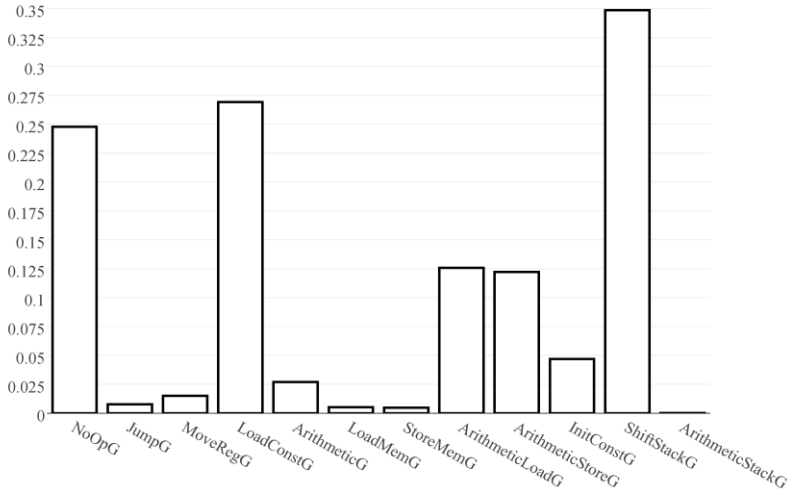


Рис. 2. Отношение числа классифицированных гаджетов данного типа к общему числу найденных гаджетов для архитектуры x86

Fig. 2. The ratio of classified gadgets of a certain type to all found gadgets for x86 architecture

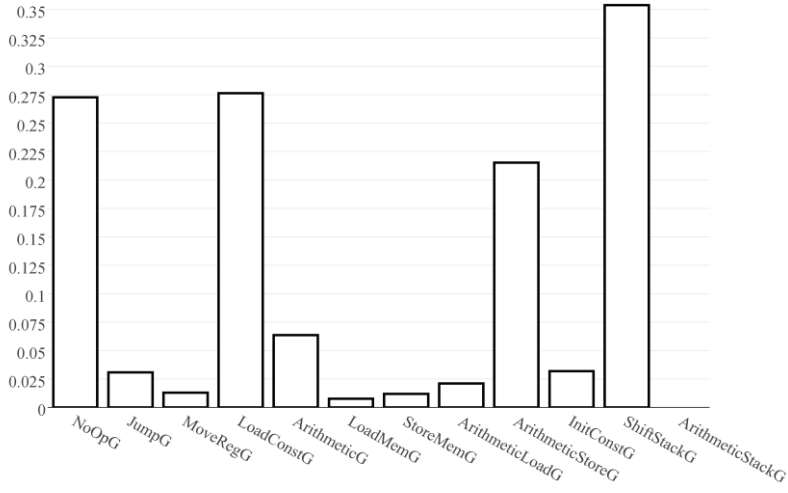


Рис. 3. Отношение числа классифицированных гаджетов данного типа к общему числу найденных гаджетов для архитектуры x86-64

Fig. 3. The ratio of classified gadgets of a certain type to all found gadgets for x86-64 architecture

Классификация гаджета производится на основе анализа эффектов выполнения гаджета на конкретных входных данных. Инструкции гаджета транслируются в промежуточное представление Pivot [5]. Далее запускается процесс интерпретации Pivot-представления. Во время интерпретации отслеживаются обращения к регистрам и памяти. Если происходит первое чтение регистра или области памяти, считанное значение генерируется случайным образом. В результате интерпретации будут получены начальные и конечные значения регистров и памяти. На основе этой информации делается вывод о возможной принадлежности гаджета тому или иному типу. Например, для принадлежности типу MoveRegG должна существовать такая пара регистров, что начальное значение первого регистра равно конечному значению второго. В результате анализа составляется список всех удовлетворяющих гаджету классов и их параметров (список кандидатов). Затем производится еще несколько запусков конкретного выполнения с различными входными данными, в результате которых из списка кандидатов удаляются ошибочно определенные классы.

Предложенный метод основывается на результатах выполнения гаджета на ограниченном количестве наборов конкретных входных данных, что в общем случае не гарантирует соответствие семантике результата выполнения гаджета на произвольных входных данных. Таким образом, возможна неверная классификация гаджета. Однако доля неверно классифицированных гаджетов на практике незначительна. Метод был реализован в виде программного инструмента.

6. Апробация метода

Метод был апробирован на исполняемых файлах из каталога /usr/bin дистрибутива операционной системы Ubuntu 14.04. Классификатор гаджетов показал быструю работу на приложениях архитектур x86 и x86-64. Статистика по распределению типов гаджетов приводится на рис. 2-3.

С использованием классификатора гаджетов была подтверждена возможность эксплуатации следующих примеров в условиях работы современных защитных механизмов.

Эксплуатация DEP и ASLR. На основе результата классификации гаджетов, найденных в 32-х битной программе zsnes с уязвимостью переполнения буфера на стеке (strcpy), была вручную составлена ROP-цепочка, вызывающая интерпретатор командной строки (табл. 2).

Для вызова интерпретатора командной строки используется функция system из библиотеки libc. Так как адрес функции system рандомизирован, используется техника возврата в рандомизированную libc [6]. Механизм ленивого связывания в Linux осуществляет загрузку адреса библиотечной функции в GOT при первом ее вызове. Zsnes не импортирует функцию system, поэтому ее адрес вычисляется во время выполнения программы с помощью ROP, используя адрес какой-либо другой функции, чей адрес уже загружен в

GOT к моменту переполнения. В приведенном примере такой функцией является функция access. Для вычисления адреса используется предположение о том, что смещение двух функций из libc относительно друг друга остается постоянным. Тогда адрес system вычисляется как сумма фактического адреса access, считанного из GOT, и смещения функции system относительно access в библиотеке libc.

Табл. 2. ROP-цепочка для zsnes. Got_addr – адрес GOT, access_got_addr – адрес ячейки GOT с адресом функции access

Table 2. ROP-chain for zsnes. Got_addr – address of GOT, access_got_addr – access GOT entry address

`eax = got_addr`	`LoadConstG : eax ← [esp]`
`ecx = “\xcdbin”`	`LoadConstG : ecx ← [esp]`
`cl += ch  # ecx = “/bin”`	`ArithmeticG : cl ← cl + ch`
`[eax] = ecx`	`StoreMemG : [eax] ← ecx`
`eax = got_addr + 4`	`LoadConstG : eax ← [esp]`
`ecx = 0xf837acd7`	`LoadConstG : ecx ← [esp]`
`[eax] = ecx`	`StoreMemG : [eax] ← ecx`
`[eax] += eax  # [eax] = “/sh\x00”`	`ArithmeticStoreG : [eax] +← eax`
`ecx = access_got_addr`	`LoadConstG : ecx ← [esp]`
`eax = &system - &access`	`LoadConstG : eax ← [esp]`
`eax += [ecx]`	`ArithmeticLoadG : eax +← [ecx]`
`jmp eax`	`JumpG : eip ← eax`

Также необходимо разместить строковый аргумент «/bin/sh» в глобальной памяти программы. В рассматриваемом примере строка размещается в первых двух ячейках GOT. Во время составления цепочки следует учесть ограничения, накладываемые структурой программы на символы ROP-цепочки. В zsnes поток управления зависит от входных данных – последний символ ‘/’ или ‘\’ заменяется нулем, следовательно, ROP-цепочка не должна содержать данных символов. Более того, ROP-цепочка не должна содержать нулевых байтов, т.к. она является частью нуль-терминированной строки.

Для полученной ROP-цепочки удалось продемонстрировать эксплуатацию уязвимости переполнения буфера на стеке в условиях одновременного функционирования защит DEP и ASLR.

Эксплуатация DEP, ASLR и «канарейки». Если в программе присутствует уязвимость переполнения буфера на стеке при условии «write-what-where» [7], то возможна перезапись ячейки таблицы GOT некоторой вызываемой функции адресом гаджета-трамплина (ShiftStackG). После вызова функции гаджет-трамплин передаст управление на оставшуюся часть ROP-цепочки. С помощью классификатора гаджетов в модельном примере был успешно

найден гаджет ShiftStackG для передачи управления на ROP-цепочку, что позволило подтвердить возможность эксплуатации уязвимости в присутствии «канарейки», DEP и ASLR.

7. Заключение

В статье представлен метод классификации ROP гаджетов. Применение предложенного метода позволяет выявить среди обнаруженных в программах уязвимостей переполнения буфера на стеке наиболее приоритетные для исправления, а именно, те уязвимости, для которых возможна эксплуатация. Предложенный метод был реализован в виде программного инструмента.

Возможными направлениями для дальнейших исследований являются дедуктивная верификация семантики гаджета и оценка эффективности современных защит от ROP.

Список литературы

[1]. One A. Smashing the stack for fun and profit. Phrack magazine, v. 7, №. 49, 1996, pp. 14-16.

[2]. Hovav Shacham. The Geometry of Innocent Flash on the Bone: Return-into-libc without Function Calls (on the x86). 2007 ACM Conference on Computer and Communications Security (CCS), Proceedings of CCS 2007, pp. 552-561.

[3]. Edward J. Schwartz, Thanassis Avgerinos, David Brumley. Q: Exploit Hardening Made Easy. 2011 Usenix Security Symposium (SEC), Proceedings of SEC 2011.

[4]. Инструмент ROPgadget. https://github.com/JonathanSalwan/ROPgadget

[5]. V. A. Padaryan, M. A. Solovyev, A. I. Kononov. Simulation of operational semantics of machine instructions. Program. Comput. Software, vol. 37, № 3, 2011, pp. 161-170. DOI: 10.1134/S0361768811030030

[6]. G. F. Roglia, L. Martignoni, R. Paleari, D. Bruschi. Surgically Returning to Randomized lib(c). 2009 Annual Computer Security Applications Conference (ACSAC), Proceedings of ACSAC 2009, pp. 60-69.

[7]. CWE-123: Write-what-where Condition. http://cwe.mitre.org/data/definitions/123.html

Classification of ROP gadgets

A.V. Vishnyakov <vishnya@ispras.ru>
Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Return-oriented programming (ROP) is a dangerous exploitation technique which can be used to bypass modern defense mechanisms. ROP reuses code chunks ending with control transfer instruction from a program binary to form a chain corresponding some payload. These code chunks are called gadgets. Though, a certain set of gadgets should be available to exploit a vulnerability. Determining gadgets that can be used to form a ROP chain can be done by gadgets search and classification. This paper introduces a method for ROP gadgets classification that allows one to evaluate whether or not ROP technique can be used to exploit a program vulnerability. Classification is based on side-effects analysis of gadget execution with concrete inputs. Gadget instructions are translated into IR which is interpreted to track registers and memory usage. Initial registers and memory values are randomly generated. According to initial and final values of registers and memory gadget semantics can be explored. Classification performs several executions to determine gadget semantics. Proposed method is applied to program binaries and its capabilities were demonstrated on 32-bit and 64-bit binaries from Ubuntu 14.04. Using classification results program exploitability was confirmed for several examples. Furthermore, a possible exploitation of stack buffer overflow vulnerability in presence of write-what-where condition was shown on a model example demonstrating a bypass of canary, DEP and ASLR.

Keywords: vulnerability; ROP; classification.

DOI: 10.15514/ISPRAS-2016-28(6)-2

For citation: Vishnyakov A.V. Classification of ROP gadgets. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 27-36 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-2

References

[1]. One A. Smashing the stack for fun and profit. Phrack magazine, vol. 7, №. 49, pp. 14-16.
[2]. Hovav Shacham. The Geometry of Innocent Flash on the Bone: Return-into-libc without Function Calls (on the x86). 2007 ACM Conference on Computer and Communications Security (CCS), Proceedings of CCS 2007, pp. 552-561.
[3]. Edward J. Schwartz, Thanassis Avgerinos, David Brumley. Q: Exploit Hardening Made Easy. 2011 Usenix Security Symposium (SEC), Proceedings of SEC 2011.
[4]. ROPgadget. https://github.com/JonathanSalwan/ROPgadget
[5]. V. A. Padaryan, M. A. Solovyev, A. I. Kononov. Simulation of operational semantics of machine instructions. Program. Comput. Software, vol. 37, № 3, 2011, pp. 161-170. DOI: 10.1134/S0361768811030030

[6]. G. F. Roglia, L. Martignoni, R. Paleari, D. Bruschi. Surgically Returning to Randomized lib(c). 2009 Annual Computer Security Applications Conference (ACSAC), Proceedings of ACSAC 2009, pp. 60-69.
[7]. CWE-123: Write-what-where Condition. http://cwe.mitre.org/data/definitions/123.html

Динамическая компиляция SQL-запросов для СУБД PostgreSQL

[1] *Р.А. Бучацкий <ruben@ispras.ru>*
[1, 2] *Е.Ю. Шарыгин <eush@ispras.ru>,*
[2] *Л.В. Скворцов <leonidxo@gmail.com>*
[1] *Р.А. Жуйков <zhroma@ispras.ru>,*
[1] *Д.М. Мельник <dm@ispras.ru>*
[3] *Р.В. Баев <baev@ispras.ru>*
[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1, стр. 52, факультет ВМК.
[3] *Московский физико-технический институт (государственный университет)*
141701, Московская область, г. Долгопрудный, Институтский переулок, д.9

Аннотация. В последние годы по мере увеличения производительности и роста объема оперативной и внешней памяти производительность СУБД для некоторых классов запросов определяется непосредственно скоростью обработки запросов процессором. Для исполнения SQL-запросов в большинстве современных реляционных СУБД используется модель итераторов (Volcano-модель), которая удобна в реализации в рамках интерпретатора запросов, но сопряжена с существенными накладными расходами при выполнении плана, например, связанными с большим количеством ветвлений, неявными вызовами функций-обработчиков и выполнением лишних проверок, избежать которых довольно сложно при использовании механизма интерпретации. Одно из решений – динамическая компиляция запросов. В рамках данной работы рассматривается метод динамической компиляции запросов с применением альтернативной модели выполнения запроса в СУБД, что подразумевает отказ от используемой в PostgreSQL итеративной Volcano-модели, и его реализация для СУБД PostgreSQL с помощью компиляторной инфраструктуры LLVM. Динамический компилятор запросов реализован в виде расширения к СУБД PostgreSQL и не требует изменения исходного кода СУБД. Результаты проведенного тестирования показывают, что динамическая компиляция запросов с помощью JIT-компилятора LLVM позволяет получить ускорение в несколько раз на тестовом наборе TPC-H.

Ключевые слова: динамическая компиляция; JIT-компиляция; СУБД; PostgreSQL; LLVM; языки запросов.

DOI: 10.15514/ISPRAS-2016-28(6)-3

Для цитирования: Бучацкий Р.А., Шарыгин Е.Ю., Скворцов Л.В., Жуйков Р.А., Мельник Д.М., Баев Р.В. Динамическая компиляция SQL-запросов для СУБД PostgreSQL. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 37-48. DOI: 10.15514/ISPRAS-2016-28(6)-3

1. Введение

Среди систем управления базами данных идёт постоянная борьба за производительность. Работы по улучшению производительности большинства реляционных СУБД традиционно были в основном направлены на оптимизацию доступа к внешней памяти, поскольку именно скорость доступа к данным обычно является узким местом при выполнении запросов. В последнее время, в связи с ростом объёмов и улучшением операционных характеристик доступа к оперативной памяти, стала актуальна задача более эффективного использования процессора.

Предлагаемый в данной работе метод динамической компиляции запросов позволит значительно увеличить производительность СУБД на запросах, скорость обработки которых в первую очередь определяется эффективностью использования процессора. Динамическая компиляция может быть выполнена с использованием информации, доступной только во время выполнения запроса, это позволяет генерировать машинный код, специализированный под конкретный запрос, и использовать модель выполнения, которая не накладывает дополнительных расходов во время выполнения. Этот подход позволяет добиться более эффективного использования процессора, сохранив при этом общую архитектуру СУБД и её подсистем, изменив только модуль вычисления запросов. Кроме того, динамическая компиляция открывает новые возможности для оптимизации, связанные с подстановкой констант и вычислением арифметических выражений, традиционно выполняемом при помощи интерпретации.

В работе [1,2] описывается алгоритм генерации эффективного машинного кода для запросов к реляционной СУБД на языке SQL с использованием компиляторной инфраструктуры LLVM [3]. Аргументируется отказ от модели итераторов и приводятся экспериментальные данные, согласно которым использование динамической компиляции запросов позволяет добиться ускорения в 2 раза, а замена модели выполнения на модель явных циклов (data-centric) – ещё в 3–4 раза. Данные методы реализованы в коммерческой реляционной СУБД с закрытым исходным кодом HyPer [4].

Для СУБД PostgreSQL [5] разработано коммерческое расширение Vitesse DB [6] с закрытым исходным кодом, в котором реализована динамическая компиляция запросов с использованием LLVM. На запросе Q1 из набора тестов TPC-H [7] компиляция предикатов позволила получить ускорение в 2 раза, а компиляция всего запроса в одну функцию – ускорение в 8 раз. Дальнейшая оптимизация с привлечением параллелизма и колоночного хранилища позволила получить ускорение до 180 раз.

В [8] описывается метод компиляции выражений в PostgreSQL в машинный код с использованием инфраструктуры LLVM. В работе приведены результаты профилирования СУБД на тестовом наборе TPC-H, согласно которым вычисление арифметических выражений занимает от 32% до 70% от общего времени выполнения запроса. Исследователи не затрагивают вопросы оптимизации операторов и вызовов функций PostgreSQL, но предлагаемый в работе метод компиляции позволяет избежать повторной загрузки атрибутов из кортежа за счёт переиспользования регистров LLVM. На наборе тестов TPC-H представленный метод позволил получить ускорение до 37%.

В данной работе рассматривается динамическая компиляция SQL-запросов для СУБД PostgreSQL с помощью компиляторной инфраструктуры LLVM.

2. Динамическая компиляция запросов в PostgreSQL

Основной алгоритм выполнения SQL-запроса в реляционных СУБД состоит из четырёх этапов. На первом этапе СУБД выполняет лексический и синтаксический анализ SQL запроса и строит дерево разбора. На следующем этапе процедура преобразования принимает от анализатора дерево разбора и выполняет семантический анализ с дальнейшим построением дерева запроса.

На третьем этапе на основе дерева запроса составляется план выполнения запроса путем выбора наиболее эффективного пути выполнения. Итоговый план является наиболее эффективным с точки зрения имеющихся оценок затрат на его выполнение.

Финальным этапом является выполнение плана, которое реализовано при помощи модели итераторов, также известной как Volcano Style Processing [9]. Модель итераторов подробно описана в разделе 2.1.

JIT-компилятор запросов для PostgreSQL, о котором идёт речь в этой статье, реализован с использованием компиляторной инфраструктуры LLVM. Инфраструктура LLVM предоставляет богатый API для анализа и оптимизации программ, а во встроенном модуле MCJIT [10] реализованы механизмы для машинно-зависимой оптимизации и динамической генерации кода под различные платформы. В LLVM используется низкоуровневое типизированное платформонезависимое промежуточное представление LLVM IR, основанное на SSA-форме.

JIT-компилятор реализован в виде расширения к СУБД. Механизм расширений в PostgreSQL предоставляет весьма широкие возможности: при помощи расширений можно определять новые типы данных, типы индексов (access methods), новые функции и операторы для использования в SQL-запросах, а также перехватывать управление на определённых этапах обработки запроса при помощи регистрации функций-обработчиков.

Во время загрузки расширение регистрирует обработчик выполнения запроса, который вызывается после этапа оптимизации непосредственно перед выполнением плана. В обработчике проверяется, поддерживаются ли все

операторы, функции и выражения, используемые в запросе, в случае чего производится динамическая компиляция и выполнение кода, оптимизированного под конкретный запрос.

При разработке JIT-компилятора было решено переписать с использованием LLVM C API основные операторы PostgreSQL. Несмотря на несколько возросшую сложность, такое переписывание позволило:

- пересмотреть используемую вычислительную модель – замена абстракции модели итераторов (Volcano-модели) на абстракцию, более подходящую для генерации кода под конкретный запрос (разделы 2.1, 2.2);
- динамически компилировать и оптимизировать код вычисления арифметических выражений и предикатов (раздел 2.3);
- спроектировать и реализовать ряд оптимизаций, возможных только в динамически компилируемом окружении [11].

2.1 Переход к альтернативной модели выполнения запросов

В используемой в большинстве современных СУБД, в том числе PostgreSQL, Volcano-модели, также известной как *pull-based*, каждый оператор реализуется при помощи итератора с интерфейсом *open(), next(), close()*. Метод *open()* инициализирует внешние ресурсы, такие как память и открытие файла, а метод *close()* освобождает их.

Выполнение плана запроса происходит рекурсивно сверху вниз по дереву, при этом каждый узел в дереве плана вызывает метод *next()* от узлов ниже для получения входных данных, обрабатывает и возвращает один кортеж на узел выше. Вызов метода *next()* каждого оператора в дереве запроса производит один новый кортеж, полученный путем рекурсивного вызова метода *next()* от дочерних операторов. Таким образом, для каждого конкретного запроса операторы в Volcano-модели организуются в конвейер, в котором поток данных управляется корневым оператором запроса, через цепочку вызовов *next()* продвигающим циклы сканирования на следующую итерацию.

Описанная модель позволяет упростить реализацию и поддержку реляционных операторов за счёт простоты абстракции итератора, но приводит к существенным накладным расходам, проявляющимся даже на простых запросах. Во-первых, метод *next()* реализован с помощью неявного вызова функции, который представляет из себя косвенный переход, и, как правило, вызывает ошибочное прогнозирование перехода (branch misprediction), что может привести к большим накладным расходам.

Во-вторых, для оператора сканирования таблиц необходимо сохранение состояния между вызовами *next()*, что означает, что для каждого считываемого из таблицы кортежа необходимо вначале загрузить переменные состояния, в том числе счётчики циклов, и только потом продолжить

выполнение с нужной итерации, при этом записав обновлённые значения переменных для последующих вызовов.

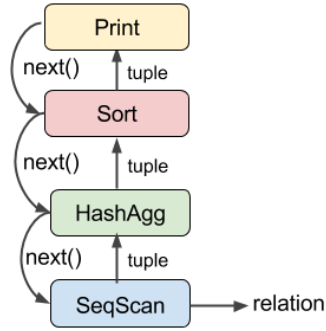


Рис. 1. Модель итераторов для запроса

Fig. 1. Iterator model for example query

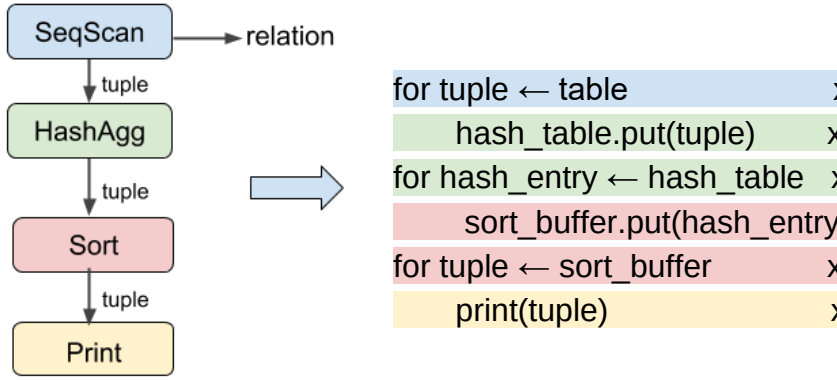


Рис. 2. Push-based модель для запроса

Fig. 2. Push-based model for example query

В плане выполнения запроса на примере СУБД PostgreSQL на рис. 1 для запроса вида select <columns> from <table> group by <column> order by <column>; узел Print будет обращаться за входными данными к дочернему узлу Sort, который, в свою очередь, будет обращаться к своему дочернему узлу HashAgg, который, в свою очередь, обратится к узлу SeqScan, представляющему последовательное чтение таблицы. В результате выполнения узла SeqScan исполнитель выберет одну строку из таблицы и вернет её вызывающему узлу HashAgg и так по цепочке до узла Print.

Альтернативой pull-based модели итераторов является модель явных циклов, также называемая push-based моделью (рис. 2).

Реализация этой модели позволяет представить счётчики циклов и другие переменные состояния локальными переменными на стеке или регистрах процессора и загружать их только при необходимости, но является существенно более сложной в реализации в рамках интерпретатора запросов.

Для JIT-компилятора запросов была выбрана именно push-based модель. Её реализации посвящен следующий раздел.

2.2 Реализация push-based модели в динамическом компиляторе

В конструкции предлагаемого в данной статье динамического компилятора запросов с использованием push-based модели генерация кода выполняется во время обхода дерева плана в прямом порядке, во время которого для каждого оператора вызываются функции *consume()* и *finalize()* и сгенерированные ими LLVM-функции передаются дочерним операторам запроса. Для каждого оператора соответствующие функции *consume()* и *finalize()* реализованы с использованием LLVM C API и вызываются для генерации реализующего его алгебраическую модель кода на LLVM IR, в котором функция *llvm.consume* родительского оператора вызывается для каждого результирующего кортежа, а *llvm.finalize* – после формирования последнего результирующего кортежа.

Таким образом, после обхода дерева плана сгенерированный код будет состоять из нескольких циклов, самым первым из которых является цикл одного из операторов сканирования таблицы.

Для запроса, приведённого в разделе 2.1, для генерации кода на языке LLVM IR в JIT-компиляторе будут вызваны функции *Sort.consume* и *Sort.finalize*, результат выполнения которых будет использован в функциях *HashAgg.consume* и *HashAgg.finalize*, результат выполнения которых, в свою очередь, будет использован в функции SeqScan при генерации внешнего цикла по таблице (см. рис. 3).

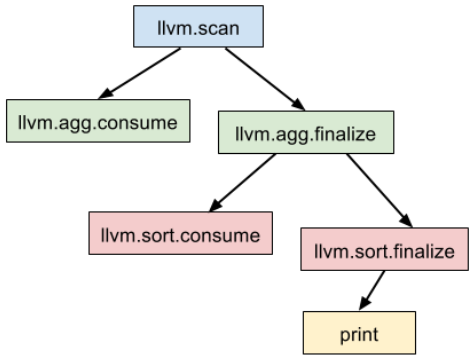
```
llvm.sort.consume = Sort.consume()
llvm.sort.finalize = Sort.finalize(print, null)
    llvm.agg.consume = HashAgg.consume()
    llvm.agg.finalize = HashAgg.finalize(llvm.sort.consume, llvm.sort.finalize)
        llvm.scan = SeqScan(llvm.agg.consume, llvm.agg.finalize)
```

Рис. 3. Цепочка вызовов функций-генераторов кода

Fig. 3. Calls to generator functions

Граф вызовов для сгенерированных LLVM-функции представлен на рис. 4 слева, а сами LLVM-функции – на рис. 4 справа. Предложенный подход позволяет избавиться от неявных вызовов функций и сохранения состояния

между вызовами. После встраивания в *llvm.scan* вызываемых ею функций (рис. 4, справа) получается код, эквивалентный показанному на рис. 2.




```
llvm.sort.consume(tuple) {
    sort_buffer.put(tuple)
}
llvm.sort.finalize() {
    for tuple ← sort_buffer
        print(tuple)
}
llvm.agg.consume(tuple) {
    hash_table.put(tuple)
}
llvm.agg.finalize() {
    for hash_entry ← hash_table
        llvm.sort.consume(tuple)
    llvm.sort.finalize()
}
llvm.scan() {
    for tuple ← table
        llvm.agg.consume(tuple)
    llvm.agg.finalize()
}
```

Рис. 4. Слева – граф вызовов LLVM-функций для запроса, справа – псевдокод LLVM-функций

Fig. 4. On the left: call graph for LLVM functions for example query, on the right: generated functions in pseudo-code

2.3 Динамическая компиляция выражений

Для вычисления выражений PostgreSQL выполняет интерпретацию дерева выражений, где каждое выражение состоит из дерева отдельных операторов и функций. Каждая вершина дерева вызывает функции соответствующих дочерних вершин неявным образом, через указатель на функцию. Это приводит к большим накладным расходам во время выполнения. Неявные вызовы не позволяют выполнять оптимизацию встраивания функций (inlining), тем самым ограничивая возможности компилятора для дальнейшей оптимизации.

Поскольку во время выполнения доступна информация о вызываемых функциях и операциях, можно использовать кодогенерацию для замены неявных вызовов функций на явные, которые в дальнейшем могут быть встроены.

Для динамической компиляции выражений производится рекурсивный обход дерева выражений в обратном порядке и вызов функций-генераторов для генерации кода операций на языке LLVM IR.

В результате этого обхода генерируется код в виде функции на языке LLVM IR (ExecQual на рис. 5). С дальнейшим использованием оптимизации встраивания функций, код для дерева выражений становится линейным и

может быть динамически скомпилирован и выполнен без каких-либо накладных расходов на неявные вызовы функций.

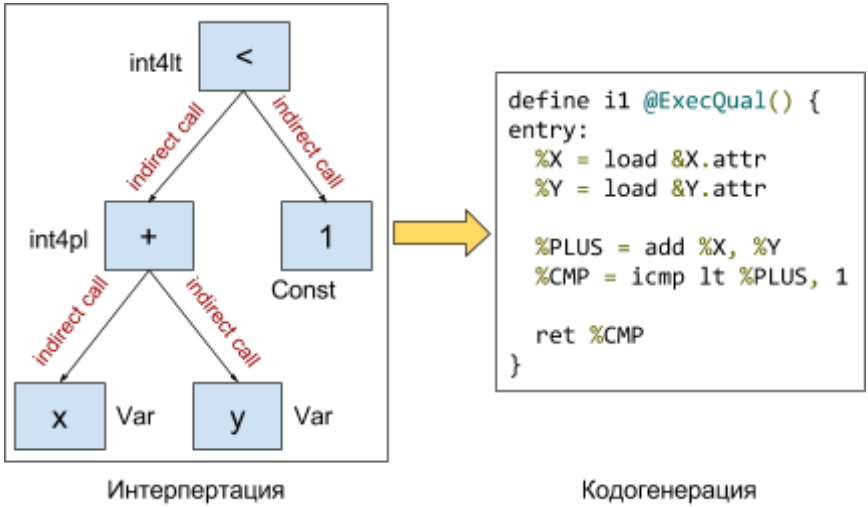


Рис. 5. Слева – интерпретация дерева выражений, справа – сгенерированный LLVM IR

Fig. 5. Left – interpretation of an expression tree, right – generated LLVM IR

Для вычисления операций вызываются встроенные функции PostgreSQL. Для получения функций-генераторов встроенных функций на языке LLVM IR был разработан метод предварительной компиляции с использованием библиотеки CPPBackend из состава LLVM (до версии 3.8), которая переводит LLVM-биткод в соответствующий код на языке C++, использующий функции LLVM C++ API для генерации модуля исходного кода LLVM IR.

Метод работает следующим образом: множество файлов исходного кода PostgreSQL, содержащих встроенные функции, с помощью компилятора *clang* [12] транслируется в объектные файлы биткода LLVM, которые компонуются в единый биткод-файл и оптимизируются модульным оптимизатором *opt* [13]. На основе оптимизированного биткода статический компилятор *llc* [14], в котором реализован интерфейс библиотеки CPPBackend (-march=cpp), строит файл на языке C++, содержащий функции-генераторы на LLVM C++ API, вызовы которых генерируют код на языке LLVM IR соответствующих встроенных функций PostgreSQL. Общая схема метода показана на рис. 6.

Стоит отметить, что генерация объектных (биткод) файлов, их компоновка, оптимизация, трансляция в C++-файл и дальнейшая компиляция этого файла происходят один раз во время сборки расширения.

Преимуществами данного метода являются простота и универсальность реализации, упрощенная поддержка, поскольку отпадает необходимость в

ручной реализации каждой встроенной функции и отслеживании изменений в коде PostgreSQL.

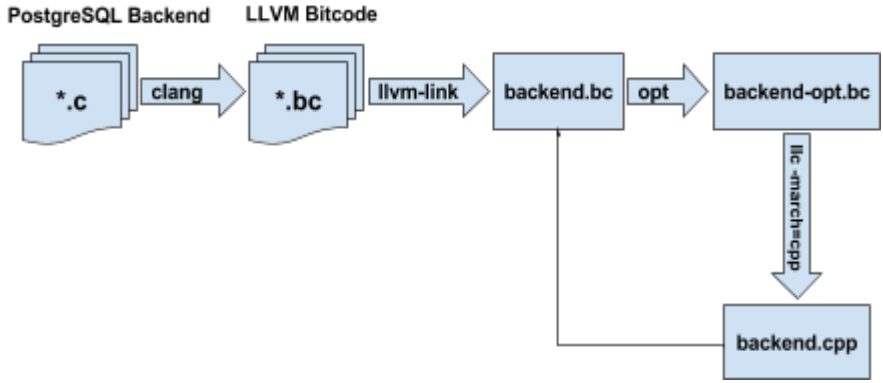


Рис. 6. Схема работы метода предкомпиляции функций PostgreSQL

Fig. 6. Scheme of the PostgreSQL backend function precompilation method

3. Результаты

Для тестирования производительности динамического компилятора (глава 2) использовался набор тестов TPC-H. Этот набор состоит из аналитических запросов и считается достаточно близким к реальным запросам к СУБД.

Тестирование производительности выполнялось на компьютере с 18-ядерным процессором Intel Xeon E7-8890 с тактовой частотой 2.5 ГГц и с 3 терабайтами оперативной памяти под управлением 64-битной операционной системы CentOS Linux release 7.2.1511. Размер базы данных TPC-H – 100 гигабайт, при тестировании база данных располагалась в оперативной памяти. Надо отметить, что типы колонок в таблицах базы данных TPC-H были модифицированы соответствующим образом: тип CHAR(1) был изменен на тип ENUM, тип NUMERIC на DOUBLE PRECISION. Данная модификация позволяет использовать встроенные типы LLVM во время динамической компиляции.

Табл. 1. Сравнение времени выполнения JIT компилятора (глава 2) на тестовом наборе TPC-H.

Table 1. Comparison of execution times of JIT compiler (chapter 2) on TPC-H benchmark.

TPC-H 100 Гбайт	Q1	Q3	Q6	Q13	Q14	Q17	Q19	Q22
PG, сек	431,81	212,06	112,52	252,17	127,36	163,56	9,03	16,47
JIT, сек	100,52	103,38	36,71	175,93	44,43	100,4	7,07	15,29
Ускорение, раз	**4,30**	**2,05**	**3,07**	1,43	**2,87**	1,63	1,28	1,08

Время выполнения измерялось путём многократного выполнения запроса и подсчёта медианы полученных результатов. Результаты тестирования для некоторых запросов из тестового набора TPC-H отражены в табл. 1. Таким образом, время выполнения запроса Q1 сократилось в 4,3 раза с использованием динамического компилятора запросов по сравнению с версией PostgreSQL 9.6 Beta 2 с отключенным параллелизмом.

Тестирование запросов Q6 и Q14 проводилось с отключенными операторами PostgreSQL BitmapHeapScan, Material и MergeJoin, так как данные операторы на момент написания статьи не были реализованы в динамическом компиляторе запросов.

4. Заключение

В данной работе рассмотрен метод динамической компиляции запросов как одно из средств, позволяющих значительно увеличить производительность СУБД на запросах, скорость обработки которых в первую очередь определяется эффективностью использования процессора.

Метод применён к существующей СУБД PostgreSQL. Результаты проведенного тестирования показывают, что динамическая компиляция запросов с помощью JIT-компилятора LLVM позволяет получить ускорение в несколько раз на тестах из набора TPC-H.

В данной работе были разработаны и реализованы LLVM-аналоги основных операторов PostgreSQL, была заменена абстракция модели итераторов (open(), next(), close()) на абстракцию, более подходящую для генерации кода под конкретный запрос и позволяющую реализовывать новые операторы и совмещать несколько операторов в рамках одного запроса. Изменение модели выполнения в сочетании с применением динамической компиляции позволило получить более эффективный код.

Исходный код динамического компилятора выражений, описанного в разделе 2.3, опубликован в открытом доступе (open source) [15]. Компилятор запросов, включающий, помимо компиляции выражений, также измененную модель выполнения запроса (разделы 2.1, 2.2), находится в стадии подготовки к публикации.

Список литературы

[1]. Neumann T. Efficiently compiling efficient query plans for modern hardware. Proc. VLDB Endow. , vol. 4, no. 9, pp. 539–550, Jun. 2011.

[2]. Neumann T., Leis V. Compiling Database Queries into Machine Code. IEEE Data Engineering Bulletin, March 2014.

[3]. Lattner C. LLVM: An Infrastructure for Multi-Stage Optimization. Master's thesis, Computer Science Dept., University of Illinois at Urbana-Champaign, Urbana, IL.

[4]. HyPer – A Hybrid OLTP&OLAP High Performance DBMS, http://www.hyper-db.de

[5]. PostgreSQL, an open source object-relational database system. https://www.postgresql.org
[6]. Vitesse DB, сайт проекта. http://vitessedata.com/vitesse-db
[7]. TPC-H, an ad-hoc, decision support benchmark. http://www.tpc.org/tpch/
[8]. Butterstein D., Grust T. Precision Performance Surgery for PostgreSQL – LLVM-based Expression Compilation, Just in Time. Proceedings of the 42nd Int'l Conference on Very Large Databases (VLDB 2016), New Delhi, India, August 2016.
[9]. Graefe G. Volcano – an extensible and parallel query evaluation system. IEEE Trans. Knowl. Data Eng., 6(1): 120–135, 1994.
[10]. MCJIT Design and Implementation. http://llvm.org/docs/MCJITDesignAndImplementation.html
[11]. Шарыгин Е.Ю., Бучацкий Р.А., Скворцов Л.В., Жуйков Р.А., Мельник Д.М. Динамическая компиляция выражений в SQL-запросах для СУБД PostgreSQL. Труды ИСП РАН, том 28, вып. 4, 2016, стр. 217-240. DOI: 10.15514/ISPRAS-2016-28(4)-13
[12]. clang – a C language family frontend for LLVM, http://clang.llvm.org/
[13]. opt – modular LLVM optimizer and analyzer, http://llvm.org/releases/3.7.1/docs/CommandGuide/opt.html
[14]. llc – LLVM static compiler, http://llvm.org/releases/3.7.1/docs/CommandGuide/llc.html
[15]. PostgreSQL с динамической компиляцией выражений – исходные коды проекта на сайте github.com. https://github.com/ispras/postgres/tree/llvm-expressions

Dynamic compilation of SQL queries for PostgreSQL

[1] *R.A. Buchatskiy <ruben@ispras.ru>*
[1,2] *E.Y. Sharygin <eush@ispras.ru>*
[2] *L.V. Skvortsov <leonidxo@gmail.com>*
[1] *R.A. Zhuykov <zhroma@ispras.ru>*
[1] *D.M. Melnik <dm@ispras.ru>*
[3] *R.V. Baev <baev@ispras.ru>*

[1] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, CMC Department bldg. 52, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*
[3] *Moscow Institute of Physics and Technology (State University) 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia*

Abstract. In recent years, as performance and capacity of main and external memory grow, performance of database management systems (DBMSes) on certain kinds of queries is more determined by raw CPU speed. Currently, PostgreSQL uses the interpreter to execute SQL queries. This yields an overhead caused by indirect calls to handler functions and runtime checks, which could be avoided if the query were compiled into native code "on-the-fly", i.e. just-in-time (JIT) compiled: at run time the specific table structure is known as well as data types and built-in functions used in the query as well as the query itself. This is especially

important for complex queries, performance of which is CPU-bound. We have developed a PostgreSQL extension that implements SQL query JIT compilation using LLVM compiler infrastructure. In this paper we show how to implement LLVM-analogues of the main operators of the PostgreSQL, how to replace Volcano iterator model abstraction (open(), next(), close()) by the abstraction that is more suitable to generate code for a particular query. Currently, with LLVM JIT we achieve up to 4.3x speedup on TPC-H Q1 query as compared to original PostgreSQL interpreter.

Keywords: dynamic compilation; just-in-time compilation; database management system engines; PostgreSQL; LLVM; query languages.

DOI: 10.15514/ISPRAS-2016-28(6)-3

For citation: Buchatskiy R.A., Sharygin E.Y., Skvortsov L.V., Zhuykov R.A., Melnik D.M., Baev R.V. Dynamic compilation of SQL queries for PostgreSQL. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 37-48 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-3

References

[1]. Neumann T. Efficiently compiling efficient query plans for modern hardware. Proc. VLDB Endow. , vol. 4, no. 9, pp. 539–550, Jun. 2011.
[2]. Neumann T., Leis V. Compiling Database Queries into Machine Code. IEEE Data Engineering Bulletin, March 2014.
[3]. Lattner C. LLVM: An Infrastructure for Multi-Stage Optimization. Master's thesis, Computer Science Dept., University of Illinois at Urbana-Champaign, Urbana, IL.
[4]. HyPer – A Hybrid OLTP&OLAP High Performance DBMS, http://www.hyper-db.de
[5]. PostgreSQL, an open source object-relational database system. https://www.postgresql.org
[6]. Vitesse DB website. http://vitessedata.com/vitesse-db
[7]. TPC-H, an ad-hoc, decision support benchmark. http://www.tpc.org/tpch/
[8]. Butterstein D., Grust T. Precision Performance Surgery for PostgreSQL – LLVM-based Expression Compilation, Just in Time. Proceedings of the 42nd Int'l Conference on Very Large Databases (VLDB 2016), New Delhi, India, August 2016.
[9]. Graefe G. Volcano – an extensible and parallel query evaluation system. IEEE Trans. Knowl. Data Eng., 6(1): 120–135, 1994.
[10]. MCJIT Design and Implementation. http://llvm.org/docs/MCJITDesignAndImplementation.html
[11]. Sharygin E.Y., Buchatskiy R.A, Skvortsov L.V., Zhuykov R.A., Melnik D.M. Dynamic compilation of expressions in SQL queries for PostgreSQL. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 4, 2016, pp. 217-240 (in Russian). DOI: 10.15514/ISPRAS-2016-28(4)-13
[12]. clang – a C language family frontend for LLVM, http://clang.llvm.org/
[13]. opt – modular LLVM optimizer and analyzer, http://llvm.org/releases/3.7.1/docs/CommandGuide/opt.html
[14]. llc – LLVM static compiler, http://llvm.org/releases/3.7.1/docs/CommandGuide/llc.html
PostgreSQL with JIT compiler for expressions – project source code at github.com website. https://github.com/ispras/postgres/tree/llvm-expressions

Моделирование семейств программных систем[1]

[1,2] *Е.М. Лаврищева <lavr@ispras.ru>*
[1,3,4] *А.К.Петренко <petrenko@ispras.ru>*
[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] *Московский физико-технический институт (гос. университет),*
141701, Московская область, г. Долгопрудный, Институтский переулок, д.9
[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[4] *Национальный исследовательский университет Высшая школа экономики,*
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Большие программные системы всегда создаются достаточно долго, в несколько этапов, это приводит к необходимости появления версий (релизов) систем. Кроме того, у большой системы всегда есть несколько, а иногда много, конфигураций установки, что обуславливается либо разным набором программно-аппаратного окружения, либо требованиями пользователя системы. Тем самым результатом разработки можно рассматривать не отдельную систему, а цепочку и семейство программных систем или программных продуктов (Product Lines/Product Families). Появление этого понятия можно рассматривать как развитие методов повышения доли повторно-используемого программного обеспечения (re-use). Однако, в отличие от ранних работ по повторно-используемому ПО исследования по семействам программ рассматривают весь спектр работ и задач создания ПО, то есть не только собственно проектирование и программирование, но и документирование, верификацию, поддержку эксплуатации, в частности, инсталляцию и так далее. Одной из работ, выполняемых в ходе создания семейства программ, является моделирование. Статья рассматривает современные подходы к моделированию семейств программных систем, а также, подробнее, одно из направлений этого исследования — моделирование семейств операционных систем (описывая задачи исследования, поддержанного грантом РФФИ).

Ключевые слова: проверка моделей; уточнение моделей; вариабельность; верификация; тестирование; семейство программных систем; управление моделями; конфигурационная сборка, UML, MDA, MDD, MDE, PIM, PSM.

[1] Работа поддержана грантом РФФИ 16-01-00352.

DOI: 10.15514/ISPRAS-2016-28(6)-4

Для цитирования: Лаврищева Е.М., Петренко А.К. Моделирование семейств программных систем. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 49-64. DOI: 10.15514/ISPRAS-2016-28(6)-4

1. Введение

Теме моделирования программных систем уделяется достаточно много внимания, начиная, по крайней мере, с 70-х годов XX века (в качестве примера можно указать на SDL). Моделирование аппаратных и программно-аппаратных устройств зародилось и активно развивалось практически сразу с возникновением ЭВМ. Новое направление работ по моделированию программных систем возникло в начале 90-х годов, в основном, оно связано с появлением UML (Unified Modeling Language, 1994) [7].

Моделирование в этом языке состоит в построении графовой объектной модели, в вершинах которой находятся отдельные объекты системы, а на дугах отношения (связи) между ними. Элементы модели трансформируются в программный код, который тестируется и документируется. И то, и другое не было принципиально новым. К тому времени было несколько графических или текстово-графических языков моделирования, некоторые языки (например, тот же SDL) позволяли генерировать программный код готовый к выполнению или требующий лишь небольшой доделки. Принципиально новым в UML было рассмотрение широкого спектра задач жизненного цикла программного продукта и, соответственно, многих аспектов моделирования, например, моделирования архитектуры в целях обеспечения гибкости, удобства адаптации системы к различных платформам и к ожидаемым требованиям по расширению функциональности, моделирования сценариев работы пользователя, моделирования поведения собственно системы, ее отдельных компонентов, моделирования протоколов взаимодействия подсистем и так далее [6].

Собственно UML является лишь нотацией предназначенной для построения моделей. На основе этой нотации (или альтернатив) строятся методологии или даже инструментальные комплексы для построения или анализа программных систем с теми или иными целями. Так известным направлением в конце 90-годов стала методология MDA (Model Driven Architecture). В рамках этой методологии основное внимание уделялось выделению консервативного (re-usable) архитектурного каркаса некоторого класса систем (примерами могут быть: интернет-магазин, система заказа билетов и т.д.). На основе такого каркаса строится платформо-независимая модель (PIM), которая потом транслируется в платформо-зависимую — PSM.

Методология MDA не получила тотального распространения, на что рассчитывали компании, которые их активно продвигали. В качестве основных причин можно указать, с одной стороны, на громоздкость

методологии, сложность композиции моделей-каркасов и, с другой стороны, в игнорировании специфики сложившихся технологий разработки программ.

Однако нужно отметить, что, хотя попытки тотального внедрения методов разработки и верификации программ на основе моделей можно признать неудачными, работы по совершенствованию языков и инструментов моделирования, а также новые средства использования моделей развиваются активно. В первую очередь нужно отметить направления MDD (Model Driven Development) и MDE (Model Driven Engineering), которые широко используются в создании встроенных, критических по безопасности систем.

Специфика задач моделирования программных семейств состоит в том, что цель моделирования состоит не только в выделении консервативного ядра системы, но и в четком описании вариантных частей и механизмов управления вариантностью. Цель анализа модели программного семейства состоит в анализе корректности (или других характеристик системы, например, производительности) при выборе того или иного набора характеристик (features) или задании конфигурации компиляции/сборки/инсталляции системы.

Тема моделирования программных семейств активно обсуждается на научных конференциях и в научных журналах. Основным форумом можно назвать конференцию MODELS (1999-2016), а также эта тема постоянно присутствует на конференциях VAMOS (2004 –2015); ICTERI (2006 –2013); SAM (System Analysis and Modeling), ICSR (International Conference on Software Reuse) и др.

2. Моделирование изменяемых систем

Как уже отмечалось, объект и цели моделирования программных продуктов (ПП) отличаются от сущностей, с которыми приходится работать при моделировании собственно программных систем. Далее дается краткое описание подходов к моделированию программных систем, базирующееся на работах SEI (Software Engineering Institute) и работах одного из авторов статьи [1,3,4,5,16].

2.1 Вариабельность продуктов и систем

В SEI (2005) был предложен метод SPLE (Software Product Line Engineering) [1,3] для изготовления семейств программных систем (СПС), основанный на модели вариабельности характеристик (Feature Model - MF) [8,9,11] и модели конфигурационной сборки готовых ресурсов (которые в англоязычной литературе называются как artifacts, components, reuses, assets, services и др.).

К. Похл [2] определил понятие вариабельности в SPLE, как модели MF для элементов системы, помечаемых вариантными точками.

Вариабельность — это способность продукта (системы) к расширению, изменению, приспособлению или конфигурированию с целью использования

в определенном контексте и обеспечения последующей его эволюции. Модель MF формируется в процессе разработки ПП аналитиками и разработчиками и включает общие функциональные и нефункциональные характеристики элементов системы [15]. К главным аспектам обеспечения вариабельности продукта системы относятся:

- моделирование вариабельности на уровне артефактов и целых подсистем с вариантными характеристиками [11];
- управление (планирование, контроль и регуляция) конфигурацией системы по модели архитектуры системы и модели вариабельности [10,18].

Множество значений одной характеристики (*вариантной характеристики*), образует коллекцию. Каждый член коллекции соответствует требованию к разработанным артефактам в ПП и присоединяется к ядру ПП в точках вариантности.

Точка вариантности — это место в системе, в котором осуществляется выбор варианта reuse-компонента. Вариантная характеристика косвенно определяет количество точек вариантности системы, присоединяемых к ядру системы.

Основу концепции изменяемости/вариабельности ПП составляют внешние характеристики и свойства функций системы, которые могут отличаться в разных вариантах ПП. По модели архитектуры системы и MF создается конфигурационный файл системы для инсталляции, запуска и выполнения системы. Совокупность систем и подсистем с общим множеством характеристик и требований к ним образуют ПП. В принципе, может быть поставлена задача получения и перечисления всех членов семейства или определения наличия определенных свойств (например, совместимости, безопасности и т.д.) для всех членов семейства [12,13].

2.2 Модель вариабельности СПС

В SPLE модель вариабельности имеет следующий вид:

$$MF_{var} = (SV; AV),$$

где SV — подмодель вариабельности артефактов системы,
AV — подмодель архитектуры продуктов системы.

Подмодель $SV = ((G_t, TR_t), Con, Dep)$,

где G_t — граф артефактов типа t (требования, компоненты, тесты и др.);
TR_t — связи артефактов типа t;
Con, Dep — предикаты на декартовом произведении множеств артефактов, которые определяют ограничения и зависимости между функциями и показателями качества СПС.

Подмодель $AV = (G, TR, G_t)$,

где G — граф архитектуры системы СПС из артефактов, переиспользуемых элементов сборки - reuses, а также из набора вариантных характеристик;

TR — связь элементов архитектуры и артефактов G_t.

Модель SV конкретизируется в специальную линию разработки ПП и конфигурируется из артефактов и элементов архитектуры в код системы. Точки вариантности позволяют управлять трансформацией элементов графа и заменой одних артефактов другими, новыми функциональными или более корректными элементами архитектуры.

2.3 Управление вариабельностью

Объектами управления вариабельностью в СПС и ПП являются:

- точка вариантности или вариантный артефакт, т.е., его формальное представление и возможность его реализации для разных СПС;
- вариант — элементарный артефакт СПС одного типа с соответствующим ему вариантным артефактом, который представляет собой описание заданного СПС;
- ограничение — предикат, определенный на декартовом произведении декартовых квадратов множества точек вариантности и вариантов;
- зависимость — предикат, который определяет допустимые взаимосвязи между точкой вариантности и множеством вариантов для СПС.

В работах Е. Дьоминга предложен метод управления организацией разработки СПС, в котором выделено четыре функции управления вариабельностью [18]:

1) *F1* — функция планирования вариабельности в артефактах СПС (на уровнях инженерии домена и приложений);
2) *F2* — функция реализации вариабельности в архитектуре СПС;
3) *F3* — функция системного мониторинга вариабельности СПС;
4) *F4* — функция актуализации вариабельной СПС.

В основе управления вариабельностью СПС и ПП лежат требования *R1-R5* к системе:

R1) обоснованность — наличие объективных оснований принятия решений (assets) для *F1-F4*;

R2) согласованность — одинаковость способа выработки и реализации решений на всех уровнях абстракции и на всех этапах процесса разработки СПС;

R3) масштабируемость — независимость способа выработки и реализации этих решений от объема функциональных возможностей СПС;

R4) трассируемость — возможность отслеживания связей между характеристиками вариабельности на всех процессах разработки ПП;

R5) визуализация свойств вариабельности и связей между характеристиками.

Приведенные требования *R1–R5* для функций *F1–F4* обеспечивают модельную среду их реализации. Необходимым элементом этой среды может быть интегрированная модель вариабельности (*VM*) семейства СПС, которая объединяет модели подсистем семейства и позволяет дать оценку уровню вариабельности системы и степени ее соответствия требованиям.

3. Направление работ по моделированию сложных систем ИСП РАН

В 2016 г. РФФИ поддержал проект «Теория и методы разработки вариабельных программных и операционных систем» (16-01-00352) [32].

Цель данного проекта состоит в разработке

- теории и методов моделирования операционных систем и веб-систем, которые рассматриваются как результат сборки из готовых ресурсов (reuses, модулей, объектов, компонентов, аспектов, сервисов и др.) и их вариантов;
- методов извлечения из готовых СПС (например, ОС Linux или другие унаследованных систем) элементов архитектуры и построения новой модели операционной системы и модели вариабельности для управления вариантами ПП и СПС.

Эти два класса систем моделирования данного проекта РФФИ являются разнородными в плане видов ресурсов, используемых в них, и близкими по показателям сложности.

3.1 Направления исследования по проекту

Исследование систем обоих типов и разработка теории и методов построения этих типов систем проводится в проекте по двум направлениям:

1) Анализ формализмов, используемых для описания моделей вариабельности и конфигурации систем различного типа, включающих изменяемые, вариабельные элементы и характеристики системы, набор правил их конфигурации в системы и верификации моделей, артефактов и систем. Результатом анализа является конкретизация модели вариабельности и формализмов описания конфигурационной сборки современных операционных систем и прикладных Веб-систем из готовых ресурсов [16].

2) Исследование методов моделирования, конфигурации и верификации сформированных конфигураций операционных систем и методов извлечения моделей вариабельности с артефактов и приложений, а также извлечения готовых ресурсов из конфигурации операционной системы для построения вариантов таких систем под конкретные

применения и Веб-систем в общем проведено в [16,27]. В качестве операционной системы используется ОС Linux, а в качестве прикладной системы бизнес-система с Веб-сервисов.

Исследование по данному проекту состоит в анализе базовых направлений подходов к моделированию ПП и их вариантов (Grid, WebSphere IBM, MS Product Line, Onion Architecture Web-system и др.), а также в усовершенствовании и определении новых моделей моделирования для заданного класса систем. Авторами предложена новая модель системы, требующая доработки функций системы, внесения изменений и проведения верификации и анализа вариабельности операционных систем по новым точкам вариантности.

3.2 Метод конфигурации

Предлагаемый в рамках данного проекта метод конфигурации обеспечивает:

- выполнение операций задания и пополнения требований, проведение их трассирования на соответствие выполняемым функциям и порождение новых вариантов архитектуры СПС из готовых, заданных точками вариантности и с учетом заданных ограничений;
- подготовку эффективных решений (assets) по разработке новых вариантов артефактов и готовых ресурсов путем анализа опыта, накопленного в базе знаний проекта системы.

Сущность управления конфигурацией состоит в**:**

- конфигурационной сборке ресурсов по моделям систем (контроль, верификация ресурсов, тестов);
- систематическом отслеживании внесенных изменений в отдельные составные элементы конфигурации системы, проведение аудита изменений и автоматизированного контроля изменений;
- поддержке целостности конфигурации и ее аудита;
- ревизии конфигурации методом проверки программных или аппаратных элементов в соответствии с версиями конфигурации и требованиями к системе;
- трассировке изменений в схему конфигурации на этапах сопровождения и эксплуатации систем;
- доказательстве наличия изоморфного отображения объектных и компонентных моделей систем в программные структуры методом трансформации моделей (PIM, PSM) и функций объектов, их данных к платформе выполнения операционной или прикладной системы.

3.3 Анализ, верификация и тестирование вариабельных систем

Методы анализа моделей вариабельности делятся на четыре группы [9]:

- На основе пропозициональной логики [19]. В ней ограничения модели характеристик транслируются в логические формулы, которые затем обрабатываются с помощью решателей и инструментов автоматического доказательства различных видов (SADT, BDD и др.) или инструментов с использованием формальных языков типа VDM, Alloy, Event-B, Z и др.
- На основе онтологий. Этот подход основан на трансляции модели вариабельности в модель онтологии. Например, в [20] производится трансляция в OWL DL (Ontology Web Language Description Logic), обладающее достаточной выразительной способностью. После трансляции используются автоматизированные инструменты анализа онтологий, такие как RACER.
- На основе программирования в ограничениях. В этом подходе ограничения модели вариабельности транслируются в описание задачи CSP (Constraint Satisfaction Problem), которая затем проверяется с помощью существующих инструментов программирования с ограничениями (constraint programming) [21,22].
- На основе проверки моделей. Этот анализ сводится к задачам проверки моделей (model checking), с помощью имеющихся инструментов [23,24] и др. Для ряда языков описания конфигураций, например, Kconfig [14], поиск несогласованностей и недопустимости конфигураций выполняются автоматически только для моделей с ограничениями, не включающими характеристики, не имеющие общего родителя. Пользователи должны вручную отслеживать соблюдение такого рода ограничений. Имеется ряд методов на основе анализа моделей программ (software model checking), использующего автоматическую генерацию абстрактных моделей из кода реальных программ, например, на языке Си с последующей верификацией этих абстрактных моделей и, если необходимо, автоматическую детализацию получаемых моделей. Обзор этих методов можно найти в [30].

Методы верификации вариантных СПС включают методы известных специалистов: Hamlet, McGregor, Kishi и др. [26,27,28], использующие тестирование на основе диаграмм характеристик, проверки модели вариабельности и конфигурационной модели создаваемых систем путем проверки корректности протоколов взаимодействия ресурсов, готовые тесты для отдельных ресурсов и функций. Методы верификации СПС на основе программных моделей развиваются несколькими группами исследователей. В

первую очередь нужно отметить работы группы С. Апеля [31] из университета Пассау.

3.4 Методы извлечения объектов из операционных и наследованных (legacy) систем

Современные операционные системы (ОС), такие как Linux, имеют большой набор разнообразных конфигурационных опций, позволяющих получать варианты системы для различных видов оборудования и в различной комплектации ресурсов. Например, в ядре Linux имеется несколько тысяч конфигурационных опций. В версии 4.2 это число составило 7872 опции (архитектура x86_64). В другой операционной системе FreeBSD, для версии v.8.0.0 определено 1 203 опции (все архитектуры).

Сложность управления многообразием конфигураций ОС довольно высока, что часто приводит к чисто конфигурационным ошибкам[2] [25,29], связанным, например, с включением несовместимых опций или исключением необходимой для заданной опции соответствующего окружения. Для более эффективного решения задач управления конфигурациями существуют специализированные формализмы и языки, позволяющие описывать многообразие конфигураций, а также правила корректного построения конфигураций. Вместе с тем, имеющиеся механизмы управления конфигурациями не дают непосредственной возможности представить и анализировать модель вариабельности в целом. Во многом проблема управления конфигурациями типична не только для операционных систем, но для многих наследованных систем.

В рамках исследований, поддержанных грантом РФФИ, планируется в первую очередь решить три следующие задачи:

- собрать информацию о точках вариабельности ОС Linux и провести их классификацию;
- провести исследование характерных ошибок, которые можно отнести к типу «конфигурационных ошибок»;
- разработать методы и соответствующие инструменты для обнаружения конфигурационных ошибок.

Кроме того, ставится общая задача построения общей модели вариабельности ОС Linux, объединяющей модель архитектуры системы и модель ее характеристик.

[2] «конфигурационными ошибками» называют программные конструкции, которые являются корректными в одних конфигурациях и некорректными в других.

4. Заключение

В статье рассмотрены подходы к моделированию семейств программных систем. Рассмотрены основные задачи фундаментального проекта «Теория и методы разработки вариабельных программных и операционных систем» (16-01-00352). Приведен анализ и дана характеристика основных подходов к моделированию семейств программных систем. Сформулированы задачи моделирования, верификации и анализа современных СПС с учетом их вариабельности и конфигурирования из готовых ресурсов. Поставлена задача извлечения модели вариабельности ядра операционной системы Linux и рассмотрены варианты постановки задачи верификации СПС Linux.

Авторы выражают глубокую признательность В.В. Кулямину за обсуждения и помощь в подготовке данной статьи.

Список литературы

[1]. Clements P., Northrop L. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering, Addison-Wesley, 2001. ISBN-13: 978-0201703320.

[2]. Pohl K., Böckle G., van der Linden F. J. Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag, 2005. DOI: 10.1007/3-540-28901-1.

[3]. Bachmann F., Clements P. Variability in software product lines. CMU/SEI Technical Report CMU/SEI-2005-TR-012, 2005.

[4]. Лаврищева Е. М., Коваль Г.И., Слабоспицькая О.О., Колесник А.Л. Особенности процессов управления при создании семейств программных систем. Проблемы программирования, (3):40-49, 2009.

[5]. Лаврищева Е.М., Слабоспицькая О.А., Коваль Г.И., Колесник А.А. Теоретические аспекты управления вариабельностью в семействах программных систем. Весник КНУ, серия физ.–мат. наук, (1):151-158, 2011.

[6]. Jacobson I., Griss M., Jonsson P. Software Reuse, Architecture, Process and Organization for Business Success. Addison-Wesley, 1997. ISBN-13: 978-0201924763.

[7]. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. М.: ДМК, 2000, 432 с.

[8]. Kang K., Cohen S., Hess J., Novak W., Peterson S. Feature-oriented domain analysis (FODA) feasibility study. CMU/SEI Technical Report CMU/SEI-90-TR-21, 1990.

[9]. Benavides D., Segura S., Ruiz-Cortés A. Automated analysis of feature models 20 years later: a literature review. Information Systems, 35(6):615–636, 2010. DOI: 10.1016/j.is.2010.01.001.

[10]. Chen L., Babar M.A. A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology, 53(4):344–362, 2011. DOI: 10.1016/j.infsof.2010.12.006.

[11]. Berger T., She S., Lotufo R., Wąsowski A., Czarnecki K. A study of variability models and languages in the systems software domain. IEEE Transactions on Software Engineering, 39(12):1611-1640, 2013. DOI: 10.1109/TSE.2013.34.

[12]. Thüm T., Apel S., Kästner C., Kuhlemann M., Schaefer I., Saake G. A classification and survey of analysis strategies for software product lines. ACM Computing Surveys, 47(1):article 6, 2014. DOI: 10.1145/2580950.

[13]. Liebig J., von Rhein A., Kästner C., Apel S., Dörre J., Lengauer C. Scalable analysis of variable software. Proceedings of the 2013 9-th Joint Meeting on Foundations of Software Engineering, pp. 81-91. ACM, 2013. DOI: 10.1145/2491411.2491437.

[14]. She S., Berger T. Formal semantics of the Kconfig language. Technical note, University of Waterloo, 2010.
https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt.

[15]. Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и JAVA RMI. М.: Мир, 2002. 510с.

[16]. Лаврищева Е.М. Теория объектно-компонентного моделирования изменяемых программных систем. ИСП РАН, 2015.
http://www.ispras.ru/preprints/docs/prep_29_2015.pdf.

[17]. Berger T., Rublack R., Nair D., Atlee J.M., Becker M., Czarnecki K., Wąsowski A. A survey of variability modeling in industrial practice. Proc. of the 7-th Intl. Workshop on Variability Modelling of Software-intensive Systems (VaMoS'2013), article No. 7, ACM 2013. DOI: 10.1145/2430502.2430513.

[18]. Лаврищева Е.М. Программная инженерия. Парадигмы, технологии, CASE-средства. Изд. Юрайт, М.: 2015, 280 с.

[19]. Batory D. Feature models, grammars, and propositional formulas. Proc. of the 9-th Intl. Conf. on Software Product Lines (SPLC'05), LNCS 3714, pp. 7-20, 2005. DOI: 10.1007/11554844_3.

[20]. Wang H., Li Y., Sun J., Zhang H., Pan J. A semantic web approach to feature modeling and verification. Proc. of Workshop on Semantic Web Enabled Software Engineering (SWESE'05), p. 44, 2005.

[21]. Benavides D., Segura S., Trinidad P., Ruiz-Cortés A. Using Java CSP solvers in the automated analyses of feature models. Generative and Transformational Techniques in Software Engineering, LNCS 4143:399-408. Springer, 2006. DOI: 10.1007/11877028_16.

[22]. White J., Doughtery B., Schmidt D., Benavides D. Automated reasoning for multi-step software product-line configuration problems. Proc. of the 13-th Sofware Product Line Conference, pp. 11-20, 2009.

[23]. Zhang W., Mei H., Zhao H. Feature-driven requirement dependency analysis and high-level software design. Requirements Engineering, 11(3):205-220, 2006. DOI: 10.1007/s00766-006-0033-x.

[24]. Dietrich C., Tartler R., Schröder-Preikshat W., Lohmann D. Understanding Linux feature distribution. Proceedings of the 2012 Workshop on Modularity in Systems Software, pp. 15-20. ACM, 2012. DOI: 10.1145/2162024.2162030.

[25]. Melo J., Flesborg E., Brabrand C., Wąsowski A. A quantitative analysis of variability warnings in Linux. Proceedings of the 10-th International Workshop on Variability Modelling of Software-intensive Systems, pp. 3-8. ACM, 2016. DOI: 10.1145/2866614.2866615.

[26]. Sayyad A. S., Ingram J., Menzies T., Ammar H. Scalable product line configuration: a straw to break the camel's back. Proc. of IEEE/ACM 28-th International Conference on Automated Software Engineering (ASE 2013), pp. 465-474. IEEE, 2013. DOI: 10.1109/ASE.2013.6693104.

[27]. Nguyen T., Colman A., Han J. A web services variability description language (WSVL) for business users oriented service customization. Proc. Int. Workshop on User–Focused Service Engineering, Consumption and Aggregation (USECA 2011).

[28]. Meinicke J., Thüm T., Schröter R., Benduhn F., Saake G. An overview on analysis tools for software product lines. Proc. of the 18-th Intl. Software Product Line Conf.: Companion Vol. for Workshops, Demonstrations and Tools, vol. 2, pp. 94-101. ACM, 2014. DOI: 10.1145/2647908.2655972.

[29]. Hemakumar A. Finding Contradictions in Feature Models. Proc. of 12-th Intl. Conf. on Software Product Lines (SPLC'2008), v. 2, pp. 183-190, 2008.

[30]. Бейер Д., Петренко А.К. Верификация драйверов операционной системы Linux. Труды ИСП РАН, Том 23, 2012 г., стр. 405-412. DOI: 10.15514/ISPRAS-2012-23-23.

[31]. Apel S., Beyer D., et al. Strategies for product-line verification: case studies and experiments. Proceeding ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, pp. 482-491.

[32]. Кулямин В.В., Лаврищева Е.М., Мутилин В.С., Петренко А.К. Верификация и анализ вариабельных операционных систем. Труды ИСП РАН, 28(3):189-208, 2016. DOI: 10.15514/ISPRAS-2016-28(3)-12.

Software Product Lines Modeling

[1,2] *K.M. Lavrischeva, <lavr@ispras.ru>*
[1,3,4] *A.K. Petrenko <petrenko@ispras.ru>*
[1] *Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, Russia, 109004.
[2] *Moscow Institute of Physics and Technology,*
9, Institutsky per., Dolgoprudny, Moscow Region, Russia, 141701.
[3] *Lomonosov Moscow State University.*
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
[4] *NIU HSE, Miasnitskaya, 20, Moscow, Russia.*

Absract. Complex software systems always exist for a long time, sometimes changing, and this leads to a variety of versions of such a system. In additional complex software systems usually have different (sometimes a lot) configurations due to different hardware and software environments, where they are intended to operate, or due to different user types with specific requirements. So, a complex software system can be regarded more correctly as a software system family or a software product line. Taking software families in consideration helps to increase reuse of their components and other software development artifacts. In difference with earlier works on software reuse, mostly focused on code or design reuse, software system family development tries to expand reuse on all kinds of development artifacts and activities, including documentation, verification, operation support, deployment, etc. One of the software system family development activities is modeling of family variability. This paper considers modern methods and approaches to such modeling, especially focusing on modeling of operating systems families variability. The research, which results are presented in this paper, is supported by RFBR.

Keywords: model checking; model refinement; variability; verification; testing; software system family; model management; configuration building, UML, MDA, MDD, MDE, PIM, PSM.

DOI: 10.15514/ISPRAS-2016-28(6)-4

For citation: Lavrischeva K.M., Petrenko A.K. Software Product Lines Modeling. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 49-64 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-4

References

[1]. Clements P., Northrop L. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering, Addison-Wesley, 2001. ISBN-13: 978-0201703320.

[2]. Pohl K., Böckle G., van der Linden F. J. Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag, 2005. DOI: 10.1007/3-540-28901-1.

[3]. Bachmann F., Clements P. Variability in software product lines. CMU/SEI Technical Report CMU/SEI-2005-TR-012, 2005.
[4]. Lavrischeva E.M., Koval' G.I., Slabospitskaya O.O., Kolesnik A.L. Product Line Development Management Specifics. Problemy programmiovaniya [Problems of Software Developement], (3):40-49, 2009 (in Ukrainian).
[5]. Lavrischeva E.M., Slabospitskaya O.O., Koval' G.I., Kolesnik A.L. Theoretical Aspects of Variability Management in Product Lines. Vesnik KNU seria fiz.-mat. nauk [Notes of KNU, series on maths and physics], (1):151-158, 2011 (in Ukrainian).
[6]. Jacobson I., Griss M., Jonsson P. Software Reuse, Architecture, Process and Organization for Business Success. Addison-Wesley, 1997. ISBN-13: 978-0201924763.
[7]. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language Reference Guide. 2-nd edition. Addison-Wesley Professional, 2005.
[8]. Kang K., Cohen S., Hess J., Novak W., Peterson S. Feature-oriented domain analysis (FODA) feasibility study. CMU/SEI Technical Report CMU/SEI-90-TR-21, 1990.
[9]. Benavides D., Segura S., Ruiz-Cortés A. Automated analysis of feature models 20 years later: a literature review. Information Systems, 35(6):615–636, 2010. DOI: 10.1016/j.is.2010.01.001.
[10]. Chen L., Babar M.A. A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology, 53(4):344–362, 2011. DOI: 10.1016/j.infsof.2010.12.006.
[11]. Berger T., She S., Lotufo R., Wąsowski A., Czarnecki K. A study of variability models and languages in the systems software domain. IEEE Transactions on Software Engineering, 39(12):1611-1640, 2013. DOI: 10.1109/TSE.2013.34.
[12]. Thüm T., Apel S., Kästner C., Kuhlemann M., Schaefer I., Saake G. A classification and survey of analysis strategies for software product lines. ACM Computing Surveys, 47(1):article 6, 2014. DOI: 10.1145/2580950.
[13]. Liebig J., von Rhein A., Kästner C., Apel S., Dörre J., Lengauer C. Scalable analysis of variable software. Proceedings of the 2013 9-th Joint Meeting on Foundations of Software Engineering, pp. 81-91. ACM, 2013. DOI: 10.1145/2491411.2491437.
[14]. She S., Berger T. Formal semantics of the Kconfig language. Technical note, University of Waterloo, 2010. https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt.
[15]. Emmerich W. Engineering Distributed Objects. Wiley, 2000.
[16]. Lavrischeva E.M. Theory of object-component modeling of variable software systems. ISPRAS, 2015 (in Russian). www/ispras.ru/preprints/docs/prep_29_2015.pdf.
[17]. Berger T., Rublack R., Nair D., Atlee J.M., Becker M., Czarnecki K., Wąsowski A. A survey of variability modeling in industrial practice. Proc. of the 7-th Intl. Workshop on Variability Modelling of Software-intensive Systems (VaMoS'2013), article No. 7, ACM 2013. DOI: 10.1145/2430502.2430513.
[18]. Lavrischeva E.M. Software engineering. Paradigms, technologies, CASE-tools. Urait, Moscow, 2015 (in Russian).
[19]. Batory D. Feature models, grammars, and propositional formulas. Proc. of the 9-th Intl. Conf. on Software Product Lines (SPLC'05), LNCS 3714, pp. 7-20, 2005. DOI: 10.1007/11554844_3.
[20]. Wang H., Li Y., Sun J., Zhang H., Pan J. A semantic web approach to feature modeling and verification. Proc. of Workshop on Semantic Web Enabled Software Engineering (SWESE'05), p. 44, 2005.

[21]. Benavides D., Segura S., Trinidad P., Ruiz-Cortés A. Using Java CSP solvers in the automated analyses of feature models. Generative and Transformational Techniques in Software Engineering, LNCS 4143:399-408. Springer, 2006. DOI: 10.1007/11877028_16.

[22]. White J., Doughtery B., Schmidt D., Benavides D. Automated reasoning for multi-step software product-line configuration problems. Proc. of the 13-th Sofware Product Line Conference, pp. 11-20, 2009.

[23]. Zhang W., Mei H., Zhao H. Feature-driven requirement dependency analysis and high-level software design. Requirements Engineering, 11(3):205-220, 2006. DOI: 10.1007/s00766-006-0033-x.

[24]. Dietrich C., Tartler R., Schröder-Preikshat W., Lohmann D. Understanding Linux feature distribution. Proceedings of the 2012 Workshop on Modularity in Systems Software, pp. 15-20. ACM, 2012. DOI: 10.1145/2162024.2162030.

[25]. Melo J., Flesborg E., Brabrand C., Wąsowski A. A quantitative analysis of variability warnings in Linux. Proceedings of the 10-th International Workshop on Variability Modelling of Software-intensive Systems, pp. 3-8. ACM, 2016. DOI: 10.1145/2866614.2866615.

[26]. Sayyad A. S., Ingram J., Menzies T., Ammar H. Scalable product line configuration: a straw to break the camel's back. Proc. of IEEE/ACM 28-th International Conference on Automated Software Engineering (ASE 2013), pp. 465-474. IEEE, 2013. DOI: 10.1109/ASE.2013.6693104.

[27]. Nguyen T., Colman A., Han J. A web services variability description language (WSVL) for business users oriented service customization. Proc. Int. Workshop on User–Focused Service Engineering, Consumption and Aggregation (USECA 2011).

[28]. Meinicke J., Thüm T., Schröter R., Benduhn F., Saake G. An overview on analysis tools for software product lines. Proc. of the 18-th Intl. Software Product Line Conf.: Companion Vol. for Workshops, Demonstrations and Tools, vol. 2, pp. 94-101. ACM, 2014. DOI: 10.1145/2647908.2655972.

[29]. Hemakumar A. Finding Contradictions in Feature Models. Proc. of 12-th Intl. Conf. on Software Product Lines (SPLC'2008), v. 2, pp. 183-190, 2008.

[30]. Beyer D., Petrenko A.K. Verification of Linux operating system drivers. Trudy ISP RAN/Proc. ISP RAS, 23:405-412, 2012 (in Russian). DOI: 10.15514/ISPRAS-2012-23-23.

[31]. Apel S., Beyer D., et al. Strategies for product-line verification: case studies and experiments. Proceeding ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, pp. 482-491.

[32]. Kuliamin V.V., Lavrischeva E.M., Mutilin V.S., Petrenko A.K. Verification and analysis of variable operating systems. Trudy ISP RAN/Proc. ISP RAS, 28(3):189-208, 2016 (in Russian). DOI: 10.15514/ISPRAS-2016-1(2)-12.

Конфигурируемый метод поиска состояний гонок в операционных системах с использованием предикатных абстракций[1]

[1]П.С. Андрианов <andrianov@ispras.ru>
[1]В.С. Мутилин <mutilin@ispras.ru>
[1,2,3,4]А.В. Хорошилов <khoroshilov@ispras.ru>
[1]Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, дом 25
[2]Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1.
[3]Московский физико-технический институт (государственный университет),
141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9
[4]Национальный исследовательский университет «Высшая школа экономики»
101000, Россия, Москва, ул. Мясницкая, д. 20

Аннотация. В статье представлен конфигурируемый метод для поиска состояний гонок. Метод позволяет настраивать требуемую точность анализа, выбирая баланс между затрачиваемыми ресурсами и количеством ложных предупреждений подключением двух расширений: уточнением путей на основе предикатных абстракций и анализом потоков. Метод основан на алгоритме Lockset и использует упрощенную модель памяти для уменьшения количества ложных предупреждений. Предлагаемый подход был реализован в инструменте CPALockator, который был апробирован на модулях ядра операционной системы Linux, что позволило обнаружить несколько состояний гонок, которые были признаны и исправлены разработчиками.

Ключевые слова: статический анализ; состояние гонки; ядро операционной системы.

DOI: 10.15514/ISPRAS-2016-28(6)-5

Для цитирования: Андрианов П.С., Мутилин В.С., Хорошилов А.В. Конфигурируемый метод поиска состояний гонок в операционных системах с

[1] Исследования проводились при финансовой поддержке Министерства образования РФ, уникальный идентификатор соглашения RFMEFI61614X0015.

использованием предикатных абстракций. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 65-86. DOI: 10.15514/ISPRAS-2016-28(6)-5

1. Введение

Ошибки, связанные с параллельным выполнением кода, традиционно считаются сложными для поиска и исправления. Наиболее многочисленным классом таких ошибок остается состояние гонки, при котором возможен доступ к одной области памяти одновременно из нескольких потоков [1]. Такая ситуация может привести к серьезным последствиям вплоть до падения всей системы [2, 3].

Существует множество инструментов для автоматизации процесса поиска состояний гонки. Мы не будем останавливаться на динамических инструментах, которые анализируют программу в процессе ее работы. Среди статических инструментов выделяют два класса: легковесные и тяжеловесные. Первые обходятся в своей работе незначительными ресурсами, но при этом демонстрируют невысокий процент истинных ошибок и/или частый их пропуск. Тяжеловесные инструменты позволяют доказывать отсутствие ошибок, но при этом потребляют значительный объем ресурсов, в первую очередь времени. Это приводит к тому, что тяжеловесные инструменты становится невозможно использовать на реальных программных системах.

Основной задачей, стоящей перед нами, была разработка легковесного статического анализатора, пригодного для применения к ядру операционной системы. Для минимизации ложных предупреждений ключевой задачей является повышение точности анализа в тех рамках, чтобы инструмент мог быть использован при верификации сложных программных комплексов, каким является ядро операционной системы ОС Linux. В работе описан легковесный метод, который был успешно расширен тяжеловесными подходами, которые обыкновенно применяются для проверки небольших программ тщательным образом.

Разработанный инструмент был успешно апробирован на драйверах ОС Linux.

2. Основные проблемы статического анализа при поиске состояний гонок

Сначала скажем несколько слов о статическом анализе. Программа моделируется с помощью графа потока управления, состоящего из некоторого множества состояний и переходов между ними, которые связаны с операторами программы. Под состоянием программы понимают состояние ее памяти, в которое включается и программный счетчик, который определяет конкретную точку программы. Задача статического анализа — найти множество достижимых состояний из некоторого заданного состояния с помощью переходов. Если среди всех достижимых состояний есть состояния определенного вида (ошибочные), то считается, что была найдена ошибка. Так

как состояний программы, как правило, слишком много, то в анализе используется некоторая абстракция состояний. От устройства абстракции состояния будет зависеть точность всего анализа. Например, если абстрактное состояние программы будет представлять собой значения всех переменных программы для всех ее потоков, то анализ будет очень точным, но крайне медленным. И наоборот, если не учитывать значения никаких переменных, то анализ будет очень быстрым, но неточным.

Теперь определим тот тип ошибки, который мы будем искать. В общем случае состоянием гонки принято называть ситуацию, при которой поведение некоторой системы зависит от последовательности внутренних неконтролируемых событий. Это означает, что в общем случае поведение такой системы не определено. В программных системах состояния гонки, которые происходят при одновременном доступе к одним и тем же областям памяти из разных потоков, при этом хотя бы один из доступов является записью, называются состояниями гонки на данных (англ. data race).

Состояния гонки не обязательно приводят к некорректному поведению программы. Например, одновременная модификация некоторого счетчика статистики может привести лишь к несущественным погрешностям. Такие ситуации называются безобидными (англ. benign) состояниями гонки. Однако известны случаи, в которых стоимость одной такой ошибки становилась катастрофической.

Поиск подобных ошибок является достаточно сложной задачей, в первую очередь потому, что ошибки, связанные с состояниями гонки, возникают недетерминированно. Поэтому любое сколь угодно качественное тестирование не сможет гарантировать, что многопоточная программа не упадет на тех же самых тестах в следующий раз.

Задача статического поиска гонок также является достаточно сложной, так как в общем случае для доказательства отсутствия ошибок необходимо рассмотреть все возможные варианты выполнения нескольких потоков — это задача экспоненциальной сложности. Методы статического анализа с использованием переключений (англ. interleavings) решают эту задачу для очень небольших программ [4, 5, 6, 7].

Многие инструменты нацелены на поиск как можно большего числа ошибок, не стремясь доказывать их отсутствие. В таких инструментах статического анализа также возникает ряд сложностей. Рассмотрим две наиболее важные проблемы: необходимость в модели памяти и присутствие определенного процента ложных срабатываний.

В отличие от динамического анализа, в котором всегда известно, к какой именно области памяти производится доступ, в статическом анализе неизвестно значение указателей, и точно определить, что доступ производится к одной и той же области памяти крайне сложно. Существуют различные методы для решения такой проблемы: точная модель памяти, анализ

синонимов и др. Результатом анализа синонимов является отображение указателей во множество областей памяти, на которые они указывают. Проблема заключается в том, что при консервативном подходе множество возможных областей памяти быстро разрастается при активном использовании указателей. В результате чего анализ тратит значительный объем ресурсов и используется только в тяжеловесных инструментах, для которых больше важна точность. Легковесные инструменты, для которых важна скорость, могут не использовать такие методы или использовать эвристики, пропускающие ошибки в некоторых случаях.

Одним из способов сокращения потенциального множества переменных, для которых возможны состояния гонки, является введение дополнительного анализа разделяемых данных. Этот анализ выдает множество потенциальных разделяемых переменных, которые могут быть вовлечены в состояние гонки. Основным вопросом остается консервативность данного анализа. Анализ разделяемых данных не способен отследить сложные случаи, связанные, например, с адресной арифметикой. Это означает, что он способен упустить некоторые разделяемые переменные, а значит, способен привести к пропуску ошибок. С другой стороны, если анализ разделяемых данных будет излишне консервативен, тогда результирующее множество разделяемых переменных будет не сильно отличаться от множества всех исходных переменных.

Второй важной проблемой статического анализа является значительно больший процент ложных предупреждений по сравнению с динамическим анализом. Рассмотрим основные причины, по которым инструмент может выдавать ложные предупреждения об ошибках. Предположим, что найдено некоторое потенциальное состояние гонки, которое описывается одним путем выполнения программы, включающем операции в нескольких потоках. Пусть инструмент выдал трассу, которая приводит к двум доступам к одной и той же памяти. Возникает вопрос, возможно ли такое выполнение программы в реальности. Далее мы подробно опишем те причины, которые приводят к наибольшему числу ложных предупреждений.

Первой причиной того, что предупреждение оказывается ложным, может быть то, что любой путь, ведущий к некоторому доступу к памяти, не может быть получен при реальном выполнении из-за несовместности некоторых условий ветвления. Например, не было учтено значение некоторой переменной. В таком случае говорят, что путь недостижим. Зачастую недостижимость может быть связана только с операциями в одном из потоков. Если из большого пути выполнения программы, включающем пути выполнения для нескольких потоков, можно выделить один локальный путь, который является недостижимым, тогда говорят о локальной недостижимости пути. В ином случае имеет место глобальная недостижимость. Существуют несколько подходов к решению задачи недостижимости, но так или иначе они все связаны с построением логических формул и доказательства их выполнимости. Как известно, задача SAT является NP-полной, и для сложного

пути с большим количеством переменных и особенно указателей время на решение этой задачи может становиться запредельным.

Глобально недостижимый путь возможен, если, например, на поведение одного потока влияет значение разделяемой переменной, которое устанавливается в другом. Вторым вариантом такого поведения может стать создание в разных ветках некоторого условного оператора двух потоков, в которых производится доступ к разделяемой области памяти. В этом случае потоки не могут выполняться параллельно, так как обе ветки условного оператора не могут выполниться одновременно, хотя локальный путь внутри созданных потоков будет достижимым. Такие случаи определить и исключить еще сложнее, чем локально-недостижимые пути.

Второй причиной ложных предупреждений является определение возможных примитивов синхронизации, которые могут быть использованы для организации доступа к разделяемым данным. В первую очередь все примитивы синхронизации также являются некоторыми объектами в памяти, на которые могут указывать сразу несколько указателей. Вопрос тождественности указателей снова приводит нас к необходимости анализа синонимов. Следующей проблемой могут стать различные условные действия, например, mutex_trylock. Поведение программы зависит от возможности захватить блокировку, но блокировка будет захвачена, только если эта функция вернет код успешного завершения (обычно 0). Для того чтобы можно было проверить, верно ли анализ учел код возврата и пошел по нужному пути, необходимо связать каким-то образом возвращаемое значение с захваченной блокировкой.

Третьей важной причиной ложных срабатываний является необходимость в анализе потоков. Зачастую программа сначала выполняет некоторые подготовительные действия (инициализацию) из одного потока, а затем создается необходимое количество вспомогательных потоков. В этом случае выдавать предупреждения для тех переменных, один из доступов к которым был из блока инициализации, не нужно. Задачей анализа потоков как раз и является определение тех участков кода, которые могут выполняться параллельно.

3. Идея метода

Как уже было отмечено, состоянием программы является состояние ее памяти, включающее в себя программный счетчик. Если выполнение программы возможно в несколько потоков, тогда в состояние памяти включается множество программных счетчиков для каждого потока управления. Будем говорить, что для состояния многопоточной программы можно взять проекцию на поток. Проекция включает в себя состояние памяти, которое доступно только конкретному потоку: его локальные переменные и общие разделяемые данные. Для того чтобы не абстрагироваться от деталей

взаимодействия потоков между собой, далее будем рассматривать анализ на проекциях. При таком анализе каждый поток рассматривается отдельно от других, которые являются для него окружением. Определим две проекции, как совместные, если существует такое глобальное состояние, проекциями которого они являются.

Состояние гонки — это такое состояние программы, в котором возможны два доступа к одной области памяти из разных потоков, хотя бы один из доступов является записью. Для того чтобы определить состояние гонки на проекциях, необходимо уже два состояния-проекции. Так, состоянием гонки будем называть такую пару проекций, которые являются совместными и в соответствующем им конкретном состоянии программы возможна гонка.

При анализе обычно рассматривают некоторые абстрактные состояния, включающие в себя некоторое множество состояний программы. Так как мы рассматриваем анализ на проекциях, то в дальнейшем под абстрактным состоянием мы будем понимать абстрактное состояние-проекцию.

Предлагаемый метод основан на хорошо известном алгоритме Lockset [8]. Этот алгоритм изначально был реализован в инструменте динамического анализа, в состоянии которого хранилось множество захваченных блокировок для каждого потока. В процессе анализа полученное множество сохранялось для каждого доступа к разделяемой памяти. Если находилась такая пара обращений к памяти, множества блокировок для которых не пересекались, то считалось, что состояния совместны и выдавалось предупреждение о состоянии гонки.

Наш анализ устроен похожим образом: в абстрактном состоянии сохраняется информация о захваченных в данный момент блокировках. После построения всего графа достижимых состояний для каждого доступа к памяти имеется абстрактное состояние, содержащее множество захваченных блокировок. Аналогично алгоритму Lockset совместность состояний определяется по пересечению множества блокировок.

Вычисление множества блокировок в процессе анализа устроено очень просто: если встречается функция захвата блокировки, то в абстрактное состояние добавляется соответствующая блокировка, если встречается функция освобождения, то блокировка удаляется.

Мы используем упрощенную модель памяти и считаем, что указатель указывает на область памяти, однозначно связанную с его именем: разные указатели указывают на разную память, а один и тот же указатель указывает на одинаковую память независимо от точки в программе. Кроме того, при анализе указателей, записанных в поля структур, учитывается только имя поля, но не имя указателя. Это означает, что анализ не сможет разделить память, на которую указывают два указателя A→a и B→a, так как имя поля у них одинаково, и будет считать, что они указывают на одну память. Стоит заметить, что поля не различаются только для структур одного типа, если же

структуры A и B имеют разный тип, то будет считаться, что их поля, даже если они носят одно имя, указывают на различные области памяти.

Рассмотрим пример работы анализа, который отвечает за сбор блокировок (см. рис 1). Заметим, что на рисунке представлены только состояния программы, в которых хранится информация о захваченных блокировках. В таком примере будет обнаружено состояние гонки: существует переход, осуществляющий доступ к глобальной переменной global из состояния, в котором отсутствуют любые блокировки, а также из состояния, в котором захвачена блокировка lock. Пересечение этих множеств пусто, значит, имеет место потенциальное состояние гонки.

```
1  int global;
2  int func(int var) {
3    if (var) {
4      lock();
5    }
6    global++;
7    if (var) {
8      unlock();
9    }
10 }
```

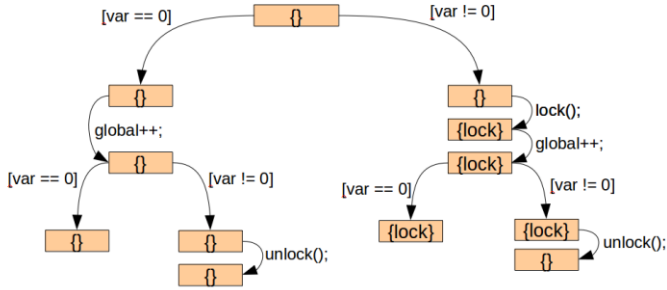


Рис. 1. Пример анализа примитивов синхронизации

Fig. 1. Example of analysis of synchronization primitives

Необходимо отметить, что в нашем инструменте используется эвристика для снижения количества ложных предупреждений об ошибках. Она основана на предположении, что случай, при котором работа с некоторыми разделяемыми данными не защищается вообще никогда, маловероятен. Обычно для большинства доступов применяется необходимая защита, и только в некоторых случаях она бывает пропущена. Таким образом, в соответствии с нашей эвристикой предупреждения не будут выдаваться для тех переменных, работа с которыми производится вообще без примитивов синхронизации, хотя, с точки зрения классического алгоритма Lockset, эта ситуация рассматривается как потенциальное состояние гонки.

Похожим образом устроен анализ разделяемых данных, задача которого состоит в отслеживании множества указателей, которые указывают на локальные данные в данной точке программы. Например, после выделения памяти с помощью функции malloc указатель не может указывать на разделяемые данные. В абстрактном состоянии для этого типа анализа хранится информация о том, какие доступные в данной точке программы области памяти являются локальными, а какие разделяемыми.

4. Схема реализации метода

Описываемый метод был реализован с использованием концепции CPA (англ. Configurable Program Analysis) [9]. Эта концепция конфигурируемого статического анализа позволяет легко запускать различные виды анализа совместно: последовательно или параллельно. При этом каждый анализ способен самостоятельно выбирать баланс между точностью и затрачиваемыми ресурсами.

Рис. 2. Схема реализации метода

Fig. 2. Implementation of the method

Рассмотрим схему анализа, представленную на рис. 2. Сначала запускается анализ разделяемых данных. Результатом его работы является множество неразделяемых областей памяти. Дальнейший анализ для переменных, указывающих на память из этого множества, производиться не будет. После анализа разделяемых данных запускается параллельная композиция трех анализов: анализа примитивов синхронизации, анализа потоков и анализа предикатных абстракций. На этом шаге отслеживается и сохраняется информация об используемых примитивах синхронизации, а также определяются точки создания потоков. После завершения анализа программы, вычисляется множество предупреждений о потенциальных ошибках. Для всех обнаруженных потенциальных ошибок запускается следующий этап анализа — уточнение. На этой стадии производится более детальный анализ с целью отбросить ложные предупреждения. В первую очередь производится проверка логической выполнимости всех условий на пути. В случае, если один из путей, ведущих к ошибке, оказался ложным, например, не было учтено значение условия, то путь считается ложным и исключается из абстракции, а переменная, значение которой повлияло на недостижимость пути, добавляется

в абстракцию, и дальнейший анализ будет учитывать ее значение. Эта операция называется уточнением абстракции. После этого анализ повторяется для более точной абстракции, то есть с учетом значений добавленных переменных. Новые найденные предупреждения также подвергаются проверке при уточнении. Так продолжается до тех пор, пока процесс не сойдется. Стоит заметить, что операция уточнения абстракции позволяет в конечном итоге учитывать только те условия, которые влияют на достижимость потенциальных состояний гонки.

Теоретически можно придумать пример, на котором число необходимых уточнений будет бесконечным, однако на практике обычно встречаются программы, для анализа которых требуется некоторое конечное число уточнений. Однако для большого объема исходного кода одна итерация уточнения может занимать несколько десятков секунд. Поэтому обычно используется некоторое ограничение по времени на анализ, по истечении которого все найденные предупреждения, если только не было доказано, что они ложные, выдаются пользователю. Проведенные эксперименты показали, что во многих случаях ложные предупреждения отсеивались за небольшое количество итераций уточнения, а остальное время тратилось на то, чтобы подтвердить, что остальные предупреждения являются истинными с точки зрения достижимости пути.

Анализ разделяемых данных и анализ примитивов синхронизации был представлен ранее в статье [10]. Анализ предикатов является стандартным анализом, уже реализованным в концепции CPA [11]. Детальное описание процесса уточнения и анализа потоков будет представлено в соответствующих разделах.

5. Описание метода уточнения

Задачей этой части анализа является исключение ложных предупреждений, связанных с наличием недостижимых путей выполнения. Важно отметить, что здесь решается только задача локальной достижимости, и взаимодействие потоков друг с другом не учитывается.

Рассмотрим пример на рис. 1. В случае, если анализ не учитывает возможные значение переменных, то будут рассмотрены 4 пути выполнения. Один из них, в котором сначала захватывается блокировка (строка 4), а затем не освобождается, влияет на весь дальнейший анализ. Однако он является недостижимым, так как условие var является одинаковым в обоих ветвях условного оператора (строки 3 и 7). Соответственно, при реальном выполнении возможно только два пути выполнения, на каждом из которых в конце будет отсутствовать захваченная блокировка.

Метод уточнения, применяемый при поиске состояний гонки, основан на классическом алгоритме CEGAR — Counterexample Guided Abstraction Refinement [12]. Сначала опишем исходный алгоритм. Основная идея этого

подхода заключается в том, что некоторое выбранное свойство доказывается не на исходной системе, которая обыкновенно устроена достаточно сложно, а на некоторой упрощенной ее модели, которая называется абстракцией. Такая абстракция может быть очень грубой, то есть упускать значительное число деталей, но она должна оставаться корректной. Другими словами все состояния, достижимые в исходной системе, должны иметь соответствующие достижимые состояния в абстракции.

Число состояний в упрощенной модели — абстракции — обыкновенно значительно меньше, что упрощает анализ, но из-за грубости модели могут возникнуть ложные предупреждения об ошибках. В таком случае строится контрпример — пример пути, который может привести к ошибке. После этого выполняется его проверка на исходной системе. Если он выполним на исходной системе, значит, найденная ошибка истинная. В противном случае это означает, что контрпример был получен из-за неточности абстракции. Тогда абстракция уточняется на основе этого контрпримера, то есть в нее добавляются те детали, из-за отсутствия которых был получен данный контрпример. После этого анализ возобновляется на уточненной абстракции. Цикл повторяется до тех пор, пока либо не будет доказана корректность системы относительно выбранного свойства, либо не будет найдена ошибка.

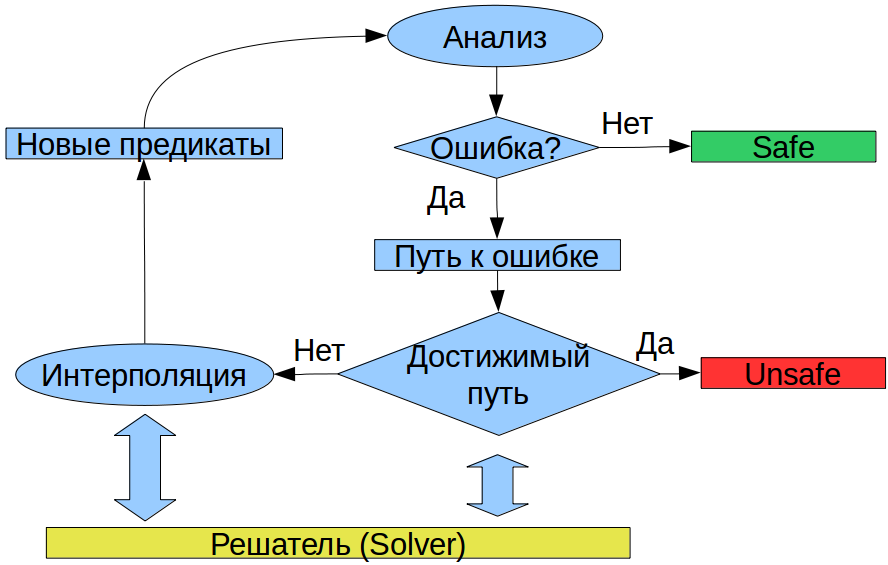


Рис. 3. Схема традиционного метода CEGAR

Fig. 3. CEGAR (Counterexample Guided Abstraction Refinement)

Абстракцию программы можно делать различными способами, но наиболее часто встречается предикатная абстракция, основанная на разбиении

множества состояний программы (значений ее переменных) на подмножества с одинаковым значением выбранных предикатов. Можно выделить декартову абстракцию, в которой предикаты комбинируются только с помощью конъюнкции, и булеву абстракцию, в которой возможна комбинация произвольных логических формул.

Для проверки контрпримера и пересчета абстракции необходимо представить последовательность инструкций исходной программы в виде некоторых логических формул. Для этого используются различные техники, такие как пред- и постусловия или формулы пути на основе представления SSA. Для проверки этих формул на выполнимость используются различные SMT-решатели (англ. Satisfiability Modulo Theories). В случае, если формула пути является невыполнимой, необходимо извлечь из нее получить те условия, которые в последствии будут добавлены к анализу. Эти условия выражаются предикатами. Существуют различные способы для выделения необходимых предикатов из формулы пути, например, синтаксические методы [13] и интерполяция Крейга [14, 15].

Традиционный метод CEGAR успешно реализован во многих инструментах статической верификации. Там он успешно применяется для решения задачи достижимости, при которой производится проверка достижимости некоторой ошибочной метки, которая присутствует в исходном коде, например, вызова специальной функции. Процесс уточнения запускается, если был найден путь, ведущий от некоторой точки входа до этой ошибочной метки.

Теперь перейдем к описанию модифицированного алгоритма CEGAR, который применяется для уточнения найденных потенциальных состояний гонки. Для наличия состояния гонки требуется два совместных состояния. Необходимым условием для этого является наличие двух локально-достижимых состояний. Соответственно, для каждого потенциального состояния гонки существуют как минимум два локальных пути, которые необходимо проверить на достижимость. Заметим, что каждый путь по отдельности не может считаться ошибочным, так как доступ к разделяемым данным без блокировок сам по себе может не быть ошибкой, необходимо, чтобы нашелся второй доступ к этим же данным.

При уточнении примера ложного пути, рассмотренного выше, будет построена формула пути, в которой встретится такая часть $[var == 1] \wedge [var == 0]$. Полученная формула будет передана специальному компоненту — решателю (англ. Solver), который выдаст вердикт, что формула невыполнима. Вместе с вердиктом решатель выдаст интерполянты, которые по сути являются противоречивой частью формулы. Для нашего примера интерполянтом является, например, $[var == 0]$. При дальнейшем анализе значение этой переменной будет учитываться, и подобная недостижимая трасса больше не появится. На рис 4 представлен граф абстрактных состояний

для анализа предикатов и анализа примитивов синхронизации для программы, изображенной на рис. 1. В фигурных скобках будем заключать абстрактное состояние анализа примитивов синхронизации, а в квадратных скобках – абстрактное состояние анализа предикатов.

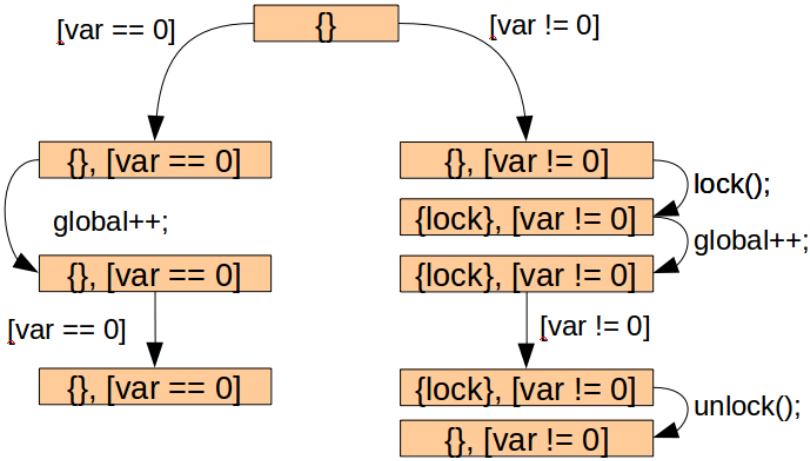


Рис 4. Пример анализа предикатов совместно с анализом примитивов синхронизации

Fig. 4. Example of predicate analysis together with analysis of synchronization primitives

Потенциально в программе может быть не одно состояние гонки. Тогда существует два варианта поведения при обнаружении потенциальной ошибки: переход к уточнению сразу или продолжение анализа для поиска следующих ошибок. Рассмотрим эти две стратегии подробнее.

1. Уточнение после обнаружения потенциального состояния гонки. При таком способе анализ продолжается до тех пор, пока не будет найдена пара совместных состояний с доступами к одним разделяемым данным. После обнаружения такой пары запускается уточнение, которое проверяет, достижим ли каждый из путей по-отдельности. Такой способ, в отличие от предыдущего, производит только необходимые проверки, и не получит излишне точную абстракцию. Однако, большая проблема заключается в том, что нужно определить, какие еще доступы к переменным следует удалить, если путь оказался недостижим. В общем случае доступ к одним данным в одной и той же строке исходного кода возможен по разным путям. Допустим, путь к некоторому доступу к переменной оказался ложным. Тогда, некоторое множество доступов к другим данным на этом же пути тоже могут оказаться недостижимыми, значит, их следует удалить. Но возможно, что к некоторым существуют другие истинные пути. В итоге для определения множества недостижимых доступов к данным,

необходимо заново произвести анализ программы целиком, что сводит на нет все преимущества данного подхода.

2. Уточнение всех полученных предупреждений после полного анализа программы. В этом варианте весь анализ проводится целиком, после чего уточняются все полученные пути, приводящие к состояниям гонки. При необходимости анализ повторяется, при этом вся информация о доступах к памяти собирается заново, таким образом, этот способ не имеет проблем с удалением зависимых доступов к памяти. Минусом данного подхода является большой объем повторяющейся работы. Так, для некоторого класса путей, например, проходящих через одну функцию, результат уточнения может быть одинаков, и нет необходимости проводить уточнение для каждого. Кроме того, могут быть участки кода, в которых не было найдено ни одного предупреждения, тогда их не нужно анализировать заново.

Были проведены некоторые эксперименты и с одной стратегией уточнения и с другой. И тот, и другой метод показал неплохие результаты, однако второй вариант обладал большей гибкостью: можно было остановить весь анализ при достижении некоторого уровня точности абстракции. Тогда как первый вариант требовал построения детальной абстракции для получения результатов.

Описанный метод уточнения позволяет исключать из выдаваемых предупреждений такие, которые образованы локально-недостижимыми путями. Однако он требует значительного количества времени, так как для исключения всех ложных путей требуется достаточно точная абстракция, что означает наличие большого количества предикатов. Например, если на пути встретился цикл, то при уточнении придется рассматривать все его итерации. По результатам экспериментов можно сказать, что многие ложные предупреждения отбрасываются достаточно быстро, а большую часть времени уточнения (~99%) тратится на то, чтобы доказать истинность остальных предупреждений.

6. Анализ потоков

Зачастую возникают ситуации, при которых доступ к некоторым разделяемым данным допускается без использования явных примитивов синхронизации. Основная причина заключается в том, что в этот момент возможно выполнение только единственного потока. Например, при инициализации данных одним потоком другие, как правило, даже не созданы. Однако, наш анализ не учитывал зависимость между точками создания потоков и считал, что все они созданы одновременно. Это порождало большое число ложных срабатываний. Рассмотрим такой пример: один поток порождает другой. Очевидно, что оба эти потока могут работать параллельно, вместе с этим часть

кода первого потока до создания второго потока никак не может быть выполнена параллельно. Задачей разработанного анализа потоков является вычисление множества точек программы, которые потенциально могут работать параллельно друг с другом.

Рассмотрим пример искусственного ложного срабатывания.

```
1 int global;
2
3 int worker(void* arg) {
4   …
5   global++;
6   …
7 }
```

```
8  int start() {
9    …
10   global = 0;
11   …
12   pthread_create(&thread, .., worker, ..);
13   …
14   pthread_join(&thread);
15   result = global;
16   …
17 }
```

В этом примере стартовая функция start выполняет некоторую инициализацию, а затем создает поток, в котором происходит работа с глобальной переменной. Так как поток не может выполниться до того, как он будет создан, одновременный доступ к глобальной переменной невозможен. Этот пример схематично можно изобразить на следующем рисунке (рис. 5).

На рисунке вертикальными линиями показано выполнение потоков, а горизонтальный стрелки означают создание нового потока и его завершение.

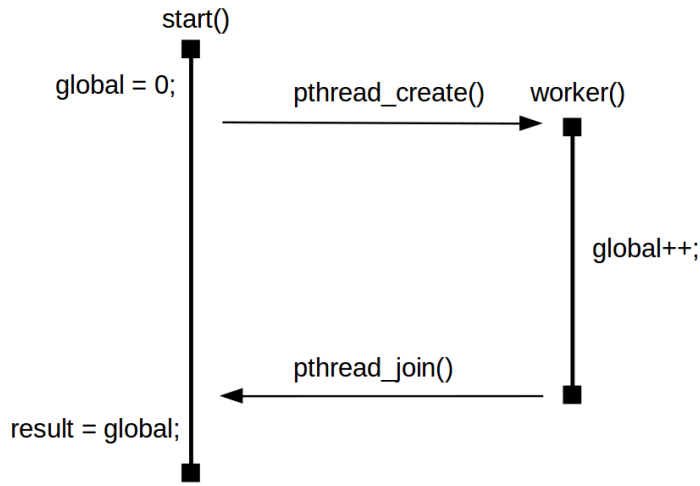


Рис. 5. Схема параллельной работы функций

Fig. 5. Example of parallel execution of functions

Абстрактное состояние содержит в себе множество меток, которое описывает множество активных в данный момент потоков. Совместность состояний определяется как наличие совместных меток, что означает, что состояния соответствуют коду, работающему параллельно. Множество меток модифицируется в точках создания или завершения потоков. Метка идентифицируется уникальным именем, привязанным к потоку, и имеет две части, отличающиеся друг от друга бинарным флагом. При создании потока в множество меток родительского потока добавляется флаг метки 0, а в множество меток дочернего потока — флаг метки 1. Рассмотрим пример: пусть создается некоторый поток. Ему присваивается уникальный идентификатор 2. Тогда во множество меток родительского потока заносится метка 2.0. А во множество меток созданного потока заносится метка 2.1. Два оператора могут быть выполнены параллельно друг с другом, если во множестве меток, им соответствующим, найдутся различные флаги одной метки. Для рассмотренного примера это означает, что состояния, в которых есть метка 2.0 несовместны, а если в одном из состояний есть метка 2.0, а в другом 2.1, то такие состояния являются совместными. По сути это означает, что в какой-то момент был создан некоторый поток, после этого родительский поток пришел к одному использованию, а дочерний — к другому.

Гораздо сложнее ситуация с операцией ожидания завершения потоков (оператора join). В общем случае алгоритм распространения меток становится достаточно сложным, поэтому мы рассмотрим упрощенный вариант, при котором делается предположение, что поток завершается в том же родительском потоке, который его и создал. В таком случае при завершении потока соответствующая метка удаляется из множества меток родительского потока. При рассматриваемых ограничениях (операция join выполняется из родительского потока) в состоянии родительского потока должна быть метка завершающегося потока с флагом 0, а в дочернем потоке — метка с флагом 1. Никаких других меток быть не должно, кроме тех, которые уже есть в родительском состоянии. Например, возможна такая ситуация. Родительский поток с идентификатором 1, имея множество таких меток {1.1, 2.0}, выполняет операцию join для дочернего потока с идентификатором 2, который имеет состояние {1.1, 2.1}. Результатом этой операции будет множество меток {1.1}.

Теперь более формально опишем алгоритм вычисления множества меток. В состоянии для этого типа анализа хранится два множества: Tset и Rset. Первое — это множество активных меток, а второе — это множество удаленных меток. Нам потребуется расширенная операция пересечения множеств меток, которую будем обозначать так:

$$Tset_1 * Tset_2 = \{Lid : \exists(Lid, flag_1) \in Tset_1 \wedge \exists(Lid, flag_2) \in Tset_2 \wedge (flag_1 \neq flag_2)\}$$

Эта операция возвращает множество тех меток, у которых различаются флаги в пересекаемых множествах. При создании потока (операция pthread_create) в

новое множество Tset родительского потока добавляется метка с флагом 0, а во множество Tset дочернего потока — метка с флагом 1. Множество Rset при этой операции не изменяется. При операции ожидания завершения потока (pthread_join) новое значение множеств $\hat{Tset}$ и $\hat{Rset}$ вычисляются следующим образом:

$$\hat{Rset} = Rset_1 \cup Rset_2 \cup (Tset_1 * Tset_2),$$
$$\hat{Tset} = \{(Lid, flag) : Lid \in (Tset_1 \cup Tset_2) / \hat{Rset}\},$$

где $Tset_1, Tset_2$ — это множества из двух состояний родительского потока и потока, который завершается.

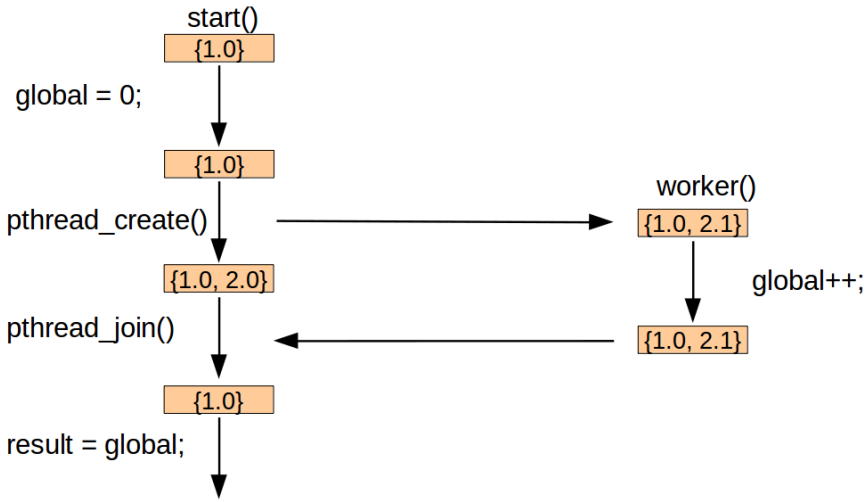


Рис. 6. Пример состояний анализа потоков

Fig. 6. Example of abstract states in thread analysis

Для рассмотренного примера ложного предупреждения анализ получит, что для первого доступа к этому полю будет вычислено следующее множество меток {1.0}, означающее, что в этот момент работает только один поток. Для доступа к этому полю из функции worker() будет получено множество {1.0, 2.1}, означающее, что в этот момент работает два потока. Однако эти два состояния несовместны, так как в них нет метки с различными флагами (метка для первого потока встречается в обоих состояниях с одним и тем же флагом). Отсюда следует, что предупреждение ложное.

Нужно отметить, что описанный метод может быть применен только в простых случаях, в которых поток завершается в том же потоке, в котором был создан. Более хитрые ситуации, в которых, например, родительский поток завершается в дочернем не рассматриваются. Теоретически метод расстановки

меток позволяет проводить анализ таких программ, однако для этого требуется значительная переработка реализации инструмента. Однако, более сложным ситуациям довольно сложно найти применение в реальном программном обеспечении, поэтому такое ограничение реализованного метода выглядит достаточно естественным.

7. Эксперименты

Эксперименты были проведены на наборе задач, основанных на модулях ядра ОС Linux 4.5-rc1 подсистемы drivers/net/wireless. Для формирования задач из исходного кода использовалась инфраструктура инструмента LDV [16, 17], с помощью которой для драйвера формировалась модель окружения [18]. Были проведены запуски в четырех вариантах работы инструмента: с одним из анализов (анализ с уточнениями и анализ потоков), с обоими и без них (табл. 1).

В указанной подсистеме было проанализировано 113 модулей. Было несколько причин, которые приводили к незавершенному анализу, например, превышение ограничений по времени, но большая часть из них относилась к этапу подготовки модуля к верификации, то есть падению некоторых компонентов системы LDV.

По результатам видно, что анализ потоков не сильно увеличивает время на анализ, но значительно повышает расход памяти. А уточнение ведет себя с точностью до наоборот. Это связано с тем, что анализ потоков требует для себя отдельного пространства состояний, а уточнение — лишь проверяет, правильно ли вычислены достижимые состояния.

Для запуска с двумя включенными дополнительными анализами был проведен анализ результатов. Все 5 найденных предупреждений связаны с тем, что некоторые данные достаются из разделяемого списка под блокировками, при этом они удаляются из списка, а после этого работа с ними осуществляется без блокировок. Таким образом, данные случаи не являются состояниями гонки.

Табл. 1. Результаты запуска инструмента на drivers/net/wireless/

Table 1. Results of launch of the tool on drivers/net/wireless/

	Предупреждения	Незавершенный анализ	Корректные модули	Время, ч	Память, Гб
+ Потоки, + уточнение	5	61	51	3.2	8.1
- Потоки, + уточнение	6	67	44	4.1	4.0

+ Потоки, - уточнение	27	57	49	2.3	8.2
- Потоки, - уточнение	186	54	43	2.1	3.5

Был проведен запуск инструмента на наборе задач, подготовленных на основе всей папке drivers/ ядра ОС Linux, в конфигурации с анализом потоков и уточнением. Было получено 2219 предупреждений. Наиболее важной причиной ложных срабатываний стало несовершенство модели окружений, из-за которой было получено более 50 % ложных предупреждений. Например, некоторые функции-обработчики драйвера при реальном выполнении вызываются под блокировками, в то время как в модели этот факт не учитывается. Кроме того зачастую возникают неявные зависимости между обработчиками разных структур, которые не позволяют им выполняться одновременно и которые также не учитываются в модели окружения. Следует еще раз отметить, что модель окружения относится к этапу подготовки задачи, а не к методу анализу непосредственно.

Возникали ситуации, в которых структуры одного типа использовались для разного назначения, которое предполагало наличие различных блокировок. Таким образом, упрощенная модель памяти, которую используется в методе, принесла около 10 % ложных предупреждений.

Еще 10% ложных предупреждений связаны с ситуациями, в которых из разделяемого множества достается объект под блокировками, а работа с ним осуществляется без них. Около 10% связаны с неточностями анализа: это и несовершенство анализа функциональных указателей, и проблемы в анализе примитивов синхронизации (в т. ч. отсутствие некоторых блокировок в конфигурации), и неточности в анализе разделяемых данных.

Около 15% предупреждений оказались истинными. Следует отметить, что зачастую для одного драйвера, содержащего состояние гонки, может выдаваться более 10 предупреждений на разные переменные. То есть, предупреждение означает лишь проявление ошибки. Так, найденные 290 предупреждений соответствовали лишь 32 ошибкам. Эти ошибки были сообщены разработчикам. Часть из них были признаны, некоторые даже исправлены, и на некоторые сообщения об ошибках ответ не был получен. Список исправленных ошибок приведен на странице http://linuxtesting.org/results/ldv, категория data race.

Таким образом, представленный метод поиска состояний гонок в операционных системах продемонстрировал практически значимые результаты с приемлемым уровнем ложных срабатываний. Являясь, по сути, легковесным методом, он допускает более гибкую настройку баланса между ресурсами и точностью анализа.

Результаты анализа причин ложных срабатываний позволяют сделать вывод, что основные усилия при применении инструмента к модулям ядра операционной системы Linux необходимо направить на повышение точности используемых моделей окружения. Развитие собственно предложенного метода может продолжаться в нескольких направлениях: повышение точности входящих в него анализов и применяемой модели памяти, поддержка новых примитивов синхронизации, например, RCU, применение новых подходов с целью ускорения анализа. Кроме того, отдельной задачей является исследование возможности практического применения разработанного инструмента к другим классам целевых программ.

Список литературы

[1]. Мутилин В.С., Новиков Е.М., Хорошилов А.В. Анализ типовых ошибок в драйверах операционной системы Linux. Труды ИСП РАН, том 22, 2012 г., стр. 349-374. DOI: 10.15514/ISPRAS-2012-22-19

[2]. Nancy Levenson, Safeware: System Safety and Computers, 1995

[3]. Nachum Dershowitz, Software horror stories, http://www.cs.tau.ac.il/~nachumd/pub.html

[4]. Cordeiro, L., Fischer, B.: Verifying Multi-Threaded Software using SMT-based Context-Bounded Model Checking. In: ICSE, pp. 331–340 (2011)

[5]. E. Clarke, D. Kroening, N. Sharygina, K. Yorav. SATABS: SAT-based predicate abstraction for ANSI-C. Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05), pp. 570-574, 2005.

[6]. A. Gupta, C. Popeea, A. Rybalchenko. Predicate abstraction and refinement for verifying multi-threaded programs In Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2011), pp. 331-344, 2011.

[7]. A. Gupta, C. Popeea, A. Rybalchenko. Threader: a constraint-based verifier for multi-threaded programs In Proceedings of the 23rd International Conference on Computer Aided Verification (CAV 2011), LNCS, vol. 6806, pp. 412-417, 2011.

[8]. Stefan Savage, Michael Burrows, Greg Nelson, Patric Sobalvarro, Thomas Anderson Eraser: A Dynamic Data Race Detector for Multithreaded Programs ACM Transactions on Computer Systems, Vol. 15, No. 4, November 1997, Pages 391–411.

[9]. Dirk Beyer, Thomas A. Henzinger, and Gregory Theoduloz, Configurable Software Verification: Concretizing the Convergence of Model Checking and Program Analysis, ACM Transactions on Computer Systems, Vol. 15, No. 4, November 1997, Pages 391–411.

[10]. Андрианов П.С., Мутилин В.С., Хорошилов А.В. Метод легковесного статического анализа для поиска состояний гонок. Труды ИСП РАН, том 27, вып. 5, 2015 г., стр. 87-116. DOI: 10.15514/ISPRAS-2015-27(5)-6

[11]. Dirk Beyer , M. Erkan Keremoglu , Philipp Wendler, Predicate abstraction with adjustable-block encoding, Proceedings of the 2010 Conference on Formal Methods in Computer-Aided Design, October 20-23, 2010, Lugano, Switzerland

[12]. Мандрыкин М.У., Мутилин В.С., Хорошилов А.В. Введение в метод CEGAR — уточнение абстракции по контрпримерам. Труды ИСП РАН, том 24, 2013, стр. 219-292. DOI: 10.15514/ISPRAS-2013-24-12

[13]. Shostak R., Deciding Combinations of Theories, Journal of the ACM, 1-12, 1984
[14]. Beyer Dirk, Zuerey Damien, Majumdar Rupak., Csisat: Interpolation for la+euf, CAV, 304-308, 2008
[15]. Bruttomesso Roberto, Cimatti Alessandro, Franzen Anders, Griggio Alberto, Sebastiani Roberto., The mathsat 4 smt solver, CAV, 299-303, 2008
[16]. Alexey Khoroshilov, Mikhail Mandrykin, Vadim Mutilin, Eugene Novikov, Alexander Petrenko, Ilya Zakharov. Configurable toolset for static verification of operating systems kernel modules. Programming and Computer Software, vol. 41, № 1, 2015, pp. 49-64. DOI: 10.1134/S0361768815010065
[17]. Alexey Khoroshilov, Mikhail Mandrykin, Vadim Mutilin, Eugene Novikov, Pavel Shved. Using Linux Device Drivers for Static Verification Tools Benchmarking. Programming and Computer Software, vol. 38, № 5, 2012, pages 245-256. DOI: 10.1134/S0361768812050039
[18]. Alexey Khoroshilov,Vadim Mutilin, Ilya Zakharov. Pattern-based environment modeling for static verification of Linux kernel modules. Programming and Computer Software, vol. 41, № 3, 2015, pages 183-195. DOI: 10.1134/S036176881503007X

П.С. Андрианов, В.С. Мутилин, А.В. Хорошилов. Конфигурируемый метод поиска состояний гонок в операционных системах с использованием предикатных абстракций. *Труды ИСП РАН*, том 28, вып. 6, 2016, стр. 65-86.

Adjustable method with predicate abstraction for detection of race conditions in operating systems[2]

[1]*P.S. Andrianov <andrianov@ispras.ru>*,
[1]*V.S. Mutilin <mutilin@ispras.ru>*,
[1,2,3,4]*A.V. Khoroshilov <khoroshilov@ispras.ru>*,
[1]*Institute for System Programming of the RAS,*
25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation
[2]*Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[3]*Moscow Institute of Physics and Technology (State University)*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
[4]*National Research University Higher School of Economics (HSE)*
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. The paper presents a configurable method of static data race detection that is trying to keep a balance between resource consumption and a number of false alarms. The method is based on well known Lockset approach. It uses simplified memory model to be fast enough. At the same time it includes advanced techniques aimed to achieve acceptable false alarms rate. The key techniques are thread analysis and predicate abstraction based refinement. The method was implemented in CPALockator tool built on top of CPAchecker framework. The tool was evaluated on Linux kernel modules and it has detected several actual data races, which were approved by developers and were fixed in upstream Linux kernel.

Key words: static analysis; race condition; kernel of operating system.

DOI: 10.15514/ISPRAS-2016-28(6)-5

For citation: Andrianov P.S., Mutilin V.S., Khoroshilov A.V. Adjustable method with predicate abstraction for detection of race conditions in operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 65-86 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-5

References

[1]. Mutilin V.S., Novikov E.M., Khoroshilov A.V. Analysis of typical faults in Linux operating system drivers. Trudy ISP RAN / Proc. ISP RAS, vol. 22, 2012, pp. 349–374 (in Russian). DOI: 10.15514/ISPRAS-2012-22-19

[2] The research was carried out with funding from the Ministry of Education and Science of Russia (the project unique identifier is RFMEFI61614X0015)

[2]. Nancy Levenson, Safeware: System Safety and Computers, 1995
[3]. Nachum Dershowitz, Software horror stories, http://www.cs.tau.ac.il/~nachumd/pub.html
[4]. Cordeiro, L., Fischer, B.: Verifying Multi-Threaded Software using SMT-based Context-Bounded Model Checking. In: ICSE, pp. 331–340 (2011)
[5]. E. Clarke, D. Kroening, N. Sharygina, K. Yorav. SATABS: SAT-based predicate abstraction for ANSI-C. Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05), pp. 570-574, 2005.
[6]. A. Gupta, C. Popeea, A. Rybalchenko. Predicate abstraction and refinement for verifying multi-threaded programs In Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2011), pp. 331-344, 2011.
[7]. A. Gupta, C. Popeea, A. Rybalchenko. Threader: a constraint-based verifier for multi-threaded programs In Proceedings of the 23rd International Conference on Computer Aided Verification (CAV 2011), LNCS, vol. 6806, pp. 412-417, 2011.
[8]. Stefan Savage, Michael Burrows, Greg Nelson, Patric Sobalvarro, Thomas Anderson Eraser: A Dynamic Data Race Detector for Multithreaded Programs ACM Transactions on Computer Systems, Vol. 15, No. 4, November 1997, Pages 391–411.
[9]. Dirk Beyer, Thomas A. Henzinger, and Gregory Theoduloz, Configurable Software Verification: Concretizing the Convergence of Model Checking and Program Analysis, ACM Transactions on Computer Systems, Vol. 15, No. 4, November 1997, Pages 391–411.
[10]. Andrianov P.S., Mutilin V.S., Khoroshilov A.V. Lightweight Static Analysis for Data Race Detection in Operating System Kernels. Trudy ISP RAN / Proc. ISP RAS, vol. 27, 2015, pp. 87-116 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-6
[11]. Dirk Beyer , M. Erkan Keremoglu , Philipp Wendler, Predicate abstraction with adjustable-block encoding, Proceedings of the 2010 Conference on Formal Methods in Computer-Aided Design, October 20-23, 2010, Lugano, Switzerland
[12]. Mandrykin M.U., Mutilin V.S., Khoroshilov A.V. Introduction to CEGAR — Counter-Example Guided Abstraction Refinement. Trudy ISP RAN / Proc. ISP RAS, vol. 24, 2013, pp. 219-292 (in Russian). DOI: 10.15514/ISPRAS-2013-24-12
[13]. Shostak R., Deciding Combinations of Theories, Journal of the ACM, 1-12, 1984
[14]. Beyer Dirk, Zuerey Damien, Majumdar Rupak., Csisat: Interpolation for la+euf, CAV, 304-308, 2008
[15]. Bruttomesso Roberto, Cimatti Alessandro, Franzen Anders, Griggio Alberto, Sebastiani Roberto., The mathsat 4 smt solver, CAV, 299-303, 2008
[16]. Alexey Khoroshilov, Mikhail Mandrykin, Vadim Mutilin, Eugene Novikov, Alexander Petrenko, Ilya Zakharov. Configurable toolset for static verification of operating systems kernel modules. Programming and Computer Software, vol. 41, № 1, 2015, pp. 49-64. DOI: 10.1134/S0361768815010065
[17]. Alexey Khoroshilov, Mikhail Mandrykin, Vadim Mutilin, Eugene Novikov, Pavel Shved. Using Linux Device Drivers for Static Verification Tools Benchmarking. Programming and Computer Software, vol. 38, № 5, 2012, pages 245-256. DOI: 10.1134/S0361768812050039
[18]. Alexey Khoroshilov,Vadim Mutilin, Ilya Zakharov. Pattern-based environment modeling for static verification of Linux kernel modules. Programming and Computer Software, vol. 41, № 3, 2015, pages 183-195. DOI: 10.1134/S036176881503007X

Генератор тестовых программ для архитектуры ARMv8 на основе инструмента MicroTESK

[1, 2, 3] А.С. Камкин <kamkin@ispras.ru>
[1] А.М. Коцыняк <kotsynyak@ispras.ru>
[1] А.С. Проценко <protsenko@ispras.ru>
[1] А.Д. Татарников <andrewt@ispras.ru>
[1] М.М. Чупилко <chupilko@ispras.ru>
[1] Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] Московский государственный университет имени М.В. Ломоносова,
119991, Россия, г. Москва, Ленинские горы, д. 1
[3] Московский физико-технический институт,
141700, Россия, Московская обл., г. Долгопрудный, Институтский пер., д. 9

Аннотация. ARM — это семейство микропроцессорных архитектур, разработанных в одноименной компании. Новейшая архитектура этого семейства, ARMv8, содержит большое число команд разных типов и отличается сложной организацией виртуальной памяти (включающей аппаратную поддержку многоуровневой трансляции адресов и виртуализации); все это делает функциональную верификацию микропроцессоров этой архитектуры крайне трудной технической задачей. Неотъемлемой частью верификации микропроцессора является генерация тестовых программ — программ на языке ассемблера, создающих разнообразные ситуации (исключения, блокировки конвейера, неверные предсказания переходов, вытеснения данных из кэш-памяти и т.п.). В статье описываются требования, предъявляемые к промышленным генераторам тестовых программ, и представляется генератор для микропроцессоров архитектуры ARMv8, разработанный с использованием инструмента MicroTESK (Microprocessor TEsting and Specification Kit). Генератор поддерживает подмножество команд, характерное для мобильных приложений (около 400 команд) и состоит из двух основных частей: (1) архитектурно независимого ядра и (2) формальной спецификации ARMv8 (точнее, модели, автоматически построенной по формальным спецификациям). При такой организации процесс разработки генератора тестовых программ состоит преимущественно в создании формальных спецификаций, что экономит усилия за счет повторного использования архитектурно независимых компонентов. Архитектура описывается на языках nML и MMUSL: первый язык позволяет описывать регистры микропроцессора, синтаксис и семантику команд; второй — устройство подсистемы памяти (адресные пространства, разного рода буферы и таблицы, алгоритмы

трансляции адресов и т.п.). В статье приводятся характеристики разработанного генератора и делается сравнение с существующими аналогами.

Ключевые слова: микропроцессоры; спецификация системы команд; функциональная верификация; генерация тестовых программ; ARM; nML; MicroTESK.

DOI: 10.15514/ISPRAS-2016-28(6)-6

Для цитирования: Камкин А.С., Коцыняк А.М., Проценко А.С., Татарников А.Д., Чупилко М.М. Генератор тестовых программ для архитектуры ARMv8 на основе инструмента MicroTESK. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 87-102. DOI: 10.15514/ISPRAS-2016-28(6)-6

1. Введение

ARM (изначально Acorn RISC Machine, сейчас Advanced RISC Machine) — это семейство микропроцессорных архитектур, разработанных в одноименной компании [1]. Микропроцессоры этого семейства чрезвычайно популярны: с 1990 года было выпущено более 86 миллиардов чипов [2]; 95% смартфонов базируются на ARM (данные за 2010 год) [3]; микросхемы ARM находят применение во встроенных системах и центрах обработки данных. Разработкой ARM-совместимых ядер занимается не только ARM, но и другие компании, включая Qualcomm, Apple, Samsung, NVIDIA и Huawei [4]; очевидно, что эффективная реализация микропроцессора есть важнейший инструмент конкурентной борьбы. Новая архитектура ARMv8(-A) отличается большим числом команд (около 1000) и сложной организацией виртуальной памяти (VMSA, Virtual Memory System Architecture) [5].

Проектирование ARMv8-совместимого микропроцессора, оптимизированного по производительности, энергопотреблению и стоимости, чревато ошибками. Для их выявления применяют разные методы: инспекция кода, тестирование, формальная верификация. Основным подходом является исполнение на модели микропроцессора тестовых программ и сравнение результатов исполнения (трасс) с результатами, полученными при исполнении тех же программ на программном эмуляторе (эталонной модели) [6]. Общепринятая практика создания тестовых программ базируется на рандомизированной генерации на основе шаблонов: пользователь определяет шаблон — описание структуры программы и ограничений на используемые данные, — генератор же строит тест, рандомизируя элементы шаблона, не заданные жестко [7].

Задача генерации тестовых программ для микропроцессора не так проста, как может показаться на первый взгляд. При конструировании тестов нужно учитывать не только формат, но семантику команд: сгенерированные данные не должны приводить к неопределенному поведению (такие ситуации особо выделяются в руководствах по архитектурам); программы не должны зацикливаться; поток управления не должен переходить в секцию данных и

т.д. и т.п. Кроме того, генератор должен принимать в расчет такие механизмы микропроцессора, как конвейер, обработка исключений, виртуальная память, многоядерность.

В работе рассматривается генератор тестовых программ для подмножества ARMv8 (около 400 команд), разработанный с использованием инструмента MicroTESK (Microprocessor TEsting and Specification Kit) [8]. Генератор состоит из двух основных частей: архитектурно независимого ядра (библиотечной части) и формальной спецификации ARMv8 на языках nML (система команд) [9] и MMUSL (подсистема памяти) [10]. При такой организации создание генератора состоит в основном в написании спецификаций, что, во-первых, экономит усилия за счет использования общих компонентов, а во-вторых, упрощает адаптацию генератора к модификациям архитектуры (добавлению новых команд, изменениям в подсистеме памяти и т.п.); помимо прочего, использование формальных спецификаций позволяет автоматизировать решение проблем, обозначенных выше.

Оставшаяся часть статьи структурирована следующим образом. В разделе 2 рассматриваются существующие решения в области генерации тестовых программ для микропроцессоров: RIS (ARM), RAVEN (ARM) и Genesys-Pro (IBM). Раздел 3 посвящен генератору тестовых программ MicroTESK-ARMv8; здесь описываются базовые принципы используемого подхода (MicroTESK) и особенности его применения к архитектуре ARMv8. В разделе 4 приводятся характеристики разработанного генератора (поддерживаемые команды, объем спецификаций и т.п.), а также сведения о трудоемкости разработки. Раздел 5 резюмирует работу и обрисовывают направления дальнейших исследований.

2. Обзор существующих решений

Компания ARM разрабатывает собственные инструменты генерации тестовых программ. К настоящему времени разработано несколько генераторов, получивших название RIS (Random Instruction Sequence) [11]. Это набор узкоспециализированных средств, предназначенных для тестирования таких механизмов, как многоядерность [12] и управление памятью [13]. Настройка RIS осуществляется путем задания используемых команд, их весов (распределений вероятности), ограничений на операнды, способа размещения кода и данных в памяти, а также цепочек команд, решающих специальные задачи (вытеснение данных из кэш-памяти и т.п.).

Другим инструментом, используемым в ARM, является RAVEN (Random Architecture Verification Machine) [14], разработанный в компании Obsidian Software, поглощенной ARM в 2011 году. Это универсальное средство, применимое для широкого спектра архитектур, в котором ядро, реализующее логику генерации, отделено от конфигурации для конкретной архитектуры. RAVEN позволяет создавать как случайные, так и нацеленные тесты. В процессе работы инструмент отслеживает состояние микропроцессора путем симуляции построенных программ на внешней эталонной модели.

Еще одним универсальным средством генерации тестовых программ является Genesys-Pro [7] от IBM Research. Этот инструмент использует входные данные двух типов: модель, описывающая архитектуру микропроцессора, и шаблоны, формулирующие задачи тестирования. Шаблоны создаются на выразительном языке, имеющем конструкции для описания цепочек команд, распределений вероятностей и ограничений. Как и в RAVEN, состояние микропроцессора в Genesys-Pro отслеживается с использованием внешней эталонной модели. Возможности инструмента могут быть расширены средством DeepTrans [15], предназначенным для верификации механизмов трансляции адресов.

Подводя итог, отметим следующие недостатки существующих решений. Инструменты типа RIS жестко привязаны к архитектуре, что делает их непригодными для верификации микропроцессоров с новой системой команд. Инструменты типа RAVEN и Genesys-Pro этого недостатка лишены — их можно адаптировать к новой архитектуре; однако эта работа не является простой и требует определенной квалификации. Так, при расширении архитектуры необходимо добавить знание о новых командах в генератор, а также реализовать эти команды в эталонной модели. Предлагаемый подход нацелен на максимальное упрощение процессов разработки и сопровождения генераторов тестовых за счет извлечения всей необходимой информации из единого источника — формальных спецификаций.

3. Генератор тестовых программ MicroTESK-ARMv8

3.1 Инструмент MicroTESK

Описываемый в статье генератор тестовых программ для микропроцессоров архитектуры ARMv8 реализован с помощью инструмента MicroTESK [8], разрабатываемого в ИСП РАН. Инструмент включает в себя две основные части: (1) *среду моделирования* и (2) *среду генерации*. Среда моделирования отвечает за построение модели микропроцессора на основе формальных спецификаций. Задача среды генерации — построение тестовых программ для микропроцессора по сценариям, описанным в тестовых шаблонах. Структура инструмента MicroTESK изображена на рис. 1.

В настоящее время среда моделирования поддерживает два типа *формальных спецификаций*: (1) *спецификации системы команд микропроцессора* на языке nML [9] и (2) *спецификации подсистемы памяти микропроцессора* на языке MMUSL [10]. Для каждого типа поддерживаются свои анализаторы и генераторы кода модели (их набор может быть расширен). Модель микропроцессора, генерируемая по спецификациям, включает в себя: (1) *метаданные*, хранящие информацию о доступных регистрах и командах; (2) *симулятор*, позволяющий программно эмулировать исполнение команд и получать информацию о состоянии микропроцессора; (3) *модель тестового покрытия*, содержащая информацию о возможных путях исполнения команд.

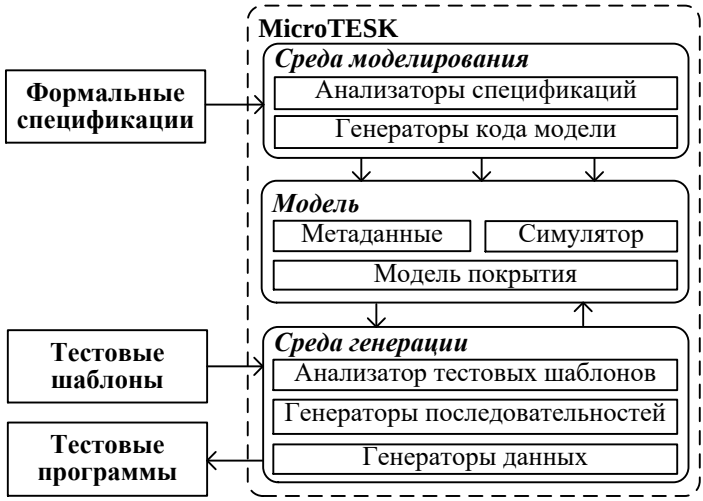


Рис. 1. Архитектура инструмента MicroTESK

Fig. 1. Architecture of the MicroTESK instrument

Среда генерации использует *тестовые шаблоны* на специальном языке, основанном на Ruby [16]. Язык позволяет описывать сценарии тестирования на достаточно абстрактном уровне, когда состав, порядок и операнды команд не фиксируются, а выбираются динамически в зависимости от поставленной задачи. В шаблонах можно «обращаться» к регистрам и «вызывать» команды микропроцессора, как это делается на языке ассемблера — для этого на основе метаданных инструмент автоматически создается оберточные функции.

Процесс генерации тестовых программ (потока команд) по шаблону состоит из следующих стадий:

1. анализ тестового шаблона и построение внутреннего представления;
2. перебор абстрактных последовательностей команд и осуществление следующих действий для каждой из них:
    a. конкретизация последовательности команд: выбор регистров, генерация данных, построение инициализирующего кода;
    b. исполнение построенных команд на симуляторе;
    c. вставка в последовательность проверок на основе данных из симулятора (при генерации самопроверяющих программ);
    d. добавление последовательности в поток команд;
3. разбиение потока на программы и их печать в ассемблерном формате.

3.2 Спецификация команд ARMv8

Архитектура ARMv8 описывается на примерно 4000 страницах руководства «*ARM Architecture Reference Manual*» [5]. Описание включает архитектурные особенности (уровни исключений и т.п.), систему команд (около 1000 команд разных типов) и организацию подсистемы памяти (см. раздел 3.3). Всю совокупность команд можно разбить на следующие группы:

- *команды ветвления* (условные и безусловные переходы, исключения, возврат из обработчика исключений);
- *команды загрузки и сохранения данных* (обычные загрузки/сохранения для различных типов данных, загрузки/сохранения с блокировками);
- *вычислительные команды* (инициализация регистров, целочисленная арифметика и арифметика с плавающей точкой);
- *векторные команды* (SIMD — Single Instruction, Multiple Data),
- *системные команды* (работа с системными регистрами).

Состояние микропроцессора (точнее, одного его ядра) описывается набором значений регистров, соответствующих текущему уровню исполнения команд (EL — Exception Level). Всего поддерживается четыре уровня, переключение между которыми осуществляется специальными системными командами:

- EL0 — уровень пользовательских приложений (непривилегированное исполнение);
- EL1 — уровень ядра операционной системы (привилегированное исполнение);
- EL2 — уровень гипервизора;
- EL3 — уровень защищенного монитора.

Архитектурой определены следующие регистры:

- R0-R30 — 64-битные регистры общего назначения (с доступом ко всем 64 битам или только к младшим 32);
- SP — указатель стека текущего EL;
- PC — счетчик команд (недоступный программно);
- V0-V31 — 128-битные регистры для хранения чисел с плавающей точкой (с поддержкой разных форматов и возможностью векторной интерпретации одного регистра);
- FPCR и FPSR — контрольный и статусный регистры для операций над числами с плавающей точкой;
- несколько сотен системных регистров.

Все регистры ARMv8 и около 40% команд (см. раздел 4), характерных для мобильных устройств (все команды за исключением SIMD и арифметики с плавающей точкой), были специфицированы на языке nML [9].

Ниже в качестве иллюстрации приведено описание команды MOVZ (Move Wide with Zero), перемещающей 16-битные данные в регистр и обнуляющей 48 оставшихся битов.

```
// Константы, соответствующие разным режимам перемещения
let MoveWideOp_N = 0b00 // Перемещение с инверсией всех битов
let MoveWideOp_Z = 0b10 // Перемещение с обнулением остальных битов
let MoveWideOp_K = 0b11 // Перемещение без изменения остальных битов

// Общая операция (функция) для команд перемещения 16-битных данных
op MovWideImmGeneral (rd: REG, imm: HWORD, shift: card(2), opcode: card(2))
  action = {
    // Учет режима перемещения (обнуление битов)
    if opcode != MoveWideOp_K then rd = 0; endif;

    // Выбор поля регистра для записи
    if shift == 0 then
      rd<15..0> = imm;
    elif shift == 1 then
      rd<31..16> = imm;
    elif shift == 2 then
      rd<47..32> = imm;
    elif shift == 3 then
      rd<63..48> = imm;
    endif;

    // Учет режима перемещения (инверсия битов)
    if opcode == MoveWideOp_N then rd = ~rd; endif;
  }

// Команда перемещения 16-битных данных с обнулением остальных битов
op movz (rd: REG, imm: HWORD, shift: card(2))
  // Ассемблерный синтаксис
  syntax = format("movz %s, #0x%x, LSL #%d",
                  rd.syntax, imm, coerce(BYTE, shift) * 16)
  // Двоичный формат
  image = format("%s10100101%2s%16s%5s", coerce(BIT, 1), shift, imm, rd.image)
  // Совершаемые командой действия
  action = {
    MovWideImmGeneral(rd, imm, shift, MoveWideOp_Z).action;
  }
```

3.3 Спецификация подсистемы памяти ARMv8

Описание виртуальной памяти ARMv8 (VMSAv8-64) занимает около 600 страниц. Значительная часть документации посвящена механизму трансляции адресов. В VMSAv8-64 определены четыре режима трансляции: защищенный EL3, незащищенный EL2, EL1&0 в защищенном и незащищенном вариантах. Оба варианта режимов ELl&0 состоят в традиционном преобразовании виртуального адреса в физический. Режимы EL3 и EL2 включают два этапа:

сначала виртуальный адрес преобразуется в промежуточный физический, а затем промежуточный физический адрес — в физический. Каждое преобразование вызывает обход таблиц трансляции и может привести к четырем дополнительным обращениям к памяти. Для ускорения этого процесса таблицы трансляции кэшируются в ассоциативном буфере трансляции (TLB — Translation Lookaside Buffer).

Для спецификации подсистемы памяти MicroTESK использует специальный язык MMUSL, расширяющий возможности nML средствами описания типов адресов, сегментов, буферов и таблиц, а также логики обработки запросов к памяти. Были специфицированы TLB, таблицы трансляции, двухуровневая кэш-памяти и логика обработки запросов к памяти во всех режимах.

Ниже приведен пример, описывающий тип виртуального адреса — структуру, содержащую адрес и дополнительные данные о запросе к памяти.

```
// Тип виртуального адреса
address VA (
  // Виртуальный адрес
  addr : 64,
  // Дополнительные данные
  acctype: 4, iswrite: 1, wasaligned: 1, size: 6
)
```

Сегмент памяти задает отображение диапазона адресов некоторого типа на множество адресов другого типа. В следующем примере описан сегмент VA_LO_UNMAPPED, преобразующий виртуальные адреса в физические напрямую (без использования TLB и таблиц трансляции).

```
// Сегмент памяти (адресное пространство)
segment VA_LO_UNMAPPED (va: VA) = (pa: PA)
  // Диапазон отображаемых виртуальных адресов
  range = (0x0000000000000000, 0x0000ffffFFFFffff)
  // Способ преобразования виртуальных адресов в физические
  read = { pa.addrdesc.paddress.physicaladdress = va.vaddress<47..0>; }
```

Буферы (TLB, кэш-память, таблицы страниц и т.п.) характеризуются объемом, ассоциативностью, стратегией вытеснения данных и другими параметрами. Ниже приведен фрагмент спецификации TLB. Формат записи определен архитектурой, в то время как значения других параметров реализационно-зависимы. Ключевое слово **register** говорит о том, что буфер отображается на регистры, что делает его доступным из спецификации системы команд.

```
// Буфер ассоциативной трансляции, отображаемый на регистры
register buffer TLB (va: VA)
 // Ассоциативность
 ways = 64
 // Число множеств
 sets = 1
 // Формат записи
 entry = (addr: 36, nG: 1, contiguous: 1, level: 2, blocksize: 64,
          perms: Permissions, addrdesc: AddressDescriptor)
 // Предикат, проверяющий попадание в буфер
 match = (addr == va.addr<47..12> && addrdesc.fault.type == Fault_None)
 // Политика вытеснения данных
 policy = LRU
```

С помощью ключевого слова **memory** можно задать буферы, отображаемые на память: каждое обращение к ним вызывает «рекурсивный» доступ в память. Примером таких буферов являются таблицы трансляции VMSAv8-64.

```
// Таблица трансляции, отображаемая на память
memory buffer TranslationTable(va: VA)
 entry = (IGNORED: 9, XN: 1, PXN: 1, Contiguous: 1, RES0: 4, pa: 36, nG: 1,
          AF: 1, SH: 2, AP: 2, NS: 1, AttrIndx: 3, page: 1, valid: 1)
```

Логика загрузки и сохранения данных описывается запросами к сегментам и буферам. Синтаксис схож с nML, но допускает конструкции вида B(addr).**hit** (буфер B содержит запись для адреса addr), E = B(addr) (запись для адреса addr считывается из буфера B и сохраняется в E) и т.п. Ниже приведен фрагмент спецификации подсистемы памяти VMSAv8-64.

```
// Управляющая логика подсистемы памяти
mmu pmem (va: VA) = (data: 64)
  var pa: PA;
  var l1Entry: L1.entry;
  var line: 256;
  // Обработка запросов загрузки данных
  read = {
    if va.acctype != AccType_PTW then
     pa.addrdesc = AArch64TranslateAddress(va.vaddress, va.acctype, 0, 1, 8);
    else
      pa.addrdesc.paddress.physicaladdress = va.vaddress<47..0>;
      pa.size = va.size;
      pa.acctype = va.acctype;
    endif;
    if (va.acctype == AccType_AT) then update_PAR_EL1(pa); endif;
    if L1(pa).hit then
      l1Entry = L1(pa);
      line = l1Entry.DATA;
    else
      if va.acctype != AccType_PTW && va.acctype != AccType_AT then
        line = AArch64MemSingleRead(pa.addrdesc, 8, va.acctype, 1);
      else
        line = M(pa);
      endif;
      l1Entry.TAG = pa.addrdesc.paddress.physicaladdress<47..12>;
      l1Entry.DATA = line;
      L1(pa) = l1Entry;
    endif;
    data = get_word_from_line(va.vaddress<4..3>, line);
    if (va.acctype == AccType_IFETCH) then
      data<47..0> = pa.addrdesc.paddress.physicaladdress;
      data<63..48> = 0;
    endif;
  }

  // Обработка запросов сохранения данных
  write = {...}
```

3.4. Генерация тестовых программ для ARMv8

После того как разработаны спецификации и по ним автоматически построена модель микропроцессора, MicroTESK можно использовать для генерации тестовых программ. По сути, результатом анализа спецификаций является генератор тестовых программ, настроенный на конкретную архитектуру, или даже на конкретный микропроцессор. Программы строятся по шаблонам на языке Ruby, описывающим сценарии тестирования с помощью специально введенных в язык конструкций [16]. Важно отметить, что для команд

микропроцессора создается оберточные функции, что позволяет использовать в шаблонах нотацию, максимально приближенную к ассемблерной.

MicroTESK использует разные техники построения тестовых программ: случайные, комбинаторные, основанные на разрешении систем ограничений. Например, приведенный ниже шаблон порождает 10 тестовых воздействий, каждый из которых состоит из одной команды ADD (add_sh_reg), номера регистров которой и их значения генерируются случайно.

```
// Класс на языке Ruby, определяющий тестовый шаблон
class RandomTemplate < ArmV8BaseTemplate
  // Главный метод тестового шаблона
  def run
    // Описание распределения вероятностей для значений регистров
    integer_dist = dist(range(:value => 0x00000000..0x0000000F, :bias => 50),
                        range(:value => 0xffffFFF0..0xffffFFFF, :bias => 50))

    // Описание структуры тестового воздействия
    sequence {
      add_sh_reg reg(_), reg(_), reg(_), LSL, 0
        // Задание способа генерации тестовых данных
        do situation('random_biased', :dist => integer_dist) end
    }.run 10 // Число тестовых воздействий данного типа
  end
end
```

Следующий пример демонстрирует комбинаторную генерацию и генерацию на основе ограничений. Входные данные для команд ADD (adds_sh_reg) и SUB (subs_sh_reg) строятся таким образом, чтобы при их исполнении либо гарантированно не возникало никаких исключений (ситуация normal), либо гарантированно возникало переполнение (ситуация overflow). Конструкция **block** комбинирует (параметр combinator) и смешивает (параметр compositor) последовательности команд, возвращаемые вложенными блоками.

```
// Блок внешнего уровня, комбинирующий и смешивающий
// последовательности, возвращаемые вложенными блоками
block(:combinator => 'product', :compositor => 'random') {
  // Вложенный блок, возвращающий 2 последовательности,
  // каждая из которых содержит одну команду ADD
  iterate {
    // Команда ADD, не вызывающая исключений
    adds_sh_reg reg(_), x0, x1, LSL, 0 do situation('normal') end
    // Команда ADD, вызывающая переполнение
    adds_sh_reg reg(_), x2, x3, LSL, 0 do situation('overflow') end
  }
  // Вложенный блок, возвращающий 2 последовательности,
  // каждая из которых содержит одну команду SUB
  iterate {
    // Команда SUB, не вызывающая исключений
    subs_sh_reg reg(_), x0, x1, LSL, 0 do situation('normal') end
    // Команда SUB, вызывающая переполнение
    subs_sh_reg reg(_), x2, x3, LSL, 0 do situation('overflow') end
  }
}.run
```

Как отмечалось в разделе 3.1, для каждого тестового воздействия строится инициализирующий код, устанавливающий требуемое состояние регистров, буферов и памяти. Для этого используются так называемые препараторы, определенные в базовом тестовом шаблоне. Препаратор — это фрагмент кода, направленный на достижение той или иной цели: инициализации регистра, буфера или памяти. Как правило, число препараторов невелико; они задаются вручную.

Ниже представлен препаратор для начальной таблицы трансляции уровня EL1. Он размещает специальным образом отформатированную запись (entry) по определенному адресу (address).

```
// Препаратор начальной таблицы трансляции
buffer_preparator (:target => 'TranslationTable', :level => 0) {
  mrs x0, ttbr0_el1
  movz x1, address, 0
  add_sh_reg x0, x0, x1, LSL, 3
  movk x2, entry(0, 15), 0
  movk x2, entry(16, 31), 1
  movk x2, entry(32, 47), 2
  movk x2, entry(48, 63), 3
  stlr x2, x0
}
```

4. Характеристики генератора MicroTESK-ARMv8

Разработанный генератор тестовых программ для ARMv8 отвечает основным требованиями, предъявляемым к промышленным инструментам такого рода:

- поддержка разных техник построения тестовых программ;
- возможность построения самопроверяющих тестов;
- учет архитектурных и микроархитектрных особенностей (условий возникновения исключений, числа вычислительных ядер и потоков, параметров кэш-памяти и т.п.).

В работе по созданию формальных спецификаций ARMv8 участвовало 2 человека; продолжительность работ составила 10 месяцев. За это время были описаны 382 команды (все команды за исключением SIMD и арифметики с плавающей точкой) и подсистема памяти. В целом были специфицированы уровни EL0 и EL1. Объем кода составил около 7800 строк для системы команд (nML) и около 2000 строк для подсистемы памяти (MMUSL). В таблице 1 приведены сведения о специфицированных командах.

Табл. 1. Статистика по специфицированным командам

Table 1. Statistics on specified commands

Класс команд	**Число специфицированных команд**
(1) Передача управления	12
(2) Сохранение в память и загрузка из памяти	85
(3) Арифметика с непосредственными операндами	68
(4) Арифметика с регистровыми операндами	121
(5) Расширенная арифметика	22
(6) Арифметики с плавающей точкой	42 (из 168)
(7) Команды SIMD	0 (из 477)
(8) Системные команды	32
Всего	**382 (из 985)**

В настоящее время MicroTESK-ARMv8 успешно используется в известной международной компании, разрабатывающей собственные микропроцессоры архитектуры ARMv8.

5. Заключение

В работе рассмотрен генератор тестовых программ для архитектуры ARMv8, построенный с использованием инструмента MicroTESK. Решение базируется на формальных спецификациях системы команд и подсистемы памяти. По своим функциям MicroTESK-ARMv8 не уступает аналогам, но превосходит их в гибкости настройки на целевую архитектуру (архитектура ARMv8 является молодой и постоянно развивается: выходят дополнения, например, ARMv8.1-A, которые должны отражаться в генераторе).

В будущем мы планируем реализовать дополнительные средства MicroTESK, полезные при промышленном использовании этого инструмента: генераторы отчетов о тестовом покрытии, достигаемом сгенерированными программами; средства автоматического построения типовых тестов; инструментарий для конструирования online-генераторов тестовых программ, т.е. генераторов,

работающих на «железе» (ПЛИС-прототипе микропроцессора или опытном образце СБИС).

Список литературы

[1]. Сайт компании ARM. http://www.arm.com.
[2]. Mallya H. The Backstory of How ARM Reached a Milestone of 86 Billion Chips in 25 Years. July 19, 2016 (https://yourstory.com/2016/07/arm-holdings-story/).
[3]. Morgan T.P. ARM Holdings Eager for PC and Server Expansion. Record 2010, Looking for Intel Killer 2020. February 1, 2011 (http://www.theregister.co.uk/2011/02/01/arm_holdings_q4_2010_numbers/).
[4]. Sims G. Custom Cores versus ARM Cores, What Is It All About? January 7, 2016 (http://www.androidauthority.com/arm-cortex-core-custom-core-kryo-explained-664777/).
[5]. ARM Architecture Reference Manual. ARM DDI 0487A.f, ARM Corporation, 2015. 5886 p.
[6]. Камкин А.С. Генерация тестовых программ для микропроцессоров. Труды ИСП РАН, том 14, часть 2, 2008. с. 23-64.
[7]. Adir A., Almog E., Fournier L., Marcus E., Rimon M., Vinov M., Ziv A. Genesys-Pro: Innovations in Test Program Generation for Functional Processor Verification. Design & Test of Computers, 21(2), 2004. pp. 84-93. DOI: 10.1109/MDT.2004.1277900.
[8]. Kamkin A., Tatarnikov A. MicroTESK: An ADL-Based Reconfigurable Test Program Generator for Microprocessors. Spring/Summer Young Researchers' Colloquium on Software Engineering, 2012. pp. 64-69. DOI: 10.15514/SYRCOSE-2012-6-8.
[9]. Freericks M. The nML Machine Description Formalism. Technical Report TR SM-IMP/DIST/08, TU Berlin CS Department, 1993.
[10]. Chupilko M., Kamkin A., Kotsynyak A., Protsenko A., Smolov S., Tatarnikov A. Specification-Based Test Program Generation for ARM VMSAv8-64 Memory Management Units. Workshop on Microprocessor Test and Verification, 2015. pp. 1-6. DOI: 10.1109/MTV.2015.13.
[11]. Hrishikesh M.S., Rajagopalan M., Sriram S., Mantri R. System Validation at ARM — Enabling our Partners to Build Better Systems. White Paper. April 2016 (http://www.arm.com/files/pdf/System_Validation_at_ARM_Enabling_our_partners_to_build_better_systems.pdf).
[12]. Venkatesan D., Nagarajan P. A Case Study of Multiprocessor Bugs Found Using RIS Generators and Memory Usage Techniques. Workshop on Microprocessor Test and Verification, 2014. pp. 4-9. DOI: 10.1109/MTV.2014.28.
[13]. Hudson J., Kurucheti G. A Configurable Random Instruction Sequence (RIS) Tool for Memory Coherence in Multi-processor Systems. Workshop on Microprocessor Test and Verification, 2014. pp. 98-101. DOI: 10.1109/MTV.2014.26.
[14]. RAVEN test program generator (http://www.slideshare.net/DVClub/introducing-obsidian-software-andravengcs-for-powerpc).
[15]. Adir A., Fournier L., Katz Y., Koyfman A. DeepTrans – Extending the Model-based Approach to Functional Verification of Address Translation Mechanisms. High-Level Design Validation and Test Workshop, 2006. pp. 102-110.
[16]. Tatarnikov A.D. Language for Describing Templates for Test Program Generation for Microprocessors. Труды ИСП РАН, vol. 28, issue 4, 2016. pp. 81-102. DOI: 10.15514/ISPRAS-2016-28(4)-5

MicroTESK-Based Test Program Generator for the ARMv8 Architecture

[1,2,3] *A.S. Kamkin <kamkin@ispras.ru>*
[1] *A.M. Kotsynyak <kotsynyak@ispras.ru>*
[1] *A.S. Protsenko <protsenko@ispras.ru>*
[1] *A.D. Tatarnikov <andrewt@ispras.ru>*
[1] *M.M. Chupilko <chupilko@ispras.ru>*
[1] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*
[3] *Moscow Institute of Physics and Technology, 9 Institutskiy per., Moscow Region, Dolgoprudny, 141700, Russia.*

Abstract. ARM is a family of microprocessor instruction set architectures developed in a company with the same name. The newest architecture of this family, ARMv8, contains a large number of instructions of various types and is notable for its complex organization of virtual memory, which includes hardware support for multilevel address translation and virtualization. All of this makes functional verification of microprocessors with this architecture an extremely difficult technical task. An integral part of microprocessor verification is generation of test programs, i.e. programs in the assembly language, which cause various situations (exceptions, pipeline stalls, branch mispredictions, data evictions in caches, etc.). The article describes the requirements for industrial test program generators and presents a generator for microprocessors with the ARMv8 architecture, which has been developed with the help of MicroTESK (Microprocessor TEsting and Specification Kit). The generator supports an instruction subset typical for mobile applications (about 400 instructions) and consists of two main parts: (1) an architecture-independent core and (2) formal specifications of ARMv8 or, more precisely, a model automatically constructed on the basis of the formal specifications. With such a structure, the process of test program generator development consists mainly in creation of formal specifications, which saves efforts by reusing architecture-independent components. An architecture is described using the nML and MMUSL languages. The first one allows describing the microprocessor registers and syntax and semantics of the instructions. The second one is used to specify the memory subsystem organization (address spaces, various buffers and tables, address translation algorithms, etc.) The article describes characteristics of the developed generator and gives a comparison with the existing analogs.

Keywords: microprocessors; instruction set specification; functional verification; test program generation; ARM; nML; MicroTESK.

DOI: 10.15514/ISPRAS-2016-28(6)-6

For citation: Kamkin A.S., Kotsynyak A.M., Protsenko A.S., Tatarnikov A.D., Chupilko M.M. MicroTESK-Based Test Program Generator for the ARMv8 Architecture. *Trudy ISP RAN/Proc. ISP RAS*, vol. 28, issue 6, 2016, pp. 87-102 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-6

References

[1]. ARM site. http://www.arm.com.

[2]. Mallya H. The Backstory of How ARM Reached a Milestone of 86 Billion Chips in 25 Years. July 19, 2016 (https://yourstory.com/2016/07/arm-holdings-story/).

[3]. Morgan T.P. ARM Holdings Eager for PC and Server Expansion. Record 2010, Looking for Intel Killer 2020. February 1, 2011 (http://www.theregister.co.uk/2011/02/01/arm_holdings_q4_2010_numbers/).

[4]. Sims G. Custom Cores versus ARM Cores, What Is It All About? January 7, 2016 (http://www.androidauthority.com/arm-cortex-core-custom-core-kryo-explained-664777/).

[5]. ARM Architecture Reference Manual. ARM DDI 0487A.f, ARM Corporation, 2015. 5886 p.

[6]. Kamkin A. Test Program Generation for Microprocessors. Trudy ISP RAN / Proc. ISP RAS, volume 14, part 2, 2008, pp. 23-64 (in Russian).

[7]. Adir A., Almog E., Fournier L., Marcus E., Rimon M., Vinov M., Ziv A. Genesys-Pro: Innovations in Test Program Generation for Functional Processor Verification. Design & Test of Computers, 21(2), 2004. pp. 84-93. DOI: 10.1109/MDT.2004.1277900.

[8]. Kamkin A., Tatarnikov A. MicroTESK: An ADL-Based Reconfigurable Test Program Generator for Microprocessors. Spring/Summer Young Researchers' Colloquium on Software Engineering, 2012. pp. 64-69. DOI: 10.15514/SYRCOSE-2012-6-8.

[9]. Freericks M. The nML Machine Description Formalism. Technical Report TR SM-IMP/DIST/08, TU Berlin CS Department, 1993.

[10]. Chupilko M., Kamkin A., Kotsynyak A., Protsenko A., Smolov S., Tatarnikov A. Specification-Based Test Program Generation for ARM VMSAv8-64 Memory Management Units. Workshop on Microprocessor Test and Verification, 2015. pp. 1-6. DOI: 10.1109/MTV.2015.13.

[11]. Hrishikesh M.S., Rajagopalan M., Sriram S., Mantri R. System Validation at ARM — Enabling our Partners to Build Better Systems. White Paper. April 2016 (http://www.arm.com/files/pdf/System_Validation_at_ARM_Enabling_our_partners_to_build_better_systems.pdf).

[12]. Venkatesan D., Nagarajan P. A Case Study of Multiprocessor Bugs Found Using RIS Generators and Memory Usage Techniques. Workshop on Microprocessor Test and Verification, 2014. pp. 4-9. DOI: 10.1109/MTV.2014.28.

[13]. Hudson J., Kurucheti G. A Configurable Random Instruction Sequence (RIS) Tool for Memory Coherence in Multi-processor Systems. Workshop on Microprocessor Test and Verification, 2014. pp. 98-101. DOI: 10.1109/MTV.2014.26.

[14]. RAVEN test program generator (http://www.slideshare.net/DVClub/introducing-obsidian-software-andravengcs-for-powerpc).

[15]. Adir A., Fournier L., Katz Y., Koyfman A. DeepTrans – Extending the Model-based Approach to Functional Verification of Address Translation Mechanisms. High-Level Design Validation and Test Workshop, 2006. pp. 102-110.

[16]. Tatarnikov A.D. Language for Describing Templates for Test Program Generation for Microprocessors. Trudy ISP RAN / Proc. ISP RAS, volume 28, part 4, 2016. pp. 81-102. DOI: 10.15514/ISPRAS-2016-28(4)-5

Подходы к представлению результатов анализа сетевого трафика*

[1] *А. И. Гетьман <thorin@ispras.ru>,*
[1] *Ю. В. Маркин <ustas@ispras.ru>,*
[1] *Д. О. Обыденков, <obydenkov@ispras.ru>,*
[1,2] *В. А. Падарян <vartan@ispras.ru>,*
[1] *А. Ю. Тихонов <fireboo@ispras.ru>*
[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына,д. 25
[2]*Московский государственный университет имени М.В. Ломоносова,*
119991 ГСП-1, Москва, Ленинские горы

Аннотация. В статье предложены различные способы представления результатов анализа сетевого трафика, необходимость в которых возникает прежде всего в задачах обеспечения сетевой информационной безопасности. Рассмотрена возможность построения полного графа сетевых взаимодействий, а также создания временной диаграммы передачи пакетов. Эти компоненты используются при расследовании инцидентов нарушения ИБ. Временная диаграмма также применяется при анализе туннельных протоколов, поскольку позволяет аналитику определить, какие именно заголовки протоколов необходимо визуализировать. Для задач, связанных с обратной инженерией, а также отладкой сетевых протоколов, предлагается использовать журнал, в котором фиксируются ошибки разбора заголовков протоколов. Представленные графические компоненты либо не имеют аналогов среди opensource-инструментов, либо улучшают уже существующие opensource-решения.

Ключевые слова: анализ сетевого трафика; отладка сетевых протоколов; граф сетевых взаимодействий; визуализация; журнал ошибок разбора.

DOI: 10.15514/ISPRAS-2016-28(6)-7

Для цитирования: Гетьман А.И., Маркин Ю.В. Обыденков Д.О., Падарян В.А., Тихонов А.Ю. Подходы к представлению результатов анализа сетевого трафика. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 103-110. DOI: 10.15514/ISPRAS-2016-28(6)-7

* Работа поддержана грантом РФФИ 15-07-07652 А

1. Введение

Во многих задачах обеспечения сетевой информационной безопасности требуется детальный анализ сетевого трафика. Среди таких задач можно выделить:

- расследование инцидентов нарушения ИБ
- анализ сетевой обстановки
- обратная инженерия/отладка сетевых протоколов

Подобные задачи, как правило, решаются не "на потоке", а путем выделения некоторого фрагмента трафика с помощью программы-сниффера и его последующего анализа. При решении этих задач критическими факторами, влияющими на скорость и эффективность, являются наличие в среде анализа графических компонент, позволяющих визуализировать различные аспекты сетевых взаимодействий, и возможности по переключению и синхронизации между этими компонентами.

2. Обзор существующих средств анализа

Большинство популярных инструментов анализа сетевого трафика либо не имеют графического интерфейса (Snort [1], The Bro Network Security Monitor [2]), либо изначально разрабатывались для решения других задач и удовлетворяют указанным требованиям лишь частично [3]. Наиболее популярным инструментом из второй группы является инструмент Wireshark [4]. Основным средством представления сетевой трассы в нём являются разобранные пакеты в виде списка, при этом только для одного выделенного пакета отображается полный стек протоколов и значения полей в заголовках этих протоколов. Пакет, как элемент списка, представляется посредством строки, состоящей из значений фиксированного набора полей, выделенных в заголовке протокола сетевого уровня (IP-адреса), а также полей заголовка протокола самого высокого уровня, который удалось разобрать. Фиксированность представления пакетов может вызывать трудности во многих случаях. В частности, это проявляется при анализе туннельных протоколов. Так протокол GRE [5] (рис. 1) предназначен для инкапсуляции пакетов сетевого уровня модели OSI в IP-пакеты: сетевой пакет, таким образом, содержит два заголовка протокола IP. В списке пакетов инструмент Wireshark отобразит поля последнего (верхнего) IP-заголовка, который с точки зрения туннельного соединения не является репрезентативным: чтобы понять, какой хост из внутренней сети инициировал взаимодействие, аналитику придется выделять каждый пакет и проверять значение поля сетевого адреса нижележащего IP-заголовка.

Другим способом визуализации сетевых соединений в Wireshark является просмотр иерархий протоколов [6], присутствующих в трассе. Однако отображение осуществляется в виде общего дерева протоколов с указанием некоторой суммарной статистики соединений, но без возможности просмотра

и перехода к конкретным представителям соединений, соответствующих заданному типу вложенности протоколов, для просмотра параметров этих соединений и дальнейшего их анализа (рис. 2).

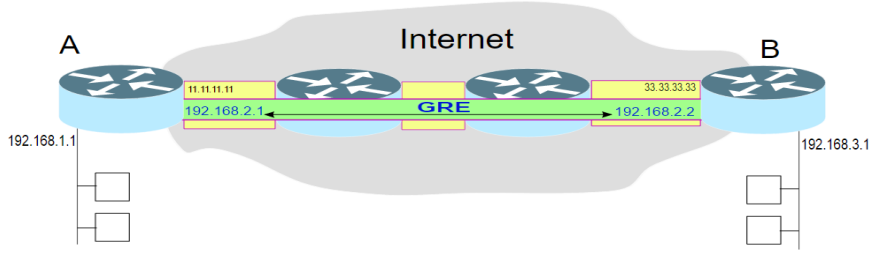


Рис. 1. Пример организации GRE-туннеля

Fig. 1. GRE tunneling scheme.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	3000	100.0	2803776	160 k	0	0	0
Ethernet	100.0	3000	1.5	42000	2401	0	0	0
Internet Protocol Version 4	99.9	2996	2.1	59920	3426	0	0	0
User Datagram Protocol	34.0	1021	0.3	8168	467	0	0	0
Remote Procedure Call	33.9	1016	58.9	1651128	94 k	0	0	0
Network File System	33.9	1016	57.1	1602320	91 k	1016	1602320	91 k
Transmission Control Protocol	8.5	256	0.5	14244	814	103	3296	188
SSH Protocol	2.6	77	0.2	4752	271	77	4752	271
Rlogin Protocol	2.5	76	0.0	1300	74	76	1300	74
Stream Control Transmission Protocol	1.6	47	0.1	2540	145	33	1508	86
MTP 3 User Adaptation Layer	0.5	14	0.0	640	36	6	64	3
Signalling Connection Control Part	0.3	8	0.0	244	13	0	0	0
Malformed Packet	0.0	1	0.0	0	0	1	0	0
Internet Control Message Protocol	0.1	3	0.0	228	13	3	228	13
Data	56.0	1681	85.8	2406409	137 k	1681	2406409	137 k
Address Resolution Protocol	0.1	4	0.0	112	6	4	112	6

Рис. 2. Пример отображения вложенности взаимодействий в Wireshark.

Fig. 2. Wireshark's protocol hierarchy statistics.

3. Реализованные формы представления результатов анализа и их применение

При расследовании инцидента нарушения ИБ необходимо локализовать сетевые соединения, посредством которых этот инцидент возник и развивался во времени: аналитик должен обладать некоторым критерием (или множеством таких критериев) на содержимое сетевых пакетов. Одним из подходов к решению задачи локализации является представление сетевых взаимодействий посредством графа, в котором вершинам соответствуют стороны сетевого взаимодействия, а ребра отображают факт взаимодействия и возможно некоторые его характеристики, такие как интенсивность. При этом одна и та же сторона может участвовать сразу в нескольких взаимодействиях. Далее требуется провести детальный анализ выделенных соединений:

- проследить за порядком отправки/получения пакетов
- просмотреть значения полей интересующих протоколов

- восстановить данные протоколов прикладного уровня

Рассмотренные opensource-инструменты не предоставляют графических компонентов для работы с такими сценариями.

Предлагаемые компоненты представления результатов разбора опираются на модель описания данных [7], используемую ядром системы анализа, разрабатываемой в ИСП РАН. В отличие от Wireshark все сетевые пакеты (а не только выбранный пользователем) отображаются посредством дерева с выделенными заголовками инкапсулируемых протоколов (рис. 3). Таким образом, не возникает трудностей при работе с туннельными протоколами.

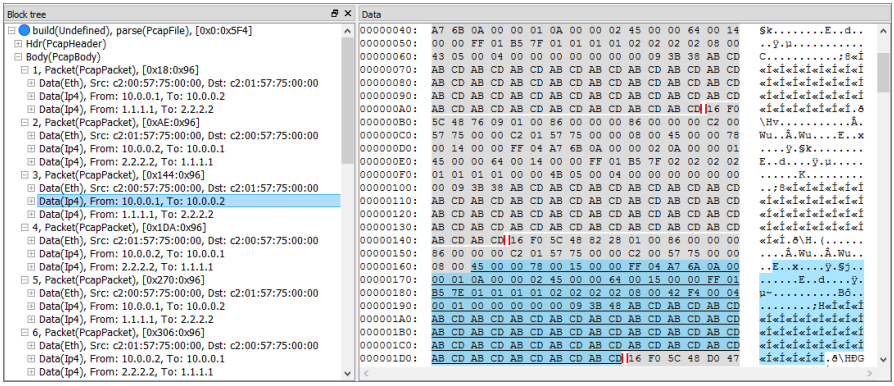


Рис. 3. Пример отображения стека протоколов для нескольких пакетов.

Fig. 3. ProtoSphere's protocol stack visualization for several packets.

Предлагается два способа визуализации сетевых взаимодействий:

- граф оконечных узлов (*Endpoints*)
- граф, детализирующий сетевые взаимодействия выбранного оконечного узла (*Nodes*)

Оба графа строятся по *дереву сетевых узлов*. Сетевой узел – это обобщение понятий отправителя и получателя для сетевых протоколов. Например, для протокола IPv4 сетевой узел описывает IP-адрес, тогда как для протокола TCP – порт. Вершина (сетевой узел) **B** дерева является дочерней по отношению к вершине **A**, если **B** характеризует отправителя (получателя) в заголовке некоторого протокола, вложенном (в рамках сетевого пакета) в заголовок нижележащего (согласно модели OSI) протокола, в котором выделен отправитель (получатель) **A**.

Граф *Endpoints* (рис. 4а) отображает сетевые соединения, относящиеся к протоколу самого низкого уровня в анализируемом файле. В качестве вершин здесь как правило выступают MAC- или IP-адреса. Ребра соединяют сетевые узлы, между которыми был передан хотя бы один сетевой пакет.

Граф *Nodes* (рис. 4б) детализирует сетевые взаимодействия заданного оконечного узла. При этом для каждого взаимодействия отображается весь стек сетевых протоколов. Дополнительно осуществляется фильтрация графа *Nodes* по содержимому сетевых пакетов, передаваемых между узлами.

Заметим, что комбинация графов *Endpoints* и *Nodes* также позволяет визуализировать результаты задачи расследования сетевой обстановки, когда по трафику требуется определить, какие сетевые службы (приложения) запущены на том или ином компьютере в сети.

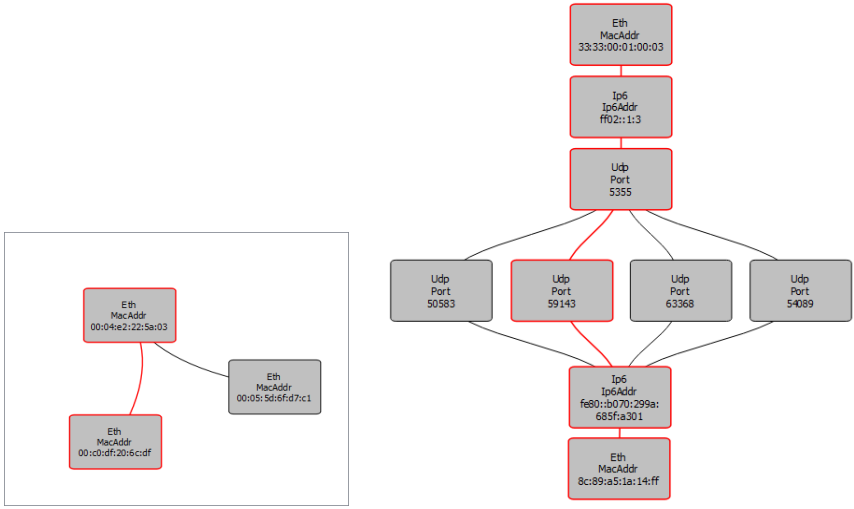


Рис. 4. (а) Пример графа Endpoints (б) Пример графа Nodes.

Fig. 4. (a) Endpoints graph (b) Nodes graph.

Для проведения детального анализа отдельного взаимодействия предлагается временна́я диаграмма, где каждый пакет отображается в виде стрелки с указанием отправителя и получателя (рис. 5). При этом можно указать, какие заголовки протоколов и какие поля в них должны отображаться над стрелками. Таким образом аналитик может адаптировать графический компонент под свои нужды. Следует отметить, что в анализаторе Wireshark есть аналогичный компонент [8], однако в нём отсутствует возможность настройки полей, значения которых будут отображаться для каждого пакета, что может быть неудобно, если интересующее поле отсутствует, или приводить к перегруженности отображения, если полей слишком много.

Вложенность взаимодействий с указанием параметров соединений отображается в дереве разбора (рис. 6). Именно в таком виде ядро инструмента хранит результаты анализа.

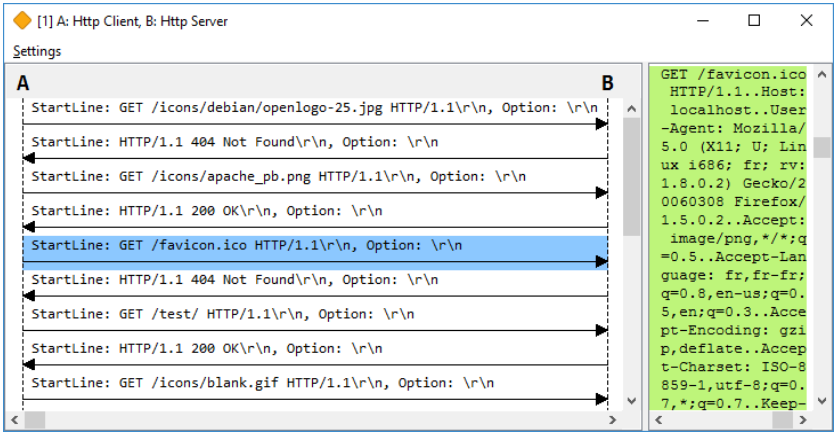


Рис. 5. Пример временно́й диаграммы.

Fig. 5. Time-based graph.

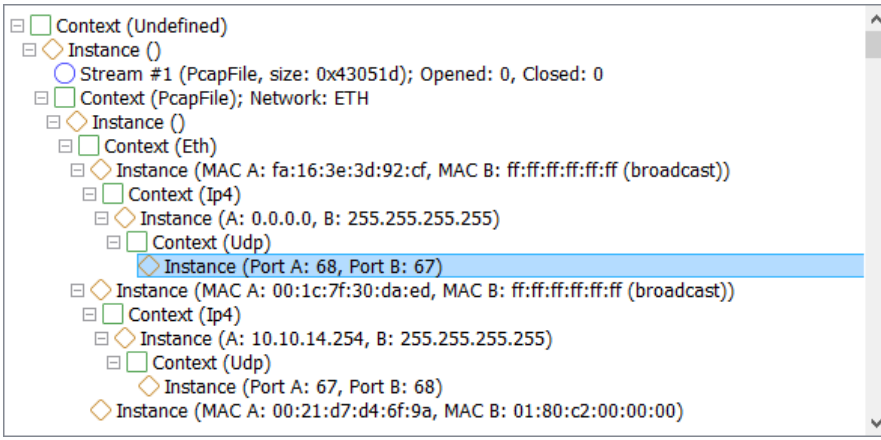


Рис. 6. Пример дерева разбора с отображением параметров соединений.

Fig. 6. ProtoSphere's protocol hierarchy statistics with display of connection settings.

При решении задач, связанных с обратной инженерией, а также отладкой сетевых протоколов, возникает необходимость в фиксации ошибок, возникающих при разборе заголовков этих протоколов. Анализаторы сетевого трафика, как правило, имеют модульную структуру: со временем появляются новые сетевые протоколы, и их необходимо поддерживать. Поддержка заключается в создании модуля, в котором локализована функциональность по работе с новым протоколом. При эксплуатации модуля могут проявляться ошибки разбора – несоответствия между кодом разборщика и данными, разбор которых осуществляется посредством данного разборщика. Журнал

ошибок, реализованный в системе, позволяет быстро локализовать такого рода ошибки. Ошибка разбора описывается текстовым сообщением и ссылкой на место ей возникновения – соответствующий пакет или сетевой сеанс. При работе с журналом поддерживается возможность фильтрации ошибок по протоколам.

Описанные графические компоненты синхронизированы друг с другом и допускают быстрое переключение между различными представлениями для повышения эффективности решения прикладных задач анализа сетевого трафика.

Список литературы

[1]. Snort. https://www.snort.org/, дата обращения: 10.10.2016
[2]. The Bro Network Security Monitor. https://www.bro.org/, дата обращения: 10.10.2016
[3]. Ю. В. Маркин, А. С. Санаров. Обзор современных инструментов анализа сетевого трафика. Препринты ИСП РАН, № 27, 2014
[4]. Wireshark. https://www.wireshark.org/, дата обращения: 10.10.2016
[5]. IETF RFC 2784. D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, Generic Routing Encapsulation, March 2000
[6]. The Protocol Hierarchy window. https://www.wireshark.org/docs/wsug_html_chunked/ChStatHierarchy.html, дата обращения: 10.10.2016
[7]. Гетьман А.И., Маркин Ю.В., Падарян В.А., Тихонов А.Ю.. Модель представления данных при проведении глубокого анализа сетевого трафика. Труды ИСП РАН, том 27, вып. 4, 2015 г., стр. 5-22. DOI: 10.15514/ISPRAS-2015-27(4)-1
[8]. Robert Shimonski. The Wireshark Field Guide: Analyzing and Troubleshooting Network Traffic. Elsevier Science & Technology Books, 2013, 128 p.

Methods of presenting the results of network traffic analysis★

[1]A. I. Get'man <thorin@ispras.ru>,
[1]Yu. V. Markin <ustas@ispras.ru>,
[1]D. O. Obydenkov <obydenkov@ispras.ru>,
[1,2]V. A. Padaryan <vartan@ispras.ru>,
[1]A. Yu. Tikhonov <fireboo@ispras.ru>
[1] Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[2]Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

★ This work is supported by RFBR grant 15-07-07652 A

Abstract. The article proposes different methods of presenting network traffic analysis results, the need for which arises primarily in the area of network security. One of the most important tasks is to identify malicious traffic. For this purpose both the complete graph of network interactions and time-based packet diagram are presented. These components are used during investigation of information security violation incidents. The timing diagram is also used in analysis of tunneling protocols because it allows the analyst to determine which protocol headers are necessary to visualize. For tasks associated with reverse engineering and debugging of network protocols, it is proposed to use a journal which records protocol header parsing errors. Presented graphic components either have no analogues among the opensource tools or improve on existing opensource solutions.

Keywords: network traffic analysis, network protocols debugging, graph of network interactions, visualization, error log.

DOI: 10.15514/ISPRAS-2016-28(6)-7

For citation: Get'man A. I., Markin Yu. V., Obydenkov D. O., Padaryan V. A., Tikhonov A. Yu. Methods of presenting the results of network traffic analysis. *Trudy ISP RAN/Proc. ISP RAS*, vol. 28, issue 6, 2016, pp. 103-110 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-7

References

[1]. Snort. https://www.snort.org/, accessed 10.10.2016
[2]. The Bro Network Security Monitor. https://www.bro.org/, accessed 10.10.2016
[3]. Yu. V. Markin, A. S. Sanarov. The modern network traffic analyzers overview. Preprinty ISP RAN [Preprints of ISP RAS], №27, 2014 (in Russian)
[4]. Wireshark. https://www.wireshark.org/, accessed 10.10.2016
[5]. IETF RFC 2784. D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, Generic Routing Encapsulation, March 2000
[6]. The Protocol Hierarchy window. https://www.wireshark.org/docs/wsug_html_chunked/ChStatHierarchy.html, accessed 10.10.2016
[7]. Get'man A. I., Markin Yu. V., Padaryan V. A., Tikhonov A. Yu. Model of data handling for in-depth analysis of network traffic. Trudy ISP RAN / Proc. ISP RAS, 2015, vol. 27, issue. 4, pp. 5-22 (in Russian). DOI: 10.15514/ISPRAS-2015-27(4)-1
[8]. Robert Shimonski. The Wireshark Field Guide: Analyzing and Troubleshooting Network Traffic. Elsevier Science & Technology Books, 2013, 128 p.

Создание виртуальных кластеров Apache Spark в облачных средах с использованием систем оркестрации[1]

[1]О. Д. Борисенко <al@somestuff.ru>
[1]Р. К. Пастухов <pastukhov@ispras.ru>
[1,2,3]С. Д. Кузнецов <kuzloc@ispras.ru>
[1]Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, дом 25
[2]Московский государственный университет имени М.В. Ломоносова,
119991 ГСП-1 Москва, Ленинские горы, МГУ имени М.В. Ломоносова, 2-й учебный корпус, факультет ВМК
[3]Московский физико-технический институт,
141700, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. Apache Spark является одним из наиболее производительных распределенных фреймворков для обработки больших данных в парадигме Map-Reduce. С распространением облачных технологий и предоставления ресурсов по запросу все более актуальной становится задача построения виртуальных вычислительных кластеров для конкретной задачи. В работе представлен краткий обзор разработанного решения для создания виртуальных кластеров Apache Spark в облачной среде Openstack и подведение итогов исследования о способах создания виртуальных кластеров Apache Spark в открытых облачных средах. Решение построено с использованием системы оркестрации Ansible. В работе будет проведено качественное сравнение разработанных в ИСП РАН подходов к решению задачи.

Ключевые слова: Apache Spark; Openstack; Amazon EC2; Map-Reduce; HDFS; виртуальные кластеры; облачные вычисления; Big Data; Apache Ignite.

DOI: 10.15514/ISPRAS-2016-28(6)-8

[1] Работа выполнена при поддержке гранта РФФИ No14-07-00602 А «Исследование и разработка методов автоматизации масштабирования и разворачивания виртуальных кластеров для обработки сверхбольших объёмов данных в облачной среде Openstack».

Для цитирования: Борисенко О.Д., Пастухов Р.К., Кузнецов С.Д. Создание виртуальных кластеров Apache Spark в облачных средах с использованием систем оркестрации. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 111-120. DOI: 10.15514/ISPRAS-2016-28(6)-8

1. Введение

Проект Apache Spark [1] является одной из наиболее развитых и производительных [2] реализаций подхода Map-Reduce [3]. В то же время данный проект является частью инфраструктуры Apache Software Foundation для обработки больших данных и обладает возможностями совместной работы с другими проектами этой инфраструктуры (такими как Apache Hadoop [4], YARN [5], Mesos [6], Ignite [7] и другими). Каждое из перечисленных программных решений масштабируется с использованием распределенного режима работы.

Однако настройка каждого из решений в распределенном окружении является очень трудоемкой задачей, требующей глубокого понимания принципов работы и точек взаимодействия каждой из систем. Кроме того, настройка взаимодействия инструментов анализа больших данных в облачной среде в ручном режиме является экономически неоправданной, поскольку облачные среды предоставляют ресурсы по запросу с оплатой за время использования ресурса. При таком подходе ручная настройка виртуального кластера обладает сразу несколькими недостатками: во-первых, пользователь системы должен оплачивать фактический простой вычислительных ресурсов; причем чем больше вычислительный кластер, тем дольше происходит настройка без средств автоматизации процесса. Во-вторых, пользователь вынужден уничтожать виртуальный кластер после расчетов, поскольку оплата берется не за фактически использованные, а за зарезервированные ресурсы. В этом случае при появлении новых задач кластер необходимо настраивать заново.

Предлагаемые в работе подходы позволяют избежать процесса настройки виртуальных кластеров в открытой облачной среде и избежать траты временных и физических ресурсов множества исследователей. В качестве облачной среды выбран проект Openstack [8] как наиболее динамически развивающийся и предоставляющий наиболее широкий спектр возможностей среди аналогов.

2. Построение решения

На момент начала исследований в 2014 году существовало ровно одно решение для автоматизации создания виртуальных кластеров Apache Spark. Оно было предложено разработчиками Apache Spark и было совместимо исключительно со средой Amazon EC2 [9]. Настройка окружения подразумевала использование заранее настроенного образа виртуальной

машины (AMI) на базе Red Hat Enterprise Linux 5.3. Предлагаемый разработчиками Apache Spark способ обладал следующими недостатками:

- решение использовало API закрытой облачной платформы Amazon EC2 и было непереносимым на другие облачные среды;
- использование заранее настроенного образа системы существенно снижало выбор компонентов пользователем;
- базовый образ на основе RHEL 5.3 ограничивал использование образа пользователем с лицензионной точки зрения и мог использоваться только в среде Amazon;
- среда Amazon EC2 не позволяла выгружать образы виртуальных машин из собственной среды.

В ходе исследований были изучены основные подходы к созданию виртуальных кластеров в Openstack.

Первый подход [10] заключался в переносе функциональности решения от разработчиков Apache Spark для Amazon EC2 на использование API Openstack, исследовании взаимосвязей компонентов и построении аналогичного решения, но с использованием образов операционных систем без каких-либо предустановленных компонентов (с целью воспроизводимости). Решение было основано на клиентских библиотеках Openstack, библиотеки paramiko для обеспечения защищенных соединений с управляемыми узлами и реализовано на языке Python.

Второй подход [11] заключался в прямой интеграции с внутренней экосистемой облака Openstack. В октябре 2014 года произошел первый публичный релиз подсистемы Openstack Sahara [12]. Система поддерживала возможность создания виртуальных кластеров с дистрибутивами Apache Hadoop от вендоров Cloudera, Hortonworks, MapR (т.е. в версиях, отличных от версий Apache Software Foundation). В состав подобных кластеров входили некоторые дополнительные инструменты, например, Apache Hive, но интеграции с Apache Spark проект не предоставлял. В ходе исследований мы разработали набор компонентов для прямой интеграции Apache Spark с компонентами виртуальных кластеров, создаваемых Openstack Sahara, и кроме того, позволяющих использовать Apache Spark с использованием подхода PaaS. Это означает, что от пользователя скрывается внутренняя структура кластера и предоставлялся специальный слой управления, который позволяет исполнять программы Apache Spark через веб-интерфейс облачной среды или через REST API без прямого подключения к вычислительному кластеру. Также был интегрирован слой, позволяющий прозрачно использовать объектное хранилище Openstack Swift в качестве прозрачной замены распределенной файловой системы HDFS. Наше решение было включено в официальный релиз Openstack Liberty и присутствует в основной кодовой базе проекта Openstack с сентября 2015 года. Данный подход, тем не менее,

обладает рядом недостатков, которые будут рассмотрены в следующем разделе.

Третий подход заключается в использовании внешних средств оркестрации для создания виртуальных кластеров. Данный подход был реализован с использованием системы оркестрации Ansible и служебной программы, предоставляющей командный интерфейс для задания параметров кластера. В ходе разработки данного решения удалось разделить слои управления облачной средой, настройки компонентов операционной системы, настройки компонентов Apache Spark, настройки дополнительных инструментов и использовать необходимые комбинации этих. Среди дополнительных компонентов представлены Apache Hadoop, NFS, Apache Ignite, слой интеграции с Openstack Swift, Ganglia, Jupyter. Важно отметить, что явной привязки к дистрибутиву и версии операционной системы нет (на текущий момент поддерживаются Ubuntu 12.04, 14.04, 16.04; добавление RHEL-подобных дистрибутивов является сравнительно простой задачей). Решение попало в список одобренных [13] разработчиками Apache Spark в отличие от предыдущих подходов. Также стоит отметить, что за счет разделения сценариев, это решение сравнительно легко можно дополнить поддержкой любой другой облачной среды.

3. Сравнение решений

Каждый из описанных подходов (и их реализаций) обладает рядом достоинств и недостатков. Рассмотрим их по порядку.

Решение на основе Python и использования HTTP REST API Openstack

Достоинства:

- Первое подобное решение для открытых облачных сред.
- За счет использования REST API Openstack имеет минимальное количество зависимостей и работает быстрее остальных.

Недостатки:

- Из-за быстрой смены кодовой базы Openstack и изменения API работает только до версии Openstack Kilo и только с плоской организацией сетей на базе Nova Networking (т.е. нет поддержки Openstack Neutron: на момент реализации, Openstack Neutron назывался Openstack Quantum и не обладал стабилизированным API).
- Очень трудоемок процесс интеграции дополнительных инструментов для анализа данных (таких как Apache Hive, Ignite и т.п).

Решение, интегрированное в Openstack

Достоинства:

- Поддерживается большим сообществом и с момента релиза Openstack Liberty существует и развивается самостоятельно.
- Интегрировано с большим количеством инструментов для работы с большими данными.
- Предоставляет уровень абстракции “обработка данных как услуга” и не требует особенных знаний от исследователя при должной настройке облачной среды.
- Обеспечивается возможность при необходимости увеличивать размер вычислительного кластера.
- Хорошо подходит для промышленных окружений с налаженными сценариями вычислений.

Недостатки:

- Существенное отставание от актуальных версий Apache Spark и невозможность получить более новую версию без вмешательства в кодовую базу Openstack. Более того, одновременно предоставляется только две последних минорных версии Apache Spark на момент выпуска версии Openstack. Openstack выходит 2 раза в год, Apache Spark — 3-4 в зависимости от года.
- Использование дистрибутивов инструментов анализа данных от вендоров (отличаются как между собой, так и от версии сообщества)
- Процедура первичного создания шаблона вычислительного кластера требует вмешательства инженера, понимающего специфику вычислительной задачи и устройство Apache Spark.
- Существенно меньший контроль над ходом исполнения задач: подразумевается, что запущенная программа никогда не содержит ошибок; процесс нахождения ошибок чрезвычайно сложен.
- Требует дополнительных ресурсов для облачной среды: подразумевается, что этот сервис запущен постоянно на одном из контроллеров облака. Кроме того, у этого сервиса есть явные зависимости от других необязательных сервисов Openstack (например, Openstack Heat), из-за чего этот сервис сравнительно редко встречается в действующих облачных средах на основе Openstack.
- Процесс создания кластера является длительным процессом по сравнению с остальными решениями, как за счет устройства самой Openstack Sahara, так и за счет большого количества промежуточных абстракций, обеспечивающих интеграцию между инструментами обработки данных.

Решение на основе технологий оркестрации

Достоинства:

- Очень легко расширяется по сравнению с первыми двумя решениями.

- Поддерживает все существующие версии Apache Spark, начиная с 1.0 в любых поддерживаемых комбинациях с Apache Hadoop. Процесс добавления новых (или собственных) версий требует минимальных усилий.
- Сравнительно легко добавлять другие облачные среды в качестве инфраструктурной основы.
- Содержит множество дополнительных инструментов (Apache Hadoop, Apache Ignite, Jupyter, Ganglia, NFS, интеграция с Swift) и позволяет легко добавлять новые за счет слабой связности кодовой базы.
- Работает с любой версией Openstack и, наиболее вероятно, не устареет, поскольку основные библиотеки поддерживаются в том числе Openstack Foundation.
- Одобрено разработчиками Apache Spark.

Недостатки:

- Обладает меньшей гибкостью интеграции с дополнительными инструментами обработки больших данных за счет меньшей связности компонентов.
- Не предоставляет уровень “вычислений как сервиса”.
- Для более тонкой настройки системы, человек должен вмешиваться в шаблоны конфигурационных файлов или добавлять новые параметры к решению (например, для изменения степени репликации HDFS).

Каждый из походов предоставляет собственные преимущества, и их совокупность полностью покрывает возможные потребности исследователей.

4. Достигнутые результаты

В ходе проекта был полностью реализован весь спектр подходов по созданию виртуальных кластеров Apache Spark в среде Openstack, и два из трех решений получили поддержку сообщества. Все результаты опубликованы под открытой и свободной лицензией Apache 2.0 и доступны любому желающему как для использования, так и для модификации под свои нужды.

Список литературы

[1]. Shanahan J. and Dai L. Large Scale Distributed Data Science using Apache Spark. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15). ACM, New York USA, pp. 2323-2324.

[2]. Li M., Tan J., Wang Y., Zhang L., Salapura V. SparkBench: a comprehensive benchmarking suite for in memory data analytic platform Spark. In Proceedings of the 12th ACM International Conference on Computing Frontiers (CF '15). ACM, New York USA, Article 53.

[3]. Jeffrey D., Sanjay G. MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.M. Bhandarkar, "MapReduce programming with

apache Hadoop," Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on, Atlanta, GA, 2010, pp. 1-1.

[4]. Vavilapalli V., Murthy A., Douglas C., Agarwal S., Konar M., Evans R., Graves T., Lowe J., Shah H., Seth S., Saha B., Curino C., O'Malley O., Radia S., Reed B., Baldeschwieler E. Apache Hadoop YARN: yet another resource negotiator. In Proceedings of the 4th annual Symposium on Cloud Computing (SOCC '13). ACM, New York USA, 2013, Article 5.

[5]. Страница проекта Apache Mesos: http://mesos.apache.org

[6]. Guller, Mohammed. Cluster Managers. Big Data Analytics with Spark. Apress, 2015. 231-242.

[7]. Dinsmore, Thomas W. In-Memory Analytics. Disruptive Analytics. Apress, 2016, pp. 97-116.

[8]. Sefraoui, Aissaoui O, Eleuldj M. OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications 55.3, 2012.

[9]. Hazelhurst, Scott. Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud. Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology. ACM, 2008.

[10]. Борисенко О.Д., Лагута А.В., Турдаков Д.Ю., Кузнецов С.Д., Автоматическое создание виртуальных кластеров Apache Spark в облачной среде Openstack, Труды ИСП РАН, том 26, вып. 4, 2014 г., стр. 33-44. DOI: 10.15514/ISPRAS-2014-26(4)-4

[11]. Aleksiyants A., Borisenko O., Turdakov D., Sher A., Kuznetsov S. Implementing Apache Spark Jobs Execution and Apache Spark Cluster Creation for Openstack Sahara. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 35-48. DOI: 10.15514/ISPRAS-2015-27(5)-3.

[12]. Ibrahim, Asmaa, Nawawy. A study of adopting big data to cloud computing. Technology Innovation and Entrepreneurship Center, Egypt Technology Innovation and Entrepreneurship Center, Egypt, 2015, pp. 1-7.

[13]. Список одобренных проектов, связанных с Apache Spark. https://cwiki.apache.org/confluence/display/SPARK/Third+Party+Projects

Deploying Apache Spark virtual clusters in cloud environments using orchestration technologies

[1]*Borisenko O. <al@somestuff.ru>*
[1]*Pastukhov R. <pastukhov@ispras.ru>*
[1,2,3]*Kuznetsov S. <kuzloc@ispras.ru>*
[1]*Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[2]*Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
[3]*Moscow Institute of Physics and Technology (State University)*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia

Abstract. Apache Spark is a framework providing fast computations on Big Data using MapReduce model. With cloud environments Big Data processing becomes more flexible since they allow to create virtual clusters on-demand. One of the most powerful open-source cloud environments is Openstack. The main goal of this project is to provide an ability to create virtual clusters with Apache Spark and other Big Data tools in Openstack. There exist three approaches to do it. The first one is to use Openstack REST APIs to create instances and then deploy the environment. This approach is used by Apache Spark core team to create clusters in propriatary Amazon EC2 cloud. Almost the same method has been implemented for Openstack environments. Although since Openstack API changes frequently this solution is deprecated since Kilo release. The second approach is to integrate virtual clusters creation as a built-in service for Openstack. ISP RAS has provided several patches implementing universal Spark Job engine for Openstack Sahara and Openstack Swift integration with Apache Spark as a drop-in replacement for Apache Hadoop. This approach allows to use Spark clusters as a service in PaaS service model. Since Openstack releases are less frequent than Apache Spark this approach may be not convenient for developers using the latest releases. The third solution implemented uses Ansible for orchestration purposes. We implement the solution in loosely coupled way and provide an ability to add any auxiliary tool or even to use another cloud environment. Also, it provides an ability to choose any Apache Spark and Apache Hadoop versions to deploy in virtual clusters. All the listed approaches are available under Apache 2.0 license.

Keywords: Apache Spark, Openstack, Amazon EC2, Map-Reduce, HDFS, virtual cluster, cloud computing, Big Data, Apache Ignite.

DOI: 10.15514/ISPRAS-2016-28(6)-8

For citation: Borisenko O., Pastukhov R., Kuznetsov S. Deploying Apache Spark virtual clusters in cloud environments using orchestration technologies. *Trudy ISP RAN/Proc.ISP RAS*, vol. 28, issue 6, 2016. pp. 111-120 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-8

References

[1]. Shanahan J. and Dai L. Large Scale Distributed Data Science using Apache Spark. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15). ACM, New York USA, pp. 2323-2324.

[2]. Li M., Tan J., Wang Y., Zhang L., Salapura V. SparkBench: a comprehensive benchmarking suite for in memory data analytic platform Spark. In Proceedings of the 12th ACM International Conference on Computing Frontiers (CF '15). ACM, New York USA, Article 53.

[3]. Jeffrey D., Sanjay G. MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.M. Bhandarkar, "MapReduce programming with apache Hadoop," Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on, Atlanta, GA, 2010, pp. 1-1.

[4]. Vavilapalli V., Murthy A., Douglas C., Agarwal S., Konar M., Evans R., Graves T., Lowe J., Shah H., Seth S., Saha B., Curino C., O'Malley O., Radia S., Reed B., Baldeschwieler E. Apache Hadoop YARN: yet another resource negotiator. In Proceedings of the 4th annual Symposium on Cloud Computing (SOCC '13). ACM, New York USA, 2013, Article 5.

[5]. Apache Mesos project home page: http://mesos.apache.org

[6]. Guller, Mohammed. Cluster Managers. Big Data Analytics with Spark. Apress, 2015. 231-242.

[7]. Dinsmore, Thomas W. In-Memory Analytics. Disruptive Analytics. Apress, 2016, pp. 97-116.

[8]. Sefraoui, Aissaoui O, Eleuldj M. OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications 55.3, 2012.

[9]. Hazelhurst, Scott. Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud. Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology. ACM, 2008.

[10]. Borisenko O., Laguta A., Turdakov D., Kuznetsov S, Automating cluster creation and management for Apache Spark in Openstack cloud, Trudy ISP RAN/Proc. ISP RAS, vol 26, issue 4, 2014, pp. 33-44 (in Russian). DOI: 10.15514/ISPRAS-2014-26(4)-4

[11]. Aleksiyants A., Borisenko O., Turdakov D., Sher A., Kuznetsov S. Implementing Apache Spark Jobs Execution and Apache Spark Cluster Creation for Openstack Sahara. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 35-48. DOI: 10.15514/ISPRAS-2015-27(5)-3.

[12]. Ibrahim, Asmaa, Nawawy. A study of adopting big data to cloud computing. Technology Innovation and Entrepreneurship Center, Egypt Technology Innovation and Entrepreneurship Center, Egypt, 2015, pp. 1-7.

[13]. List of approved third-party project for Apache Spark. https://cwiki.apache.org/confluence/display/SPARK/Third+Party+Projects

Применение облачных вычислений для анализа данных большого объема в умных городах

[1] *Рензо Массобрио <renzom@fing.edu.uy>*
[1] *Серхио Несмачнов <sergion@fing.edu.uy>*
[2] *Андрей Черных <chernykh@cicese.mx>*
[3] *Арутюн Аветисян <arut@ispras.ru>*
[4] *Глеб Радченко <gleb.radchenko@susu.ru>*
[1] *Республиканский университет Уругвая, Монтевидео 11300, Уругвай*
[2] *Центр научных исследований и высшего образования, Энсенада, B.C. 22860, Мексика*
[3] *Институт системного программирования Российской академии наук, Москва, 109004, Россия*
[4] *Южно-Уральский государственный университет, Челябинск, 454080, Россия*

Аннотация. В этой статье рассматривается вопрос применения анализа данных большого объема с использованием облачных вычислений для решения задач анализа дорожного траффика в контексте «умных» городов. Предложенное решение базируется на модели параллельных вычислений MapReduce, реализованной на платформе Hadoop. Анализируются два экспериментальных случая: оценка качества общественного транспорта на основе анализа истории местоположения автобусов, и оценка мобильности пассажиров при помощи анализа истории покупок билетов с транспортных карт. Оба эксперимента используют реальную базу данных системы общественного транспорта Монтевидео в Уругвае. Результаты эксперимента показали, что рассмотренная модель действительно позволяет эффективно обрабатывать большие объемы данных.

Ключевые слова: облачные вычисления; big data; умные города; интеллектуальные транспортные системы, большие объемы данных.

DOI: 10.15514/ISPRAS-2016-28(6)-9

Для цитирования: Массобрио Р., Несмачнов С., Черных А., Аветисян А., Радченко Г. Применение облачных вычислений для анализа данных большого объема в умных

городах. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 121-140. DOI: 10.15514/ISPRAS-2016-28(6)-9

1. Введение

Одной из задач умных городов является использование информационных и коммуникационных технологий для управления ресурсами городского хозяйства с целью поднятия качества услуг, предоставляемых горожанам. Использование этих технологий позволяет снизить инфраструктурные и эксплуатационные расходы, повысить эффективность использования ресурсов и способствовать улучшению взаимодействия жителей и администрации [1]. Чаще всего такие технологии применяются в области транспортного сервиса из-за его ключевой роли в жизни современного города.

Сегодня во многих развитых городах общественный транспорт решает одну из важнейших задач в формировании городской инфраструктуры, обеспечивая мобильность жителей [2]. Особенно большую роль общественный транспорт играет в густонаселенных районах. Однако, многие транспортные системы не соответствуют постоянно растущим требованиям. Чтобы решить эту проблему, правительство должно иметь глубокое понимание всей ситуации и деталей, включая ежедневную статистику использования транспорта [3]. Однако, ссылаясь на недостаток финансов и человеческих ресурсов, администрация обычно оперирует чрезвычайно ограниченным объемом частитчно устаревшей информации для принятия решений. Чаще всего, данные собираются, но не анализируются, что приводит к тому, что они используются для улучшения инфраструктуры общественного или личного транспорта в «сыром» виде. По этой причине, процесс принятия решений относительно транспортной ситуации обычно затягивается. Модель умного города позволяет анализировать данные из многих источников, чтобы понять транспортную ситуацию в городах.

Интеллектуальные транспортные системы (ИТС) играют ключевую роль в транспортных системах умных городов. ИТС – это системы, объединяющие в себе синергетические технологии, искуственный интеллект и инженерные принципы, применяемые к транспортным системам для увеличения пропускной способности, безопасности, эффективности и уменьшения воздействия на окружающую среду [4]. ИТС позволяют собирать данные о транспорте и способах передвижении в городах [5]. В больших городских зонах ИТС генерирует огромное количество информации, из которой необходимо извлечь полезную информацию о передвижении жителей.

В этой статье представлена платформа для эффективного анализа больших данных ИТС за счёт преимуществ облачных вычислений. Облачные вычисления включают в себя систему методов, которые позволяют использовать все преимущества множества вычислительных элементов для комплексного решения задач посредством распределенных вычислений. Когда исследователи сталкиваются со сложными задачами, такими как обработка

больших объёмов данных, распознавание образов, глубинное обучение, на помощь приходят распределенные вычисления с высокой производительностью, достигаемой кооперативным подходом. Такой эффект достигается разделением большой задачи на множество маленьких подзадач, решаемых параллельно на разных вычислительных элементах, для увеличения скорости обработки [6]. За последние десять лет были разработаны различные платформы для анализа больших данных с использованием распределенных вычислений на базе облачных систем [7].

В текущей работе мы адаптировали стандартную платформу для обработки данных большого объема, содержащихся в ИТС умного города. Данный эксперимент – это наш первый шаг к внедрению и увеличению эффективности обработки больших данных в рамках умных городов, что будет полезно и жителям, и администрациям городов.

Нами были проанализированы два экспериментальных случая: оценка качества общественного транспорта на основе анализа истории местоположения автобусов и оценка мобильности пассажиров при помощи анализа истории покупок билетов с транспортных карт. Оба эксперимента используют реальную базу данных системы общественного транспорта Монтевидео в Уругвае.

Статья организована следующим образом. В разделе 2 описываются основные концепции облачных вычислений, представляется парадигма MapReduce и платформа Hadoop. В разделе 3 представлен обзор смежных источников по вопросам, связанным с умными городами и обработки данных ИТС. Отдельный акцент сделан на распределенной и облачной обработке таких данных. В разделе 4 представлена модель облачной системы анализа данных ИТС. Раздел 5 посвящен обзору двух экспериментальных случаев и анализу полученных результатов. В разделе 6 представлены основные результаты исследования, определены основные направления дальнейшего развития работы.

2. Облачные вычисления, парадигма MapReduce и платформа Hadoop

В этом разделе описаны основные концепции облачных вычислений, модель MapReduce и ее реализация в рамках платформы Hadoop.

2.1 Распределенные вычисления и облачные вычисления

Распределенные вычисления – это обобщающее понятие для вычислительной модели и перечня программных алгоритмов, основанных на использовании множества вычислительных элементов, соединенных в сеть для решения задач, требующих больших ресурсов [8]. Распределенные процессы, выполняемые на базе данных вычислительных элементов, взаимодействуют, синхронизируют и координируют вычислительный процесс посредством

функциональной декомпозиции или же декомпозиции предметной области. За последние двадцать лет, распределенные вычисления были успешно внедрены в различных прикладных областях [6].

Облачные вычисления являются особым типом распределенных вычислений – они используют географически распределенные ресурсы и Интернет для выполнения запросов пользователей [9]. Суть облачных вычислений заключается в использовании сервисов по запросу, а пользователи могут получить к ним доступ с любого устройства из любой точки мира, используя общий пул реконфигурируемых ресурсов (таких как вычислительная мощность, память, сеть, приложения и др.).

Облачные вычисления очень часто используются и для работы с большими объемами данных. Одна из самых распространенных вычислительных моделей для обработки больших объемов данных – это модель MapReduce.

2.2 MapReduce

MapReduce – это парадигма программирования для обработки данных большого объема при помощи параллельных (или распределенных) алгоритмов на базе вычислительных кластеров, вычислительных сетей и облачных вычислительных систем.

Парадигма MapReduce состоит из двух простых операций. Первая операция, с помощью функции распределения (Map), выполняет (параллельно, на множестве вычислительных элементов) перечень задач, таких как фильтрация, сортировка и (или) непосредственно вычисления. Вторая операция - это сведение (Reduce) результатов выполнения функции распределения [10].

Вычисления, которые используют парадигму MapReduce, называют задачами MapReduce. Такая задача имеет два этапа решения: распределение и сведение. Первый этап выполняется в четыре шага: чтение данных, анализ распределения, комбинирование и непосредственное распределение. Результат этих процедур – список узлов и данных, которые обрабатываются на этих узлах. Второй этап состоит также из четырех шагов: перемещение, сортировка, сведение и вывод результата вычислений.

Библиотеки для парадигмы MapReduce написаны на множестве языков программирования. Самая распространенная - Apache Hadoop, описана ниже.

2.3 Hadoop

Сегодня Hadoop – это одна из самых популярных платформ, использующих модель MapReduce для обработки больших объемов данных. Hadoop – это распределенная вычислительная система, которая использует файловую систему с открытым исходным кодом Hadoop Distributed File System (HDFS) [11].

Hadoop предоставляет высокий уровень абстракции для описания задач. Это облегчает работу пользователям по реализации распределенных алгоритмов,

даже тем, у кого недостаточный уровень знаний в области распределенных вычислений. Такие задачи могут быть запущены на множестве вычислительных узлов, и обрабатывать большие объемы данных в реальном времени. Hadoop включает в себя экосистему, которая предоставляет дополнительные возможности для управления задачами, распределенного программирования, интерфейсы баз данных и другие функции.

2.4 Реализация парадигмы MapReduce на платформе Hadoop

Реализация парадигмы MapReduce на платформе Hadoop является самой распространенной и известной. Хотя MapReduce и проста для понимания, ее не всегда просто реализовать в виде алгоритмов для функций распределения и сведения.

В рамках платформы Hadoop, распределитель обычно принимает сырые данные в файловой системе HDFS. Данные, по умолчанию в текстовом формате, для распределителя представляют собой строки с ключом, равным байтовому смещению начала строки от начала файла. Задача MapReduce состоит из входных данных, кода программы и процедур для распределения. Пользователь может реализовать свои модули разделения, методы чтения записей, форматы входных данных и комбинаторы в зависимости от потребностей [7].

Вычислительный кластер в Hadoop состоит из корневого узла и подчиненных ему узлов. Корневой узел поддерживает несколько возможных ролей: менеджер задач, исполнитель, узел имен, и узел хранения. Подчиненные узлы могуть играть одну из двух ролей – узел хранения или исполнитель. Менеджер задач координирует все задачи системы с помощью списка задач исполнителей. Исполнитель запускает задачи и отправляет отчет о прогрессе работы менеджеру задач, который ведет статистику прогресса каждой задачи. Если задача завершается неправильно, менеджер задач может перезапустить ее на другом исполнителе. Именные узлы и узлы хранения относятся к кластеру HDFS.

Выполнение задачи MapReduce на платформе Hadoop выглядит следующим образом. В первую очередь, задача создается на клиентском узле, который запущен на виртуальной машине Java Virtual Machine (JVM). Далее, эта задача передает новую задачу менеджеру задач, который анализирует весь список запущенных и возвращает идентификатор новой задачи, после чегофайл, который необходимо выполнить и его кэш копируются на узлы. Далее, когда задача распределена, менеджер задач с помощью идентификатора инициализирует процесс обработки и подает входные данные. Исполнители возвращают менеджеру информацию о возможности запуска и доступной мощности. В зависимости от ответа менеджер задач назначает узлу выполнение или сведение. Узел-исполнитель извлекает ресурсы для обработки задачи и запускает новую JVM и выполняет функцию выполнения или сведения.

3. Обзор работ по смежным тематикам

Ряд статей, выпущенных в недавнее время, описывают применение распределенных и облачных вычислений для обработки больших данных, сгенерированных различными ИТС. В этом разделе мы проведем краткий обзор этих работ.

Преимущества анализа больших данных для систем общественного транспорта были представлены в работе [12]. Авторы проанализировали различные источники информации: траектория движения транспорта (координаты GPS, скорость передвижения), отчеты о неисправностях, передвижения людей (с помощью GPS и Wi-Fi сигналов), социальные сети (текстовые записи, адреса) и веб-логи (идентификаторы пользователей, комментарии). В статье описаны все преимущества и недостатки каждого источника информации. Также рассмотрено несколько новых идей для улучшения системы общественного транспорта с применением парадигмы ИТС, включая краудсорсинг для сбора и анализа актуальных данных о движении транспорта, сопровождение водителя и анализ поведения пассажиров. Кроме того, в статье представлен вывод о том, как внедрить технологию в систему общественного транспорта, чтобы она была совместима со следующими поколениями ИТС, которые повысят безопасность и эффективность поездок.

Методы интеллектуального вычисления недавно нашли применение при проектировании ИТС.

В работе [13] предложен метод последовательного поиска для прогнозирования ситуации на дорогах с использованием системы обнаружения транспортных средств (Vehicle Detection System) и алгоритма классификации k ближайших соседей (k nearest neighbors - kNN). Такое сочетание значительно превосходит традиционное использование алгоритма kNN, обеспечивая более точные результаты, сохраняя при этом высокую эффективность и стабильность.

В работе [14] представлено применение метода анализа данных на основе случайных лесов и Байесовского вывода для обработки больших объемов данных в системе микроволнового обнаружения транспорта (Microwave Vehicle Detection System). Главной целью работы является обнаружение факторов, провоцирующих аварии, в реальном времени. При анализе данных в час-пик, применяется модель надежности, учитывается увеличение объема и снижение средней скорости транспортного потока, индекс затора. Основной вывод анализа заключается в том, что пробки чаще всего являются основной причиной столкновений сзади.

В работе [15] представлен подход обучения с учителем, с использованием метода опорных векторов и Байесовской классификации, для построения системы прогноза транспортного потока в реальном времени. Сначала сырые данные проходят два этапа подготовки и фильтрацию шума. Это стандартный подход для подготовки данных в подобных исследованиях, и мы также будем

применять его в нашем эксперименте. Затем, модель транспортного потока исследуется Байесовской платформой. Методами регрессионного анализа моделируется пространственно-временная зависимость и отношения между дорогами. Эффективность такого метода исследуется на транспортных данных Кёнбусона, железной дороги Сеул-Пусан в Южной Корее. Результаты эксперимента показали, что подход с использованием метода опорных векторов в для оценки превосходит традиционные методы линейной регрессии с точки зрения точности.

В работе [16] была предложена модель для эффективного прогнозирования скорости движения на заданной местности. Она использует историю из различных источников, в том числе данные из различных ИТС, погодные условия и особые события, происходящие в городе. Для получения точных результатов модель прогнозирования должна часто обновляться, чтобы использовать самые последние данные. Модель прогнозирования сочетает в себе алгоритм классификации k ближайших соседей и регрессию на основе Гауссовского процесса. Кроме того, результаты рассчитываются с использованием модели MapReduce, реализованной на платформе Hadoop. Экспериментальная оценка была выполнена на основе реального сценария, используя данные, полученные на Research Data Exchange – платформе для публикации данных ИТС. Данные собраны на участке дороги I5N в Сан Диего, штат Калифорния, США. Информация содержит скорость и величину расхода топлива, полученные с помощью петлевого детектора на дороге, а также данные о видимости, полученные с ближайшей метеорологической станции. Результаты экспериментов показали, что предложенный метод может точно предсказать скорость движения потока со средней ошибкой менее двух миль в час. Кроме того, за счет использования платформы Hadoop в кластерной инфраструктуре, а не на одной вычислительной машине, было достигнуто уменьшение времени обработки на 69%.

В работе [17] представлены результаты исследований проблем краткосрочного прогнозирования транспортного потока в реальном времени. В решении использовался алгоритм k ближайших соседей в распределенной среде MapReduce на платформе Hadoop. Предлагаемое решение рассматривает пространственно-временную корреляцию в транспортном потоке, т.е. текущий поток на определенном участке дороги зависит от прошлого (временное измерение) и от потока на соседних участках дороги (пространственное измерение). В реализованном алгоритме, эти два фактора можно контролировать с помощью весов. Экспериментальный анализ проводился с использованием данных траекторий более 12 000 такси в Пекине, оборудованных GPS-датчиками, в 15-дневный период в ноябре 2012 года. Первые 14 дней данных использовались для обучения системы, а последний – для вычисления результатов. Предложенный алгоритм позволил уменьшить среднюю абсолютную ошибку от 8,5% до 11,5% в среднем в сравнении с тремя существующими методами, основанными на алгоритме k ближайших

соседей. Кроме того, предлагаемое решение в лучшем случае достигает вычислительную эффективность 0,84.

В работе [23] предложено использовать многокритериальные ячеистые генетические алгоритмы MOcell для оптимизации расписаний автобусного парка с автобусами различной вместимости.

В представленном обзоре, нами были определены наиболее распространенные методы анализа больших данных для обработки данных ИТС. Для определения моделей транспортного потока и генерации полезной информации для прогнозирования, применяются такие методы машинного обучения, как метод регрессии, k ближайших соседей и Байесовский вывод. Тем не менее, есть несколько работ, уделяющих особое внимание улучшению системы общественного транспорта, особенно с точки зрения пассажиров. В этом контексте, наша работа продолжает тему статей, предлагая конкретную модель для анализа данных ИТС в облаке, улучшающую систему общественного транспорта.

4. Предлагаемая модель

В этом разделе описана предполагаемая структура платформы распределенных облачных вычислений для обработки данных ИТС с целью улучшения систем общественного транспорта.

4.1 Архитектура системы

Задача разбивается на два этапа: 1) этап предварительных вычислений для подготовки входных данных к следующему шагу; 2) этап обработки данных ИТС с использованием распределенных облачных вычислений.

Для организации вычислений и определения структуры используется подход Master/Slave (ведущий/ведомый). На рис. 1 представлена концептуальная схема разрабатываемой платформы.

На этапе предварительных вычислений ведущий процесс подготавливает данные, фильтруя те записи, которые содержат ненужную информацию, а затем передает записи на этап распределенных вычислений. Процесс фильтрации может различаться в зависимости от поставленной задачи (см. раздел 5 "Практические примеры").

На этапе распределенных вычислений используется подход декомпозиции по данным. После этапа предварительных вычислений отфильтрованные и отсортированные данные разбиваются на части и передаются нескольким вычислительным элементам. Ведущий процесс является ответственным за распределение данных и назначение каждой части данных ведомым процессам для обработки. Каждый ведомый процесс получает от ведущего часть данных. Такая структура соответствует категории вычислительных систем Single Program Multiple Data (SPMD), согласно которой ведомые

процессы совместно выполняют одну и ту же задачу на разных блоках данных.

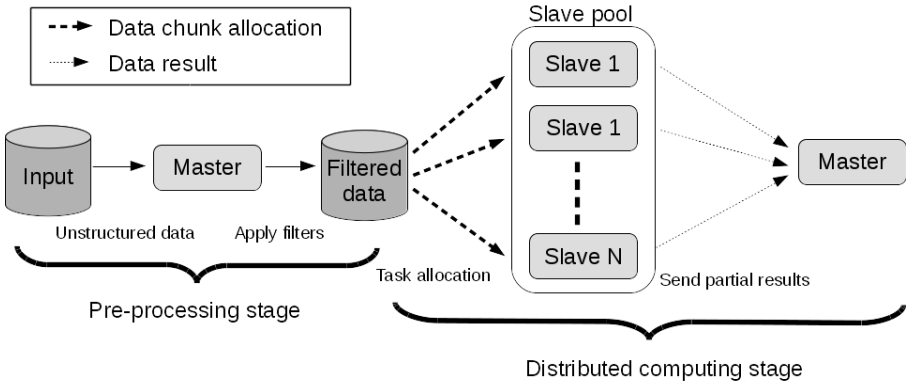


Рис. 1. Концептуальная схема разрабатываемой платформы

Fig. 1. Conceptual scheme of the developed platform

4.2 Реализация с применением MapReduce

При разработке распределенной вычислительной системы для анализа данных ИТС использовалась платформа Hadoop и была применена парадигма MapReduce. Система соответствует данной парадигме, так как между ведомыми процессами взаимодействий не происходит, а взаимодействия между ведущим и ведомыми процессами ограничены лишь распределением данных и сбором результатов вычислений. В Hadoop реализован стандартный подход MapReduce, в котором используется один главный узел и несколько рабочих узлов. Главный узел, используя процесс JobTracker, посылает задачи различным процессам TaskTracker, каждый из которых связан с определенным рабочим узлом. Как только все ведомые процессы заканчивают назначенные им задания, каждый процесс TaskTracker передает результаты обратно процессу JobTracker в главный узел.

В Hadoop реализован механизм обеспечения отказоустойчивости. Дополнительно, для улучшения работы механизма отказоустойчивости при работе с данными ИТС используются следующие особенности: 1) возможность отбрасывания поврежденных входных данных в случае наличия поврежденной информации в записи; 2) встроенный механизм репликации HDFS для хранения данных в различных вычислительных узлах.

5. Практические примеры

В этом разделе описывается применение разработанной платформы для обработки двух разных типов данных ИТС. В обоих случаях

экспериментальная оценка проводится с использованием соответствующего набора реальных данных ИТС г. Монтевидео, Уругвай.

5.1 Метрики качества обслуживания транспортной системы с использованием данных о местоположении автобусов

В этом практическом примере рассматривается задача вычисления метрик качества обслуживания системы общественного транспорта с использованием данных GPS-навигаторов, установленных в автобусах [24]. Данные содержат информацию о местоположении каждого автобуса в конкретный момент времени. Эта информация обновляется каждые 10-30 секунд. Для определения эффективности системы общественного транспорта основной задачей является введение соответствующих метрик, таких как: 1) реальное время, затрачиваемое каждым автобусом на путь между заранее заданными и отмеченными местами в городе; 2) статистическая информация о задержках каждого автобуса на конкретном маршруте для определения перегруженных мест. Данные должны быть должным образом организованы для возможности определения закономерностей объема потока пассажиров и загруженности дорог в различные дни недели, а также в различное время дня.

Существует как минимум две целевые группы, которым разработанная система принесет пользу: пассажиры и городская администрация. С помощью информации, которая получена на основе обработки исторических данных и данных реального времени, пользователь системы общественного транспорта сможет принять более выгодные решения о собственном перемещении (например, выбрать определенный автобусный маршрут, совершить пересадку). Эта информация может предоставляться посредством интеллектуального мобильного приложения или сайта. Для городской администрации такая информация полезна для планирования долгосрочных изменений автобусных маршрутов, расписания движения, положения автобусных остановок, а также для выявления конкретных проблемных ситуаций.

Схема разработанной системы представлена на рис. 2. Автобусы посылают данные о местоположении на облачный сервер. На сервере происходит MapReduce-обработка собранных данных GPS различных автобусов в реальном времени. Результаты вычислений передаются в мобильное приложения для использования конечными пользователями и мониторинговое приложение для использования городской администрацией.

На этапе предварительных вычислений, ведущий процесс подготавливает данные, фильтруя те записи, которые хранят ненужную информацию (например, поврежденные данные GPS). Дополнительно данные фильтруются таким образом, чтобы включать только записи из временного промежутка, задаваемого пользователем. В конце этапа записи сортируются по идентификационному номеру автобуса, что позволяет увеличить эффективность работы следующего этапа.

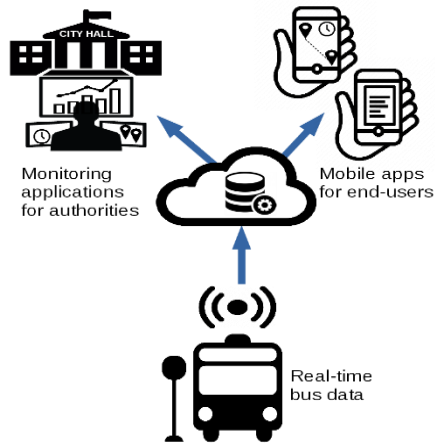


Рис. 2. Архитектура системы облачной обработки данных ИТС

Fig. 2. Architecture of the ITS cloud data processing system

На этапе распределенных вычислений, ведущий процесс разделяет набор отфильтрованных записей GPS, полученных на предыдущем этапе, на части и распределяет их между ведомыми процессами. Каждая часть включает записи о местоположении и времени конкретного автобусного маршрута. Таким образом, обеспечивается независимость ведомых процессов, что увеличивает вычислительную производительность. Каждый ведомый процесс вычисляет время, затраченное на путь между каждыми отмеченными местами на маршруте. На шаге свертки, результаты вычислений используются для определения статистики качества обслуживания, которая будет предоставлена пользователю. Результаты сортируются по маршрутам автобусов и по отмеченным местам. В конце этапа, каждый ведомый процесс возвращает полученные результаты ведущему процессу, который должен сгруппировать результаты и вывести итоговый результат пользователю.

Экспериментальная оценка производится на основе облачной инфраструктуры Cluster FING, предоставленной Республиканским университетом Уругвая [18], для вычислений используются 24-ядерные процессоры AMD Opteron 6172 Magny Cours с тактовой частотой 2,26 ГГц, 24 Гб оперативной памяти и операционная система CentOS Linux 5.2.

Для экспериментального анализа используются реальные исторические данные ИТС, предоставленные правительством г. Монтевидео, Уругвай. Автобусные компании Монтевидео обязаны посылать данные о местоположении автобусов и о продажах билетов правительству города. Автобусная сеть Монтевидео достаточно сложна и насчитывает 1383 автобусных маршрута и 4718 автобусных остановок.

В этом практическом примере рассматривается полный набор данных о продажах билетов и местоположении автобусов за 2015 год, в котором содержится около 200 Гб данных. Данные о местоположении автобуса содержат информацию о месте нахождения каждого автобуса с интервалами от 10 до 30 секунд.

Полный набор данных разделяется на части, которые используются для определения различных сценариев, представляющихся для оценки производительности разработанной системы с различными размерами входных файлов, различными временными промежутками и различным количеством процессов предварительной обработки (Map) и процессов распределенных вычислений (Reduce).

Таблица 1. Результаты замеров производительности

Table 1. Performance measurement results.

#I	#D	#M	#R	T_1(s)	T_N(s)	S_N	E_N
10	3	14	8	1333.9	253.1	5.27	0.22
10	3	22	22	1333.9	143	9.33	0.39
10	30	14	8	2108.6	178	11.84	0.49
10	30	22	22	2108.6	187.3	11.26	0.47
20	3	14	8	2449	351.1	6.98	0.29
20	3	22	22	2449	189.8	12.9	0.54
20	30	14	8	3324.5	275.6	12.06	0.5
20	30	22	22	3324.5	238.8	13.92	0.58
20	60	14	8	4762	300.8	15.83	0.66
20	60	22	22	4762	264.7	17.99	0.75
30	3	14	8	3588.5	546.9	6.56	0.27
30	3	22	22	3588.5	179.6	19.99	0.83
30	30	14	8	5052.9	359.6	14.05	0.59

#I	#D	#M	#R	T_1(s)	T_N(s)	S_N	E_N
30	30	22	22	5052.9	281.1	17.98	0.75
30	60	14	8	5927.9	383.4	15.46	0.64
30	60	22	22	5927.9	311.4	19.04	0.79
30	90	14	8	7536.9	416.6	18.09	0.75
30	90	22	22	7536.9	349.2	21.58	0.9
60	3	14	8	7249.6	944	7.68	0.32
60	3	22	22	7249.6	362.1	20.02	0.83
60	60	14	8	10037.1	672.6	14.92	0.62
60	60	22	22	10037.1	531.4	18.89	0.79
60	90	14	8	11941.6	709.6	16.83	0.7
60	90	22	22	11941.6	648.9	18.4	0.77
60	180	14	8	19060.8	913.7	20.86	0.87
60	180	22	22	19060.8	860.3	22.16	0.92

Работа происходит на наборах данных ИТС объемом 10, 20, 30 и 60 Гб и на временных промежутках длительностью 3 дня, а также 1, 2, 3 и 6 месяцев. Для оценки производительности параллельного алгоритма используются стандартные метрики ускорения и эффективности.

В таблице 1 представлены результаты замеров вычислительной производительности системы при использовании различных сценариев и следующих изменяемых параметров: размер входных данных в Гб (#I), временной промежуток в днях (#D), количество процессов Map (#M) и Reduce (#R). Для каждого сценария определялись такие результаты, как время выполнения в секундах с использованием 1 ядра (T1) и 24 ядер (TN),

ускорение (SN) и эффективность (EN). В таблице приведены средние значения после пяти независимых запусков каждого сценария.

Результаты в таблице 1 демонстрируют значительное увеличение производительности при использовании выбранного подхода по сравнению с последовательной реализацией, особенно на больших наборах данных. Ускорение при использовании 24 ядер достигает значения 22.16, что соответствует значению эффективности 0.92. Распределенная реализация позволяет сократить время работы на больших объемах данных с 6 часов до 14 минут. Такое увеличение производительности просто необходимо для быстрого реагирования городской администрации на возникновение непредвиденных ситуаций и для анализа различных метрик и сценариев, как обычными пользователями, так и администраторами системы.

Использование 22 процессов Map и 22 процессов Reduce позволяет достигнуть наибольшей эффективности. Время выполнения возрастает на 15% в лучшем случае и на 9% в среднем по сравнению со сценариями, в которых используются 14 процессов Map и 8 процессов Reduce. При работе с малыми объемами данных процессы низко нагружены с вычислительной и пространственной точек зрения, и заметного выигрыша во времени выполнения по сравнению с последовательной реализацией нет.

5.2 Построение матриц отправления-прибытия с использованием исторических данных смарт-карт

Для решения задач оптимизации систем городского транспорта необходимо понимать шаблоны передвижения и распределения горожан. Обычно эта информация представляется в виде следующих матриц: 1) матриц отправления прибытия (Origin-Destination - OD-матриц), которые отражают количество людей, передвигающихся из конкретной точки города в определенный временной промежуток [19]; 2) матриц распределения, которые отражают количество автобусных билетов, проданных на каждой остановке в городе.

Обычно эти матрицы строятся на основе опросов пассажиров и водителей. Однако такой подход не дает полного видения ситуации, не предоставляет актуальную информацию и требует больших вложений от городского правительства.

В этом практическом примере предлагается иной подход для построения матриц распределения и OD-матриц [25]. Мы вычисляем и обновляем их, используя обработку таких данных ИТС, как количество билетов, проданных с использованием смарт-карт и без их использования, а также данных о местоположении автобусов. Учитывая высокую вычислительную сложность обработки больших объемов данных ИТС, нами будет применяться модель распределенных вычислений, описанная в разделе 4.

Основной проблемой при генерации матриц распределения и OD-матриц с использованием данных о продажах билетов является то, что пассажиры

применяют смарт-карту только при посадке и не используют ее при выходе из автобуса. Следовательно, хоть начальный пункт каждой поездки известен, необходимо определить и конечный пункт. Более того, пассажиры, у которых есть смарт-карта, не обязаны применять ее для оплаты билета, они могут расплатиться наличными. Следовательно, записи о продажах билетов не хранят полную информацию, связанную с конкретным пассажиром, поэтому нельзя отследить несколько поездок одного и того же пассажира.

Модель для построения OD-матриц основана на восстановлении последовательности поездок для пассажиров, использующих смарт-карту. Подобный подход описан в соответствующей литературе [20, 21, 22]. Мы предполагаем, что у каждой смарт-карты есть только один пользователь. Рассматриваемый подход основан на обработке каждой поездки, получении данных о начальном пункте поездки и точном/приближенном определении конечного пункта поездки. Для приближенного определения вводятся следующие понятия: поездка с пересадками и прямая поездка. Основные детали вводимых понятий описаны далее.

Поездка с пересадками. В такой поездке пассажир при посадке в первый автобус оплачивает поездку смарт-картой, которая имеет уникальный идентификационный номер. Далее пассажир за ограниченный промежуток времени, указанный в билете, может сделать одну и более пересадок без покупки нового билета, а просто предоставив свою смарт-карту.

Такой подход позволяет отследить, хранит ли запись информацию о новой поездке (т.е. о покупке нового билета) или же она хранит информацию о пересадке (т.е. о подтверждении смарт-карты). Мы предполагаем, что пассажиры не делают длительных пеших перемещений во время поездок и что пассажир, совершающий пересадку, выходит на остановке, которая находится ближе всего к той остановке, на которой он сядет на следующий автобус. Посадка на автобус после пересадки записывается в систему. Приблизительное место пересадки с одного автобуса на другой определяется как ближайшая автобусная остановка конкретного маршрута.

Прямая поездка. В такой поездке не происходит пересадок. Дополнительно, последний этап поездки с пересадками рассматривается в качестве прямой поездки. В обоих случаях сложность состоит в точном определении пункта назначения. Для определения пунктов назначения мы вводим два предположения, которые часто используются в соответствующей литературе: 1) пассажиры начинают новую поездку на автобусной остановке, находящейся ближе всего к пункту назначения предыдущей поездки; 2) в конце дня пассажиры возвращаются на автобусную остановку, которая была пунктом отправки первой поездки текущего дня. Для определения пунктов назначения мы пытаемся построить "цепь" поездок каждого пассажира в определенный день. Для этого просматриваются все поездки, совершенные каждым пассажиром за 24 часа. Для каждой новой поездки мы пытаемся определить точку высадки, для этого ищется автобусная остановка, находящаяся в заранее

заданном радиусе от остановки предыдущей поездки. Также хранится лог поездок, которые не могут быть выстроены в "цепь" для определения эффективности метода.

В этом практическом примере на этапе предварительных вычислений, описанном в разделе 4, фильтруются те данные о продажах, которые несут неточную информацию. Мы предполагаем, что измерение GPS неточно, если положение автобуса удалено более чем на 50 метров от его маршрута. Отфильтрованные данные разбиваются на части по идентификационным номерам смарт-карт и передаются ведомым процессам. Таким образом, ведомый процесс обрабатывает информацию обо всех поездках определенного пассажира, поэтому взаимодействие между ведомыми процессами отсутствует. Для улучшения производительности в конце этапа предварительных вычислений записи сортируются по дате для последовательной передачи ведомым процессам.

Для экспериментального анализа использовался полный набор данных ИТС за январь 2015 года, который включал данные о продажах билетов и местоположении автобусов. В наборе данных находилась информация о более чем 500 тысячах смарт-карт (что соответствует более чем 13 миллионам поездок).

В описанной параллельной модели ведущий/ведомый необходимо выбрать объем задач, назначаемый каждому ведомому процессу. Подходящий объем задач предоставляет хорошее распределение нагрузки и уменьшение количества ненужных связей между ведущим и ведомыми процессами. Были проведены пять независимых вычислений с различным объемом задач и использованием различного числа ядер.

Полученные экспериментальные результаты показывают, что применение модели распределенных вычислений позволяет значительно улучшить эффективность обработки данных по сравнению с последовательной реализацией. Для прямых поездок с использованием объема задач в 5000 поездок и 24 ядер было получено значение ускорения 16.41. Результаты подтверждают улучшение времени работы с использованием распределенной вычислительной системы и нескольких вычислительных узлов.

Также результаты показывают, что размер объема задач значительно влияет на общее время работы алгоритма, при меньших размерах достигаются лучшие результаты. Дальнейшие эксперименты должны быть направлены на установление краевых значений, после которых затраты на связи между ведущими и ведомыми процессами негативно сказываются на времени выполнения.

6. Заключение и дальнейшие исследования

В этой работе была разработана и реализована распределенная вычислительная система для облачной обработки больших объемов данных

ИТС с использованием парадигмы MapReduce на платформе Hadoop. Выполнен обзор литературы с обсуждением предыдущих попыток использования распределенных вычислений для обработки данных ИТС в контексте умных городов. Эффективность разработанной модели отражена и с использованием двух практических примеров: 1) вычисление метрик качества обслуживания системы общественного транспорта с использованием данных о местоположении автобусов; 2) построение OD-матриц на основе данных о продажах билетов. В обоих случаях распределенная модель позволяет значительно уменьшить время обработки больших объемов исторических данных. Экспериментальный анализ в обоих случаях был произведен с использованием реальных данных ИТС г. Монтевидео, Уругвай.

Возможные направления дальнейших исследований: 1) использование различных источников данных ИТС; 2) разработка приложения для горожан с интуитивно понятным доступом к полученной информации; 3) применение полученной информации для решения оптимизационных задач, например, прокладка автобусных маршрутов, изменение мест автобусных остановок, расписание автобусов и т.д.

Благодарности. Работа выполнена при поддержке Правительства Российской Федерации, Акт 211, контракт № 02.A03.21.0011 и CONACYT (Consejo Nacional de Ciencia y Tecnología, México), грант номер 178415.

Список литературы

[1]. Deakin, M., & Al Waer, H. (2011). From intelligent to smart cities. Intelligent Buildings International, 3(3), 140-152.
[2]. Grava, S. (2003). Urban transportation systems. Choices for communities.
[3]. Chen, C., Ma, J., Susilo, Y., Liu, Y., Wang, M.: (2016). The promises of big data and small data for travel behavior (aka human mobility) analysis. Transportation Research Part C: Emerging Technologies 68, 285–299.
[4]. Sussman, J. S. (2008). Perspectives on intelligent transportation systems (ITS). Springer Science & Business Media.
[5]. Figueiredo, L., Jesus, I., Machado, J. T., Ferreira, J., & de Carvalho, J. M. (2001). Towards the development of intelligent transportation systems. In Intelligent transportation systems (Vol. 88, pp. 1206-1211).
[6]. Foster I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[7]. White T. (2009). *Hadoop: The Definitive Guide* (1st ed.). O'Reilly Media, Inc..
[8]. Attiya H. & Welch J. (2004). *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons.
[9]. Buyya R., Broberg J., & Goscinski. A. M. (2011). *Cloud Computing Principles and Paradigms*. Wiley Publishing.
[10]. Dean J. & Ghemawat S. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107-113.

[11]. Shafer, J., Rixner, S., & Cox, A. L. (2010). The hadoop distributed filesystem: Balancing portability and performance. In *IEEE International Symposium on Performance Analysis of Systems & Software* (pp. 122-133).

[12]. Zheng, X., Chen, W., Wang, P., Shen, D., Chen, S., Wang, X., ... & Yang, L. (2016). Big data for social transportation. IEEE Transactions on Intelligent Transportation Systems, 17(3), 620-630.

[13]. Oh, S., Byon, Y. J., & Yeo, H. (2016). Improvement of Search Strategy With K-Nearest Neighbors Approach for Traffic State Prediction. IEEE Transactions on Intelligent Transportation Systems, 17(4), 1146-1156.

[14]. Shi, Q., & Abdel-Aty, M. (2015). Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. Transportation Research Part C: Emerging Technologies, 58, 380-394.

[15]. Ahn, J., Ko, E., & Kim, E. Y. (2016). Highway traffic flow prediction using support vector regression and Bayesian classifier. In 2016 International Conference on Big Data and Smart Computing (BigComp) (pp. 239-244). IEEE.

[16]. Chen, X. Y., Pao, H. K., & Lee, Y. J. (2014). Efficient traffic speed forecasting based on massive heterogenous historical data. In Big Data (Big Data), 2014 IEEE International Conference on (pp. 10-17). IEEE.

[17]. Xia, D., Wang, B., Li, H., Li, Y., & Zhang, Z. (2016). A distributed spatial–temporal weighted model on MapReduce for short-term traffic flow forecasting. Neurocomputing, 179, 246-263.

[18]. Nesmachnow S. (2010). Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. Revista de la Asociación de Ingenieros del Uruguay 61 (1), 12-15.

[19]. Yang H., Sasaki T., Iida Y., Asakura Y. (1992). Estimation of origin-destination matrices from link traffic counts on congested networks, Transportation Research Part B: Methodological, Volume 26, Issue 6, Pages 417-434.

[20]. Trépanier, M., Tranchant, N., & Chapleau, R. (2007). Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems*, *11*(1), 1-14.

[21]. Wang, W., Attanucci, J. P., & Wilson, N. H. (2011). Bus passenger origin-destination estimation and related analyses using automated data collection systems. *Journal of Public Transportation*, *14*(4), 7.

[22]. Munizaga, M. A., & Palma, C. (2012). Estimation of a disaggregate multimodal public transport Origin–Destination matrix from passive smartcard data from Santiago, Chile. *Transportation Research Part C: Emerging Technologies*, *24*, 9-18.

[23]. Peña D., Tchernykh A., Nesmachnow S., Massobrio S., Drozdov A. Y., Garichev S. N. (2016). Multiobjective vehicle type and size scheduling problem in urban public transport using MOCell. IEEE International conference Engineering & Telecommunications, Moscow, Russia.

[24]. R. Massobrio, A. Pías, N. Vázquez, & S. Nesmachnow (2016). Map-Reduce for Processing GPS Data from Public Transport in Montevideo, Uruguay. In 2do Simposio Argentino de Grandes Datos.

[25]. E. Fabbiani, P. Vidal, R. Massobrio, & S. Nesmachnow (2016). Distributed Big Data analysis for mobility estimation in Intelligent Transportation Systems. In Latin American High Performance Computing Conference.

Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Cities

[1] *Renzo Massobrio <renzom@fing.edu.uy>*
[1] *Sergio Nesmachnow <sergion@fing.edu.uy>*
[2] *Andrei Tchernykh <chernykh@cicese.mx>*
[3] *Arutyun Avetisyan <arut@ispras.ru>*
[4] *Gleb Radchenko <gleb.radchenko@susu.ru>*
[1] *Universidad de la República, Montevideo 11300, Uruguay.*
[2] *CICESE Research Center, Ensenada , B.C. 22860, México*
[3] *Institute for System Programming of the RAS, Moscow, 109004, Russia*
[4] *South Ural State University, Chelyabinsk, 454080, Russia.*

Abstract. In this paper, we present a Big Data analysis paradigm related to smart cities using cloud computing infrastructures. The proposed architecture follows the MapReduce parallel model implemented using the Hadoop framework. We analyse two case studies: a quality-of-service assessment of public transportation system using historical bus location data, and a passenger-mobility estimation using ticket sales data from smartcards. Both case studies use real data from the transportation system of Montevideo, Uruguay. The experimental evaluation demonstrates that the proposed model allows processing large volumes of data efficiently.

Keywords: cloud computing; big data; smart cities; intelligent transportation systems.

DOI: 10.15514/ISPRAS-2016-28(6)-9

For citation: Massobrio R., Nesmachnow S., Tchernykh A., Avetisyan A., Radchenko G. Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Cities. *Trudy ISP RAN/Proc.ISP RAS*, vol. 28, issue 6, 2016. pp. 121-140 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-9

Acknowledgment. This work is partially supported by Government of the Russian Federation, Act 211, contract № 02.A03.21.0011, and CONACYT (Consejo Nacional de Ciencia y Tecnología, México), grant no. 178415. Datasets used in this paper are from Intendencia de Montevideo.

References

[1]. Deakin, M., & Al Waer, H. (2011). From intelligent to smart cities. Intelligent Buildings International, 3(3), 140-152.
[2]. Grava, S. (2003). Urban transportation systems. Choices for communities.
[3]. Chen, C., Ma, J., Susilo, Y., Liu, Y., Wang, M.: (2016). The promises of big data and small data for travel behavior (aka human mobility) analysis. Transportation Research Part C: Emerging Technologies 68, 285–299.

[4]. Sussman, J. S. (2008). Perspectives on intelligent transportation systems (ITS). Springer Science & Business Media.

[5]. Figueiredo, L., Jesus, I., Machado, J. T., Ferreira, J., & de Carvalho, J. M. (2001). Towards the development of intelligent transportation systems. In Intelligent transportation systems (Vol. 88, pp. 1206-1211).

[6]. Foster I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[7]. White T. (2009). *Hadoop: The Definitive Guide* (1st ed.). O'Reilly Media, Inc..

[8]. Attiya H. & Welch J. (2004). *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons.

[9]. Buyya R., Broberg J., & Goscinski. A. M. (2011). *Cloud Computing Principles and Paradigms*. Wiley Publishing.

[10]. Dean J. & Ghemawat S. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107-113.

[11]. Shafer, J., Rixner, S., & Cox, A. L. (2010). The hadoop distributed filesystem: Balancing portability and performance. In *IEEE International Symposium on Performance Analysis of Systems & Software* (pp. 122-133).

[12]. Zheng, X., Chen, W., Wang, P., Shen, D., Chen, S., Wang, X., ... & Yang, L. (2016). Big data for social transportation. IEEE Transactions on Intelligent Transportation Systems, 17(3), 620-630.

[13]. Oh, S., Byon, Y. J., & Yeo, H. (2016). Improvement of Search Strategy With K-Nearest Neighbors Approach for Traffic State Prediction. IEEE Transactions on Intelligent Transportation Systems, 17(4), 1146-1156.

[14]. Shi, Q., & Abdel-Aty, M. (2015). Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. Transportation Research Part C: Emerging Technologies, 58, 380-394.

[15]. Ahn, J., Ko, E., & Kim, E. Y. (2016). Highway traffic flow prediction using support vector regression and Bayesian classifier. In 2016 International Conference on Big Data and Smart Computing (BigComp) (pp. 239-244). IEEE.

[16]. Chen, X. Y., Pao, H. K., & Lee, Y. J. (2014). Efficient traffic speed forecasting based on massive heterogenous historical data. In Big Data (Big Data), 2014 IEEE International Conference on (pp. 10-17). IEEE.

[17]. Xia, D., Wang, B., Li, H., Li, Y., & Zhang, Z. (2016). A distributed spatial–temporal weighted model on MapReduce for short-term traffic flow forecasting. Neurocomputing, 179, 246-263.

[18]. Nesmachnow S. (2010). Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. Revista de la Asociación de Ingenieros del Uruguay 61 (1), 12-15.

[19]. Yang H., Sasaki T., Iida Y., Asakura Y. (1992). Estimation of origin-destination matrices from link traffic counts on congested networks, Transportation Research Part B: Methodological, Volume 26, Issue 6, Pages 417-434.

[20]. Trépanier, M., Tranchant, N., & Chapleau, R. (2007). Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems*, *11*(1), 1-14.

[21]. Wang, W., Attanucci, J. P., & Wilson, N. H. (2011). Bus passenger origin-destination estimation and related analyses using automated data collection systems. *Journal of Public Transportation*, *14*(4), 7.

[22]. Munizaga, M. A., & Palma, C. (2012). Estimation of a disaggregate multimodal public transport Origin–Destination matrix from passive smartcard data from Santiago, Chile. *Transportation Research Part C: Emerging Technologies*, *24*, 9-18.

[23]. Peña D., Tchernykh A., Nesmachnow S., Massobrio S., Drozdov A. Y., Garichev S. N. (2016). Multiobjective vehicle type and size scheduling problem in urban public transport using MOCell. IEEE International conference Engineering & Telecommunications, Moscow, Russia.

[24]. R. Massobrio, A. Pías, N. Vázquez, & S. Nesmachnow (2016). Map-Reduce for Processing GPS Data from Public Transport in Montevideo, Uruguay. In 2do Simposio Argentino de Grandes Datos.

[25]. E. Fabbiani, P. Vidal, R. Massobrio, & S. Nesmachnow (2016). Distributed Big Data analysis for mobility estimation in Intelligent Transportation Systems. In Latin American High Performance Computing Conference.

Automatic Analysis, Decomposition and Parallel Optimization of Large Homogeneous Networks

[1] *D.Yu. Ignatov <ignatov.dmitry@huawei.com>*
[1] *A.N. Filippov <filippov.alexander@huawei.com>*
[2] *A.D. Ignatov <ihnatova@student.ethz.ch>*
[1] *X. Zhang <zhangxuecang@huawei.com>*
[1] *Russian Research Center, Huawei Technologies, 1, Altufevskoe sh., Moscow, 127106, Russia*
[2] *Swiss Federal Institute of Technology in Zurich, Department of Computer Science, 101, Rämistrasse, Zurich, 8092, Switzerland*

Abstract. The life of the modern world essentially depends on the work of the large artificial homogeneous networks, such as wired and wireless communication systems, networks of roads and pipelines. The support of their effective continuous functioning requires automatic screening and permanent optimization with processing of the huge amount of data by high-performance distributed systems. We propose new meta-algorithm of large homogeneous network analysis, its decomposition into alternative sets of loosely connected subnets, and parallel optimization of the most independent elements. This algorithm is based on a network-specific correlation function, Simulated Annealing technique, and is adapted to work in the computer cluster. On the example of large wireless network, we show that proposed algorithm essentially increases speed of parallel optimization. The elaborated general approach can be used for analysis and optimization of the wide range of networks, including such specific types as artificial neural networks or organized in networks physiological systems of living organisms.

Keywords: homogeneous network; decomposition; optimization; distributed computing

DOI: 10.15514/ISPRAS-2016-28(6)-10

For citation: Ignatov D.Yu., Filippov A.N., Ignatov A.D., Zhang X. Automatic Analysis, Decomposition and Parallel Optimization of Large Homogeneous Networks. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 141-152. DOI: 10.15514/ISPRAS-2016-28(6)-10

1. Introduction

There are many large homogeneous networks consisting of the elements of the same type, which influence our everyday life, such as wired or wireless networks, network of switches in datacenter or artificial neural networks. In addition to mentioned information systems, there are networks transferring physical objects, such as a road network, different types of pipe networks or even organized in networks physiological systems of living organisms. The support of the effective continuous functioning of such networks requires their permanent screening and optimization. It is important that optimization of artificial networks begins on the stage of their planning, continues during all time of their usage and includes balancing of activity of network elements. Thus, whenever we begin an optimization, the network is already partially optimized and its elements have relatively similar level of activity.

An optimization of networks can be described as a maximization of objective function, which simulates activity of network. Objective function takes as input network parameters and provides estimation of performance and/or quality of service provided by network. The homogeneous networks have useful properties – since they consist of uniform elements, the network or its part can be optimized with the usage of the same objective function. Therefore, network can be decomposed into the relatively independent subnets, which are optimized in parallel processes. The quality objective function can be specified, for example, as an average level of radio signal in the area covered by the wireless network, or as an average speed of traffic in switches of datacenter or on crossroads of road network. In general, for homogeneous network decomposed into non-overlapping subnets, the quality objective function can be represented as an average value of objective functions of subnets:

$$Q\left(\underset{w}{\rightarrow}\right) = \frac{1}{n}\sum_{i=1}^{n} Q\left(\underset{w_i}{\rightarrow}\right),$$

where Q – quality of service provided by network, $\underset{w}{\rightarrow}$ – vector of network parameters, n – number of non-overlapping subnets, $\underset{w_i}{\rightarrow}$ – vector of parameters of i-th subnet.

The decomposition of complex systems before their optimization is an old idea raised with development of large-scale algorithms. Beginning from early works [1, 2] until now [3-5] the main decomposition approach is the detection of the mostly independent subnets with minimal strength of interactions between their elements. The network analysis leads to the calculation of strength of these interactions, and then decomposition is performed according to specific for network criterion. Typically, decomposition of network falls under the category of NP-hard problems, which are solved with graph partitioning algorithms. Obtained subnets can be optimized on multiple-core computer or cluster with such general technique as Simulated Annealing [4] or network-specific optimization algorithm [5].

The decomposition allows to do the optimization in parallel as well as to reduce optimization complexity by discarding of weak interactions [4], and, in such way, essentially increases speed of optimization. Remarkable, that decomposition of large network into the weakly interacting parts can reduce the complexity of error surface relief, and in such way decreases probability of stuck in local optimum [6]. Therefore, decomposition of network reduces optimization complexity as well as probability of optimization stuck in local optimum.

The quantity of networks, consisting of uniform elements, rapidly grows in modern industry, especially in the sphere of information technologies. Such growth stimulates the development of new efficient algorithms of network decomposition, which can provide maximal decrease of the optimization complexity and efficient parallelization of optimization process. In this paper, we propose optimization methods, which combine idea of independent optimization with the alternative decomposition approach. They are intended to solve problems with following characteristics: first, optimized network consists of the uniform elements; second, optimization of network is carried out in multi-core and distributed systems.

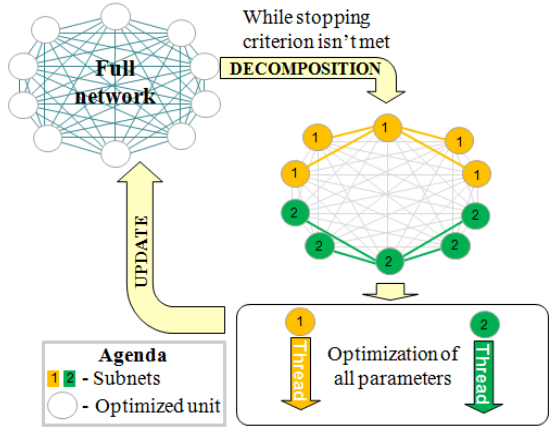


Fig. 1. Automatic sector planning for parallel optimization of network.

2. Network Decomposition approaches

With the purpose to find solution of the Maximal Independent Set and related problems the big variety of decomposition algorithms are elaborated [1-5]. They break a complex task of network optimization down into subtasks by decomposition of network into one set of relatively independent subnets. In order to perform decomposition, network is firstly represented with the weighted complete graph, where each vertex corresponds to a network element and each edge has weight equals to the rank of correlation between pair of correspondent elements. Hereby the rank of correlation is calculated by network-specific correlation function and represents the strength of elements interactions within network. The weights on

edges are used in the process of network decomposition into subnets weakly connected with each other, while data in vertex – for optimization of obtained subnets.

For example, described in invention [3] method of automatic sector planning of network for parallel optimization (Fig. 1) decomposes network into a predefined number of subnets by the rule of minimal sum of the crossing edges weights. After decomposition the subnets are optimized independently with the same objective function. Optimization of subnets is performed for all regulated parameters in distributed-centralized mode.

Main drawback of such approaches is an ignorance of crossing edges, while some of them appear to be significant. Thus, this ignorance negatively affects accuracy and time of optimization process.

3. Alternative decomposition with independent optimization

The alternative decomposition (Fig. 2) as well as existing approach (Fig. 1) is based on the representation of network data and structure with the weighted complete graph.

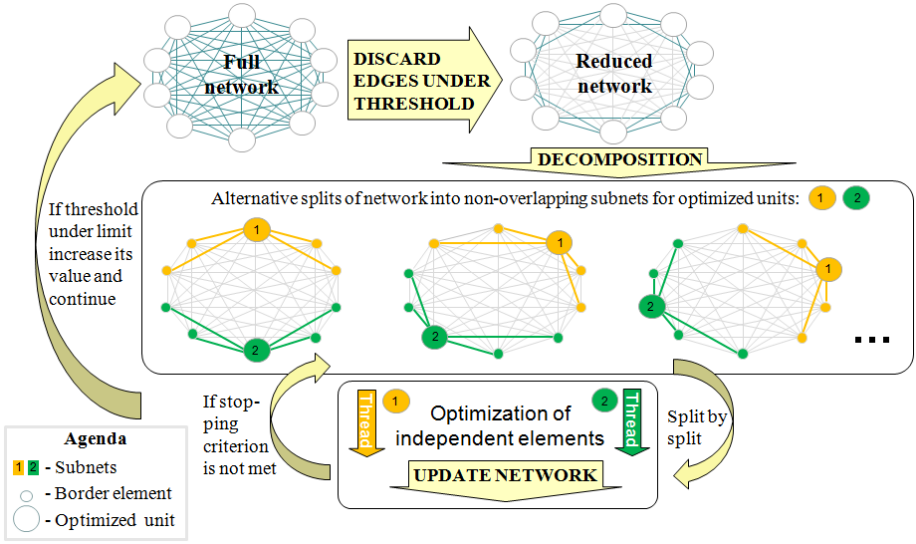


Fig. 2. Alternative decomposition of network with independent optimization.

The decomposition consists of two steps:

1. The weak correlations between elements are discarded by removing of edges with weights under network-specific filtering threshold.
2. Reduced network is decomposed into the set of alternative splits containing non-overlapping subnets.

Last step includes two stages (Fig. 3):

a) Selection of subnet for every optimized unit. This subnet consists of the unit and connected to it elements.

b) Finding (i.e. brute force search) of combinations of non-overlapping subnets – alternative splits of network. Every split is generated from at least one untapped subnet and covers as much vertices as possible.

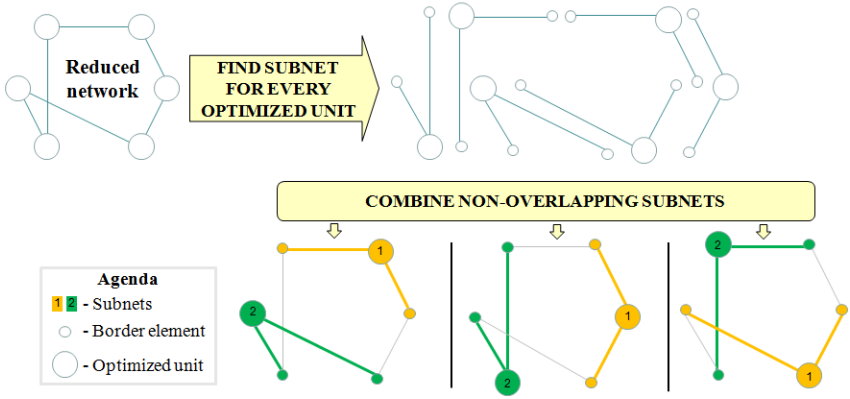


Fig. 3. Details of alternative decomposition.

Therefore, in order to reduce complexity of the optimization process the most irrelevant interactions between elements of network are removed. Based on remaining connections, the network is further split into relatively independent subnets. The result of splitting process is a set of alternative splits, where every split consists of all possible non-overlapping subnets within network. The obtained splits are optimized one by one in loop (Fig. 2, bottom). Within each iteration, the following steps are executed:

1. The next alternative split is selected.
2. The non-overlapping subnets are apportioned to the cores available on computer/cluster.
3. The optimization of subnets is performed in parallel processes by optimizing procedure.
4. The full network is updated with the values of optimized parameters.

Herewith the optimizing procedure is implementation of Simulated Annealing probabilistic technique of global optimization, which randomly searches for optimal solution in the space of all available alternatives [7]. We modified this method by adding new variable – step regulated by precision parameter (P), which restricts the area of neighbors search.

procedure optimize(S_0, **P**) {
  S_{new} := S_0
  step := **maxStep** * (1 – **P**)

S_{gen} := random neighbor of S_0 within **step**
T := temperature(1 – **P**)
if A(E(S_0), E(S_{gen}), T) ≥ random(0, 1) then S_{new} := S_{gen}
Output: state S_{new}
}

where S_0, S_{gen} and S_{new} – current, generated and new states of subnet, correspondently; maxStep – maximal value of step; temperature – monotonically increasing function mentioned in Simulated Annealing method; E and A – energy and acceptance functions of Simulated Annealing method, correspondently.

This optimizing procedure is performed on alternative splits in cycle as long as better states are founded. After optimizing procedure (Fig. 2, bottom) is finished, we increase filtering threshold value and repeat the cycle: network decomposition and further iterative optimizing procedure. Finally, when the filtering threshold reaches the maximal value, then the optimization process is stopped and optimized parameters are ready to be used for adjustment of physical network.

The effectiveness of this algorithm (Fig. 2) is provided by the parallel optimization of the most independent elements only, whereas in original approach (Fig. 1) all elements of network are optimized at once. The following regulation of optimization precision gives additional increase of optimization speed and quality. We use precision parameter P to control the precision of optimization by regulation of the step of Simulated Annealing algorithm as well as for calculation of threshold for filtering out weak connections between elements. At the beginning of the process, value of P is set to its minimum (i.e. zero) and is progressively increased with predefined constant up to the maximum (i.e. one). Before every decomposition procedure, the value of P is increased

$$P = P + const$$

and filtering threshold is calculated

$$Th = Th_{min} + (Th_{max} - Th_{min}) \times P,$$

where Th – filtering threshold parameter of decomposition process, Th_{min} – network-specific minimal value of Th, Th_{max} – network-specific maximal value of Th, P – precision parameter of optimization process.

Fig. 2 and 3 represent an example of simple network decomposition into set of alternative splits, where optimized unit consists of one element. In the case of large networks, decomposition could be based on the complex optimized units – group of strongly correlated elements, which are optimized together within subnet. The size of optimized unit is determined by the size of network and number of cores available on computer or cluster.

The peculiarities of optimization of alternative splits in distributed system are described bellow (Fig. 4). At the beginning of optimization process the copies of network-specific optimizing procedure are distributed on all nodes. Then the network is decomposed into alternative splits of subnets. Herewith for every split

while the number of subnets is lower than the total quantity of cores in computer cluster, subnets are randomly duplicated. When the number of subnets becomes equal to the quantity of cores in cluster, then optimization begins and subnets are sent to every node proportionally to quantity of cores on it. Optimization is carried out independently on every node without data shift. After optimization is finished every node sends back only optimized (i.e. changed) parameters of subnets, and if the alternative solutions exist for some subnet – the best of them is selected. Then network is updated with optimized parameters and the next split is processed.

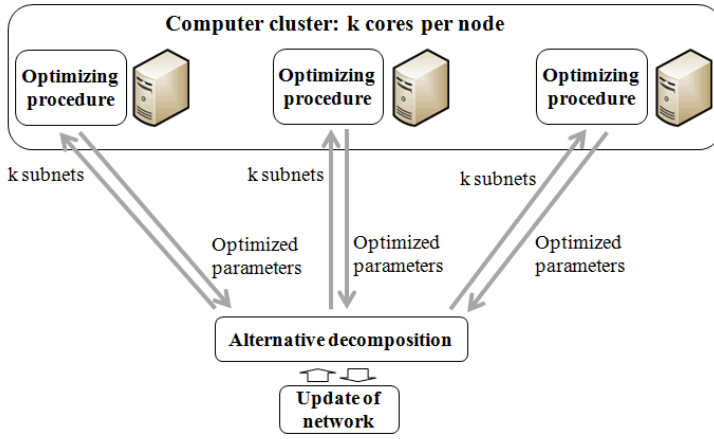


Fig. 4. Optimization of alternatives on computer cluster.

Optimization continues as long as objective function produces better result. If optimization of predefined number of splits does not give improvement, the optimization loop is finished, filtering threshold is increased and optimization continues on the next level of splitting. It is important, that in every split the optimized parameters are selected in such way, that they influence activity of full network. Thus, if optimization of predefined number of splits cannot improve network, the probability that the next split will give significant improvement is quite small. The adjustment of physical network is provided continuously – at the end of every optimization loop – or once, when the P value reaches its maximum.

4. Evaluation

For demonstration of the efficiency of proposed approach, the optimization of wireless network is implemented with visualization of quality of radio signal (Fig. 5). The quality is estimated with Signal to Interference plus Noise Ratio (SINR) [8]. An optimizing procedure is represented by modified Simulated Annealing algorithm. Objective function calculates the average value of SINR in covered by network area on the basis of Okumura-Hata, COST-231 and Stanford University Interim radio propagation models [9], which take into account the type of area and radio-frequency diapason. The result of radio propagation model is adjusted

according to tilt and azimuth propagation functions [10]. The experimental model of network consists of 100 sites with 3 sector antennas in each of them (total 300 antennas). Every antenna has 4 regulated parameters: power, height, tilt and azimuth (total 1200 regulated parameters). The rank of correlation between pair of antennas is calculated as 1 / (distance between antennas).

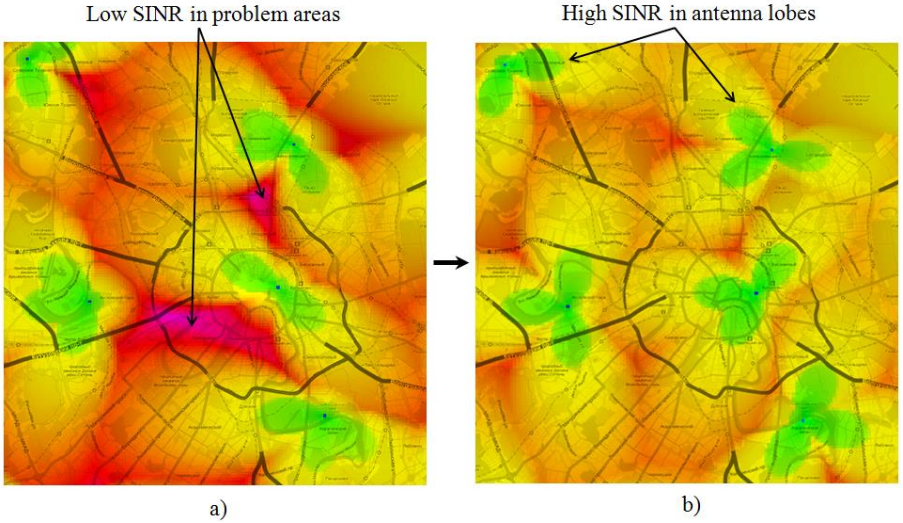


Fig. 5. Visual representation of Signal to Interference plus Noise Ratio (SINR) in initial (a) and optimized (b) subnet. Crimson and red territories – problem areas with low level of SINR, orange and yellow – sufficient level of SINR, and green – high level of SINR.

The SINR is automatically registered during optimization and its average values for 30 experiments are represented in logarithmic scale (Fig. 6). We can see, that after 3600 s the values (± 95 % conf. interval) of SINR reaches for sector planning 35.0 ± 1.7 dB and for independent optimization of alternatives 40.1 ± 1.9 dB, and don't change within experimental error after this time. Difference between represented values are significant ($p < 0.01$) and for independent optimization of alternatives 15 % higher, than in case of sector planning. According to approximation curve, the independent optimization of alternatives reaches the value 35.0 dB at ~ 400 s of experiment (Fig. 6, dashed line), so it is ~ 9 times faster, than sector planning algorithm with full optimization. Thus, by the example of large wireless network optimization we show that compare to recently published technique [3] the proposed algorithm (Fig. 2) gives 9 times speed-up or after long time optimization – demonstrates better accuracy.

The main principles, which according to our opinion underlay the speed-up of optimization, are represented on diagram (Fig. 7). According to these principles, optimization process begins with rough search of global optimum in small number

of complex subnets with the aim to avoid stuck in local optimum. During optimization the complexity of subnets is progressively decreased without loosing of optimization precision. Such decrease is reasonable, because during optimization process, the amplitude of oscillation of regulated parameters is decreased, and as a result, the strength of close interaction of element within network can essentially prevalent over the strength of distant interactions. According to our experiments (Fig. 6), this strategy gives accurate result of the optimization process.

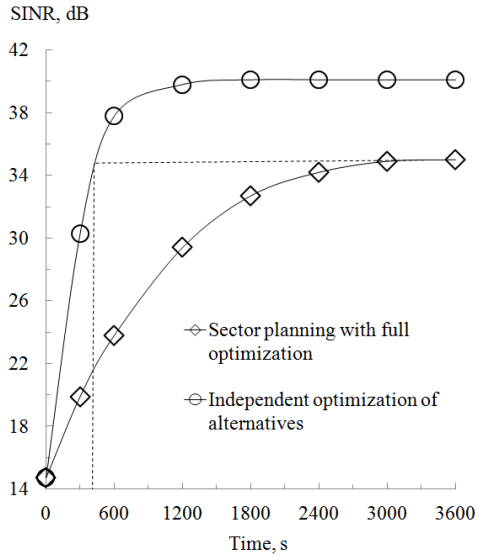


Fig. 6. Dependence of average value of Signal to Interference plus Noise Ratio (SINR) on time of optimization of wireless network.

From other side, automatic regulation of quantity of alternative calculations allows us to use all available computational resources for optimization. It is important that at the beginning of optimization process the rough search of global optimum produces essential oscillation of optimized parameter. Therefore, in this situation the alternative calculations are reasonable for selection of the best solution. Oppositely, at the end of the process, the optimization in small subnets produces almost the same results and the quantity of alternative calculations, which are not justifiable any more, are decreased (Fig. 7) to minimum. Thus, proposed algorithm accelerates the optimization process through regulation of precision parameter together with rational usage of all available calculation resources at different time of optimization process.

5. Conclusion

This paper presents a new meta-algorithm of network alternative decomposition into the sets of non-overlapping subnets and parallel optimization of the most independent parameters of network on computer or cluster. The optimization is carried out with progressive decreasing of subnet complexity and increasing of optimization precision. Usage of proposed algorithm leads to the following benefits:

- the faster optimization due to reduction of optimization problem complexity and efficient usage of all calculation resources;
- the better precision of optimization due to progressive change of optimization strategy from rough search of optimum at the beginning of optimization process to precise search of optimum at the end of optimization process.

We believe that proposed approaches of independent alternative optimization and dynamic regulation of precision are providing solid basis for the implementation of highly scalable distributed solution for the wide variety of large homogeneous networks.

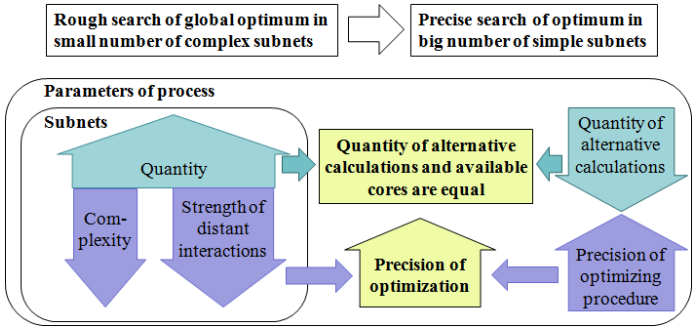


Fig. 7. Directions of progressive change of parameters of decomposition and optimization with the time. Vertical arrows: up and down – increase and decrease of parameter value, correspondently; horizontal arrows – influence of parameters on integral indices.

References

[1]. Awerbuch B., Goldberg A.V., Luby M., Plotkin S. Network decomposition and locality in distributed computation. Proc. of the 30th Annual Symposium on the Foundations of Computer Science (FOCS 89), 1989, pp. 364-369. doi: 10.1109/SFCS.1989.63504

[2]. Fantauzzi F., Gaivoronski A.A., Messina E. Decomposition Methods for Network Optimization Problems in the Presence of Uncertainty. Network Optimization, 1997, vol. 450, pp. 234-248. doi: 10.1007/978-3-642-59179-2_12

[3]. An automatic Sector Planning Method. Patent 104581745 A. Application CH 201510069770, filed February 10 2015, issued April 29 2015, 8 p.

[4]. Friesen A.L., Domingos P. Recursive Decomposition for Nonconvex Optimization. Proceedings of the 24th International Joint Conference on Artificial Intelligence, 2015, pp. 253-259.

[5]. Li M., Andersen D.G., Smola A.J. Graph Partitioning via Parallel Submodular Approximation to Accelerate Distributed Machine Learning, 2015, preprint arXiv:1505.04636. https://arxiv.org/abs/1505.04636
[6]. Singha T., Arbogasta J.E., Neagu N. An incremental approach using local-search heuristic for inventoryrouting problem in industrial gases. Computers and Chemical Engineering, 2015, vol. 80, pp. 199–210.
[7]. Aarts E., Korst J., Michiels W. Simulated Annealing / Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer Science, New York, 2013, pp. 265-285. doi: 10.1007/978-1-4614-6940-7_10
[8]. Lotfollahzadeh T., Kabiri S., Kalbkhani H., Shayesteh M.G. Femtocell base station clustering and logistic smooth transition autoregressive-based predicted signal-to-interference-plus-noise ratio for performance improvement of two-tier macro / femtocell networks. IET Signal Processing, 2016, vol. 10, issue: 1, pp. 1-11. doi: 10.1049/iet-spr.2014.0265
[9]. Kumar P., Patil B., Ram S. Selection of Radio Propagation Model for Long Term Evolution (LTE) Network. International Journal of Engineering, 2015, vol. 3, issue 1, pp. 373-379.
[10]. Kifle D.W., Wegmann B., Viering I., Klein A. Impact of Antenna Tilting on Propagation Shadowing Model. IEEE Conference on Vehicular Technology VTC, 2013, pp. 1-5. doi: 10.1109/VTCSpring.2013.6692585

Автоматический анализ, декомпозиция и параллельная оптимизация больших однородных сетей

[1] Д.Ю. Игнатов <ignatov.dmitry@huawei.com>
[1] А.Н. Филиппов <filippov.alexander@huawei.com>
[2] А.Д. Игнатов <ihnatova@student.ethz.ch>
[1] С. Чжан <zhangxuecang@huawei.com>
[1] Российский научно-исследований центр, Техкомпания Хуавэй, 127106, Россия, г. Москва, Алтуфьевское шоссе, д. 1.
[2] Швейцарская высшая техническая школа Цюриха, отдел компьютерных наук, 8092, Швейцария, г. Цюрих, Рамиштрассе, д. 101.

Аннотация. Жизнь современного мира во многом зависит от функционирования больших однородных сетей, таких как проводные и безпроводные коммуникационные системы, сети дорог и трубопроводов. Поддержание их эффективной работы требует автоматического контроля, постоянной оптимизации, включающей обработку больших объемов данных с использованием высокопроизводительных распределенных систем. Предложен новый мета-алгоритм для анализа больших однородных сетей, их альтернативного разбиения на слабосвязанные подсети и параллельной оптимизации наиболее независимых элементов подсетей. Данный подход основан на специфической для сети корреляционной функции, алгоритме имитации отжига и адаптирован для работы в вычислительном кластере. На примере безпроводной коммуникационной сети показано, что предложенный алгоритм существенно увеличивает скорость

многопоточной оптимизации. Разработанный общий подход может быть использован для анализа и оптимизации широкого спектра сетей, включая такие специфические типы как искусственные нейронные сети или организованные в виде сетей физиологические системы живых организмов.

Ключевые слова: однородные сети; декомпозиция; оптимизация; распределенные вычисления

DOI: 10.15514/ISPRAS-2016-28(6)-10

Для цитирования: Игнатов Д.Ю., Филиппов А.Н., Игнатов А.Д., Чжан С. Автоматический анализ, декомпозиция и параллельная оптимизация больших однородных сетей. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 141-152 (на английском). DOI: 10.15514/ISPRAS-2016-28(6)-10

Список литературы

[1]. Awerbuch B., Goldberg A.V., Luby M., Plotkin S. Network decomposition and locality in distributed computation. Proc. of the 30th Annual Symposium on the Foundations of Computer Science (FOCS 89), 1989, pp. 364-369. doi: 10.1109/SFCS.1989.63504

[2]. Fantauzzi F., Gaivoronski A.A., Messina E. Decomposition Methods for Network Optimization Problems in the Presence of Uncertainty. Network Optimization, 1997, vol. 450, pp. 234-248. doi: 10.1007/978-3-642-59179-2_12

[3]. An automatic Sector Planning Method. Patent 104581745 A. Application CH 201510069770, filed February 10 2015, issued April 29 2015, 8 p.

[4]. Friesen A.L., Domingos P. Recursive Decomposition for Nonconvex Optimization. Proceedings of the 24th International Joint Conference on Artificial Intelligence, 2015, pp. 253-259.

[5]. Li M., Andersen D.G., Smola A.J. Graph Partitioning via Parallel Submodular Approximation to Accelerate Distributed Machine Learning, 2015, preprint arXiv:1505.04636. https://arxiv.org/abs/1505.04636

[6]. Singha T., Arbogasta J.E., Neagu N. An incremental approach using local-search heuristic for inventoryrouting problem in industrial gases. Computers and Chemical Engineering, 2015, vol. 80, pp. 199–210.

[7]. Aarts E., Korst J., Michiels W. Simulated Annealing / Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer Science, New York, 2013, pp. 265-285. doi: 10.1007/978-1-4614-6940-7_10

[8]. Lotfollahzadeh T., Kabiri S., Kalbkhani H., Shayesteh M.G. Femtocell base station clustering and logistic smooth transition autoregressive-based predicted signal-to-interference-plus-noise ratio for performance improvement of two-tier macro / femtocell networks. IET Signal Processing, 2016, vol. 10, issue: 1, pp. 1-11. doi: 10.1049/iet-spr.2014.0265

[9]. Kumar P., Patil B., Ram S. Selection of Radio Propagation Model for Long Term Evolution (LTE) Network. International Journal of Engineering, 2015, vol. 3, issue 1, pp. 373-379.

[10]. Kifle D.W., Wegmann B., Viering I., Klein A. Impact of Antenna Tilting on Propagation Shadowing Model. IEEE Conference on Vehicular Technology VTC, 2013, pp. 1-5. doi: 10.1109/VTCSpring.2013.6692585

Parallel modularity computation for directed weighted graphs with overlapping communities[1]

[1]*Mikhail Drobyshevskiy <drobyshevsky@ispras.ru>*
[1]*Anton Korshunov <korshunov@ispras.ru>*
[1,2,3]*Denis Turdakov <turdakov@ispras.ru>*
[1]*Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[2]*Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[3]*National Research University Higher School of Economics (HSE)*
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. The paper presents new versions of modularity measure for directed weighted graphs with overlapping communities. We consider several approaches to computing modularity and try to extend them. Taking into account computational complexity, we suggest two parallelized extensions which are scalable to large graphs (more than 10^4 nodes).

Keywords: modularity; community detection; PageRank; LinkRank; belonging function; belonging coefficient.

DOI: 10.15514/ISPRAS-2016-28(6)-11

For citation: Drobyshevskiy M., Korshunov A., Turdakov D. Parallel modularity computation for directed weighted graphs with overlapping communities. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 153-170. DOI: 10.15514/ISPRAS-2016-28(6)-11

1. Introduction

The motivation of our research into modularity computation was the need to quantitatively assess and compare the quality of various clustering algorithms applied to mobile call graphs. As soon as no such graphs with ground-truth

[1]This research was collaborated with and supported by Russian Research Center, Huawei Technologies.

community structure were found, we couldn't use the most popular quality metric based on Normalized Mutual Information (NMI).

For evaluating quality of community detection methods on graphs with unknown reference communities, metrics based on probabilistic models are used. Such metrics include modularity, surprise, significance [19], ER-modularity [5]. Also, generative models from model-based community detection methods can be used to estimate likelihood of clustered graph [15, 11].

Modularity value characterizes the strength of a particular clustering of a graph. It is high when clusters are dense and sparsely connected to each other, whereas its value is low when clusters are formed at random. Besides evaluation of community cover, modularity is also used as optimization function in some community detection algorithms [16, 18]. In [12] modularity is also used for graph partitioning, but only for the case of two communities.

Here we consider modularity metric, its existing extensions for directed and weighted graphs and for the case of overlapping communities. Then we describe our extensions of modularity for overlapping communities in directed weighted graphs.

2. Notation

In this paper we will use the following notation, most of which are common in graph theory.

$G(V, E)$ – graph with nodes V and edges E, nodes $i, j, k \in V$, edge $l(i, j) \in E$;

A – adjacency matrix of graph G;

$A_{i,j}$ – an element of A;

$w_{i,j}$ – weight of edge $l(i, j)$;

k_i – degree of node i;

C – set of communities on graph G, $c \in C$ – particular community;

C_i – set of communities node i belongs to;

S – average community size in graph G, $S = \frac{1}{|C|} \sum_{c \in C} |c|$;

Σ – average square community size in graph G, $\Sigma = \frac{1}{|C|} \sum_{c \in C} |c|^2$;

We will also use V, E, C instead of $|V|, |E|, |C|$ to denote sizes of corresponding sets.

3. Existing versions of modularity

Modularity was defined by Newman and Girvan [3] to measure a quality of a partition of a graph into a set of clusters. It is the fraction of edges within the clusters minus the expected such fraction in a randomly connected graph with the same nodes and their degrees. Modularity was originally defined for undirected unweighted graphs and is given by:

$$Q = \sum_{c \in C} \left[\frac{E_c^{in}}{E} - \left(\frac{2E_c^{in} + E_c^{out}}{2E} \right)^2 \right], \quad (1)$$

where E_c^{in} – number of edges between nodes within community c, E_c^{out} – number of edges from the nodes in community c to the nodes outside c.

Modularity can equivalently be expressed via adjacency matrix A_{ij} and nodes degrees k_i:

$$Q = \frac{1}{2E} \sum_{c \in C} \sum_{i,j \in c} \left(A_{ij} - \frac{k_i k_j}{2E} \right) \quad (2)$$

There are three main directions of extension of the original modularity definition: for directed graphs, for weighted graphs, and for the case of overlapping communities.

3.1 Modularity for directed and weighted graphs

Extension of modularity (2) to directed graphs is rather straightforward [7]:

$$Q = \frac{1}{E} \sum_{c \in C} \sum_{i,j \in c} \left(A_{ij} - \frac{k_i^{out} k_j^{in}}{E} \right), \quad (3)$$

where k_i^{out} is out-degree of node i and k_j^{in} is in-degree of node j.

Modularity (2) is easily generalized to weighted graphs as well [2]:

$$Q = \frac{1}{2m} \sum_{c \in C} \sum_{i,j \in c} \left(w_{ij} - \frac{w_i w_j}{2m} \right), \quad (4)$$

where w_{ij} – weight of edge $l(i,j)$, $w_i = \sum_j w_{ij}$ is sum of all weights of edges of node i, and $m = \frac{1}{2} \sum_{i,j} w_{ij}$ is total weight of all edges.

Moreover, modularity formula (2) for both weighted and directed graphs can be written as [6]:

$$Q = \frac{1}{m} \sum_{c \in C} \sum_{i,j \in c} \left(w_{ij} - \frac{w_i^{out} w_j^{in}}{m} \right) \quad (5)$$

Finally, modularity based on LinkRank, was suggested for weighted directed graphs [9]:

$$\begin{aligned} Q &= \sum_{c \in C} \sum_{i,j \in c} (L_{ij} - \pi_i \pi_j) \\ L_{ij} &= \pi_i G_{ij} - \text{LinkRank}, \\ \vec{\pi} &= (\pi_1, \ldots, \pi_V) - \text{PageRank vector} \end{aligned} \quad (6)$$

LinkRank is an analogy of PageRank [14] for links. PageRank is the probability of a particular page (node) being visited by a random surfer and can be defined as a stationary row vector of Google Matrix G: $\vec{\pi}^T = \vec{\pi}^T G$. In case of directed graphs Google Matrix $G_{ij} = \alpha \frac{w_{ij}}{w_i^{out}} + \frac{1}{N}(\alpha g_i + 1 - \alpha)$, where α is damping parameter for PageRank (with probability $1 - \alpha$ random surfer jumps to a random node) and g_i is indicator of dangling node:

$$g_i = \begin{cases} 1 & \text{if node is dangling } (w_i^{out} = 0) \\ 0 & \text{otherwise} \end{cases}$$

This formula originates from an alternative notion of community as a group of nodes where a random surfer spends more time in average. More technically, this definition of modularity is the deviation between the fraction of time a random walker spends within communities and the expected such time.

3.2 Overlapping modularity

In the case when a node can belong to several communities, the belonging coefficients $a_{i,c}$ are introduced [8] which indicate how much a node i belongs to community c. This coefficients are non-negative and sum to one: $\forall i \in V, \forall c \in C \; a_{i,c} > 0 \; \sum_{c \in C} a_{i,c} = 1$. This relates to another extension of community detection problem, called fuzzy community detection [13]. To generalize different approaches of using belonging coefficients, a belonging function $f(a_{i,c_i}, a_{j,c_j})$ can be defined [17] to characterize an extent to what an edge (i, j) connects communities c_i and c_j respectively.

According to this, several approaches for overlapping modularity from the literature can be generalized to the following two definitions [17]:

$$\begin{aligned} Q &= \sum_{c \in C} \left[\frac{E_c^{in}}{E} - \left(\frac{2E_c^{in} + E_c^{out}}{2E} \right)^2 \right] \\ E &= \frac{1}{2} \sum_{i,j \in V} A_{ij} \\ E_c^{in} &= \frac{1}{2} \sum_{i,j \in c} A_{ij} \cdot f(a_{i,c}, a_{j,c}) \\ E_c^{out} &= \sum_{i \in c} \sum_{j \in c' \neq c} A_{ij} \cdot f(a_{i,c}, a_{j,c'}) \end{aligned} \tag{7}$$

and

$$Q = \frac{1}{2E} \sum_{c \in C} \sum_{i,j \in c} \left(A_{ij} - \frac{k_i k_j}{2E} \right) f(a_{i,c}, a_{j,c}) \tag{8}$$

where belonging coefficient can be:

$$a_{i,c} = \begin{cases} \dfrac{1}{C_i} \\ \dfrac{\sum_{k \in c} A_{ik}}{\sum_{c' \in C_i} \sum_{k \in c'} A_{ik}} \\ \dfrac{\sum_{k \in c} \frac{M^c_{ik}}{M_{ik}} A_{ik}}{\sum_{c \in C} \sum_{k \in c} \frac{M^c_{ik}}{M_{ik}} A_{ik}} \end{cases}, \tag{9}$$

where M_{ik} is the number of maximal cliques containing edge (i,j), M^c_{ik} is the number of maximal cliques containing edge (i,j) inside community c. Belonging function can be:

$$f(a,b) = \begin{cases} \frac{a+b}{2} \\ ab \\ max(a,b) \end{cases} \tag{10}$$

3.3 Further extensions of modularity

Besides the node-based extensions, there was suggested edge-based extension [10] (for directed graphs):

$$\begin{gathered} Q = \frac{1}{E} \sum_{c \in C} \sum_{i,j \in V} \left(\beta_{l(i,j),c} A_{ij} - \frac{\beta^{out}_{l(i,\cdot),c} k^{out}_i \beta^{in}_{l(\cdot,j),c} k^{in}_j}{E} \right) \\ \beta_{l(i,j),c} = f(a_{i,c}, a_{j,c}) - \text{edge belonging coefficient}, \\ \beta^{out}_{l(i,\cdot),c} = \frac{1}{V} \sum_{k \in V} f(a_{i,c}, a_{k,c}) - \text{expected of that for outcoming link}, \\ \beta^{in}_{l(\cdot,j),c} = \frac{1}{V} \sum_{k \in V} f(a_{k,c}, a_{j,c}) - \text{expected of that for incoming link}. \end{gathered} \tag{11}$$

Here edge belonging function $f(a_{i,c}, a_{j,c})$ can be any of (10), but the authors suggested this variant (together with empirically found expression for $h(x)$):

$$\begin{gathered} f(a,b) = \frac{1}{(1+e^{-h(a)})(1+e^{-h(b)})}, \\ h(x) = 2px - p, \quad p = 30. \end{gathered} \tag{12}$$

It is worth to notice that actually in the inner sum iterating of pairs of nodes i, j are done over nodes only from community c (not from the whole V), due to the form of β functions.

Authors of [17] suggested density-based version of modularity (1) for overlapping directed graphs:

$$Q_D = \sum_{c \in C} \left[\frac{E_c^{in}}{E} \cdot d_c - \left(\frac{2E_c^{in} + E_c^{out}}{2E} \cdot d_c \right)^2 - \sum_{c' \neq c} \frac{E_{c,c'}}{2E} d_{c,c'} \right]$$

$$d_c = \frac{2E_c^{in}}{\sum_{i \neq j \in c} f(a_{i,c}, a_{j,c})} - \text{internal density}$$

$$d_{c,c'} = \frac{E_{c,c'}}{\sum_{i \in c, j \in c'} f(a_{i,c}, a_{j,c'})} - \text{pair-wise density} \quad (13)$$

$$E_{c,c'} = \sum_{i \in c, j \in c'} A_{ij} \cdot f(a_{i,c}, a_{j,c'})$$

3.4 Drawbacks and limitations

The first obvious drawback is that there was not found any modularity formula, comprising all three needed properties: support of directed, weighted graphs with overlapping communities.

The second limitation is computational complexity. Aforementioned formulas of overlapping modularity are not acceptable for large graphs (with more than 10^4 nodes within community cover) due to their high computational complexity. Denoting the average number of communities by C, average community size by S and number of nodes by V, we have for (13) time complexity $O(C^2\Sigma)$, and for (11) — $O(CV^2\Sigma)$. See subsection 4.1 for more details.

It's also worth noting that LinkRank authors [9] provide some evidence that the modularity (5) can't distinguish the direction of links.

4. Our extensions of modularity

Since we focus on modularity for directed weighted graphs with overlapping communities, we actually have two possibilities of extension: make overlapping (directed) modularities support weights, or to extend directed weighted modularities to the overlapping case.

The first approach suggests naive substitution of adjacency matrix of a graph to matrix of weights and number of edges to the sum of their weights. Doing so with density formula (13) leads to unnormalization: values of modularity start to exceed the available range $(-\frac{1}{2}; 1]$. But we will still use it in experiments with unweighted graphs. On the other hand, edge-based formula seems to allow such generalization, becoming:

$$Q_E = \frac{1}{m} \sum_{c \in C} \sum_{i,j \in V} \left(\beta_{l(i,j),c} w_{ij} - \frac{\beta^{out}_{l(i,\cdot),c} w_i^{out} \beta^{in}_{l(\cdot,j),c} w_j^{in}}{m} \right) \quad (14)$$

But this is still computationally expensive.

The second approach consists in introducing belonging coefficients (9) and belonging functions (10) to simple version (5):

$$Q_S = \frac{1}{m} \sum_{c \in C} \sum_{i,j \in c} \left(w_{ij} - \frac{w_i^{out} w_j^{in}}{m} \right) f(a_{i,c}, a_{j,c}), \tag{15}$$

and to LinkRank-based version of modularity (6):

$$Q_{LR} = \sum_{c \in C} \sum_{i,j \in c} (L_{ij} - \pi_i \pi_j) f(a_{i,c}, a_{j,c}). \tag{16}$$

Since PageRank (and hence LinkRank) has fast implementations ([1, 4]), these two formulas have much lower computational complexities.

Also, we suggested to use in formulas a normalization coefficient instead of belonging function:

$$f(a_{i,c}, a_{j,c}) \mapsto \frac{1}{|C_i \cap C_j|} \tag{17}$$

The intuition is the following. If both nodes i, j belong to n communities, the term $\sum_{i,j \in c} (...)$ will encounter n times in modularity formula, once for each community, so we weigh it by the factor of $\frac{1}{n} = \frac{1}{|C_i \cap C_j|}$. It's easy to see that otherwise modularity can become unlimited: suppose that each community is actually two equal different communities, then modularity value doubles.

4.1 Computational complexity

Here we calculate computational complexities of modularity extensions Q_D, Q_E, Q_S and Q_{LR}. All complexities are present in table 1.

Firstly, denote by $O(F)$ computational complexity of $f(a_{i,c}, a_{j,c})$ – we consider it later.

In the expression for Q_D (13), the term E_c^{out} is computed in $O(C\Sigma F)$, so as d_c; E_c^{out} in $O(C\Sigma F)$; $E_{c,c'}$ and $d_{c,c'}$ in $O(S^2 F)$ time. Counting that average square community size Σ is not less than square of average size S^2, each term of summation has complexity $O(C\Sigma F)$, giving overall complexity $O(C^2 \Sigma F)$.

In the expression for Q_E (14), the hardest term is β^{in} and β^{out}, which take $O(VF)$ steps, thus resulting in $O(CV^2F^2)$ overall complexity.

Q_S (15) and Q_{LR} (16) have complexity $O(C\Sigma F)$, ignoring PageRank calculation time as insignificant. Understanding the big-O complexity of PageRank calculation requires analyzing the code of pagerank scipy method from NetworkX[2]. However, Aric Hagberg (NetworkX Lead Programmer) wrote that their implementation has

[2] http://networkx.github.io/

"linear complexity in the number of edges". In practice, PageRank computation time is negligible.

Now consider $f(a_{i,c}, a_{j,c})$. Uniform belonging coefficient $a_{i,c} = \frac{1}{C_i}$ may be computed by one operation if communities for each node are explicitly known, e.g. each node has a set of labels. But usually community detection algorithms return list of communities represented by sets of nodes. This means we need $O(C \log S)$ operations to find all communities a given node i belongs to. The same concerns fraction belonging coefficient, for which we have $O(S + C \log S)$, supposing that average node membership is not very high, i.e. $C_i = O(1)$. Therefore, intersection belonging function together with the others are $O(1)$.

Table 1: Computational complexities for modularity formulas, belonging functions and belonging coefficients.

formula	**complexity**
Q_D	$O(C^2\Sigma)$
Q_E	$O(CV^2\Sigma)$
Q_S	$O(C\Sigma)$
Q_{LR}	$O(C\Sigma)$
belonging function	
sum	$O(1)$
product	$O(1)$
intersection	$O(1)$
edge-based	$O(1)$
belonging coefficient	
uniform	$O(C \log S)$
fraction	$O(S + C \log S)$

4.2. Effects

In order to demonstrate adequacy of the estimate based on computed modularity with regard to intuitive community structure, we computed modularities of several alternative community covers of the example graph (see Fig. 1). We generated a large set of random community covers, and sort them according to the modularity value computed with formula (15). Fig. 1 demonstrates 3 covers with highest modularity and 3 covers with lowest modularity. We can see that the most intuitive cover corresponds to the highest modularity value. The same holds for formula (16).

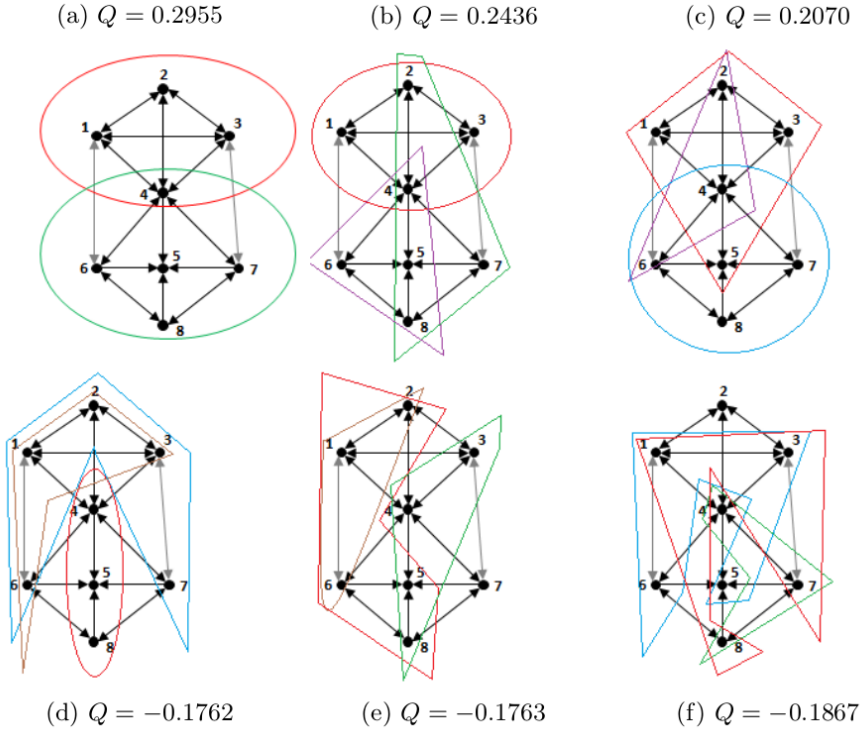


Fig. 1. Modularity (15) of different community covers for example directed weighted graph. All edges have weight 10 except (1-6) and (3-7) which have weight 0.1. Top row: covers with maximal modularity; bottom row: covers with minimal modularity.

5. Experiments

We implemented in Python four versions of modularity Q_D, Q_E, Q_S and Q_{LR} together with 4 belonging functions (see (10) and (12)):

1. sum $f(a,b) = \frac{1}{2}(a+b)$,
2. product $f(a,b) = ab$,
3. intersection $f(a,b) = \dfrac{1}{|C_i \cup C_j|}$,
4. edge-based $f(a,b) = (1+e^{-60a+30})^{-1}(1+e^{-60b+30})^{-1}$,

and two belonging coefficients:

1. uniform $a_{i,c} = \frac{1}{C_i}$,
2. fraction $a_{i,c} = \frac{\sum_{k \in c} A_{ik}}{\sum_{c' \in C_i} \sum_{k \in c'} A_{ik}}$.

Also we conducted a set of experiments: on computation time, different belonging functions and belonging coefficients, and parallelizing.

5.1 Computation time

We compared modularity value and computation time of four appropriate formulas (Q_D, Q_E, Q_S, Q_{LR}) on two graphs of different size. Since Q_D doesn't support weights and fraction belonging coefficient is undefined for directed graphs (due to possible zero in denominator), graphs were chosen undirected unweighted. Experiments with directed weighted graphs are to be conducted later. We took default belonging functions (suggested in original papers) and uniform belonging coefficient for simplicity.

The small graph was generated by CDR-GEN generator[3] and clustered by SLPA algorithm[4] with threshold $p = 0.10$. Parameters of the smaller graph are: number of nodes $|V| = 3124$, number of edges $|E| = 3913$, number of communities $|C| = 333$, average size of community $S = 10.1$ with 100% of nodes involved in communities, average membership 1.14.

The big graph was Wu et al dataset[5] clustered by MOSES algorithm[6] with 7% of nodes involved in communities ($|V| = 72111$, $|E| = 79003$, $|C| = 899$, $S = 5.2$). Results are in table 2.

Table 2 shows that as size of graph and size and number of communities grow, Q_D and Q_E become too computationally expensive, so there are only two scalable candidates, Q_S and Q_{LR}.

Table 2: Modularity value and computation time for Q_S, Q_{LR}, Q_D and Q_E on 2 undirected unweighted graphs.

formula	bel.func.	bel.coef.	complexity	Time `small`	Time `big`
Q_D	product	uniform	$O(C^3 \Sigma \log S)$	3m57s	3h12m
Q_E	edge-based	uniform	$O(C^2 V^2 \Sigma \log S)$	7m17s	53h53m
Q_S	intersection	uniform	$O(C^2 \Sigma \log S)$	0.6s	18s
Q_{LR}	intersection	uniform	$O(C^2 \Sigma \log S)$	0.7s	18s

[3] https://github.com/mayconbordin/cdr-gen
[4] https://sites.google.com/site/communitydetectionslpa/
[5] http://www.pnas.org/content/107/44/18803?tab=ds
[6] https://sites.google.com/site/aaronmcdaid/moses

Table 3: Comparison of different belonging functions and belonging coefficients for Q_D, Q_E, Q_S and Q_{LR} on undirected unweighted graph with |V | = 72146, |E| = 79003, |C| = 1894, S = 5.30 (clustered by Clique Percolation).

formula	bel.coef.	sum	product	intersection	edge-based
Q_D	uniform	0.292 (2481 s)	0.291 (2425 s)	undefined	0.291 (2266 s)
Q_D	fraction	0.292 (2355 s)	0.292 (2187 s)	undefined	0.293 (2752 s)
Q_E	uniform	0.737 (4688 s)	0.714 (3201 s)	0.763 (1488 s)	0.713 (3349 s)
Q_E	fraction	0.740 (4130 s)	0.718 (4795 s)	0.763 (1476 s)	0.719 (3683 s)
Q_S	uniform	0.737 (4 s)	0.714 (3 s)	0.760 (1 s)	0.713 (2 s)
Q_S	fraction	0.738 (3 s)	0.715 (3 s)	0.760 (1 s)	0.717 (3 s)
Q_{LR}	uniform	0.647 (4 s)	0.628 (4 s)	0.665 (2 s)	0.628 (4 s)
Q_{LR}	fraction	0.647 (5 s)	0.630 (5 s)	0.665 (2 s)	0.631 (8 s)

Table 4: Comparison of different belonging functions for Q_S and Q_{LR} on a directed weighted graph with overlapping communities (|V | = 72146, |E| = 79003, |C| = 1894, S = 5.30, clustered by Clique Percolation).

	$\|\mathbf{C}\| = \mathbf{1894}$, $\mathbf{S} = \mathbf{5.30}$ (Clique Percolation)			$\|\mathbf{C}\| = \mathbf{6731}$, $\mathbf{S} = \mathbf{9.05}$ (SLPA)		
formula	**sequential**	**parallel 1**	**parallel 2**	**sequential**	**parallel 1**	**parallel 2**
Q_S	104s	37s	38s	106m	36m	32m
Q_{LR}	105s	39s	40s	103m	37m	32m

Table 5: Comparison of times of sequential and parallel versions (N = 6 processes) of Q_S and Q_{LR} on a directed unweighted graph with |V | = 72146, |E| = 79003 clustered by Clique Percolation (covers 13% of nodes) and SLPA (covers 78% of nodes) algorithms.

	$\|\mathbf{C}\| = \mathbf{1894}$, $\mathbf{S} = \mathbf{5.30}$ (Clique Percolation)			$\|\mathbf{C}\| = \mathbf{6731}$, $\mathbf{S} = \mathbf{9.05}$ (SLPA)		
formula	**sequential**	**parallel 1**	**parallel 2**	**sequential**	**parallel 1**	**parallel 2**
Q_S	104s	37s	38s	106m	36m	32m
Q_{LR}	105s	39s	40s	103m	37m	32m

5.2 Belonging functions and belonging coefficients

Then we investigated influence of different belonging functions and belonging coefficients on values of Q_S and Q_{LR}. We used the same Wu et al dataset clustered by Clique Percolation algorithm[7] with 13% of nodes involved in communities ($|V| = 72146, |E| = 79003, |C| = 1894, S = 5.30$).

Table 3 shows that the choice of belonging function or belonging coefficient doesn't make much difference to result modularity. Meanwhile, intersection belonging function takes the lowest time. Values of Q_S are in good consistency with those of Q_E, which is widely used in papers. Q_{LR} values tend to be less than Q_S and Q_E. Q_D values differ a lot, possibly due to dissimilar formula structure, but as far as we know this formula was not compared to other ones in literature.

[7] http://www.cfinder.org/

Table 4 extends the comparison of different belonging functions for Q_S and Q_{LR} on a directed weighted graph with overlapping communities. Belonging coefficient is uniform. We see that the behavior is consistent with that of undirected unweighted case.

5.3 Parallel modularity

Computation process of Q_S and Q_{LR} naturally allows parallelization. Since each community and each node pair contributes independently to the modularity value, iterating over node pairs may be distributed between processors.

We implemented two parallel versions. The first one is rather straightforward. Iteration over communities is left sequential. Each time when community of size more than $c_0 = 100$ is encountered, $N = 6$ parallel processes are initialized. The set of all nodes pairs within the community is split into N equal chunks and are assigned to these processes (see algorithm 1).

for $c \in C$ **do**
  if $|c| > c_0$ **then**
    $\{c_1, \ldots, c_N\} \leftarrow$ split c into N equal chunks
    do in parallel $i \in \overline{1, N}$:
      computeModularity(c_i)
  end
end

Algorithm 1: *Parallel modularity version 1.*

The second parallel version is a little more complicated. The idea is to split the set of communities C into subsets between processors. But in order to balance the load, these chunks should have approximately equal sum of squares of community size since community of size s has s^2 ordered node pairs (counting self-loops). To achieve this we used a greedy algorithm, which iterates over communities in descending order and assigns each of them to a subset that has the smallest sum of size squares. The only problem here is that the biggest community may have size square much more than sum of size squares of the rest ones, i.e. the chunk which gets this community will be overloaded. To overcome this challenge we sort communities by their sizes in descending order and apply the first parallel approach to first (biggest) several communities, until we encounter community with small enough size to allow balancing of the rest ones or reach lower community size bound c_0. The rest ones are split into subsets according to the mentioned greedy algorithm. To determine whether to start balancing we use a simple condition: square of size of current biggest community should be at most $\frac{1}{N}$ of total sum of squares of sizes of communities left at the moment. Formally, having sorted sizes of

communities $s_1 \geqslant s_2 \geqslant ... \geqslant s_C$, the condition of stopping at community k is $s_k^2 \leqslant \frac{1}{N}(s_k^2 + ... + s_C^2)$. See algorithm 2.

```
sortBySizeInDescendingOrder(C)
for c_i ∈ C do
    if |c_k| ⩽ c_0 or
     |c_k|^2 ⩽ (1/N)(|c_k|^2 + ... + |c_|C||^2) then
      break
    end
    C ← C \ {c_k}
    {c_1, ..., c_N} ← split c into N equal chunks
    do in parallel i ∈ 1,N:
     computeModularity(c_i)
end
{C_1, ..., C_N} ← balanceSumOfSquares(C)
do in parallel i ∈ 1,N:  for c_ik ∈ C_i do
  computeModularity(c_ik)
end
```

Algorithm 2: *Parallel modularity version 2.*

We compared the speedup due to both versions of parallellization versus sequential computing of modularity for Q_S and Q_{LR}. See table 5. When number of communities is small ($|C| = 1894$) the first method is slightly faster due to its simplicity (results were averaged over 5 runs). In case of many communities the second version shows its benefit.

We also investigated process scalability of both parallel implementations. The results are represented in Fig. 1.

6. Conclusion

We investigated existing approaches to computing modularity measure and developed Q_S and Q_{LR} – modularity extensions for large directed weighted graphs with overlapping communities. These extensions have low computational complexity which makes them applicable to graphs with more than 10^4 nodes and they also can be computed in parallel way.

These two formulae are based on different notions of community: as group of nodes with more dense links (Q_S) or a group of nodes where a random surfer tends to spend more time (Q_{LR}). Since a surfer walks along link direction, the second formula is more sensible to direction of links in a graph.

As a future direction may be considered a possibility to use new version of modularity for overlapping community detection in directed weighted graphs.

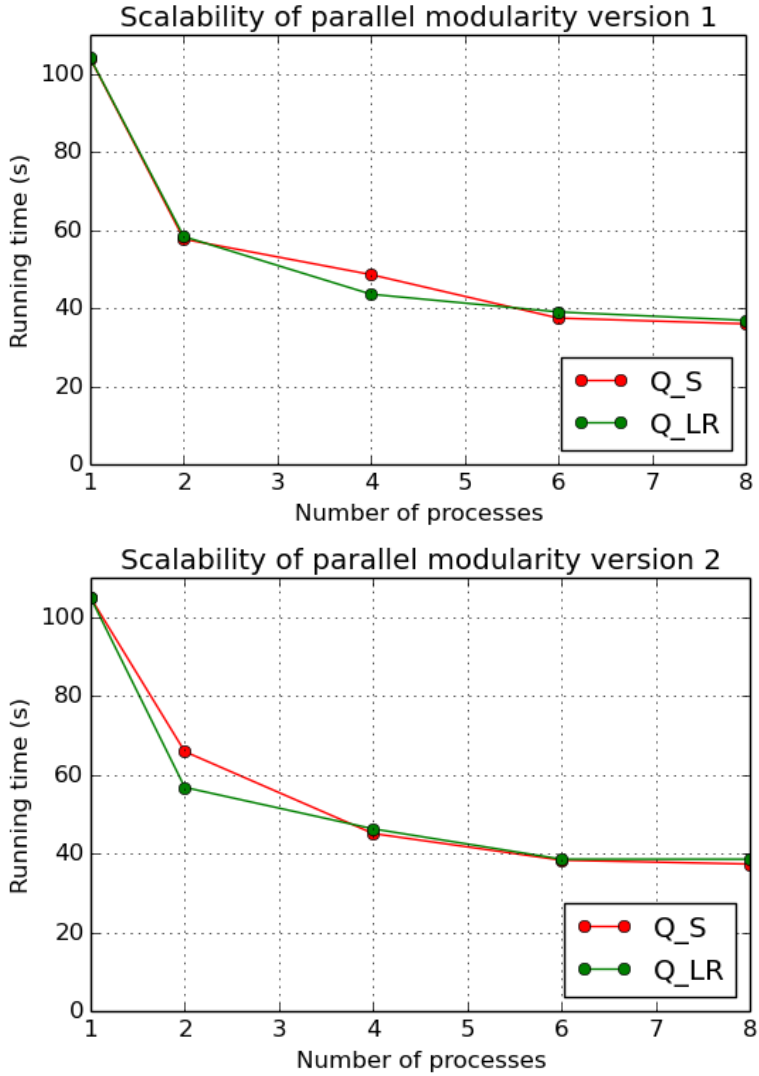


Fig. 2. Speedup of both parallel modularity versions on a directed unweighted graph with |V| = 72146, |E| = 79003, |C| = 1894, S = 5.30

References

[1]. Lawrence Page et al. “The PageRank citation ranking: bringing order to the web.” Technical Report. Stanford InfoLab. (1999).

[2]. Mark EJ Newman. “Analysis of weighted networks”. *Physical Review E* 70.5 (2004), p. 056131.

[3]. Mark EJ Newman, Michelle Girvan. “Finding and evaluating community structure in networks”. *Physical review E* 69.2 (2004), p. 026113.

[4]. Amy N Langville, Carl D Meyer. “A survey of eigenvector methods for web information retrieval”. *SIAM review* 47.1 (2005), pp. 135–161.

[5]. Jörg Reichardt, Stefan Bornholdt. “Statistical mechanics of community detection”. *Physical Review E* 74.1 (2006), p. 016110.

[6]. Alex Arenas et al. “Size reduction of complex networks preserving modularity”. *New Journal of Physics* 9.6 (2007), p. 176.

[7]. Elizabeth A Leicht, Mark EJ Newman. “Community structure in directed networks”. *Physical review letters* 100.11 (2008), p. 118703.

[8]. Tam´as Nepusz et al. “Fuzzy communities and the concept of bridgeness in complex networks”. *Physical Review E* 77.1 (2008), p. 016107.

[9]. Youngdo Kim, Seung-Woo Son, Hawoong Jeong. “Finding communities in directed networks”. *Physical Review E* 81.1 (2009), p. 016103.

[10]. Vincenzo Nicosia et al. “Extending the definition of modularity to directed graphs with overlapping communities”. *Journal of Statistical Mechanics: Theory and Experiment* 2009.03 (2009), p. 03024.

[11]. Aaron McDaid, Neil Hurley. “Detecting highly overlapping communities with model-based overlapping seed expansion”. *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE. 2010, pp. 112–119.

[12]. Yu-Teng Chang, Dimitrios Pantazis, Richard M Leahy. “Partitioning directed graphs based on modularity and information flow”. *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*. IEEE. 2011, pp. 1105–1108.

[13]. Steve Gregory. “Fuzzy overlapping communities in networks”. *Journal of Statistical Mechanics: Theory and Experiment* 2011.02 (2011), p. 02017.

[14]. Amy N Langville, Carl D Meyer. *Google’s PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011.

[15]. Jaewon Yang, Jure Leskovec. “Community-affiliation graph model for overlapping network community detection”. *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE. 2012, pp. 1170–1175.

[16]. Mingming Chen, Konstantin Kuzmin, Boleslaw K Szymanski. “Community detection via maximization of modularity and its variants”. *Computational Social Systems, IEEE Transactions on* 1.1 (2014), pp. 46– 65.

[17]. Mingming Chen, Konstantin Kuzmin, Boleslaw K Szymanski. “Extension of modularity density for overlapping community structure”. *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*. IEEE. 2014, pp. 856–863.

[18]. Nicolas Dugué, Anthony Perez. “Directed Louvain: maximizing modularity in directed networks”. PhD thesis. Universit´e d’Orléans, 2015.

[19]. Vincent A Traag, Rodrigo Aldecoa, J-C Delvenne. “Detecting communities using asymptotical surprise”. *Physical Review E* 92.2. APS, 2015, p. 022816.

Параллельное вычисление модулярности для направленных взвешенных графов с пересекающимися сообществами

[1]*Дробышевский Михаил <drobyshevsky@ispras.ru>*
[1]*Коршунов Антон <korshunov@ispras.ru>*
[1.2.3]*Турдаков Денис <turdakov@ispras.ru>*
[1]*Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2]*Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[3]*Национальный исследовательский университет «Высшая школа экономики»*
101000, Россия, Москва, ул. Мясницкая, д. 20

Аннотация. В статье представлены новые алгоритмы расчета модулярности для направленных взвешенных графов с пересекающимися сообществами. Рассматриваются несколько подходов для вычисления модулярности и их расширения. Учитывая вычислительную сложность известных подходов, предлагаются два параллельных расширения, масштабируемых на графы с более 10^4 вершин.

Ключевые слова: модулярность; поиск сообществ; пэйдж-ранк; линк-ранк; функция принадлежности; коэффициент принадлежности.

DOI: 10.15514/ISPRAS-2016-28(6)-11

Для цитирования: Дробышевский М., Коршунов А., Турдаков Д. Параллельное вычисление модулярности для направленных взвешенных графов с пересекающимися сообществами. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 153-170 (на английском). DOI: 10.15514/ISPRAS-2016-28(6)-11

Список литературы

[1]. Lawrence Page и др. “The PageRank citation ranking: bringing order to the web”. Technical Report. Stanford InfoLab. (1999).
[2]. Mark EJ Newman. “Analysis of weighted networks”. *Physical Review E* 70.5 (2004), стр. 056131.
[3]. Mark EJ Newman, Michelle Girvan. “Finding and evaluating community structure in networks”. *Physical review E* 69.2 (2004), стр. 026113.
[4]. Amy N Langville, Carl D Meyer. “A survey of eigenvector methods for web information retrieval”. *SIAM review* 47.1 (2005), стр. 135–161.
[5]. Jörg Reichardt, Stefan Bornholdt. “Statistical mechanics of community detection”. *Physical Review E* 74.1 (2006), стр. 016110.

[6]. Alex Arenas et al. “Size reduction of complex networks preserving modularity”. *New Journal of Physics* 9.6 (2007), стр. 176.

[7]. Elizabeth A Leicht, Mark EJ Newman. “Community structure in directed networks”. *Physical review letters* 100.11 (2008), стр. 118703.

[8]. Tamas Nepusz et al. “Fuzzy communities and the concept of bridgeness in complex networks”. *Physical Review E* 77.1 (2008), стр. 016107.

[9]. Youngdo Kim, Seung-Woo Son, Hawoong Jeong. “Finding communities in directed networks”. *Physical Review E* 81.1 (2009), стр. 016103.

[10]. Vincenzo Nicosia et al. “Extending the definition of modularity to directed graphs with overlapping communities”. *Journal of Statistical Mechanics: Theory and Experiment* 2009.03 (2009), стр. 03024.

[11]. Aaron McDaid, Neil Hurley. “Detecting highly overlapping communities with model-based overlapping seed expansion”. *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE. 2010, стр. 112–119.

[12]. Yu-Teng Chang, Dimitrios Pantazis, Richard M Leahy. “Partitioning directed graphs based on modularity and information flow”. *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*. IEEE. 2011, стр. 1105–1108.

[13]. Steve Gregory. “Fuzzy overlapping communities in networks”. *Journal of Statistical Mechanics: Theory and Experiment* 2011.02 (2011), стр. 02017.

[14]. Amy N Langville, Carl D Meyer. “Google’s PageRank and beyond: The science of search engine rankings”. Princeton University Press, 2011.

[15]. Jaewon Yang, Jure Leskovec. “Community-affiliation graph model for overlapping network community detection”. *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE. 2012, стр. 1170–1175.

[16]. Mingming Chen, Konstantin Kuzmin, Boleslaw K Szymanski. “Community detection via maximization of modularity and its variants”. *Computational Social Systems, IEEE Transactions on* 1.1 (2014), стр. 46– 65.

[17]. Mingming Chen, Konstantin Kuzmin, Boleslaw K Szymanski. “Extension of modularity density for overlapping community structure”. *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*. IEEE. 2014, стр. 856–863.

[18]. Nicolas Dugué, Anthony Perez. “Directed Louvain: maximizing modularity in directed networks”. PhD thesis. Universit´e d’Orléans, 2015.

[19]. Vincent A Traag, Rodrigo Aldecoa, J-C Delvenne. “Detecting communities using asymptotical surprise”. *Physical Review E* 92.2. APS, 2015, стр. 022816.

Метод автоматического определения возраста пользователей с помощью социальных связей

[1,2]А.Г. Гомзин <gomzin@ispras.ru>
[1,2,3]С.Д. Кузнецов <kuzloc@ispras.ru>
[1]Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, дом 25
[2]Московский государственный университет имени М.В. Ломоносова,
119991 ГСП-1 Москва, Ленинские горы, МГУ имени М.В. Ломоносова, 2-й учебный корпус, факультет ВМК
[3]Московский физико-технический институт,
141700, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. Работа посвящена методам определения возраста пользователей социальных сетей. Социальные сети предоставляют пользователям возможность заполнять свои профили, которые могут включать в себя возраст. В связи с тем, что профили заполняются не полностью, возникает задача предсказания неуказанных значений атрибутов пользователей. Явно указанные и предсказанные значения возраста используются в рекомендательных и маркетинговых системах, они позволяют фильтровать целевую аудиторию рекомендуемых товаров и услуг. Кроме того, предсказанные значения могут использоваться для более точного определения демографического профиля интернет-сообществ, целевую аудиторию рекламных кампаний в Интернете. В данной работе предлагается метод, предсказывающий незаполненное значение возраста пользователя. Метод использует следующую доступную информацию из социальной сети: явно указанные пользователями значения возраста и социальные связи. Метод основан на распространении меток по графу друзей и подписок пользователей на сообщества.

Ключевые слова: социальные сети; демографические атрибуты; векторная модель; социальный граф; распространение меток

DOI: 10.15514/ISPRAS-2016-28(6)-12

Для цитирования: Гомзин А.Г., Кузнецов С.Д. Метод автоматического определения возраста пользователей с помощью социальных связей. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 171-184. DOI: 10.15514/ISPRAS-2016-28(6)-12

1. Введение

Профили пользователей социальных сетей как правило включают в себя демографические атрибуты, такие как пол, возраст, семейное положение, уровень образования, религиозные, политические взгляды и т.д. Значения демографических атрибутов используются в рекомендательных и маркетинговых системах. Они позволяют фильтровать целевую аудиторию рекомендуемых товаров и услуг.

По тем или иным причинам далеко не все атрибуты заполняются пользователями. Кроме того, иногда пользователи оставляют в профиле ложные сведения. Данная работа посвящена методам, предсказывающим незаполненные значения возраста, с использованием социальных связей, явно указанным другими пользователями значениями возраста. В качестве исходных данных используются открытые данные социальной сети Вконтакте[1], такие как профили пользователей, друзья пользователей, подписки пользователей на сообщества. Методы поиска ложно указанных атрибутов в данной статье не рассматриваются.

Представленный в работе метод определения демографических атрибутов основан на использовании социального графа. Пользователи и сообщества – узлы данного графа, отношения дружбы между пользователями, подписок на сообщества – ребра. Под сообществом понимается специальная страница в социальной сети, объединяющая пользователей по интересам: пользователи подписываются на интересующие их сообщества для получения релевантной информации. Значения атрибутов предсказываются путем распространения меток в этом графе. Метки представляют собой значения возраста.

Вначале будут описаны существующие методы решения задачи определения демографических атрибутов и смежных задач. Затем описан предлагаемый метод. В конце статьи представлены результаты экспериментального исследования разработанного метода.

2. Краткий обзор существующих решений

В этом разделе представлен обзор решений задач определения демографических атрибутов пользователей Интернета.

Наибольший интерес для исследователей методов определения демографических атрибутов представляют социальные сети, такие как Facebook[2], Twitter[3] и другие. Кроме данных ресурсов, в некоторых исследованиях анализируются комментарии на Youtube [5], новости и электронные письма [2].

[1] https://vk.com
[2] https://www.facebook.com/
[3] https://twitter.com/

Наиболее распространенным подходом, используемым при решении задач определения демографических атрибутов, является извлечение признаков из текстов пользователей и применение к ним методов машинного обучения. Вначале будет описаны признаки, которые использовали авторы работ, затем перечислены используемые алгоритмы.

В работе [5] определяется пол пользователей Youtube. Сначала авторы используют метод распространения пола в графе пользователи-видео, где ребро между пользователем и видео означает факт просмотра видео пользователем. Затем в качестве признаков рассматриваются статистические признаки, такие как средняя длина комментария в символах/словах/предложениях, словесные n-граммы, возраст пользователя, а также распределение пола, полученное с помощью модели распространения атрибута «пол» в графе пользователи-видео. В работе [1] пол пользователей Twitter определяется по текстам их сообщений (твитов). Используются символьные и словесные n-граммы. В работе [8] рассматривается задача определения возраста пользователей, пишущих на голландском языке. Возраст пользователей разбивается на интервалы. В качестве признаков используются символьные и словесные 1,2 и 3-граммы. Помимо решений, в которых множество значений возраста пользователей разбивается на несколько интервалов, существуют методы, которые предсказывают числовое значение возраста [7]. Авторы [3] определяют политические предпочтения пользователей социальной сети Twitter. Рассматриваются три класса: демократы, республиканцы, неявная политическая позиция. В качестве признаков используются словесные юниграммы, хэштеги, сообщества пользователей (полученные с помощью алгоритма, основанного на распространении меток в социальном графе пользователей).

Одним из самых простых используемых алгоритмов является Наивный байесовский классификатор. Этот метод используется в работе [1]. При классификации на два класса часто используется линейный классификатор. Одним из популярных алгоритмов обучения линейного классификатора является метод опорных векторов. Его используют авторы работ [2], [1], [8], [7]. В работах также встречаются такие алгоритмы, как решающие деревья и логистическая регрессия [2]. Для определения числового значения возраста используется линейная регрессия [7].

Более полный обзор методов определения демографических атрибутов пользователей по текстам их сообщений представлен в работе [10].

Помимо текстовой информации в качестве исходных данных для определения демографических атрибутов используются социальные связи. В работе [6] анализируется университетская социальная сеть. Атрибуты определяются с использованием алгоритма кластеризации социального графа методом распространения меток. В работе [4] в качестве исходных данных рассматривается мобильная социальная сеть, в которой связи между

пользователями составляются на основе звонков и коротких сообщений между ними.

Примером использования одновременно двух видов данных (текстов пользователей и социальных связей) является работа [9]. Авторы определяют тональность сообщений пользователей Twitter. При этом строится граф, в котором присутствуют пользователи, сообщения, слова, эмотиконы, используется метод распространения меток в этом графе.

В этой работе будет предложен метод, определяющий возраст пользователей путем распространения меток в графе, включающем в себя пользователей, сообщества и связи между ними.

3. Описание предлагаемого метода

Для работы метода определения демографических атрибутов необходимы следующие данные:

- Профили пользователей, содержащие значения демографических атрибутов
- Социальные связи (достаточно одного из нижеперечисленных видов данных):
    - Списки друзей пользователей
    - Списки подписчиков сообществ

Сначала из профилей пользователей извлекаются значения возраста. Назовем данный процесс разметкой.

Затем неуказанные значения возраста определяются для всех пользователей с использованием социальных связей.

В данном разделе статьи сначала идет описание краулера, т.е. сборщика данных, затем описание разметки атрибутов, затем алгоритм определения неуказанных значений атрибутов.

3.1 Сбор данных

Сбор данных осуществлялся из социальной сети ВКонтакте.

Для сбора использовались методы VK API для разработчиков приложений (https://vk.com/dev/methods).

Сбор информации затрагивает всех пользователей, но не все сообщества. При сборе профилей пользователей, а также скачивании графа дружбы, краулер предварительно получает список идентификаторов всех пользователей из каталога (https://vk.com/catalog.php). Скачивание графа подписок на сообщества осуществляется для 1 миллиона «наиболее активных» сообществ VK. Список таких сообществ был составлен предварительно (до начала сбора данных) путем ранжирования всех доступных на тот момент сообществ по дате наиболее позднего публичного сообщения.

Для сбора **профилей пользователей** используются методы API users.get и groups.getById. Методы принимают на вход списки идентификаторов пользователей или сообществ и возвращают списки их профилей в формате JSON. За один запрос к каждому из методов скачивается 200 профилей.

Для сбора **графов дружбы и подписки** используются методы API friends.get и groups.getMembers. Методы принимают идентификатор одного пользователя или сообщества и возвращают списки идентификаторов его друзей или подписчиков.

Все используемые методы сбора данных используют версию API 5.52.

Реализация краулера данных выполнена на основе фреймворка MODIS Crawler. Данный фреймворк позволяет параллельно осуществлять множество запросов к методам VK API.

3.2 Разметка возраста

Алгоритм определения возраста пользователей использует указанные другими пользователями значения возраста.

Возраст пользователя извлекается из даты его рождения, указанной в профиле. Поле «дата рождения» может быть представлено в трех вариантах:

1. DD-MM-YYYY - доступна полная дата
2. YYYY - доступен год рождения
3. DD-MM - доступна дата без года

Здесь YYYY - год, MM - месяц, DD - день месяца.

В первых двух вариантах известен год рождения. Возраст определяется как:

$$Y_c - Y_u \tag{1}$$

Здесь Y_u - указанный в профиле год, Y_c - текущий год.

3.3 Определение возраста

Система определения возраста авторов определяет неуказанные значения атрибутов пользователей на основе информации о размеченных значениях и социальных связей (граф друзей, граф подписок на сообщества).

Социальный граф состоит из узлов и связей между ними. Узлы бывают двух видов: пользователи и сообщества. Граф включает в себя следующие связи между узлами:

- граф друзей: связи между пользователями (отношение дружбы).
- граф подписчиков: связи между пользователем и сообществом (подписка пользователя на сообщество).

Каждому узлу (пользователю или сообществу) в графе ставится в соответствие набор меток. Каждая метка соответствует определенному значению атрибута (например, «возраст=23»).

Схема алгоритма:

1. Инициализация
2. Построение векторной модели
3. Вычисление весов пользователей и сообществ, распространение меток на узлы-сообщества
4. Построение векторной модели с учетом весов
5. Распространение меток на узлы-пользователей с учетом весов

На этапе инициализации узлы-пользователи получают метки. Далее метки распространяются на узлы-сообщества. Затем метки узлов пользователей и узлов-сообществ распространяются на узлы-пользователей, у которых отсутствует метка (не указан явно возраст).

Распространение меток представляет собой вычисление метки узла на основе меток его соседей в социальном графе. Алгоритм вычисления метки представлен далее.

Инициализация

Изначально узлы-пользователи инициализируются метками в соответствии с разметкой.

Построение векторной модели

Для каждого пользователя строится распределение значений атрибута среди его соседей. На рисунке 1 приведен пример. Затем все распределения группируются по значению атрибута пользователя. После этого для каждого значения возраста вычисляется среднее распределение значений возраста соседей. В итоге получается так называемая векторная модель для возраста. Например, на рисунке 2 изображена векторная модель, в которой для каждого значения возраста задано распределение возрастов соседей.

Обозначим эту модель $Model_{avg}$.

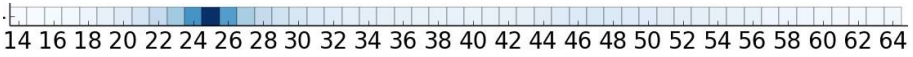


Рис. 1. Распределение значений возраста соседей пользователя. Вероятность обозначена интенсивностью цвета (чем темней, тем больше).

Fig. 1. The distribution of neighbors' ages for the user. The probability is indicated by color strength (the darker - the more).

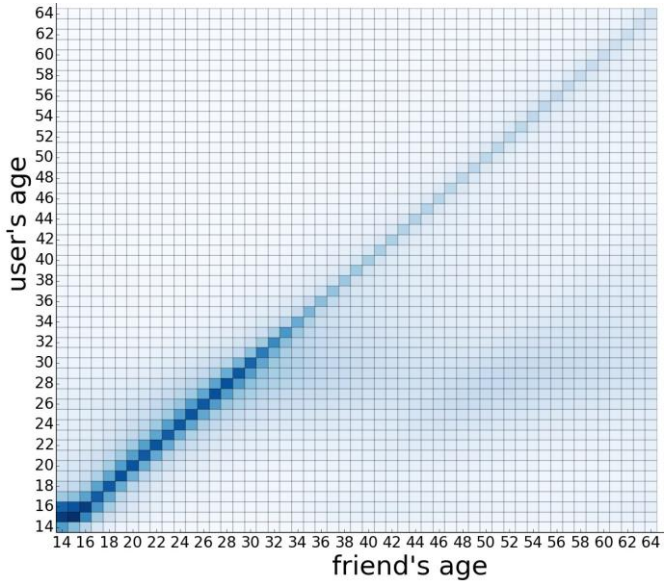


Рис. 2: Векторная модель для атрибута «возраст». В каждой строке – усредненные распределения возраста соседей.

Fig. 2. Vector model for the "age" attribute. Each line is average distribution of neighbors' ages.

Данная векторная модель, в которой производится усреднение распределений по всем данным, применяется при распространении меток в сторону пользователей. Для распространения меток в сторону узлов-сообществ используется модель, в которой распределение значений атрибутов соседей для каждого значения атрибута имеет следующий вид:

$$p(val_n|val_c) = \begin{cases} 1 & \text{если } val_n = val_c \\ 0 & \text{если } val_n \neq val_c \end{cases} \quad (2)$$

Здесь $p(val_n|val_c)$ – вероятность того, что значения атрибута соседа равно val_n, при условии что свое значение атрибута равно val_c.

Обозначим эту модель $Model_{max}$.

Векторные модели используются для оценки близости распределения соседей узла, для которого вычисляется метка, к соответствующему распределению из модели. При использовании модели $Model_{max}$ максимальная близость достигается, когда все метки соседей узла принимают одинаковое значение.

Вычисление весов сообществ, распространение меток на сообщества

На этом этапе алгоритма моделируется распространение меток по социальным связям. При этом для каждого сообщества на основе значений атрибута соседей вычисляются метка (значение атрибута) и вес.

Вес – вещественное число, определяющее, насколько метка (значение атрибута) данного пользователя или сообщества соответствует векторной модели. Его можно интерпретировать как уверенность алгоритма в своем решении.

При определении метки (значения атрибута) узла строится распределение значений данного атрибута у соседей *Distr* (см. рисунок 1), затем для каждого значения атрибута вычисляется близость данного распределения к соответствующему распределению из векторной модели. В качестве меры близости используется косинусная мера. Таким образом, значение метки вычисляется по формуле:

$$L = \arg\max_{val}(sim(Model_*(val), Distr)) \quad (3)$$

$$S = \max_{val}(sim(Model_*(val), Distr)) = sim(Model_*(L), Distr) \quad (4)$$

где:

$$sim(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} \quad (5)$$

Здесь L — значение атрибута (метка), S — близость, соответствующая максимально близкому значению атрибута *val*, *Distr* — распределение значений атрибута соседей, $Model_*(val)$ — распределение из модели для значения атрибута *val* (для пользователей используется модель $Model_{avg}$, для сообществ – $Model_{max}$).

Значения L и S вычисляются для узлов, у которых множество узлов-соседей с указанным значением атрибута непустое.

Значения S используются для определения весов узлов $W(node)$. Веса вычисляются отдельно для каждого типа узла. Вычисленные значения S сортируются по возрастанию и помещаются в массив. Затем вес узла определяется по формуле:

$$W(node) = \left(\frac{pos(S_{node})}{N}\right)^2 \quad (6)$$

Здесь $pos(S_{node})$ – порядковый номер значения S (от 1 до N) в отсортированном массиве, N – количество узлов, для которых вычислено значение S.

Для узлов, у которых не вычислено значение S вес равен 0.

Построение векторной модели и распространение меток на пользователей с учетом весов

Для каждого типа соседей (пользователи, сообщества) вручную задается вес, задающий вклад каждого источника данных (W_{User}, W_{Comm}, соответственно). Данные веса подбираются при тестировании для каждого атрибута.

После того, как метки распространены на сообщества и определены веса сообществ, запускается распространение меток для пользователей с учетом вычисленных на предыдущем шаге весов $W(node)$ (т.е. вклад каждого сообщества-соседа пользователя в распределение $Distr$ равен его весу). Для каждого типа соседей (пользователь, сообщество) отдельно вычисляется распределение значений атрибута соседей данного типа. Затем это распределение домножается на соответствующий вес (W_{User}, W_{Comm}). Полученная сумма распределений нормализуется.

При распространении меток на данном этапе алгоритма используется модель $Model_{avg}$. После распространения меток незаполненные атрибуты заполняются в соответствие с распространенными метками.

4. Тестирование

Для оценки качества определения демографических атрибутов используется кросс-валидация с разбиением данных на 10 частей. Кросс-валидация запускается при различных значениях параметров W_{User}, W_{Comm}. Каждый из этих параметров при тестировании принимает значения 1, 10 или 100.

В данном разделе сначала описывается выборка, затем метрики качества, затем результаты.

4.1 Выборка

Сначала среди всех сообществ выбираются сообщества, у которых имеется хотя бы K подписчиков с явно указанным значением возраста.

Затем в выборку попадают пользователи, у которых:

- имеется хотя бы K социальных связей: друзей с явно указанным значением атрибута, отобранных сообществ и
- размечено значение возраста

В проведенных экспериментах $K = 10$.

Количество пользователей в выборке: 28940134

4.2 Метрики

Для атрибута *возраст* определяется точность. С увеличением возраста пользователя абсолютная ошибка в предсказании его возраста становится менее критичной, поэтому при оценке точности используется величина относительной ошибки. Считается, что значение *возраста* предсказано верно, если:

$$|age_u - age_p| \leq 0,15 \cdot age_u$$

Здесь age_u – значение *возраста* пользователя из разметки, age_p – предсказанное значение *возраста*.

Для атрибута возраст также вычисляется средняя абсолютная ошибка (MAE):

$$\frac{\sum |age_u - age_p|}{N}$$

Здесь N – количество предсказанных значений.

4.3 Результаты

Тестирование проводилось при различных значениях параметров: W_{User}, W_{Comm}. Рассматриваются конфигурации данных параметров, когда они равны и когда один из них преобладает на порядок. В таблице 1 представлены значения точности и средней абсолютной ошибки.

Таблица 1. Результаты тестирования.

Table 1. Test results.

Значения весов	Метрика	Значение
$W_{User} = 1$, $W_{Comm} = 1$	точность	81,3 %
	MAE	2,79 года
$W_{User} = 1$, $W_{Comm} = 10$	точность	77,6 %
	MAE	3,28 года
$W_{User} = 10$, $W_{Comm} = 1$	точность	81,1 %
	MAE	2,81 года

Из полученной таблицы видно, что наибольший вклад в качество определения возраста приносит граф друзей.

5. Выводы

В данной работе рассматривается задача определения возраста пользователей социальных сетей. Предложен метод, позволяющий определять значения возраста пользователей, у которых имеется хотя бы один из видов данных: список друзей, список подписок на сообщества. Алгоритм основан на распространения меток в социальном графе. Пользователи и сообщества – узлы данного графа, отношения дружбы между пользователями, подписок на сообщества – ребра. Метки представляют собой значения возраста.

В результате тестирования были достигнуты приемлемые показатели качества определения возраста. В дальнейшем планируется применить данный метод к другим атрибутам пользователей, использовать генерируемый текстовый контент и зависимости между значениями различными атрибутами (например, между возрастом и уровнем образования).

Список литературы

[1]. John D Burger, John Henderson, George Kim, and Guido Zarrella. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309. Association for Computational Linguistics, 2011.

[2]. Na Cheng, Rajarathnam Chandramouli, and KP Subbalakshmi. Author gender identification from text. *Digital Investigation*, 8(1):78–88, 2011.

[3]. Michael D Conover, Bruno Gon¸calves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. Predicting the political alignment of twitter users. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 192–199. IEEE, 2011.

[4]. Yuxiao Dong, Yang Yang, Jie Tang, Yang Yang, and Nitesh V Chawla. Inferring user demographics and social strategies in mobile social networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 15–24. ACM, 2014.

[5]. Katja Filippova. User demographics and language in an implicit social network. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1478–1488. Association for Computational Linguistics, 2012.

[6]. Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260. ACM, 2010.

[7]. Dong Nguyen, Noah A Smith, and Carolyn P Ros´e. Author age prediction from text using linear regression. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 115–123. Association for Computational Linguistics, 2011.

[8]. Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. Predicting age and gender in online social networks. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pages 37–44. ACM, 2011.

[9]. Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 53–63. Association for Computational Linguistics, 2011.

[10]. Гомзин А.Г., Кузнецов С.Д. Методы построения социо-демографических профилей пользователей сети Интернет. *Труды ИСП РАН*, том 27, вып. 4, 2015, стр. 129-144. DOI: 10.15514/ISPRAS-2015-27(4)-7

A method of automatically estimating user age using social connections

[1,2]*A.G. Gomzin <gomzin@ispras.ru>*
[1,2,3]*S.D. Kuznetsov <kuzloc@ispras.ru>*
[1] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2]*Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*
[3]*Moscow Institute of Physics and Technology (State University) 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia*

Abstract. The work is devoted to methods of social network users' age detection. Social networks allow users to fill their profiles that may contain an age. Profiles are not fully filled, so the task of unknown attributes detection arises. Explicit and predicted values are used in recommender and marketing systems. Moreover, the predicted values can be used for determining online communities' demographic profiles and for inferring the target audience of marketing campaigns in the Internet. In this paper a method for predicting unfilled age values is proposed. The method uses the following information available from the social network: explicit users' ages and social graph. The graph contains nodes representing users and communities. Community is the special page in the Internet that unites users on interests. Friendship relations between users and subscriptions of users on communities represented as edges of the social graph. The method is based on the label propagation in the friendship and subscription graphs. Ages of the users are representd by labels that are propagated in the graph. The scheme of the algorithm is following: initialize user labels according to explicit profiles; build vector model that contains distributions of the neighbours' ages grouped by user age; compute weights of users and communities, propagate labels to communities; build vector model considering calculated weights; propagate labels to users that have not filled their age in the profile. The paper describes the algorithm and contains experimantal results showing that friendship relations are more useful for age prediction in the social network than communities.

Keywords: social media; demographic attributes; vector model; social graph; label propagation

DOI: 10.15514/ISPRAS-2016-28(6)-12

For citation: Gomzin A., Kuznetsov S. A method of automatically estimating user age using social connections. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 171-184 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-12

References

[1]. John D Burger, John Henderson, George Kim, and Guido Zarrella. Discriminating gender on twitter. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1301–1309. Association for Computational Linguistics, 2011.

[2]. Na Cheng, Rajarathnam Chandramouli, and KP Subbalakshmi. Author gender identification from text. Digital Investigation, 8(1):78–88, 2011.

[3]. Michael D Conover, Bruno Gon¸calves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. Predicting the political alignment of twitter users. In Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, pages 192–199. IEEE, 2011.

[4]. Yuxiao Dong, Yang Yang, Jie Tang, Yang Yang, and Nitesh V Chawla. Inferring user demographics and social strategies in mobile social networks. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 15–24. ACM, 2014.

[5]. Katja Filippova. User demographics and language in an implicit social network. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1478–1488. Association for Computational Linguistics, 2012.

[6]. Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In Proceedings of the third ACM international conference on Web search and data mining, pages 251–260. ACM, 2010.

[7]. Dong Nguyen, Noah A Smith, and Carolyn P Ros´e. Author age prediction from text using linear regression. In Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities, pages 115–123. Association for Computational Linguistics, 2011.

[8]. Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. Predicting age and gender in online social networks. In Proceedings of the 3rd international workshop on Search and mining user-generated contents, pages 37–44. ACM, 2011.

[9]. Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In Proceedings of the First workshop on Unsupervised Learning in NLP, pages 53–63. Association for Computational Linguistics, 2011.

[10]. Gomzin, A.G., Kuznetsov, S.D. Methods for Construction of Socio-Demographic Profile of Internet Users. Trudy ISP RAN/Proc. ISP RAS, vol 27, issue 4, 2015, pp. 129-144 (in Russian). DOI: 10.15514/ISPRAS-2015-27(4)-7.

Подходы к определению основного места проживания пользователей социальных сетей на основе социального графа

[1]Ю.С. Трофимович <integral@ispras.ru>
[2]И.С. Козлов <kozlov-ilya@ispras.ru>
[1,2,3]Д.Ю. Турдаков <turdakov@ispras.ru>
[1]Институт системного программирования РАН
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2]Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1
[3]Национальный исследовательский университет «Высшая школа экономики»
101000, Россия, Москва, ул. Мясницкая, д. 20

Аннотация. В статье рассматриваются подходы к определению основного места проживания пользователей социальных сетей по графу образуемому в результате установления двунаправленной связи — “дружбы”. Предложен подход, базирующийся на векторном представлении вершин графа и последующем применении алгоритма классификации на основе обучения с учителем. Приведены результаты экспериментов и сравнение с референсными подходами. Показано, что предложенный подход сопоставим по качеству с другими подходами.

Ключевые слова. определение местоположения; социальные сети; социальный граф; векторное представление вершин графа.

DOI: 10.15514/ISPRAS-2016-28(6)-13

Для цитирования: Трофимович Ю.С., Козлов И.С., Турдаков Д.Ю. Подходы к определению основного места проживания пользователей социальных сетей на основе социального графа. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 185-196. DOI: 10.15514/ISPRAS-2016-28(6)-13

1. Введение

Современные социальные сети являются не только средством общения, но так же инструментом маркетинга, рекламы и социальных исследований. Так, в статье Sakaki, Okazaki и Matsuo [1] описывается система быстрого обнаружения и оповещения о землятресениях основанная на анализе

сообщений “Твиттера” в реальном времени. В статьях Lamb, Paul и Dredze [2] и Sadilek, Kautz и Silenzio [3] используются сообщения той же социальной сети для обнаружения эпидемий гриппа и моделирования его распространения.

Для многих из этих приложений необходима информация о местоположении пользователей, которая не всегда доступна. Среди пользователей “Твиттер” лишь около 26%[4] указывают в профиле название города, около 30% пользователей “ВКонтакте” оставляют это поле пустым. Так появляется задача определения места проживания пользователей социальных сетей.

Подходы к решению задачи удобно классифицировать по информации о пользователях, на которую эти подходы опираются. Большинство подходов сконцентрированы на анализе 1) социального графа 2) пользовательского контента (фотографии, текст сообщений), либо 3) используют гибридный подход, комбинируя первые два [5].

В то время как подходы, опирающиеся на анализ контента довольно широко используют методы машинного обучения, среди подходов основанных на анализе социального графа машинное обучение не столь популярно.

Одна из сложностей, препятствующих этому — необходимость конструирования признаков на основе социального графа. Как правило методы машинного обучения с учителем принимают в качестве признаков числовой вектор конечной размерности и не могут работать с графом непосредственно. Для извлечения признаков из графа были разработаны методы, сопоставляющие вершинам графа вектора конечной размерности.

В данной статье мы приводим краткий обзор работ в области определения местоположения пользователей социальных сетей и предлагаем подход, основанный на векторном представлении вершин графа и методах машинного обучения с учителем. Стоит отметить, что предложенный подход не имеет какой-либо специфики, связанной с местоположением. Он может так же использоваться для оценки любых других демографических атрибутов, оказывающих влияние на формирование связей в обществе. Так, схожий подход использовался Perozzi и Skiena [6] для определения возраста пользователей.

Также мы приводим результаты экспериментов и сравнения с выбранными референсными подходами.

2. Обзор существующих работ

2.1 Определение места проживания по социальному графу

Начало исследованиям в области определения местоположения пользователей социальных сетей по социальному графу положила работа Backstrom, Sun и Marlow [7]. В ней авторы исследуют уже отмеченное ранее влияние расстояний на структуру социальных связей и предлагают вероятностную

модель для получения предполагаемого места проживания пользователей социальной сети “Facebook”.

В работе Jurgens [8] было предложено использовать метод распространения меток, суть которого заключается в итеративном выполнении следующих шагов:

1. определение множества пользователей, чье место проживания неизвестно;
2. выполнение метода select для пользователей, определенных на этапе 1;
3. добавление в список пользователей с известным местоположением тех пользователей, чье местоположение было определено на этапе 2.

Метод select() возвращает предполагаемое место проживания для данного пользователя основываясь на информации о его друзьях. В работе [8] рассмотрено несколько реализаций этого метода, за основной референсный подход мы взяли незначительно модифицированную реализацию под названием “Traditional Label Propagation” — ее возвращаемым значением является самое часто встречаемое место проживание среди друзей данного пользователя.

Методы машинного обучения активно используются для определения местоположения по тексту сообщений, но не нашли широкого применения в анализе социальных графов.

Jia и др. [9] в своей работе отмечают что эффективность подхода основанного на анализе социального графа во многом зависит от подмножества друзей, выбранного для анализа и предлагают использовать Random Walk with Restart (RWR) для сортировки друзей по степени “влияния”. Там же приводятся теоретические оценки максимальной точности подходов на основе анализа социального графа и их сравнение с имеющимися подходами.

Chen, Liu и Zou [10] так же используют RWR для оценки “похожести” пользователей основываясь на социальном графе, и предлагают графовую модель — Social Tie Factor Graph (STFG).

2.2 Векторное представление вершин графа

2.2.1 PCA на графах

Ранние работы по представлению вершин графа основывались на разложении матрицы Кирхгофа соответствующего графа [11]. Такой подход получил название PCA на графах. Однако, было показано что проекции не всегда подходят для использования в качестве векторного представления — в то время как проекции сильно различающихся вершин находятся далеко друг от друга, проекции похожих вершин оказываются слишком близко [12].

2.2.2 DeepWalk

Perozzi, Al-Rfou и Skiena [13] предложили модель DeepWalk. DeepWalk использует случайные блуждания для получения векторного представления вершин графа. Работа состоит из двух этапов:

1. Случайные блуждания по графу порождают последовательности вершин.
2. На основе полученных последовательностей вершин получают векторное представление вершин. Для этого используют неглубокую нейронную сеть, по аналогии с моделью word2vec [14].

2.2.3 BLM

В другом методе, предложенном в работе [12], получение векторного представления вершин графа базируется на вероятностной модели рёбер. В данном подходе каждая вершина представлена двумя векторами, $\vec{In} \in \mathbb{R}^D$ и $\vec{Out} \in \mathbb{R}^D$ (где D — размерность векторного пространства), которые являются векторным представлением входящих и исходящих рёбер этой вершины соответственно. Вероятность ребра между вершинами $v \to u$ вычисляется по формуле

$$p(v|u) = \frac{\exp(In_u^T Out_v)}{\sum_{w \in V} \exp(In_u^T Out_w)}$$

Векторное представление может быть получено за счёт максимизации вероятностей (принцип максимума правдоподобия), однако непосредственное получение векторного представления затруднено необходимостью вычислять сумму $\sum_{w \in V} \exp(In_u^T Out_w)$. Вместо этого авторы работы используют метод контрастной оптимизации [15].

3. Референсные подходы

3.1 “Случайный друг”

Местоположение каждого пользователя считается равным местоположению одного случайно выбранного друга.

3.2 Подход на основе распространения меток

Подход основан на работе Jurgens [8], используемая модификация описана в Алгоритме 1. Пусть: U — множество пользователей социальной сети, N — отображение пользователей в списки друзей, M — отображение пользователей в местоположения, k — количество друзей, при котором определение местоположения по друзьям считается возможным. Метод select() возвращает самую часто встречающуюся метку. Алгоритм возвращает в качестве результата своей работы E — итоговое отображение пользователей в местоположения. Основным отличием от подхода, описанного в разделе 2, является параметр k. В экспериментах k задавалось равным 5.

```
Data: U, N, M, k
Result: E
E = M;
E' = ∅;
while E ≠ E' do
    E' = E;
    for u ∈ U - domain(E) do
        L := пустое множество;
        for n ∈ N(u) do
            if n ∈ domain(E) then
                добавить E(n) в L;
            end
        end
        if |L| > k then
            E(u) = select(L);
        end
    end
end
```

Алгоритм 1. Алгоритм определения места проживания на основе распространения меток.

Algorithm 1. Algorithm for inferring location of residency based on label propagation

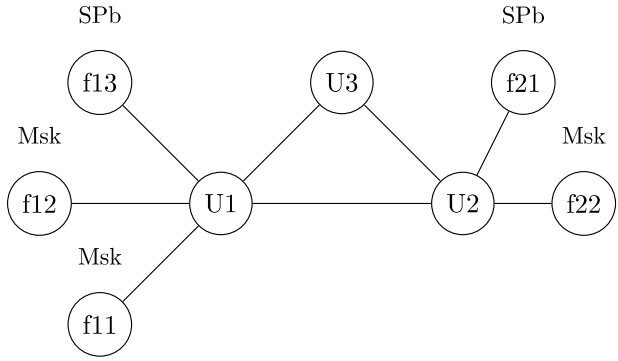


Рис. 1. Пример социального графа

Fig. 1. Example of a social graph

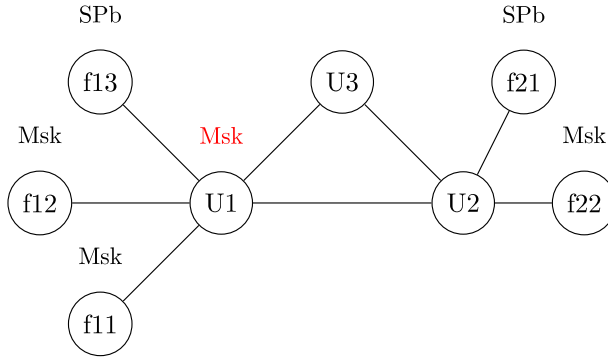


Рис. 2. Социальный граф после первой итерации подхода на основе распространения меток

Fig. 2. Social graph after first iteration of the approach based on label propagation

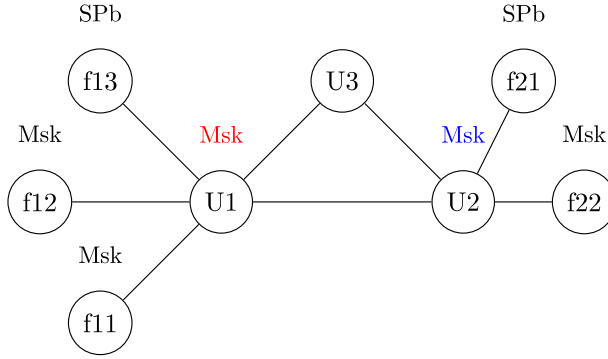


Рис. 3: Социальный граф после второй итерации подхода на основе распространения меток

Fig. 3. Social graph after second iteration of the approach based on label propagation

Небольшая иллюстрация приведена на рисунках 1-3. Для простоты предлагается взять k равным 3. На рисунке 1 приведен пример социального графа. Для пяти пользователей место проживания считается известным, для трех (U1, U2 и U3) — неизвестным. На каждой итерации выбираются пользователи, у которых минимум k друзей с известным местом проживания. При k равным 3 для данного социального графа на первой итерации будет отобран только пользователь U1. В качестве места проживания будет взято наиболее популярное местоположение среди друзей — для U1 это Msk 2. После этого начнется вторая итерация. Если считать место проживания U1 известным, становится возможным предсказать место проживания U2 — у него получается ровно три друга с известным местом проживания. После присвоения U2 Msk в качестве места проживания вторая итерация

заканчивается 3. На момент начала третьей итерации только U3 остается с неизвестным местоположением, но у него меньше k друзей с известным местоположением, поэтому в ходе третьей итерации граф остается неизменным и работа алгоритма заканчивается.

4. Подход на основе векторного представления вершин графа

Предлагаемый нами подход для определения места постоянного проживания пользователя состоит из следующих этапов:

1. Построение векторного представления вершин социального графа.
2. Обучение классификатора с использованием полученных на первом этапе векторов в качестве признаков.

В качестве метода получения векторного представления вершин был использован метод BLM, описанный в разделе 2.2.3. В социальной сети “Вконтакте” преобладают ненаправленные рёбра(связь типа дружба). По этой причине, а также в целях экономии памяти было принято $\vec{In} \equiv \vec{Out}$.

Данный подход отличает автоматическое извлечение векторов признаков из графа, что позволяет отойти от ручного конструирования признаков. После получения векторного представления вершин задача сводится к стандартной задаче машинного обучения — классификации. Каждый пользователь представлен конечномерным числовым вектором(вектор признаков), вектору признаков необходимо сопоставить один из конечного числа классов.

Нами были испробованы различные методы клас- сификации: логистическая регрессия, случайный лес, XGboost, а также многослойные нейронные сети. Результаты работы системы были оценены с помощью кросс-валидации. Наилучшие результаты показала многослойная нейронная сеть. Нами была использована библиотека Keras [16].

5. Эксперименты

5.1 Данные

Все эксперименты проводились на данных социальной сети “ВКонтакте”. За “золотой стандарт” бралось местоположение указанное самими пользователями в профиле. Данные о местоположении пользователя, предоставляемые сетью ВКонтакте, содержат в себе:

1. идентификационный номер пользователя;
2. идентификационный номер местоположения;
3. строковое описание местоположения (например, название города).

В качестве единиц местоположения для РФ были выбраны субъекты РФ, для зарубежных пользователей — страны. Каждой имеющейся паре {Идентификационный номер местоположения, Строковое описание

местоположения} был сопоставлен регион РФ или страна. Для получения региона РФ по строковому описанию местоположения использовался сервис Яндекса “Геокодер”, для получения названия страны по строковому описанию местоположения использовался сервис openstreetmap.org. Всего данные состояли из 156057700 профилей пользователей. Место проживания было известно для 99055808 из них.

5.2 Метрики

Для оценки подходов были выбраны следующие метрики:

1. доля верно определенных регионов (accuracy);
2. F1-мера с усреднением “Macro” — невзвешенное среднее значение F1-мер;
3. F1-мера с усреднением “Weighted” — взвешенное пропорционально присутствию в датасете среднее значение F1-мер;
4. точность с усреднением “Macro” — невзвешенное среднее значение точностей;
5. точность с усреднением “Weighted” — взвешенное пропорционально присутствию в датасете среднее значение точностей;
6. полнота с усреднением “Macro” — невзвешнное среднее значение полноты;
7. полнота с усреднением “Weighted” — взвешенное пропорционально присутствию в датасете среднее значение полноты.

5.3 Результаты

Результаты тестирования приведены в таблицах I и II. В таблицах используются следующие обозначения: RN — подход “Случайный друг”, LP — подход на основе распространения меток, GE — подход на основе векторного представления. В таблице 1 приведены значения метрик для всех предсказанных пользователей, вне зависимости от их активности в социальной сети. В таблице 2 приведены значения метрик только для пользователей, оставивших хотя бы один комментарий.

Для подхода на основе распространения меток использовалось сокрытие 85% известных меток. То есть местоположение 15% (15003815) выбранных случайным образом пользователей считалось известным, остальные же 85% требовалось предсказать.

Определенный интерес представляют метрики замеренные с макроусреднением, и большая разница в значениях одних и тех же метрик с усреднением “Macro” и “Weighted”. Одна из особенностей используемого датасета в неравномерности представленных классов — регионов. Редкие и слабо представленные регионы сложны для предсказания, и при неверных ответах классификатора существенно занижают значения метрик с макроусреднением. “Weighted”-метрики напротив, используют долю

пользователей из определенного региона в качестве веса этого региона, и больше подходят для оценки подходов. Небольшое отставание подхода на основе векторного представления от подхода с распространением меток по взвешенной точности, и значительно более высокие значения полноты и F1-меры позволяют заключить что предложенный подход на основе векторного представления может быть эффективно использован для определения места проживания пользователя как сам по себе, так и в комбинации с классификаторами на основе пользовательских данных.

6. Заключение

В данной статье мы привели краткий обзор работ в области определения места проживания пользователей социальных сетей, в том числе с использованием машинного обучения. Так же нами был предложен подход на основе векторного представления вершин графа и нейронной сети и приведены результаты экспериментов, показывающие применимость предложенного подхода.

Табл. 1. Результаты тестирования: все пользователи (вне зависимости от наличия комментариев)

Обозначения: RN — подход "Случайный друг", LP — подход на основе распространения меток, GE — подход на основе векторного представления

Table 1. Test results: all users (independent of comment presence)

Notation: RN — "Random friend" approach, LP — approach based on label propagation, GE — approach based on vector representation

Метрика		Метод		
		RN	LP	GE
Accuracy		0.417	0.463	**0.516**
Macro	Precision	0.357	**0.522**	0.308
	Recall	0.180	0.150	**0.167**
	F1-score	0.231	0.192	**0.203**
Weighted	Precision	0.654	**0.760**	0.729
	Recall	0.417	0.463	**0.516**
	F1-score	0.506	0.555	**0.595**

Табл. 2. Результаты тестирования: только пользовати оставившие комментарии

Table 2. Test results: only users who have posted comments

Метрика		Метод		
		RN	LP	GE
Accuracy		0.644	0.786	**0.839**
Macro	Precision	0.245	**0.374**	0.302
	Recall	0.225	0.241	**0.268**
	F1-score	0.234	0.255	**0.271**
Weighted	Precision	0.671	0.819	**0.823**
	Recall	0.644	0.786	**0.840**
	F1-score	0.657	0.788	**0.827**

Список литературы

[1]. Takeshi Sakaki, Makoto Okazaki, Yutaka Matsuo. “Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development”. *IEEE Trans. on Knowl. and Data Eng.* 25.4 (Apr. 2013), pp. 919–931.

[2]. Alex Lamb, Michael J Paul, Mark Dredze. “Separating Fact from Fear: Tracking Flu Infections on Twitter.” Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Atlanta. June 2013.

[3]. Adam Sadilek, Henry A Kautz, Vincent Silenzio. “Modeling Spread of Disease from Social Interactions.” Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012.

[4]. Zhiyuan Cheng, James Caverlee, Kyumin Lee. “You Are Where You Tweet: A Content-based Approach to Geo-locating Twitter Users”. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. CIKM ’10. Toronto, ON, Canada: ACM, 2010, pp. 759–768.

[5]. Afshin Rahimi et al. “Exploiting Text and Network Context for Geolocation of Social Media Users”. *CoRR* abs/1506.04803 (2015).

[6]. Bryan Perozzi, Steven Skiena. “Exact Age Prediction in Social Networks”. *Proceedings of the 24th International Conference on World Wide Web*. WWW ’15 Companion. Florence, Italy: ACM, 2015, pp. 91–92.

[7]. Lars Backstrom, Eric Sun, Cameron Marlow. “Find me if you can: improving geographical prediction with social and spatial proximity”. *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 61–70.

[8]. David Jurgens. “That’s What Friends Are For: Inferring Location in Online Social Media Platforms Based on Social Relationships”. *ICWSM* 13 (2013), pp. 273–282.

[9]. Yantao Jia et al. “Location Prediction: A Temporal-Spatial Bayesian Model”. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7.3 (2016), p. 31.

[10]. Jinpeng Chen, Yu Liu, Ming Zou. “Home location profiling for users in social media”. *Information & Management* 53.1 (2016), pp. 135–143.

[11]. Fan RK Chung. “Spectral graph theory”. CBMS regional conference series in mathematics, No. 92 (1996).

[12]. Oleg U Ivanov, Sergey O Bartunov. “Learning Representations in Directed Networks”. *International Conference on Analysis of Images, Social Networks and Texts*. Springer. 2015, pp. 196–207.

[13]. Bryan Perozzi, Rami Al-Rfou, Steven Skiena. “Deepwalk: Online learning of social representations”. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 701–710.

[14]. T Mikolov, J Dean. “Distributed representations of words and phrases and their compositionality”. *Advances in neural information processing systems* (2013).

[15]. Michael U Gutmann, Aapo Hyvärinen. “Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics”. *Journal of Machine Learning Research* 13.Feb (2012), pp. 307–361.

[16]. François Chollet. *Keras*. https://github.com/fchollet/keras. 2015.

Approaches to estimate location of social network users based on social graph

[1]Y.S. Trofimovich <integral@ispras.ru>
[1]I.S. Kozlov <kozlov-ilya@ispras.ru>
[1,2,3]D.Y. Turdakov <turdakov@ispras.ru>
[1]Institute for System Programming of the Russian Academy of Sciences
Moskva, Aleksandra Solzhenitsyna, 25, 109004, Russia
[2]Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[3]National Research University Higher School of Economics (HSE)
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. Many applications require information about the geolocation of users, which is not always available. Among the users of Twitter only about 26% indicate the name of the city in their profiles, about 30% of users of VKontakte leave this field blank. So there is the problem of determining the place of residence of social network users. We investigate approaches to geolocation of social network users using their mutual bidirectional ties — social graph. At first, we present a brief overview of the work in the field of geolocating users of social networks. Then we propose an approach that relies on graph nodes’ embeddings and supervised machine learning techniques. Series of experiments were conducted with proposed and baseline approaches. Experiments show that proposed approach is comparable with others. The results of experiments allow us to conclude that the proposed approach based on vector representation can be effectively used to determine the user's place of residence by itself, or in combination with classifiers based on user data It is worth noting that the proposed approach has no any specifics related to the geolocation. It can also be used to assess any other demographic attributes that influence the formation of relationships in society. Thus, a similar approach was used in Perozzi and Skiena to determine the age of the users.

Keywords: geolocation, social networks, social graph, graph embeddings

DOI: 10.15514/ISPRAS-2016-28(6)-13

For citation: Trofimovich Y.S., Kozlov I.S., Turdakov D.Y. Approaches to estimate location of social network users based on social graph. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 185-196 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-13

References

[1]. Takeshi Sakaki, Makoto Okazaki, Yutaka Matsuo. "Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development". *IEEE Trans. on Knowl. and Data Eng.* 25.4 (Apr. 2013), pp. 919–931.

[2]. Alex Lamb, Michael J Paul, Mark Dredze. "Separating Fact from Fear: Tracking Flu Infections on Twitter." Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Atlanta. June 2013.

[3]. Adam Sadilek, Henry A Kautz, Vincent Silenzio. "Modeling Spread of Disease from Social Interactions." Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012.

[4]. Zhiyuan Cheng, James Caverlee, Kyumin Lee. "You Are Where You Tweet: A Content-based Approach to Geo-locating Twitter Users". *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. CIKM '10. Toronto, ON, Canada: ACM, 2010, pp. 759–768.

[5]. Afshin Rahimi et al. "Exploiting Text and Network Context for Geolocation of Social Media Users". *CoRR* abs/1506.04803 (2015).

[6]. Bryan Perozzi, Steven Skiena. "Exact Age Prediction in Social Networks". *Proceedings of the 24th International Conference on World Wide Web*. WWW '15 Companion. Florence, Italy: ACM, 2015, pp. 91–92.

[7]. Lars Backstrom, Eric Sun, Cameron Marlow. "Find me if you can: improving geographical prediction with social and spatial proximity". *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 61–70.

[8]. David Jurgens. "That's What Friends Are For: Inferring Location in Online Social Media Platforms Based on Social Relationships". *ICWSM* 13 (2013), pp. 273–282.

[9]. Yantao Jia et al. "Location Prediction: A Temporal-Spatial Bayesian Model". *ACM Transactions on Intelligent Systems and Technology (TIST)* 7.3 (2016), p. 31.

[10]. Jinpeng Chen, Yu Liu, Ming Zou. "Home location profiling for users in social media". *Information & Management* 53.1 (2016), pp. 135–143.

[11]. Fan RK Chung. "Spectral graph theory". CBMS regional conference series in mathematics, No. 92 (1996).

[12]. Oleg U Ivanov, Sergey O Bartunov. "Learning Representations in Directed Networks". *International Conference on Analysis of Images, Social Networks and Texts*. Springer. 2015, pp. 196–207.

[13]. Bryan Perozzi, Rami Al-Rfou, Steven Skiena. "Deepwalk: Online learning of social representations". *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 701–710.

[14]. T Mikolov, J Dean. "Distributed representations of words and phrases and their compositionality". *Advances in neural information processing systems* (2013).

[15]. Michael U Gutmann, Aapo Hyvärinen. "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics". *Journal of Machine Learning Research* 13.Feb (2012), pp. 307–361.

[16]. François Chollet. *Keras*. https://github.com/fchollet/keras. 2015.

Joining Dictionaries and Word Embeddings for Ontology Induction

D.A. Ustalov <dau@imm.uran.ru>
Krasovskii Institute of Mathematics and Mechanics
16 Sofia Kovalevskaya Str., 620990 Yekaterinburg, Russia

Abstract. This paper presents an ontology induction approach that extracts the structured lexical knowledge from synonym dictionaries and establishing the semantic relations within these structures using word embeddings and their projections. The results of the preliminary experiments have also been reported showing certain strengths and weaknesses of the proposed approach.

Keywords: ontology induction; lexical resource; synonyms; word embeddings; graph clustering; projection learning.

DOI: 10.15514/ISPRAS-2016-28(6)-14

For citation: Ustalov D.A. Joining Dictionaries and Word Embeddings for Ontology Induction. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 197-206. DOI: 10.15514/ISPRAS-2016-28(6)-14

1. Introduction

A thesaurus, or a lexical ontology, is a lexical database that groups the words into the sets of synonyms called *synsets* or *concepts*, and records a number of semantic *relations* between these concepts [1]. It is a crucial resource for many natural language processing and artificial intelligence problems, which require common sense reasoning. However, the current state of the electronic thesauri for the Russian language makes it highly topical to refine and to integrate the existing openly available resources to facilitate the development of language technology [2]. In this study, an ontology induction approach that integrates the knowledge represented by the synonym dictionaries enhanced by the methods of distributional semantics has been presented and preliminarily evaluated.

2. Related Work

Given the fact that thesauri are composed of concepts and relations, the approaches for acquisition of both have been briefly reviewed. Unsupervised methods for

concept discovery are designed for clustering co-occurrence graphs to group the words having similar meanings. For instance, the methods proposed by Schütze [3] and Lin & Pantel [4] construct such graphs using large text corpora. Recently, a significant attention in lexical semantics has been paid to the specialized algorithms like Chinese Whispers [5], which is a hard clustering algorithm that assigns a word to at most one cluster at a time, and MaxMax [6], which is a soft clustering algorithm designed specifically for the word sense induction task [7].

Currently, the most widely used method for detecting hyponymy-hypernymy relations is the Hearst patterns [8]. However, these lexical-syntactic patterns offer the sparse representation of words that is less convenient than word embeddings [9]. Fu et al. [10] proposed the projection learning approach to learning hypernyms for the Chinese language. This approach assumes learning the projection matrix so that multiplying it on a hyponym vector produces a hypernym vector. The learning problem has been posed as the linear regression problem that has been then numerically approximated using stochastic gradient descent. Shwartz et al. developed an integrated method that combines the syntactic parsing features with word embeddings based on a long short-term memory network [11]. The resulting method called HypeNET has been implemented using the recurrent neural network that encodes the patterns with the embeddings.

3. Approach

The proposed ontology induction approach, depicted at Fig. 1, uses a synonym dictionary to produce the concepts, and a pre-trained word embeddings model to establish the hyponymy-hypernymy relations between the concepts. The goal of the concept discovery step is to yield a set of concepts by grouping the words with similar meanings from a synonym graph. The goal of the relation establishment step is to link these concepts to each other using the hyponymy-hypernymy relation, which is also known as the subsumption relation or *is-a* relation.

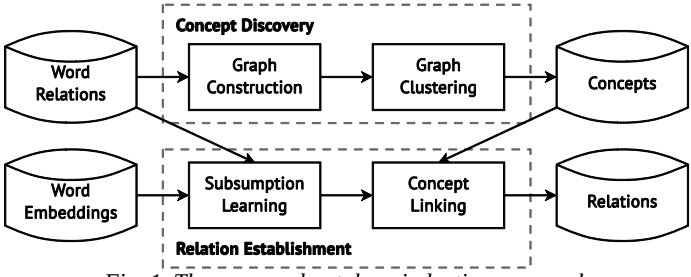


Fig. 1. The proposed ontology induction approach

3.1 Concept Discovery

A synonym dictionary is a graph the set of which vertices contains the words and the set of which edges contains the word pairs connected via the synonymy relation. The cliques in such a graph naturally form the densely connected sets of

synonyms [12] corresponding to the concepts. Given the fact that the clique problem in a graph is NP-complete [13], the Chinese Whispers [5] graph clustering algorithm has been used for finding a global segmentation of the graph. However, the hard clustering property of this algorithm does not handle the polysemous words well. To deal with that, a word sense induction procedure has been run to extract the word senses [14] which have been then combined back into a disambiguated word sense graph [15]. Finally, the disambiguated word sense graph has been clustered again using Chinese Whispers to induce the concepts from these disambiguated word senses.

3.2 Relation Establishment

To extend the lexical coverage of the available subsumption word pairs, a projection learning setup has been used [10] to compute a transformation matrix for word embeddings by projecting the hyponym vector to its possible hypernym vector. To get this done, the 100-dimensional word embeddings dataset for Russian trained by Arefyev et al. [16] using the skip-gram architecture with the sliding window of 10 words and the minimal word frequency of 100 on a text corpus of 13 billion words have been obtained (this configuration is among the best of those participated in the RUSSE study [17] despite its low computational performance requirements). The subsumption pairs from the Russian Wiktionary [18] stem both the train and test sets for learning the projection matrix, the total number of the pairs being 33 885. To avoid lexical overfitting [19], no hyponym from the train set is present in the test set. Since the specificity of the semantic relations differing in various regions of the embedding space, the hyponym embeddings have been clustered using the *k*-means algorithm tuned on a development dataset.

Having the concepts induced and the subsumptions trained, the relations between the concepts are established as follows. For each concept, every word has been projected to its hypernym embedding and the ten nearest neighbours of that projection have been obtained. These neighbours jointly form a bag of words for this concept, for which the most similar concept is computed using the cardinality of the set intersection as the similarity measure. If such a concept found, the former concept is considered as a hyponym of the latter.

4. Experiments

For evaluation purposes, the Russian Wiktionary [18], the Abramov's dictionary [20], and the Universal Dictionary of Concepts [21] have been combined into the single graph to benefit from different lexical coverage provided by the different synonym dictionaries and also to enforce the jointly observed synonymy relations. The resulting graph has 406 889 edges connecting 74 133 individual words. The RuThes-lite 2.0 lexical ontology, composed of 31.5K concepts, 111.5K lexemes and 130K relations, has been used as the gold standard [1] during these experiments.

4.1 Concept Evaluation

To assess the performance of the described concept discovery method, the same graph has been also processed by two other algorithms: Chinese Whispers [5] and MaxMax [6]. Similarly to the experimental setup used for evaluating the MaxMax clustering algorithm [6], the pairwise precision, recall and F_1-measure scores [22] and V-measure score [23] have been computed (Table 1).

Table 1. Concept evaluation

Method \ Parameter	**# of lexemes**	**# of concepts**	**Precision**	**Recall**	**F_1-measure**	**V-measure**
Chinese Whispers	73 878	16 063	0.135	0.022	0.038	0.866
MaxMax	73 878	16 870	0.181	0.004	0.007	0.835
Concept Discovery	29 650	5 984	0.193	0.039	0.065	0.860

According to the concept evaluation results, the described concept discovery method outperformed other methods on every pairwise score and showed the comparable V-measure representing the goodness of the output clustering. As a hard clustering algorithm, Chinese Whispers demonstrated good performance on grouping monosemous words into the concepts like {*компьютер* (computer), *ЭВМ* (ECM), …}, especially on named entities. Also, as anticipated, its performance degraded on polysemous words, resulting in the concepts like {*вода* (water), *акватория* (water area), *влага* (moisture), *кислород* (oxygen), *водород* (hydrogen), …}. Surprisingly, MaxMax, despite the existence of a successful case study of the Portuguese language [7], showed poor results in this study due to the possible difference of the expected graph structure. Firstly, unlike other methods being compared, it emitted a large number of the concepts grouping more than 300 words. After the investigation, these concepts have been removed from the evaluation as the non-relevant. Secondly, a substantial part of the concepts provided by MaxMax grouped the words having no obvious synonymy relation, e.g., {*прайс* (price), *бином Ньютона* (Newtonian binomial), *программный пакет* (software package), …}. In contrast, the concepts yielded by the described concept discovery method correctly reflect the polysemy phenomenon, e.g., {*пустота* (emptiness), *бессодержательность* (barrenness), *бессмысленность* (meaninglessness), …} and {*вакуум* (vacuum), *пустота* (emptiness), *ничто* (nihil), …}. Unfortunately, the disambiguation procedure being used [15] tends to miss certain underrepresented word senses, which results in their absence in the disambiguated word sense graph, and, therefore, in the output concepts as well.

4.2 Relation Evaluation

To assess the performance of the described relation establishment method, it has been applied for each of the 5 984 concepts discovered at the previous step. Each concept has been matched to the most similar RuThes-lite concept using the cardinality of the set intersection as the similarity measure. An established relation has been then considered as correct if there exists a directed path from the hyponym concept to the hypernym concept in the gold standard. Also, the performance of the projection learning setup has been compared to the using of the unmodified subsumption pairs from the Russian Wiktionary without the word embeddings. Table 2 shows the relation evaluation results.

Table 2. Relation evaluation

Dataset \ Parameter	**# of candidate relations**	**# of matched concepts**	**# of correct relations**
Russian Wiktionary	1 627	1 210	113
Projection Learning	3 918	2 293	133

According to the relation evaluation results, the projection learning setup increases the number of candidate concepts and relations, but the number of the correctly established relations did not increase substantially. However, due to the lack of the available subsumption dictionaries, it seems reasonable to try a larger word embeddings dataset, a different learning setup, or a different concept similarity measure.

5. Conclusion

In this short paper, an ontology induction approach that induces a thesaurus structure by integrating synonym dictionaries for discovering concepts and a distributional model for establishing the hyponymy-hypernymy relations between them, has been described and preliminarily evaluated. The results of this study are openly available under a libré license: https://github.com/dustalov/concept-discovery. The plans for the further study include the improving the relation establishment step by more sophisticated matching and machine learning techniques as well as applying crowdsourcing for validating the subsumption candidates.

Acknowledgements. The author is grateful to Andrew Krizhanovsky for the provided Russian Wiktionary dump, to Natalia Loukachevitch for the provided RuThes-lite dataset, and to Alexander Panchenko for the fruitful discussions on the present study. The reported study was funded by RFBR according to the research project no. 16-37-00354 мол_а. This work is supported by the Russian Foundation

for the Humanities projects no. 13-04-12020 ``New Open Electronic Thesaurus for Russian" and no. 16-04-12019 ``RussNet and YARN thesauri integration".

References

[1]. Loukachevitch, N. Thesauri in Information Retrieval Tasks. Moscow State University Press, Moscow, 2011 (in Russian)

[2]. Kiselev, Y., Porshnev, S.V., Mukhin, M. Current Status of Russian Electronic Thesauri: Quality, Completeness and Availability. Programmnaya Ingeneria, 6, 2015, pp. 34–40 (in Russian)

[3]. Schütze, H. Automatic Word Sense Discrimination. Journal of Computational Linguistics, 24, 1998 pp. 97–123

[4]. Lin, D., Pantel, P. Concept Discovery from Text. In Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, Association for Computational Linguistics, 2002, pp. 1–7

[5]. Biemann, C. Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. In Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing. TextGraphs-1, Association for Computational Linguistics, 2006 pp. 73–80

[6]. Hope, D., Keller, B.: MaxMax: A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction. In: 14th International Conference, CICLing 2013, Samos, Greece, March 24-30, 2013, Proceedings, Part I, Springer Berlin Heidelberg, 2013, pp. 368–381

[7]. Gonçalo Oliveira, H., Gomes, P. ECO and Onto.PT: a flexible approach for creating a Portuguese wordnet automatically. Language Resources and Evaluation, v. 48, № 2, 2014, pp. 373–393

[8]. Hearst, M.A. Automatic Acquisition of Hyponyms from Large Text Corpora. In: Proceedings of the 14th Conference on Computational Linguistics - Volume 2. COLING '92, Association for Computational Linguistics, 1992, pp. 539–545

[9]. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In: Advances in Neural Information Processing Systems 26. Curran Associates, Inc., 2013 pp. 3111–3119

[10]. Fu, R., Guo, J., Qin, B., Che, W., Wang, H., Liu, T. Learning Semantic Hierarchies via Word Embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2014 pp. 1199–1209

[11]. Shwartz, V., Goldberg, Y., Dagan, I. Improving Hypernymy Detection with an Integrated Path-based and Distributional Method. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2016, pp. 2389–2398

[12]. Kamps, J., Marx, M., Mokken, R.J., de Rijke, M. Using WordNet to Measure Semantic Orientations of Adjectives. In Proceedings of LREC'2004. European Language Resources Association, 2004 pp. 1115–1118

[13]. Bomze, I.M., Budinich, M., Pardalos, P. M., Pelillo, M. The maximum clique problem. In Handbook of Combinatorial Optimization. Springer, 1999, 1–74

[14]. Panchenko, A., Simon, J., Riedl, M., Biemann, C. Noun Sense Induction and Disambiguation using Graph-Based Distributional Semantics. In Proceedings of the 13th

Conference on Natural Language Processing (KONVENS 2016). Bochumer Linguistische Arbeitsberichte, 2016 pp. 192–202

[15]. Faralli, S., Panchenko, A., Biemann, C., Ponzetto, S.P. Linked Disambiguated Semantic Networks. In 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II, Springer International Publishing, 2016 pp. 56–64

[16]. Arefyev, N., Panchenko, A., Lukanin, A., Lesota, O., Romanov, P.: Evaluating Three Corpus-Based Semantic Similarity Systems for Russian. In Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue", Volume 2 of 2, Papers from special sessions, RGGU, 2015, pp. 106–118

[17]. Panchenko, A., Loukachevitch, N.V., Ustalov, D., Paperno, D., Meyer, C.M., Konstantinova, N.: RUSSE: The First Workshop on Russian Semantic Similarity. In Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue", Volume 2 of 2, Papers from special sessions, RGGU, 2015, pp. 89–105

[18]. Krizhanovsky, A.A., Smirnov, A.V.: An approach to automated construction of a general-purpose lexical ontology based on Wiktionary. Journal of Computer and Systems Sciences International, v. 52, № 2, 2013, pp. 215–225

[19]. Levy, O., Remus, S., Biemann, C., Dagan, I.: Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, 2015, pp. 970–976

[20]. Abramov, N. The dictionary of Russian synonyms and semantically related expressions, 7th edition. Moscow (1999) (in Russian)

[21]. Dikonov, V.G. Development of lexical basis for the Universal Dictionary of UNL Concepts. In Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue", Bekasovo, Russia, 2013, pp. 212–221

[22]. Powers, D.M.W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies, v. 2, № 4, 2011, pp. 37–63

[23]. Rosenberg, A., Hirschberg, J.: V-Measure. A Conditional Entropy-Based External Cluster Evaluation Measure. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Association for Computational Linguistics, 2007, pp. 410–420

Применение словарей и векторов слов для автоматического построения лексической онтологии

Д.А. Усталов <dau@imm.uran.ru>
Институт математики и механики им. Н.Н.Красовского Уральского отделения Российской академии наук,
Россия, 620990, г. Екатеринбург, ул. Софьи Ковалевской, д. 16

Аннотация. В статье представлен подход к автоматическому построению лексической онтологии путём извлечения и связывания структурированных данных, направленный на повторное использование материалов существующих лексических ресурсов неизвестного качества. Подход состоит из двух этапов. На первом этапе производится построение и кластеризация графа синонимов с целью вывода отдельных значений слов и их объединения в синонимические ряды, именуемые синсетами или понятиями. На втором этапе производится формирование родо-видовых отношений между понятиями путём сопоставления родо-видовых пар слов. С целью расширения множества доступных родо-видовых пар слов выполняется преобразование векторных представлений гипонимов в векторные представления гиперонимов при помощи проекционной матрицы. Проведены предварительные эксперименты с использованием тезауруса русского языка в качестве золотого стандарта. Проанализированы преимущества и недостатки предложенного подхода.

Ключевые слова: автоматическое построение онтологии; лексический ресурс; синоним; векторное представление слова в пространстве низкой размерности;, кластеризация графа; подбор проекционной матрицы.

DOI: 10.15514/ISPRAS-2016-28(6)-14

Для цитирования: Усталов Д.А. Применение словарей и векторов слов для автоматического построения лексической онтологии. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 197-206 (на английском). DOI: 10.15514/ISPRAS-2016-28(6)-14

Благодарности. Автор благодарит Андрея Крижановского за предоставленный машиночитаемый вариант Русского Викисловаря, Наталью Лукашевич за предоставленный машиночитаемый вариант тезауруса РуТез, а также Александра Панченко за ценные рекомендации по настоящей работе. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-37-00354 мол_а. Исследование поддержано грантами РГНФ № 13-04-12020 «Новый открытый электронный тезаурус русского языка» и № 16-04-12019 «Интеграция тезаурусов RussNet и YARN».

Список литературы

[1]. Лукашевич Н.В. Тезаурусы в задачах информационного поиска. М.: Издательство Московского университета, 2011. 512 с.

[2]. Киселёв Ю., Поршнев, С.В., Мухин М.Ю. Современное состояние электронных тезаурусов русского языка: качество, полнота и доступность. Программная инженерия, 2015, вып. 6, с. 34-40.

[3]. Schütze H. Automatic Word Sense Discrimination. Journal of Computational Linguistics, 1998, issue 24, pp. 97-123.

[4]. Lin D., Pantel P. Concept Discovery from Text. Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, pp. 1-7, 2002, Association for Computational Linguistics.

[5]. Biemann C. Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing (TextGraphs-1), pp. 73-80, 2006, Association for Computational Linguistics.

[6]. Hope D., Keller B. MaxMax: A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction. Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Part I, pp. 368-381, 2013, Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-37247-6_30.

[7]. Gonçalo Oliveira H., Gomes P. ECO and Onto.PT: a flexible approach for creating a Portuguese wordnet automatically. Language Resources and Evaluation, 2014, vol. 48, issue 2, pp. 373-393. DOI: 10.1007/s10579-013-9249-9.

[8]. Hearst M.A. Automatic Acquisition of Hyponyms from Large Text Corpora. Proceedings of the 14th Conference on Computational Linguistics - Volume 2, pp. 539-545, 1992, Association for Computational Linguistics. DOI: 10.3115/992133.992154.

[9]. Mikolov T., Sutskever I., Chen K., Corrado G.S., Dean J. Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems 26, pp. 3111-3119, 2013, Curran Associates, Inc.

[10]. Fu R., Guo J., Qin B., Che W., Wang H., Liu T. Learning Semantic Hierarchies via Word Embeddings. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1199-1209, 2014, Association for Computational Linguistics.

[11]. Shwartz V., Goldberg Y., Dagan I. Improving Hypernymy Detection with an Integrated Path-based and Distributional Method. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2389-2398, 2016, Association for Computational Linguistics.

[12]. Kamps J., Marx M., Mokken R.J., de Rijke M. Using WordNet to Measure Semantic Orientations of Adjectives. Proceedings of LREC'2004, pp. 1115-1118, 2004, European Language Resources Association.

[13]. Bomze I.M., Budinich M., Pardalos P. M., Pelillo M. The Maximum Clique Problem. Handbook of Combinatorial Optimization, 1999, pp. 1-74. DOI: 10.1007/978-1-4757-3023-4_1.

[14]. Panchenko A., Simon J., Riedl M., Biemann C. Noun Sense Induction and Disambiguation using Graph-Based Distributional Semantics. Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016), pp. 192-202, 2016, Bochumer Linguistische Arbeitsberichte.

[15]. Faralli S., Panchenko A., Biemann C., Ponzetto S.P. Linked Disambiguated Semantic Networks. Proceedings of the 15th International Semantic Web Conference - Part II, pp. 56-64, 2016, Springer International Publishing. DOI: 10.1007/978-3-319-46547-0_7.

[16]. Арефьев Н., Панченко А., Луканин А., Лесота О., Романов, П. Сравнение трёх систем семантической близости для русского языка. Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» - Том 2. Доклады специальных секций, с. 106-118, 2015, Изд-во РГГУ.

[17]. Панченко А., Лукашевич Н.В., Усталов Д., Паперно Д., Мейер К.М., Константинова Н. RUSSE: семинар по оценке семантической близости для русского языка. Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» - Том 2. Доклады специальных секций, с. 89-105, 2015, Изд-во РГГУ.

[18]. Krizhanovsky A.A., Smirnov A.V. An approach to automated construction of a general-purpose lexical ontology based on Wiktionary. Journal of Computer and Systems Sciences International, 2013, vol. 52, issue 2, pp. 215-225. DOI: 10.1134/S1064230713020068.

[19]. Levy O., Remus S., Biemann C., Dagan I. Do Supervised Distributional Methods Really Learn Lexical Inference Relations?. Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 970-976, 2015, Association for Computational Linguistics.

[20]. Абрамов Н. Словарь русских синонимов и сходных по смыслу выражений. М.: Русские словари, 1999, 528 с.

[21]. Dikonov V.G. Development of lexical basis for the Universal Dictionary of UNL Concepts. Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference “Dialogue”, pp. 212-221, 2013. RGGU.

[22]. Powers D.M.W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies, 2011, vol. 2, issue 4, pp. 37-63.

[23]. Rosenberg A., Hirschberg J. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 410-420, 2007, Association for Computational Linguistics.

Сравнительный анализ мер сходства, основанных на преобразовании скользящих аппроксимаций, в задачах классификации временных рядов

[1] *И.С. Алимова <alimovailseyar@gmail.com>*
[1] *В.Д. Соловьев <maki.solovyev@mail.ru>*
[2] *И.З. Батыршин <batyr1@gmail.com>*
[1] *Казанский федеральный университет,*
420008, Россия, г. Казань, ул. Кремлевская, д. 18
[2] *Национальный политехнический институт,*
CIC IPN, 0773, DF, Мехико, Мексика.

Аннотация. Одним из главных вопросов при решении задачи классификации временных рядов является выбор меры сходства рядов. В данной статье представлен сравнительный анализ меры сходства временных рядов, основанной на преобразовании скользящих аппроксимаций (САП трансформ), с двумя другими наиболее известными мерами: Алгоритмом Динамической Трансформации и Евклидовым расстоянием для задачи классификации. Кроме того, предложен алгоритм, улучшающий точность меры САП трансформ для временных рядов, имеющих схожие значения, но сдвинутых относительно друг друга по оси X, где координата на оси X представляет собой единицу времени.

Ключевые слова: временной ряд; классификация; мера сходства; САП трансформ; преобразование скользящих аппроксимаций.

DOI: 10.15514/ISPRAS-2016-28(6)-15

Для цитирования: Алимова И.С., Соловьев В.Д., Батыршин И.З. Сравнительный анализ мер сходства, основанных на преобразовании скользящих аппроксимаций, в задачах классификации временных рядов. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 207-222. DOI: 10.15514/ISPRAS-2016-28(6)-15

1. Введение

В связи с непрекращающимся ростом данных, представленных в виде временных рядов, возрастает интерес к анализу временных рядов с целью извлечения новой полезной информации. Временной ряд представляет собой

упорядоченную последовательность данных, собранных в разные моменты времени. Временные ряды встречаются в области финансов, метеорологии, экономики, нефтедобыче, медицине и т.д. Примерами временного ряда являются показатели курсов валют, взятые за определенный период времени, кардиограмма человека, значения температуры воздуха окружающей среды в течении суток.

Основной задачей анализа временных рядов является выявление структуры ряда для прогнозирования его дальнейших значений. Прогноз будущих значений используется для принятия решений по дальнейшей работе системы. Например, определение дальнейшей динамики цен на акции компании может помочь в решении о покупке или продаже акций компании.

Одним из методов анализа временных рядов является метод классификации. Классификация временных рядов позволяет определить группы схожих по значению рядов с помощью различных мер сходства. В подобных задачах выбор меры сходства временных рядов влияет на точность классификации в большей степени, чем выбор метода классификации, что доказано на примере меры Алгоритма Динамической Трансформации [1]. В связи с этим, проводятся исследования по эффективности мер сходства временных для задачи классификации рядов [2, 3].

В работе [4] была представлена новая мера сходства временных рядов - САП трансформ. Данная мера применялась в области метеорологии для выявления взаимосвязи между загрязнением воздуха и метеорологическими показателями [5], а также в области нефтедобычи для задачи кластеризации нефтяных вышек [6]. Однако, не известны работы о сравнении меры САП с прочими мерами на существующих наборах временных рядов, а также не предложены способы по улучшению точности классификации меры САП трансформ. В связи с этим целью данной работы является сравнительный анализ меры САП трансформ с другими наиболее распространенными мерами сходства временных рядов, а также предложен один из способов модификации меры САП трансформ для улучшения точности.

В работе представлены результаты анализа точности трех мер сходства временных рядов на задаче классификации:

- Алгоритм динамической трансформации временной шкалы (АДТ).
- Евклидово расстояние.
- Преобразование скользящих аппроксимаций (САП трансформ) в двух вариациях: вычисленную для всех размеров скользящих окон и с модификацией, предложенной в этой работе.

В следующем разделе представлено обоснование выбора данных мер. В разделе 3 описаны формулы для расчета мер и используемый в данной работе алгоритм классификации временных рядов. Раздел 4 описывает проведенные эксперименты и наборы временных рядов на которых они проводились. В разделе 5 представлены результаты проведенного исследования.

2. Выбор мер сходства для анализа

В работе [2] представлены результаты оценки эффективности классификации 49 мер на наборе временных рядов из коллекции UCR [7]. Согласно полученным результатам, наиболее точной оказалась мера АДТ с ее модификациями. Мера АДТ [8] применяется в распознавании речи, жестов, рукописного текста, робототехнике, медицине, биоинформатике [9-12]. Преимуществом данной меры является способность распознавать схожие временные ряды даже если периоды рядов сдвинуты относительно друг друга по оси. Однако, данная мера имеет ряд недостатков. В частности, мера показывает неточные результаты для временных рядов, схожих между собой, но имеющих большую разницу максимальных и минимальных значений [13].

После меры АДТ наиболее точной мерой сравнения временных рядов оказалось Евклидово расстояние с его преобразованиями. Евклидово расстояние - это хорошо известная и наиболее простая для вычисления мера [14]. Основным ее преимуществом является маленькая вычислительная сложность, как по времени (O(n)), так и по памяти (O(1)). Однако, она уступает мере АДТ по точности классификации [2].

Мера САП трансформ основана на значении углов наклона касательных к графикам временных рядов в выбранной точке. Преимуществом данной меры является ее возможность определять как положительные зависимости, когда временные ряды одновременно увеличивают или уменьшают значения, так и отрицательные зависимости, когда значения одного временного ряда убывают, а другого возрастают и наоборот.

3. Метод классификации временных рядов

Задача классификации временных рядов связана с определением групп близких по значению временных рядов. Близость временных рядов определяется различными мерами сходства. В данном разделе представлены формулы вычисления выбранных для анализа мер сходства для временных рядов $x = (x_1, \dots x_m)$, $y = (y_1, \dots y_m)$ и алгоритм классификации.

3.1 Евклидово расстояние

Евклидово расстояние представляет из себя сумму расстояний между точками временного ряда и вычисляется по формуле:

$$d_{\text{Евкл}} = \sqrt{\sum_{i=0}^{m} |x_i - y_i|^2}$$

3.2 АДТ

В данной статье вычислялась классическая мера АДТ без модификаций. На первом этапе подсчитывалась матрица расстояний:

$$d_{ij} = |x_i - y_j|, i = 1, \dots, m, j = 1, \dots, m$$

На основе полученной матрицы вычислялась матрица деформаций:

$$D_{ij} = d_{ij} + \min(D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1})$$

По матрице деформаций строился вектор трансформации W:

$$w_0 = D_{mm}$$

$$w_k = \min(D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}), k = 1, \dots, m$$

Итоговая мера АДТ вычислялась по формуле:

$$d_{\text{АДТ}} = \frac{1}{m}\sum_{i=1}^{m} w_i$$

3.3 САП

Мера САП вычислялась по формулам, описанным в [4]. Для подсчета данной меры сначала вычислялось значение локального тренда a_i:

$$a_i = \frac{6\sum_{j=0}^{k-1}(2j - k + 1)y_{i+j}}{hk(k^2 - 1)},$$

где k - размер выбранного окна, h - временной интервал (в нашем случае h=1).

После этого подсчитывалась мера ассоциаций для локальных трендов по формуле:

$$coss_k(X, Y) = \frac{\sum_{i=1}^{m} a_{yi} \cdot a_{xi}}{\sqrt{\sum_{i=1}^{m} a_{yi}^2 \cdot \sum_{j=1}^{m} a_{xj}^2}}$$

Это значение подсчитывалось для всех k = 2, ..., m. Среднее среди полученных значений является мерой САП трансформ:

$$AM(X, Y) = \frac{1}{|K|}\sum_{k=1}^{m-1} coss_k(X, Y)$$

Однако, точность базового алгоритма подсчета данной меры варьируется в зависимости от размера выбранного окна k и характеристик временных рядов. Сокращение количества и выбор наиболее оптимальных значений параметра k, а также модификации алгоритма позволили улучшить точность меры. Выбор оптимального значения параметра k и преобразование алгоритма описаны в следующем разделе.

3.4 САП с динамической трансформацией локальных трендов

В базовом варианте САП мера считается по всем возможным размерам окна, однако, сокращение их количества и выбор оптимальных размеров позволили улучшить точность меры. Кроме того, мера САП не приспособлена для выявления сходства между временными рядами, значения которых имеют схожую динамику, однако, сдвинуты относительно друг друга по оси X. С целью улучшения точности на подобных временных рядах были произведены модификации базовой меры САП трансформ, описанные в этом разделе.

Для подбора значений k временные ряды в обучающем файле были разделены на два подмножества случайным образом с одинаковым количеством рядов в каждом из них. Одно подмножество принималось за контрольную выборку, другое - за обучающее. Далее для каждого значения окна k <= 30 мы проводили классификацию обучающего множества по алгоритму, описанному ниже. Размер окна k, для которого точность имела максимальное значение, использовался при построении матрицы косинусов для классификации основного тестового множества временных рядов.

Для полученного на предыдущем этапе значения k вычислялись вектора локального тренда a_x и a_y. Для вычисленных векторов строилась матрица косинусов:

$$coss_{p,q}(a_x, a_y) = \frac{\sum_{i=1}^{m-\max\{p,q\}+1} a_{y_{i+q}} \cdot a_{x_{i+p}}}{\sqrt{\sum_{i=1}^{m-q+1} a_{y_{i+q}} \cdot \sum_{j=1}^{m-p+1} a_{x_{i+p}}}}$$

По матрице косинусов аналогично мере АДТ вычислялся вектор трансформации:

$$w_0 = coss_{mm}$$

$$w_k = w_{k-1} + \min\left(coss_{i-1,j-1}, coss_{i-1,j}, coss_{i,j-1}\right), k = 1, \dots, m$$

Итоговое значение меры вычислялось по формуле:

$$d_{\text{САП+ДТ}} = \sum_{i=1}^{m} w_i$$

Описанную в данном разделе меру в дальнейшем будем называть САП с динамической трансформацией локальных трендов (САП+ДТ).

3.5 Алгоритм классификации

Входные данные: T — тестовый набор временных рядов, S — обучающий набор данных, M — мера сходства
Результат: k — количество верно классифицированных рядов
k ← 0
Для каждого t ∈ T:
Находится s ∈ S такой, что $M(t,s)$ — минимальна
Если номера классов s и t совпадают:
k ← $k + 1$

Алгоритм 1. Алгоритм подсчета количества верно классифицированных рядов для заданной меры M.

Algorithm 1. The algorithm for counting the number of correctly classified time series for a given measure M.

Классификация временных рядов проходила по методу ближайшего соседа. Класс временного ряда выбирался следующим образом: для каждого временного ряда из тестового файла подбирался ряд из обучающего файла такой, что используемая мера для выбранных временных рядов была минимальна. Если номера классов выбранных рядов совпадали, считалось, что ряд классифицирован верно.

4. Эксперименты

Для оценки точности мер сходства временных рядов была проведена классификация временных рядов, где в качестве критерия для сравнения рядов применялись выбранные меры сходства. В данном разделе описаны наборы данных и критерии по которым проводилась оценка точности мер сходства временных рядов.

4.1 Набор данных

Мы использовали временные ряды из 43 наборов данных из коллекции UCR [7]. Коллекция содержит в себе как реальные данные, полученные в результате измерений, так и синтетические. Например, временные ряды в наборе данных GUN_POINT являются результатом измерения траектории движения центра ладони руки человека, когда он достает пистолет из кобуры на бедре, целится в мишень и кладет пистолет обратно. Временной ряд Synthetic Control получен генерированием различного вида графиков (циклических, возрастающих, убывающих и т.д.).

В табл. 1 представлена статистика по наборам, которые использованы в дальнейшем для количественного анализа мер. Подробная статистика по длине временных рядов, количеству классов и временных рядов в 43 наборах

представлена в [7]. Каждый набор данных содержит обучающий и тестовый файлы. В каждом из файлов записаны временные ряды с номером класса к которому они принадлежат.

Табл. 1. Статистика по некоторым наборам временных рядов из коллекции UCR.

Table 1. Statistics for some sets of time series of UCR collection.

Название	Количество классов	Кол-во рядов в обучающем файле	Кол-во рядов в тестовом файле	Длина рядов
50Words	50	450	455	270
Car	4	60	60	577
ECGFiveDays	2	23	861	136
FaceFour	4	24	88	350
FacesUCR	14	200	2050	131
Gun_Point	2	50	150	150
MoteStrain	2	20	1252	84
SonyAIBORobotSurface	2	20	601	70
SonyAIBORobotSurfaceII	2	27	953	65
Symbols	6	25	995	398
Trace	4	100	100	275
TwoLeadECG	2	23	1139	82
uWaveGestureLibrary_X	8	896	3582	315
WordsSynonyms	25	267	638	270

4.2 Качественный анализ классификации

Точность (Acc) классификации меры для набора данных подсчитывалась по формуле из [13]:

$$Acc = \frac{\mathrm{k}}{|\mathrm{T}|},$$

где k - количество верно классифицированных рядов, |T| - количество рядов в тестовом наборе данных.

После этого подсчитывалось среднее значение точности по всем наборам данных. Полученное число считалось точностью меры.

Для статистической оценки эффективности проведенных модификаций для меры САП трансформ был использован критерий Вилкоксона.

5. Результаты

5.1 Оценка точности мер сходства

В табл. 2 представлены макро усредненные оценки точности выбранных мер ассоциаций временных рядов, подсчитанные в данной работе на 43 наборах коллекции UCR. Наилучшие результаты согласно табл. 2 показали мера АДТ и Евклидово расстояние, следом идут САП + ДТ и САП.

Несмотря на то, что в целом меры САП трансформ уступают по точности Евклидову расстоянию и АДТ мере, предложенные модификации для меры САП позволили улучшить точность данной меры на 14 наборах данных из 43-х на которых проводились тесты. Кроме того, на 4 наборах данных из этих 14-ти мера САП+ДТ показывает большую точность, чем мера АДТ и Евклидово расстояние. Результаты для наборов данных представлены в табл. 3. Жирным шрифтом в табл. 3 выделены максимальные показатели точности для каждого набора данных.

Табл.2. Точность рассматриваемых мер по всем наборам коллекции, полученное макро усреднением.

Table 2. The average accuracies of the measures for all sets of the collection.

Мера	**Точность**
АДТ	0,91
Евклидово расстояние	0,9
САП + ДТ	0,88

САП	0,86

Для анализа вклада предложенной модификации, меры САП+ДТ и САП были подсчитаны улучшения точности меры САП+ДТ по отношению к САП в процентах. В табл. 3 полученные значения указаны в скобках в столбце для меры САП+ДТ. Согласно полученным значениям, проведенные модификации для меры САП улучшили ее точность в среднем на 9% для 14 наборов данных, на которых мера САП+ДТ превышает по точности меру САП.

Табл. 3. Значения точности рассматриваемых мер для наборов данных для временных рядов из коллекции UCR.

Table 3. The values of accuracy of the described measures for the data sets, on time series from UCR repository.

Набор данных	Евклидово расстояние	АДТ	САП	САП + ДТ
50Words	**0,8**	0,71	0,65	0,67 (+3%)
Car	**0,8**	0,73	0,65	0,67(+2%)
ECGFiveDays	0,81	0,77	0,68	**0,97** (+29%)
FaceFour	0,78	**0,84**	0,82	**0,84** (+2%)
FacesUCR	0,85	**0,94**	0,78	0,87 (+9%)
Gun_Point	0,93	0,87	0,84	**0,94** (+10%)
MoteStrain	0,88	**0,89**	0,83	0,86 (+3%)
SonyAIBORobotSurface	0,71	0,72	0,72	**0,78** (+6%)
SonyAIBORobotSurfaceII	0,87	0,85	0,85	**0,88** (+3%)
Symbols	0,9	**0,95**	0,88	0,94 (+6%)
Trace	0,8	**0,98**	0,71	0,95 (+24%)
TwoLeadECG	0,84	**0,97**	0,76	0,92 (+16%)
uWaveGestureLibrary_X	**0,79**	0,72	0,66	0,71(+5%)
WordsSynonyms	**0,72**	0,68	0,59	0,63(+4%)
Beef	1	1	1	1(0%)
Coffee	1	1	1	1(0%)
Cricket_X	1	1	1	1(0%)
Cricket_Y	1	1	1	1(0%)
Cricket_Z	1	1	1	1(0%)
ECG200	1	1	1	1(0%)

FISH	1	1	1	1(0%)
OliveOil	1	1	1	1(0%)
OSULeaf	1	1	1	1(0%)
Plane	1	1	1	1(0%)
SwedishLeaf	1	1	1	1(0%)
Herring	1	1	1	1(0%)
InsectWingbeatSound	1	1	1	1(0%)
BeetleFly	1	1	1	1(0%)
BirdChicken	1	1	1	1(0%)
PhalangesOutlineCorrect	1	1	1	1(0%)
ShapeletSim	1	1	1	1(0%)
ToeSegmentation1	1	1	1	1(0%)
ToeSegmentation2	1	1	1	1(0%)
Lighting2	1	1	1	1(0%)
wafer	1	1	1	1(0%)
FaceAll	**0,78**	0,77	0,74	0,74(0%)
ItalyPowerDemand	**0,99**	0,93	0,95	0,95(0%)
Adiac	**0,99**	0,59	0,45	0,31(-14%)
CBF	0,88	**1**	0,94	0,89(-5%)
DiatomSizeReduction	0,96	0,96	**0,97**	0,86(-11%)
Lighting7	0,6	**0,8**	0,53	0,49(-4%)
synthetic_control	0,94	**0,98**	0,88	0,87(-1%)
Two_Patterns	0,92	**1**	0,96	0,9(-6%)

Для анализа наборов данных, на которых мера САП улучшила точность, для каждого из них были построены графики. На каждом из них отображались данные двух временных рядов, принадлежащих одному классу из обучающего и тестового наборов данных.

На рис 1, 2 и 3 представлены графики временных рядов, принадлежащих одному классу, из обучающего и тестового наборов данных Gun_Point, ECGFiveDays и Symbols соответственно.

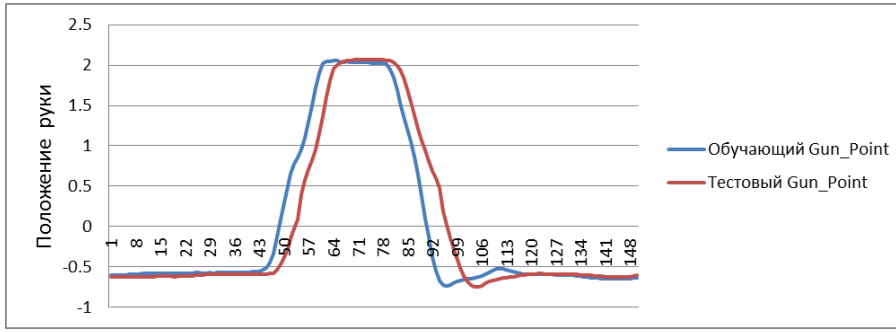


Рис. 1. График обучающего и тестового временных рядов из набора данных Gun_Point, описывающего положение руки при выстреле.

Fig. 1. The charts of the training and test sets of time series of GunPoint dataset describing the hand position in shooting.

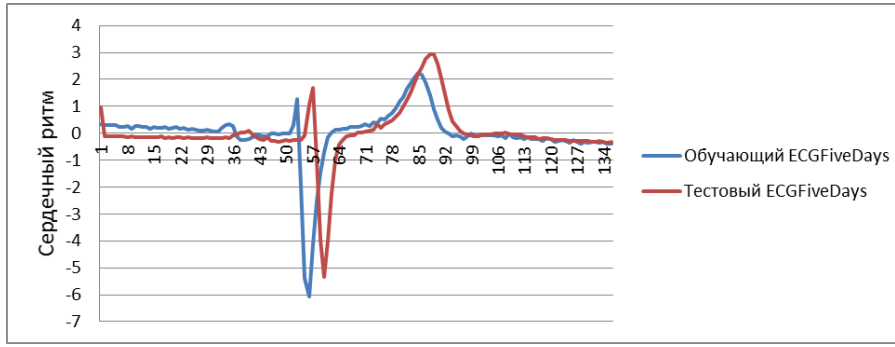


Рис. 2. График обучающего и тестового временных рядов из набора данных ECGFiveDays, describing сердечный ритм.

Fig. 2. The charts of the training and test sets of time series of ECGFiveDays dataset describing heart rate.

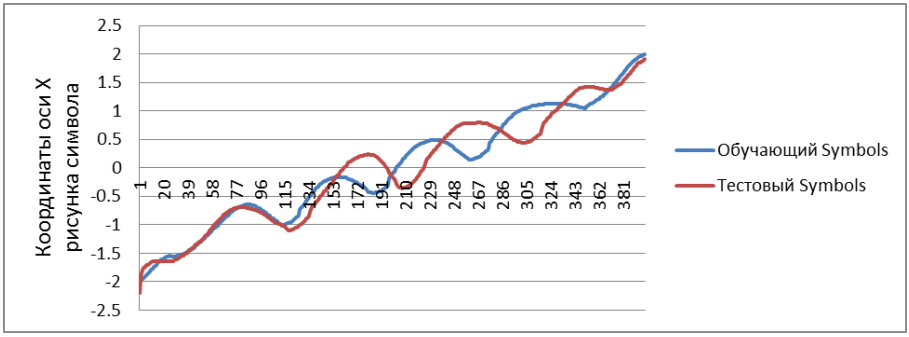


Рис. 3. График обучающего и тестового временных рядов из набора данных Symbols, описывающего координату X для рисунка символа.

Fig. 3. The charts of the training and test sets of time series of ECGFiveDays dataset describing the X-Axis motion in drawing the shape.

На рис. 1, 2 и 3 видно, что графики тестового и обучающего временных рядов имеют схожую форму, однако, сдвинуты относительно друг друга по оси X. Похожие закономерности наблюдаются для остальных 10-ти наборов данных из 14-ти, в табл. 3 они выделены курсивом. Для проверки гипотезы о том, что мера САП+ДТ улучшает точность, были выбраны дополнительно 10 рядов для которых наблюдается сдвиг по оси X относительно друг друга. В табл. 3 они так же выделены курсивом. Лишь на одном из выбранных наборов значение меры ухудшилось. Таким образом, мера САП+ДТ может быть применима при классификации рядов, для которых заранее известно, что принадлежащие к одному классу ряды имеют сдвиг относительно друг друга по оси X.

5.2 Статистическая оценка эффективности меры САП+ДТ

Для проверки гипотезы о том, что мера САП+ДТ имеет более высокую точность классификации временных рядов, была посчитана T статистика критерия Вилкоксона для значений точности мер САП и САП+ДТ. Наборы данных, для которых разность значений точности равна 0, были заранее исключены из рассмотрения. Таким образом, статистика считалась для 20 оставшихся наборов данных. В результате вычислений, было получено $T_{\text{эксп}} = 35$, при $T_{\text{крит}} = 43$ для $p = 0.01$. Поскольку $T_{\text{эксп}} < T_{\text{крит}}$, гипотеза о том, что мера САП+ДТ показывает более высокую точность, является достоверной. Исходя из полученных результатов, можно сделать вывод, что для задачи классификации временных рядов лучше подходит мера САП+ДТ, чем мера САП.

5.3 Оценка производительности алгоритмов

В табл. 4 приведены значения вычислительной сложности алгоритмов для вычисления мер сходства, рассматриваемых в данной статье. Наиболее быстрой для вычисления является Евклидово расстояние. Несмотря на то, что остальные меры имеют одинаковые оценки сложности все же мера САП вычисляется дольше, поскольку на каждом шаге алгоритма необходим проход по подпоследовательности длины k. Наиболее долгая по времени для вычислений мера САП+ДТ, так как для нее необходимы дополнительные вычисления для подбора оптимального значения длины окна k.

Табл.4.Оценки производительности алгоритмов для вычисления рассматриваемых мер сходства для временного ряда длины m.

Table 4. The estimates of productivity of the algorithms for calculating similarity measures for the time series of length m.

Мера	**Сложность**
Евклидово расстояние	$O(m)$
АДТ	$O(m^2)$
САП	$O(m^2)$
САП + ДТ	$O(m^2)$ (для фиксир. значения окна k)

6. Заключение

В данной статье проведен сравнительный анализ точности меры САП трансформ с мерами АДТ и Евклидовой на задаче классификации временных рядов. Помимо этого, предложен один из способов по улучшению точности меры САП для классификации рядов схожих по значению, однако смещенных относительно друг друга по оси X, где координата на оси X представляет собой единицу времени. Результаты исследования показали, что меры САП и САП+ДТ уступают по точности Евклидовой и АДТ мерам на наборах временных рядов различных предметных областей. Однако, для наборов временных рядов, у которых явно прослеживается сдвиг значений относительно оси X мера САП+ДТ превзошла значения точности меры САП, а для 4-х наборов данных значения точности меры САП+ДТ превосходит значения точности всех рассматриваемых в статье мер. Исходя из этого, можно сделать вывод, что для классификации временных рядов, обладающих подобным свойством, в качестве альтернативы можно рассматривать меру САП+ДТ.

7. Благодарности

Работа выполнена при финансовой поддержке проекта РФФИ 15-01-06456 и за счет средств субсидии, выделенной в рамках государственной поддержки Казанского (Приволжского) федерального университета в целях повышения его конкурентоспособности среди ведущих мировых научно-образовательных центров.

Авторы выражают благодарность Тутубалиной Елене Викторовне за помощь при подготовке статьи.

Список литературы

[1]. Weiss S. M. A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets. Proceedings of the 2012 SIAM International Conference on Data Mining, 2012, pp. 999-1010. DOI: 10.1137/1.9781611972825.86.

[2]. Giusti R., Batista G. E. An empirical comparison of dissimilarity measures for time series classification. Intelligent Systems (BRACIS), 2013 Brazilian Conference on. – IEEE, 2013, pp. 82-88.

[3]. Ding H. et al. Querying and mining of time series data: experimental comparison of representations and distance measures. Proceedings of the VLDB Endowment, 2015, vol. 1, issue 2, pp. 1542-1552.

[4]. Batyrshin, I., Herrera-Avelar, R., Sheremetov, L., & Suarez, R. Moving approximations in time series data mining. Proc. Int. Conf. Fuzzy Sets and Soft Computing in Economics and Finance FSSCEF, 2004, pp. 62-72.

[5]. Almanza V., Batyrshin I. On trend association analysis of time series of atmospheric pollutants and meteorological variables in Mexico City Metropolitan Area. Mexican Conference on Pattern Recognition. Springer Berlin Heidelberg, 2011, pp. 95-102.

[6]. Батыршин И.З., Кошульски А., Шереметов Л.Б., Климова А.С., Панов А.М. Анализ взаимодействия нефтяных скважин на основе гибридной кластеризации временных рядов продуктивности скважин. Нечеткие системы и мягкие вычисления. Тверской государственный университет, том 2, вып. 4, 2007 г., стр. 63-73.

[7]. E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. (2006) The UCR time series classification/clustering homepage. Доступно по ссылке: http://www. cs. ucr. edu/~ eamonn/time series data.

[8]. M. Muller. Dynamic time warping. Inf. Retr. Music Motion. Information retrieval for music and motion. Springer, Berlin, 2007, pp. 69–84.

[9]. Lu G. et al. A novel framework of change-point detection for machine monitoring. Mechanical Systems and Signal Processing, 2017, vol. 83, pp. 533-548.

[10]. Rath T. M., Manmatha R. Word image matching using dynamic time warping. Computer Vision and Pattern Recognition. Proceedings IEEE Computer Society Conference on, 2003, vol. 2, pp. 521-527.

[11]. Muda L., Begam M., Elamvazuthi I. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. Journal of Computing, 2010, vol. 2, issue 3, pp. 138-143.

[12]. Vakanski A. et al. Trajectory learning for robot programming by demonstration using hidden Markov model and dynamic time warping. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012, vol. 42, issue 4, pp. 1039-1052.

[13]. Keogh E. J., Pazzani M. J. Derivative Dynamic Time Warping .Sdm, 2001, vol. 1, pp. 5-7.

[14]. Faloutsos C., Ranganathan M., Manolopoulos Y. Fast subsequence matching in time-series databases. Proceedings of the 1994 ACM SIGMOD international Conference on Management of Data, 1994, vol. 23, issue 2, pp. 419-429.

Comparative analysis of the similarity measures based on the moving approximation transformation in problems of time series classification

[1] *I.S. Alimova <alimovailseyar@gmail.com>*
[1] *V.D. Solovyev <maki.solovyev@mail.ru>*
[2] *I.Z. Batyrshin <batyr1@gmail.com>*
[1] *Kazan Federal University,*
18, Kremlyovskaya st., Kazan, 420008, Russia.
[2] *Instituto Politecnico Nacional,*
CIC IPN, 0773, DF, Mexico.

Abstract. One of the major issues dealing with time-series classification problem is the choice of similarity measure. This article presents a comparative analysis of the similarity measure for time series based on moving approximations transform (MAP transforms) with other two most useful measures: Algorithm Dynamic Transformation and Euclidean distance for classification task. In addition, algorithm, that improves the precision of the measure for time series, that have similar values, but shifted relative to each other on the axis X, where coordinate on the X axis represents the time unit, is proposed.

Key words: time series; classification; similarity measure; MAP transform; Moving Approximation Transform.

DOI: 10.15514/ISPRAS-2016-28(6)-15

For citation: Alimova I.S., Solovyev V.D., Batyrshin I.Z. Comparative analysis of the similarity measures based on the moving approximation transformation in problems of time series classification. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 207-222 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-15

References

[1]. Weiss S. M. A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets. Proceedings of the 2012 SIAM International Conference on Data Mining, 2012, pp. 999-1010. DOI: 10.1137/1.9781611972825.86.

[2]. Giusti R., Batista G. E. An empirical comparison of dissimilarity measures for time series classification. Intelligent Systems (BRACIS), 2013 Brazilian Conference on. – IEEE, 2013, pp. 82-88.

[3]. Ding H. et al. Querying and mining of time series data: experimental comparison of representations and distance measures. Proceedings of the VLDB Endowment, 2015, vol. 1, issue 2, pp. 1542-1552.

[4]. Batyrshin, I., Herrera-Avelar, R., Sheremetov, L., & Suarez, R. Moving approximations in time series data mining. Proc. Int. Conf. Fuzzy Sets and Soft Computing in Economics and Finance FSSCEF, 2004, pp. 62-72.

[5]. Almanza V., Batyrshin I. On trend association analysis of time series of atmospheric pollutants and meteorological variables in Mexico City Metropolitan Area. Mexican Conference on Pattern Recognition. Springer Berlin Heidelberg, 2011, pp. 95-102.

[6]. Batyrshin I.Z., Koshulski A.1, Sheremetov L.B.2, Klimova A.S.3, Panova A.M.4. Oil wells interaction analysis based on hybrid clustering of wells productivity time series. Nechetkie sistemy i mjagkie vychislenija [Fuzzy Systems and Soft Computations]. Tverskoj gosudarstvennyj universitet [Tver State University], 2007, vol. 2, issue 4, pp. 63-73 (in Russian).

[7]. E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. (2006) The UCR time series classification/clustering homepage. Available at http://www.cs.ucr.edu/~eamonn/time_series_data.

[8]. M. Muller. Dynamic time warping. Inf. Retr. Music Motion. Information retrieval for music and motion. Springer, Berlin, 2007, pp. 69–84.

[9]. Lu G. et al. A novel framework of change-point detection for machine monitoring. Mechanical Systems and Signal Processing, 2017, vol. 83, pp. 533-548.

[10]. Rath T. M., Manmatha R. Word image matching using dynamic time warping. Computer Vision and Pattern Recognition. Proceedings IEEE Computer Society Conference on, 2003, vol. 2, pp. 521-527.

[11]. Muda L., Begam M., Elamvazuthi I. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. Journal of Computing, 2010, vol. 2, issue 3, pp. 138-143.

[12]. Vakanski A. et al. Trajectory learning for robot programming by demonstration using hidden Markov model and dynamic time warping. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012, vol. 42, issue 4, pp. 1039-1052.

[13]. Keogh E. J., Pazzani M. J. Derivative Dynamic Time Warping .Sdm, 2001, vol. 1, pp. 5-7.

[14]. Faloutsos C., Ranganathan M., Manolopoulos Y. Fast subsequence matching in time-series databases. Proceedings of the 1994 ACM SIGMOD international Conference on Management of Data, 1994, vol. 23, issue 2, pp. 419-429.

Извлечение аспектных терминов на основе условных случайных полей и векторных представлений слов*

Д.О. Машкин <danil.mashkin@gmail.com>
Е.В. Котельников <kotelnikov.ev@gmail.com>
Вятский государственный университет,
610000, г. Киров, ул. Московская, д.36

Аннотация. В интернете существует множество площадок, которые предоставляют пользователям возможность обмениваться своими мнениями и оставлять отзывы о всевозможных товарах и услугах. Эти мнения могут быть полезны не только для других пользователей, но и для компаний, которые хотят отслеживать собственную репутацию и получать своевременные отзывы о своих продуктах и услугах. Наиболее детальная постановка задачи в данной области ставится при аспектно-ориентированном анализе тональности, в котором определяется отношение пользователя не только к объекту в целом, но и к отдельным его аспектам. В настоящей работе рассмотрено решение подзадачи извлечения аспектных терминов при аспектно-ориентированном анализе тональности. Представлен обзор исследований в данной области. Подзадача извлечения аспектных терминов рассматривается как проблема разметки последовательности; для её решения применяется модель условных случайных полей (CRF). Для составления признакового описания последовательности используются векторные представления слов, полученные на основе нейросетевых моделей для русского языка, а также части речи анализируемых слов. Представлены этапы работы программной системы извлечения аспектных терминов. Эксперименты с разработанной программной системой проводились на размеченном корпусе отзывов о ресторанах, созданном в рамках международного тестирования SemEval-2016 Task 5. Исследованы зависимости качества решения подзадачи извлечения аспектных терминов от различных нейросетевых моделей и вариаций признаковых описаний. Наилучшие результаты (F1-мера = 69%) демонстрирует вариант системы, учитывающий контекст и части речи. Работа содержит подробный анализ ошибок, допущенных системой, а также предложения по возможным вариантам их коррекции. В заключении представлены направления дальнейших исследований.

* Работа выполнена при финансовой поддержке Министерства образования и науки РФ, государственное задание ВятГУ (код проекта 586).

Ключевые слова: аспектно-ориентированный анализ тональности; извлечение аспектных терминов; машинное обучение; разметка последовательностей слов; векторное представление слов; word2vec; SemEval-2016.

DOI: 10.15514/ISPRAS-2016-28(6)-16

Для цитирования: Машкин Д.О., Котельников Е.В. Извлечение аспектных терминов на основе условных случайных полей и векторных представлений слов. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 223-240. DOI: 10.15514/ISPRAS-2016-28(6)-16

1. Введение

Интернет-ресурсы являются удобной платформой для коммуникаций, обсуждения и поиска новой информации. Мнения пользователей, содержащие определенную тональность, представляют огромную ценность для компаний, которые хотят отслеживать собственную репутацию и получать своевременные отзывы о своих продуктах и услугах.

Наиболее детальной постановкой задачи в данной области является аспектно-ориентированный анализ тональности (АОАТ) [1], в котором определяется отношение пользователя не только к объекту в целом, но и к отдельным его аспектам. Например, в отзыве о ресторане *«К сервису действительно никаких нареканий и интерьер такого рода мне нравится, но говядина по-французски была ужасна»* можно выделить три аспекта – это *обслуживание*, *интерьер* и *еда*. Тональность аспектов может быть разной. Каждый аспект выражается различными словами или словосочетаниями, которые называются аспектными терминами (АТ). В приведенном примере АТ являются слова *«сервису»*, *«интерьер»* и словосочетание *«говядина по-французски»*. Таким образом, можно выделить четыре основные подзадачи АОАТ [1]:

- извлечение аспектных терминов;
- категоризация аспектных терминов по заданным аспектам;
- определение тональности аспектных терминов;
- определение тональности по отношению к аспектам в целом.

Первое тестирование систем анализа в этой области проходило в рамках семинара SemEval-2014 [2]. Первым российским стало тестирование SentiRuEval в рамках конференции Dialogue-2015 [3]. В 2016 году SemEval стал многоязычным, включив дорожки для английского, русского и других языков [4].

В данной статье описывается программная система для решения первой подзадачи АОАТ – извлечения аспектных терминов на материале русского языка. Подзадача рассматривается как проблема разметки последовательности слов и решается с применением условных случайных полей (Conditional Random Fields, CRF). При этом для обучения модели используются векторные представления слов (ВПС), а также дополнительные морфологические и

словарные признаки. Эксперименты проводятся на корпусе отзывов, предоставленном организаторами международного тестирования SemEval-2016.

Статья организована следующим образом. Во втором разделе приведен краткий обзор исследований в области извлечения аспектных терминов. В третьем разделе описываются текстовые данные, предоставленные организаторами тестирования SemEval-2016. Четвертый раздел посвящен описанию предложенной программной системы. В пятом разделе описываются эксперименты и приводится сравнение метрик качества программной системы с официальными результатами SemEval-2016. Шестой раздел содержит анализ и классификацию ошибок. Седьмой раздел посвящен выводам и описанию перспективных направлений дальнейших исследований.

2. Обзор исследований в области АОАТ

Исследования в области АОАТ получили широкое развитие в большой степени благодаря открытым тестированиям SemEval [2], [4] и SentiRuEval [3]. В работе [5] приведен обзор решений участников тестирований последних лет. В работе [1] представлен подробный обзор подходов и методов решения всех подзадач АОАТ. Для решения подзадачи извлечения АТ, которая является предметом настоящего исследования, используются следующие подходы:

- извлечение аспектов с помощью методов машинного обучения с учителем (supervised learning) [6], [7], [8], [9], [11], [12];
- поиск часто встречающихся имен существительных и именных групп (frequent nouns and noun phrases) [13], [14], [16];
- извлечение аспектов на основе связи между мнением и объектом (relation-based methods) [13];
- извлечение аспектов с помощью тематического моделирования (topic modeling) [19], [20].

Подзадача извлечения АТ подразумевает нахождение слов или словосочетаний в предложении, по отношению к которым выражено мнение пользователя. Определенные слова могут с большей вероятностью сигнализировать о том, что следующие за ними слова входят в состав АТ. Поэтому подход к решению подзадачи извлечения АТ как к проблеме разметки последовательности с помощью модели машинного обучения условные случайные поля является одним из самых распространенных [6], [7], [8], [9], [11], [12].

В работе [6] для извлечения АТ авторы использовали CRF со статистическими и синтаксическими признаками. ВПС [10], ранее использованные для нахождения семантической близости, также широко применяются для задачи извлечения АТ в качестве признаков для CRF [11], [8].

Система, описанная в работе [11], показала лучший результат в тестировании систем извлечения АТ для русского языка SentiRuEval-2015. Авторы

используют семантическую близость слов. Каждому слову из тестовой выборки ставился в соответствие вектор из нейросетевой языковой модели. Слова помечались АТ в зависимости от близости по косинусной мере их ВПС, входящим в состав АТ в обучающей выборке.

В работе [8] описана программная система, занявшая второе место в решении подзадачи извлечения АТ на SemEval-2016 среди неограниченных обучающей выборкой систем. Признаковое описание для слов из последовательности составлялось следующим образом. Использовалось контекстное окно из пяти слов – текущего и по два слова слева и справа от него. Каждому слову из контекстного окна сопоставлялось ВПС из нейросетевой модели. Если ВПС не было найдено в модели, то система назначала слову первое найденное ВПС, сопоставленное соседнему слову слева или справа. Если предыдущим словам также не было назначено ВПС, то текущему слову сопоставлялся один из четырех специально введенных векторов для первых двух слов с начала и двух слов с конца предложения. Также признаковое описание дополнялось морфологическими и лексическими признаками.

Система, показавшая лучшие результаты на SemEval-2016 [12], также основана на CRF и ВПС. Авторы использовали выходные данные рекуррентной нейронной сети в качестве дополнительных признаков.

3. Текстовый корпус

Одна из задач международного тестирования SemEval-2016 была посвящена АОАТ и заключалась в определении мнений, выраженных в отзывах пользователей по отношению к определенным сущностям и их атрибутам. Для русского языка организаторами тестирования был предоставлен корпус отзывов о ресторанах. Отзывы этой предметной области могли содержать мнения о шести сущностях (ресторан, еда, напитки, атмосфера, обслуживание, местонахождение) и пяти их атрибутах (цены, качество, стиль, разное и в целом).

Пятая задача SemEval-2016 делилась на две подзадачи – АОАТ на уровне предложения и АОАТ на уровне текста. Первая подзадача подразумевала поиск мнений пользователей в рамках каждого предложения отзыва отдельно, а вторая подзадача – анализ всего отзыва целиком. Последняя задача – АОАТ на уровне текста – в данной работе не рассматривается. Первая подзадача включала три этапа:

- этап 1: определение категории аспекта (Aspect Category Detection, ACD). Цель этого этапа – найти все пары сущностей и атрибутов, по отношению к которым выражено мнение пользователя. В данной работе пара *сущность#атрибут* называется *аспектом*;
- этап 2: определение объекта, по отношению к которому выражено мнение (Opinion Target Expression, OTE). Данный этап подразумевал нахождение слова или словосочетания, использованного в тексте для

явного указания на сущность, по отношению к которой выражено мнение. В настоящей работе такие объекты называются *аспектными терминами* (АТ);

- этап 3: семантическая полярность (Sentiment Polarity). Каждой паре *сущность#атрибут* необходимо было сопоставить одну из следующих меток полярности: *позитивная*, *негативная* или *нейтральная*.

Предоставленный организаторами размеченный корпус отзывов состоит из обучающей и тестовой выборок. Выборки представляют собой XML-файлы, в которых каждый отзыв разделен на предложения. Для каждого предложения в обучающей выборке выделены АТ (слово или словосочетание с указанием граничных символов), их категории и полярности.

В Табл. 1 приведены характеристики корпуса. Количество отзывов, предложений, слов и АТ в обучающей выборке примерно втрое больше, чем в тестовой.

Табл. 1. Количественные характеристики отзывов SemEval-2016

Table 1. Quantitative characteristics of the reviews SemEval-2016

Характеристика корпуса	Обучающая выборка	Тестовая выборка
Количество отзывов	312	103
Количество предложений	3 548	1 168
Количество слов	40 094	13 181
Количество АТ	3159	972
однословные АТ	2 580	785
многословные АТ	579	187

Следует отметить, что количество АТ, являющихся словосочетаниями в выборках, значительно меньше, чем количество однословных АТ, и составляют соответственно лишь 19.3% и 18.2% от общего числа АТ в выборках. Подобная разница сказывается на качестве обучения модели и ее прогнозам по отношению к АТ, выраженным словосочетаниями (см. раздел 5).

В данной работе описано решение второго этапа (OTE) первой подзадачи (sentence-level ABSA) пятой задачи SemEval-2016, которая подразумевает извлечение АТ, т. е. определение аспектных слов или словосочетаний с указанием их граничных символов.

4. Описание программной системы

В настоящей работе подзадача извлечения АТ рассматривается как проблема разметки последовательностей слов. Идея данного подхода состоит в том, чтобы для каждого предложения построить последовательность признаковых описаний слов. В качестве модели машинного обучения с учителем была

выбрана модель условных случайных полей CRF и ее реализация для языка Python – *pyStruct* [21].

Для представления слова использовались ВПС, часть речи, присутствие слова в словаре терминов, являющихся АТ в обучающей выборке, наличие заглавной буквы в слове и синтаксическая связь с предком. Указанные признаки формируют признаковое описание каждого слова в последовательности.

Для решения подзадачи извлечения АТ программная система должна выполнять следующие функции:

- загрузка обучающей и тестовой выборок отзывов из XML-файлов;
- токенизация – разделение предложений на слова;
- генерация признакового описания для каждого слова обучающей и тестовой выборок;
- присвоение меток признаковым описаниям обучающей выборки;
- обучение модели CRF на последовательности признаковых описаний и меток из обучающей выборки;
- предсказание меток для признаковых описаний из тестовой выборки;
- сопоставление полученным меткам слов из тестовой последовательности для преобразования их в АТ;
- сохранение отзывов тестовой выборки вместе с АТ в XML-файл.

Токенизация каждого предложения проводилась с помощью морфологического анализатора *mystem* [22]. Получая на вход целое предложение, *mystem* возвращает массив слов вместе с леммой и грамматической информацией каждого слова. Лемма используется для поиска ВПС в нейросетевой языковой модели, а часть речи добавляется в признаковое описание.

Основой для создания признакового описания стал программный инструмент *word2vec*, предназначенный для построения ВПС, основанный на нейросетевых языковых моделях. Он способен определять семантическую близость слов путем максимизации косинусной меры сходства между векторами слов, находящихся в похожих контекстах [10].

Для обучения модели на вход ей подается последовательность всех признаковых описаний из обучающей выборки, которым сопоставлены соответствующие метки АТ. В системе использовалась распространенная схема BIO-разметки [23], где B – метка токена, с которого начинается АТ, I – метка токена, входящего в АТ, но не являющегося первым, O – метка токена, не входящего в состав АТ. При обработке тестовой последовательности признаковых описаний модель CRF должна сопоставить ей наиболее вероятную последовательность меток АТ.

В нашей системе использовались общедоступные нейросетевые модели для русского языка [24], разработанные для определения семантической близости

в тестировании Russian Semantic Similarity Evaluation (RUSSE) [25]. Модели создавались с помощью нескольких корпусов:

- *News* – корпус текстов российских новостей, собранных коммерческими новостными агрегаторами. После предобработки объем корпуса составил 1300 миллионов токенов.
- *Web* – корпус текстов, составленный из русскоязычных веб-страниц. Около 9 миллионов страниц были выбраны случайным образом из индекса крупнейших поисковых систем. После предобработки объем корпуса составил примерно 620 миллионов токенов.
- *Ruscorpora* – Национальный корпус русского языка [26]. После предобработки объем корпуса составил 107 миллионов токенов.
- *RuwikiRuscorpora* – корпус, содержащий *Ruscorpora* и русскоязычную Википедию. Размер корпуса – более 280 миллионов токенов.

При создании корпуса *News* была использована архитектура нейронной сети skip-gram, для остальных моделей – CBOW. Лемматизация производилась с помощью *mystem*, после чего из корпусов были убраны стоп-слова.

5. Эксперименты

В данном разделе рассмотрен ряд экспериментов, целью которых было составить признаковое описание, наиболее подходящее для модели CRF в задаче извлечения АТ. Метрикой оценки качества была F1-мера, полученная с помощью предоставленного организаторами SemEval-2016 программного инструмента для тестирования [4]. Проведенные эксперименты были направленны на решение следующих задач:

- определение оптимального размера контекстного окна ВПС и частей речи в признаковом описании;
- определение влияния дополнительных признаков (наличие слова в словаре, заглавная буква, синтаксическая связь) на качество системы;
- определение возможности совместного использования дополнительных признаков;
- поиск нейросетевой языковой модели, обеспечивающей наилучшее качество решения подзадачи извлечения АТ.

Целью первой серии экспериментов стало определение влияния контекста слова в признаковом описании на результаты системы. Контекстом считаются слова, расположенные слева и справа в окне заданного размера от анализируемого слова. В Табл. 2 показано изменение F1-меры после расширения контекстного окна. В признаковом описании использовались ВПС из нейросетевой модели на основе корпуса *News* (этот корпус оказался лучше, см. ниже) и части речи слов. В подзадаче извлечения АТ в рамках SemEval-2016 для русского языка участниками не было представлено методов,

которые превзошли бы базовый метод (baseline) организаторов, поэтому в Табл. 2 показано сравнение с официальным базовым методом [4].

Табл. 2. Расширение контекстного окна в признаковом описании слова

Table 2. Feature set context window expansion

Новый признак в признаковом описании	Точность, %	Полнота, %	F1-мера, %	Изменение F1-меры по отношению к базовому методу, %
Базовый метод (baseline)	57.66	43.06	49.31	–
ВПС	62.23	46.74	53.38	4.07
Часть речи слова	61.77	49.05	54.68	5.37
ВПС контекста [−1; +1]	65.66	52.83	58.55	9.24
Часть речи слов контекста [−1; +1]	69.81	62.18	65.77	16.44
ВПС контекста [−2; +2]	71.68	64.07	67.66	18.35
Часть речи слов контекста [−2; +2]	71.59	64.60	67.91	18.60
ВПС контекста [−3; +3]	71.42	65.65	68.41	19.10
Часть речи слов контекста [−3; +3]	**72.00**	**66.17**	**68.96**	**19.65**
ВПС контекста и часть речи [−4; +4]	71.15	65.54	68.23	18.92

Использование только ВПС в качестве признакового описания уже превосходит базовый метод, предоставленный организаторами SemEval-2016, по F1-мере на 4.07%. Максимальное изменение F1-меры достигается после добавления к признаковому описанию ВПС (+3.87%) и частей речи (+7.22%) соседних слов. По мере расширения контекстного окна F1-мера незначительно увеличивается и достигает максимального значения 68.96% (на 19.65% больше базового метода) при использовании в признаковом описании трех соседних слов слева и справа.

Размер контекстного окна значительно сказывается на времени генерации признакового описания и потребляемой памяти, поэтому для дальнейших экспериментов использовались окна [−1; +1] и [−2; +2].

В **Ошибка! Неверная ссылка закладки.** сравниваются результаты использования в признаковом описании дополнительных признаков. В отличие от предыдущей серии экспериментов, в табл. 4 приведено изменение F1-меры при добавлении нового признака к признаковому описанию контекстного окна [−1; +1] и исключению всех остальных признаков. Первый из исследуемых признаков – наличие в слове заглавной буквы. Второй – присутствие слова в словаре аспектной лексики. В словарь аспектной лексики

вошли 552 слова, являющиеся АТ в обучающей выборке. Третьим исследуемым признаком стала синтаксическая связь с предком. Для построения дерева зависимостей использовался программный инструмент MaltParser [27], обученный на корпусе SynTagRus [26].

Табл. 3. Сравнение влияния дополнительных признаков

Table 3. Comparison of the additional features impact

Исследуемый признак	Точность, %	Полнота, %	F1-мера, %	Изменение F1-меры, %
ВПС контекста и части речи [−1; +1]	69.81	62.18	65.77	–
Есть ли заглавная буква?	70.34	62.28	66.07	0.30
Содержится ли слово в словаре лексики АТ?	68.45	**64.28**	66.30	0.53
Синтаксическая связь слова.	**73.07**	60.71	**66.32**	**0.55**

Использование дополнительных признаков в признаковом описании привело к незначительному увеличению F1-меры. Приведенное в Табл. 4 совместное использование этих признаков также не смогло увеличить F1-меру больше, чем на 0.78%.

Табл. 4. Совместное использование дополнительных признаков

Table 4. Using of additional feature set

Используемые признаки	Точность, %	Полнота, %	F1-мера, %	Изменение F1-меры %
ВПС контекста и части речи [−1; +1]	**69.81**	62.18	65.77	–
Первые два признака.	68.52	**64.70**	**66.55**	**0.78**
Все три признака.	69.64	63.13	66.22	0.45

При этом отметим, то добавление первых двух дополнительных признаков к признаковому описанию, содержащему ВПС и частей речи из контекстного окна [−3; +3], привело к ухудшению оценки качества по F1-мере на 0.39%.

В Табл. 5 приведено сравнение производительности системы при использовании разных нейросетевых моделей из работы [24]. Признаковое описание элементов последовательности содержало векторные представления слова и слов из контекстного окна [−2; +2], части речи слова и слов из контекста, а также дополнительные признаки – наличие заглавной буквы и присутствие слова в корпусе слов, являющихся АТ.

Табл. 5. Сравнение результатов программной системы с базовым методом SemEval-2016

Table 5. Comparison between system results and SemEval-2016 baseline

Метод/модель	Точность, %	Полнота, %	F1-мера, %
Базовый метод (baseline)	57.66	43.06	49.31
Модель на основе *Ruscorpora*	72.29	63.86	67.81
На основе *News*	**72.97**	**65.23**	**68.88**
На основе R*uwikiruscorpora*	71.44	63.86	67.44
На основе *Web*	71.01	64.07	67.36

Все варианты CRF-классификатора, обученного на признаковых описаниях, основанных на общедоступных моделях word2vec [24], показали примерно равные результаты и превзошли официальный базовый метод по F1-мере. Несмотря на то, что word2vec-модели готовились для тестирования по определению семантической близости, содержащиеся в них ВПС оказались применимы и для задачи извлечения АТ в АОАТ. Лучшей из представленных word2vec-моделей в этой задаче оказалась модель *News*, обученная на корпусе новостных статей.

Как было отмечено выше, отзывы могут содержать АТ, являющиеся как отдельными словами, так и словосочетаниями. В Табл. 6 приведены сравнительные показатели извлечения АТ для обоих случаев.

Табл. 6. Сравнение производительности системы для АТ, являющихся отдельными словами, и АТ из словосочетаний

Table 6. System performance comparison between one word AT and multiword AT

АТ	Точность, %	Полнота, %	F1-мера, %
Однословные	76,11	73,05	74,55
Многословные	51,81	30,64	38,51

Как и предполагалось, недостаточное количество АТ из словосочетаний в обучающей выборке сказалось на извлечении подобных АТ в тестовой последовательности. Система смогла найти меньше трети АТ из словосочетаний, а точность составила лишь 51,81%, что в итоге дало F1-меру равную 38,51%. F1-мера для АТ, выраженных отдельными словами, оказалась почти в два раза выше и составила 74.55%.

6. Анализ ошибок

Можно выделить несколько типичных ошибок, допускаемых системой. Ниже приведены примеры отзывов из тестовой выборки, распознанные с ошибками, возможные причины ошибок и примеры из обучающей выборки. Все примеры взяты из корпуса отзывов тестирования SemEval-2016.

Главной проблемой системы является пропуск АТ, при этом пропускаются как отдельные слова, являющиеся АТ, так и словосочетания: система позволяет найти лишь 30,64% АТ из словосочетаний с точностью 51,81%. Для АТ из одного слова полнота равна 73,05%, а точность – 76,11%. Вот некоторые примеры предложений из отзывов, АТ в которых не были распознаны:

1) ***«Чурчхела** не понравилась, была замороженная и безвкусная».*
2) *«зато меня порадовал **тирамису** − вкусно, но подан некрасиво в стаканчике».*
3) ***«"Пицца Pro Тесто"** − поскольку первый раз, решили сразу понять "да" или "нет" заказав фирменную пиццу самого большого размера и опять же, не ошиблись».*
4) ***«Меню** продуманное и разнообразное, замечательные десерты, есть отличные предложения торжественных **блюд** для банкетов, на заказ сделают **торт** любой сложности».*
5) *«В такой **ресторан** хочется вернуться вновь».*
6) *«Вкусно! Заказывали и **итальянскую и японскую кухню**, всем понравилось».*

В первых двух примерах АТ являются редкие слова *чурчхела* и *тирамису*, которых нет в нейросетевой языковой модели *News*. Латиница из третьего примера также не содержится в нейросетевой языковой модели. Исправить подобные ошибки можно, использовав другую нейросетевую модель, содержащую в себе необходимые термины предметной области.

В четвертом и пятом примерах пропуск АТ *меню*, *блюд, торт* и *ресторан* может быть следствием специфики предметной области и обучающей выборки, представленной организаторами. В обучающей выборке классификатор получил множество примеров, где эти слова не являются АТ. Некоторые из них приведены далее. *«Милая девушка принесла меню»* – в данном примере АТ является слово *девушка*, а не *меню*. *«И даже когда пытались его подозвать, чтобы он собрал нам оставшиеся блюда с собой, он просто отворачивался и уходил в другой конец зала»* – в этом предложении пользователь не выражает какого-либо мнения к аспекту *еда* и, в частности, к блюдам, поэтому слово *блюда* не является АТ. *«Каравай и торт мы заказали там же, нам все привезли»* – нет выражения тональности по отношению слову *торт*, поэтому оно не является АТ. Слово *ресторан* в четвертом примере является АТ и принадлежит аспекту, характеризующему ресторан в целом, однако часто это слово употребляют для описания какого-то другого аспекта. Так, например, в предложении *«Для меня это просто идеальный интерьер небольшого ресторана»* – слово *ресторан* используется для описания аспекта *интерьер*. Для исправления подобного типа ошибок система должна учитывать не только близкий контекст текущего слова, но и все предложение.

Словосочетание *итальянскую и японскую кухню* из шестого примера должно было быть распознано как АТ, однако последовательности слов в обучающей выборке редко содержали АТ, начинающиеся на слово *итальянский*. Так,

например, в предложениях *«Кухня до итальянской не дотягивает − если только размером порции»* и *«Обычный совковый ресторанчик с итальянским названием»* слово *итальянский* не является АТ. Причиной подобных ошибок программной системы является недостаточное содержание АТ из словосочетаний в обучающей выборке и специфика подхода разметки последовательностей. Часто порядок слов в АТ может быть изменен без потери смысла, это означает, что классификатор должен присваивать одним и тем же словам различные метки, в зависимости от контекста. В отличие от шестого примера, в предложении из обучающей выборки *«Очень радует, что есть японская и итальянская кухни»* другой порядок слов и присутствует АТ *японская и итальянская кухни*, а не *итальянскую и японскую кухню*. Это значит, что обучаясь, классификатор присваивает слову *японский* метку начала АТ – *B*, а слову итальянский – метку слова, входящего в состав АТ, – *I*. При разметке классификатором тестовой последовательности эти метки оказываются неправильными.

Наряду с пропуском слов, являющихся АТ, встречаются случаи неверного распознавания классификатором слов как АТ. В следующих примерах показаны подобные случаи:

7) *«**Ресторан** на Энергетиков».*
8) *«**Ресторан** был пуст!».*
9) *«Оказывается пара за соседним столиком заказала такое же **блюдо** и оно было предназначено для них».*
10) *«Заказала горячую **закуску из овощей**».*

Слова *ресторан* и *блюдо* уже обсуждались в четвертом и пятом примерах. В седьмом, восьмом и девятом примерах продемонстрирована обратная ситуация. При обучении классификатор получал противоречивые примеры для слова *ресторан*. Поэтому при назначении метки слову из тестовой выборки большую роль сыграл контекст слов, и седьмой и восьмой примеры демонстрируют, что контекст слов в признаковом описании стал причиной ошибки классификатора. В этих примерах слово *ресторан* расположено на первом месте в предложении, следовательно, в признаковом описании левая часть контекстного окна пустая (признаковое описание каждого элемента последовательности должно быть одинаковой длины, поэтому левая часть, отведенная для соседних слева слов, заполняется нулями). Обучающая выборка содержит несколько предложений, в которых слово *ресторан*, находящееся на первом месте в предложении, является АТ. Вот некоторые из них: *«Ресторан замечательный!»*, *«Ресторан хорошо подходит для романтического ужина»*. В отличии от примеров 7–10, все перечисленные предложения содержат мнения пользователей. Для того, чтобы учитывать подобные случаи, можно использовать подход извлечения АТ на основе связи между мнением и объектом [1].

В примерах 11–12 приведены примеры АТ из словосочетаний, в которых программная система вместо выделения одного АТ, выраженного словосочетанием, распознала несколько АТ из одиночных слов:

11) *«Интерьер супер, очень удобные **кабинки с мягкими подушками**».*
12) *«Понравились **с креветкой, филадельфией и овощами**».*

В примере 11 системой были выделены АТ *кабинки* и *подушками*. В примере 12 отдельными АТ стали все три слова – *креветкой*, *филадельфией*, *овощами*. Обучающая выборка содержала мало примеров с этими словами. Классификатор мог допустить ошибку из-за недообучения или сходства ВПС этих слов с представлениями слов, часто являющимися АТ.

В примерах 13–14 приведены редкие ошибки, для обработки которых необходимо использование дополнительных подходов к извлечению АТ:

13) *«Цены не кусаются, учитывая что это центр города и сравниваю с другими **суши барами** =)».*
14) *«Все оказалось вкусным, салаты, суп, колбаски из баранины – просто **песня**».*

Пример 13 содержит сравнение цен объекта с другими *суши барами*. Сравнение сложно обработать с помощью машинного обучения, для обработки подобных отзывов может быть применен подход, основанный на правилах.

В примере 14 классификатор ошибочно определил слово *песня* как АТ. В данном контексте словосочетание *просто песня* является оценочной фразой, выраженной идиомой, т. е. словосочетанием, значение которого напрямую не следует из значений составляющих его слов. Идиомы могут выражать как позитивные, так и негативные эмоции, и, как правило, достаточно сильные. Обработка подобных словосочетаний требует предварительно собранного списка всевозможных идиом.

7. Заключение

В рамках данного исследования была разработана программная система для извлечения АТ в АОАТ, основанная на модели CRF. В ходе экспериментов было составлено оптимальное признаковое описание, включающее ВПС из контекстного окна [–3; 3] и их части речи. При тестировании на корпусе отзывов семинара SemEval-2016 программная система показала качество 68.96% по F1-мере. Данный результат превосходит базовый метод, предоставленный организаторами тестирования. Анализ ошибок показал, что подобный подход не всегда обрабатывает ситуации, когда одно и то же слово зависит от контекста и может являться или не являться АТ, поэтому направлениями дальнейших исследований будут обработка подобных исключений и объединение различных подходов к решению подзадачи извлечения АТ.

Список литературы

[1]. Liu B., Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1), 2012, pp. 1–167.

[2]. Pontiki M., Galanis D., Pavlopoulos J., Papageorgiou H., Androutsopoulos I., Manandhar S. Semeval-2014 task 4: Aspect based sentiment analysis. Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), 2014, pp. 27-35.

[3]. Loukachevitch N., Blinov P., Kotelnikov E., Rubtsova Y., Ivanov V., Tutubalina E. SentiRuEval: Testing Object-oriented Sentiment Analysis Systems in Russian. Proceedings of the 21st International Conference on Computational Linguistics (Dialog-2015), 2015, volume 2, pp. 12–24.

[4]. Pontiki M., Galanis D., Papageorgiou H., Androutsopoulos I., Manandhar S., AL-Smadi M., Al-Ayyoub M., Zhao Y., Qin B., De Clercq O., Hoste V., Apidianaki M., Tannier X., Loukachevitch N., Kotelnikov E., Bel N., Zafra S. M. J., Eryigit G. Semeval-2016 task 5: Aspect based sentiment analysis. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 19–30.

[5]. Андрианов И., Майоров В., Турдаков Д. Современные методы аспектно-ориентированного анализа эмоциональной окраски. Труды ИСП РАН, том 27, вып. 5, 2015 г., стр. 5-22. DOI: 10.15514/ISPRAS-2015-27(5)-1.

[6]. Ivanov V., Tutubalina E., Mingazov N., Alimova I. Extracting Aspects, Sentiment and Categories of Aspects in User Reviews about Restaurants and Cars. Proceedings of the 21st International Conference on Computational Linguistics (Dialog-2015), 2015, volume 2, pp. 46–57.

[7]. Jakob N., Gurevych I., Extracting opinion targets in a single-and cross-domain setting with conditional random fields, Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, 2010, pp. 1035-1045.

[8]. Xenos D., Theodorakakos P., Pavlopoulos J., Malakasiotis P., Androutsopoulos I. AUEB-ABSA at SemEval-2016 Task 5: Ensembles of Classifiers and Embeddings for Aspect Based Sentiment Analysis. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 312–317.

[9]. Hamdan H. SentiSys at SemEval-2016 Task 5: Opinion Target Extraction and Sentiment Polarity Detection. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 350–355.

[10]. Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, 2013, pp. 3111–3119.

[11]. Blinov P. D., Kotelnikov E. V. Semantic Similarity for Aspect-Based Sentiment Analysis. Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015, volume 2, pp. 36–45.

[12]. Toh Z., Su J. NLANGP at SemEval-2016 Task 5: Improving Aspect Based Sentiment Analysis using Neural Network Features. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 282–288.

[13]. Hu M., Liu B. Mining and summarizing customer reviews. Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, 2004, pp. 168–177.

[14]. Popescu A. M., Nguyen B., Etzioni O. OPINE: Extracting product features and opinions from reviews. Proceedings of HLT/EMNLP on interactive demonstrations, 2005, pp. 32–33.

[15]. Turney P. D. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. Proceedings of the 12th European Conference on Machine Learning, 2001, pp. 491–502.

[16]. Scaffidi C., Bierhoff K., Chang E., Felker M., Ng H., Jin C. Red Opal: product-feature scoring from reviews. Proceedings of the 8th ACM conference on Electronic commerce, 2007, pp. 182–191.

[17]. Hofmann T. Probabilistic latent semantic indexing. Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, 1999, pp. 50–57.
[18]. Blei D. M., Ng A. Y., Jordan M. I. Latent Dirichlet Allocation. Journal of machine Learning research, 2003, pp. 993-1022.
[19]. Mukherjee A, Liu B. Aspect extraction through semi-supervised modeling. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 2012, volume 1, pp. 339-348.
[20]. Titov I., McDonald R., Modeling online reviews with multi-grain topic models. Proceedings of the 17th international conference on World Wide Web. ACM, 2008, pp 111–120.
[21]. Müller A.C., Behnke S. PyStruct: learning structured prediction in python. Journal of Machine Learning Research 15(1), 2014, pp. 2055–2060.
[22]. Segalovich I. A Fast Morphological Algorithm with Unknown Word Guessing Induced by a Dictionary for a Web Search Engine. In MLMTA, 2003, pp. 273–280.
[23]. Ramshaw L. A., Marcus M. P. Text chunking using transformation-based learning. Natural language processing using very large corpora, Springer Netherlands, 1999, pp. 157-176.
[24]. Kutuzov A, Andreev I. Texts in, meaning out: neural language models in semantic similarity task for Russian. Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015, volume 2, pp. 133–144.
[25]. Panchenko A., Loukachevitch N. V., Ustalov D., Paperno D., Meyer C. M., Konstantinova N. Russe: The first workshop on russian semantic similarity. Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015, volume 2, pp. 89-105.
[26]. Plungian V. A. Why we make Russian National Corpus? Otechestvennye Zapiski 2, 2005.
[27]. Sharov S, Nivre J. The proper place of men and machines in language technology. Processing Russian without any linguistic knowledge. Proceedings of the 17th International Conference on Computational Linguistics Dialog-2011, 2011, pp. 657–670.

Aspect term extraction based on word embedding and conditional random fields

D.O. Mashkin <danil.mashkin@gmail.com>
E.V. Kotelnikov <kotelnikov.ev@gmail.com>
Vyatka State University,
36, Moskovskaya, Kirov, 610000, Russia.

Abstract. There are many sites in the Internet that allow users to share their opinions and write reviews about all kinds of goods and services. These views may be useful not only for other users, but also for companies which want to track their own reputation and to receive timely feedback on their products and services. The most detailed statement of the problem in this area is an aspect-based sentiment analysis, which determines the user attitude not only to the object as a whole, but also to its individual aspects. In this paper we consider the solution of subtask of aspect terms extraction in aspect-based sentiment analysis. A review of research in this area is given. The subtask of aspect terms extraction is considered as a problem of sequence labeling; to solve it we apply the model of conditional random fields (CRF). To create the sequence feature description, we use distributed representations of words derived from neural network models for the Russian language and parts of speech of the analyzed words. The stages of the aspect terms extraction software system are shown. The experiments with the developed software system were carried out on the corpus of labeled reviews of restaurants, created in the International Workshop on Semantic Evaluation (SemEval-2016). We describe the dependence of the quality of aspect terms extraction subtask on various neural network models and the variations of feature descriptions. The best results (F1-measure = 69%) are shown by a version of the system, which takes into account the context and the parts of speech. This paper contains a detailed analysis of errors made by the system, as well as suggestions on possible options for their correction. Finally, future research directions are presented.

Keywords: aspect-based sentiment analysis; sentiment analysis; aspect term extraction; machine learning; sequential labeling;, word embedding; word2vec; SemEval-2016.

DOI: 10.15514/ISPRAS-2016-28(6)-16

For citation: Mashkin D.O., Kotelnikov E.V. Aspect term extraction based on word embedding and conditional random fields. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016. pp. 223-240 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-16

References

[1]. Liu B., Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1), 2012, pp. 1–167.
[2]. Pontiki M., Galanis D., Pavlopoulos J., Papageorgiou H., Androutsopoulos I., Manandhar S. Semeval-2014 task 4: Aspect based sentiment analysis. Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), 2014, pp. 27-35.

[3]. Loukachevitch N., Blinov P., Kotelnikov E., Rubtsova Y., Ivanov V., Tutubalina E. SentiRuEval: Testing Object-oriented Sentiment Analysis Systems in Russian. Proceedings of the 21st International Conference on Computational Linguistics (Dialog-2015), 2015, volume 2, pp. 12–24.

[4]. Pontiki M., Galanis D., Papageorgiou H., Androutsopoulos I., Manandhar S., AL-Smadi M., Al-Ayyoub M., Zhao Y., Qin B., De Clercq O., Hoste V., Apidianaki M., Tannier X., Loukachevitch N., Kotelnikov E., Bel N., Zafra S. M. J., Eryigit G. Semeval-2016 task 5: Aspect based sentiment analysis. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 19–30.

[5]. Andrianov I., Mayorov V., Turdakov D. Modern Approaches to Aspect-Based Sentiment Analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 5–22 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-1.

[6]. Ivanov V., Tutubalina E., Mingazov N., Alimova I. Extracting Aspects, Sentiment and Categories of Aspects in User Reviews about Restaurants and Cars. Proceedings of the 21st International Conference on Computational Linguistics (Dialog-2015), 2015, volume 2, pp. 46–57.

[7]. Jakob N., Gurevych I., Extracting opinion targets in a single-and cross-domain setting with conditional random fields, Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, 2010, pp. 1035-1045.

[8]. Xenos D., Theodorakakos P., Pavlopoulos J., Malakasiotis P., Androutsopoulos I. AUEB-ABSA at SemEval-2016 Task 5: Ensembles of Classifiers and Embeddings for Aspect Based Sentiment Analysis. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 312–317.

[9]. Hamdan H. SentiSys at SemEval-2016 Task 5: Opinion Target Extraction and Sentiment Polarity Detection. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 350–355.

[10]. Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, 2013, pp. 3111–3119.

[11]. Blinov P. D., Kotelnikov E. V. Semantic Similarity for Aspect-Based Sentiment Analysis. Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015, volume 2, pp. 36–45.

[12]. Toh Z., Su J. NLANGP at SemEval-2016 Task 5: Improving Aspect Based Sentiment Analysis using Neural Network Features. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 282–288.

[13]. Hu M., Liu B. Mining and summarizing customer reviews. Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, 2004, pp. 168–177.

[14]. Popescu A. M., Nguyen B., Etzioni O. OPINE: Extracting product features and opinions from reviews. Proceedings of HLT/EMNLP on interactive demonstrations, 2005, pp. 32–33.

[15]. Turney P. D. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. Proceedings of the 12th European Conference on Machine Learning, 2001, pp. 491–502.

[16]. Scaffidi C., Bierhoff K., Chang E., Felker M., Ng H., Jin C. Red Opal: product-feature scoring from reviews. Proceedings of the 8th ACM conference on Electronic commerce, 2007, pp. 182–191.

[17]. Hofmann T. Probabilistic latent semantic indexing. Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, 1999, pp. 50–57.
[18]. Blei D. M., Ng A. Y., Jordan M. I. Latent Dirichlet Allocation. Journal of machine Learning research, 2003, pp. 993-1022.
[19]. Mukherjee A, Liu B. Aspect extraction through semi-supervised modeling. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 2012, volume 1, pp. 339-348.
[20]. Titov I., McDonald R., Modeling online reviews with multi-grain topic models. Proceedings of the 17th international conference on World Wide Web. ACM, 2008, pp 111–120.
[21]. Müller A.C., Behnke S. PyStruct: learning structured prediction in python. Journal of Machine Learning Research 15(1), 2014, pp. 2055–2060.
[22]. Segalovich I. A Fast Morphological Algorithm with Unknown Word Guessing Induced by a Dictionary for a Web Search Engine. In MLMTA, 2003, pp. 273–280.
[23]. Ramshaw L. A., Marcus M. P. Text chunking using transformation-based learning. Natural language processing using very large corpora, Springer Netherlands, 1999, pp. 157-176.
[24]. Kutuzov A, Andreev I. Texts in, meaning out: neural language models in semantic similarity task for Russian. Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015, volume 2, pp. 133–144.
[25]. Panchenko A., Loukachevitch N. V., Ustalov D., Paperno D., Meyer C. M., Konstantinova N. Russe: The first workshop on russian semantic similarity. Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015, volume 2, pp. 89-105.
[26]. Plungian V. A. Why we make Russian National Corpus? Otechestvennye Zapiski 2, 2005.
[27]. Sharov S, Nivre J. The proper place of men and machines in language technology. Processing Russian without any linguistic knowledge. Proceedings of the 17th International Conference on Computational Linguistics Dialog-2011, 2011, pp. 657–670.