ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 8
N 6
2017

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.7 DOI: 10.17587/prin.8.243-249

А. А. Андреев, магистрант, e-mail: andreev@cs.petrsu.ru,
Петрозаводский государственный университет

Обобщенная графовая модель виртуальных частных сетей в коммуникационной инфраструктуре локального поставщика сетевых услуг

Для разработки и сопровождения программного обеспечения процессов управления сетевой инфраструктуры на разных уровнях активно используются ее описания в виде графовых моделей. В настоящей статье предложена графовая модель для описания структуры туннелируемых соединений виртуальной частной сети, развернутой в рамках сети локального поставщика сетевых услуг (ПСУ). Модель является расширением обобщенной графовой модели структуры физического, канального и сетевого уровней ПСУ. Также приведены формально обоснованные с использованием свойств рассматриваемой модели методы автоматизированного обнаружения и построения в графе виртуальных частных сетей на основе данных, получаемых из сети.

Ключевые слова: *сетевое управление, граф коммуникационной инфраструктуры, виртуальные частные сети, VPN, графовая модель*

Введение

Современные коммуникационные инфраструктуры (далее Сети) локальных поставщиков сетевых услуг (лПСУ) имеют тенденцию к увеличению масштабов, что приводит к необходимости использования сетевыми администраторами сложных технологий структуризации Сетей. В число таких технологий входят виртуальные частные сети (VPN — *virtual private networks*) — защищенные от несанкционированного доступа логические Сети, развернутые поверх других Сетей [1]. Их используют для обеспечения безопасного подключения удаленных пользователей к Сети и в целях объединения сегментов Сети, удаленных физически (например, филиалов организации) [1, 2].

При внедрении и сопровождении виртуальных частных сетей (ВЧС) в Сети лПСУ (далее — Сеть с ВЧС) встает ряд задач сетевого управления, направленных на обеспечение качества сетевых услуг: проектирование и масштабирование топологии ВЧС; обеспечение надежности, отказоустойчивости и достаточной пропускной способности соединений, используемых ВЧС, и т. д. [1—3].

Решение большинства перечисленных задач требует наличия полного и детального описания структуры Сети с ВЧС (как логической, так и физической). Наиболее распространенной формой представления такой структуры является граф, вершины которого соответствуют сетевым устройствам, портам и конечным точкам протоколов передачи данных, а ребра — связям иерархии и передачи данных.

Графовые модели, которые используются в работе [4] для обеспечения качества обслуживания в Сети с ВЧС и в работе [5] для анализа производительности и возможности масштабирования топологии ВЧС, учитывают только физическую структуру Сети и пропускную способность соединений. Игнорирование возможности блокировки связей на канальном уровне, применения агрегирования каналов, политик маршрутизации и виртуальных локальных сетей не позволяет учесть все возможные, невозможные и резервные пути передачи данных между сторонами ВЧС, а также изменение этих путей при перестроении топологии ВЧС. Модель, используемая в работе [6] для построения оптимальной структуры ВЧС с точки зрения обеспечения качества обслуживания и сокращения числа соединений в ВЧС, описывает структуру Сети без учета возможности присутствия в ней виртуальных локальных сетей, что сильно ограничивает класс современных Сетей, в которых модель может быть применена. Модели, используемые в работе [7—9] для автоматизации построения графа структуры Сети, не предоставляют возможности описания ВЧС, что не позволяет использовать эти модели в Сетях с ВЧС.

На кафедре Информатики и математического обеспечения Петрозаводского государственного университета в рамках разработки экспериментальной платформы Nest для исследования моделей и методов сетевого управления [10] проводится работа по автоматизации построения графа структуры Сети. В рамках этой работы были разработаны обобщенная графовая модель структуры Сети на физическом, ка-

нальном и сетевом уровнях модели OSI (*open systems interconnection basic reference model)*, а также алгоритм автоматизированного построения графа структуры Сети с использованием элементов модели [11—13]. На настоящий момент модель позволяет подробно описать физическую и логическую структуры Сети без ВЧС, что лишает ее недостатков перечисленных моделей.

Целью данной работы является расширение модели, описанной в работе [12], для отражения элементов и связей ВЧС в Сетях лПСУ, а также модификация процесса построения графа структуры Сети для учета ВЧС.

Классификация структур ВЧС

Рассмотрим технологии ВЧС и классифицируем их с точки зрения структуры для того, чтобы определить необходимые для описания ВЧС элементы модели.

Виртуальные частные сети реализуются с помощью технологий управления доступом к Сети (аутентификация, авторизация, шифрование передаваемых данных) и технологий установления логических сетевых соединений, в основе которых лежит туннелирование [2]. В рамках туннелирования пакеты одного протокола инкапсулируются в пакеты другого протокола того же или более высокого уровня модели OSI. Последние передаются через Сеть-посредник (которая находится между соединяемыми сегментами Сети) и декапсулируются с сохранением всех заголовков, адресов назначения и отправителя. Технологии туннелирования могут быть классифицированы по уровню несущего протокола, по назначению, по уровню инкапсулируемого протокола, по числу соединяемых сторон [1, 2, 14, 15].

По уровню несущего протокола выделяют туннели канального (например, протоколы PPPoE, PPPoA), сетевого (GRE, IPSec), транспортного (L2TP, PPTP), сеансового (SSL/TLS в рамках групп технологий OpenVPN, SSTP) и прикладного (SSH) уровней.

По назначению туннелей выделяют те, которые применяют в ВЧС удаленного доступа для связи сегмента Сети с одной вычислительной машиной (L2TP, PPTP, OpenVPN и др.), а также туннели в рамках "межучасточной" (*site-to-site*) ВЧС для соединения двух или более сегментов Сетей (GRE, IPSec и др.).

По уровню инкапсулируемого протокола выделяют туннели канального и сетевого уровней. Туннели канального уровня (L2TP, PPTP, MPPE и др.) соединяют два или более широковещательных домена в один, как показано на рис. 1, где по итогам создания туннеля в одном домене оказываются коммутаторы, изображенные квадратными фигурами со стрелками.

Туннели сетевого уровня (IPSec, GRE, OpenVPN и др.) соединяют две IP-подсети в одну, как показано на рис. 2, где по итогам создания туннеля в одной подсети оказываются маршрутизаторы, изображенные круглыми фигурами со стрелками.

По числу соединяемых сторон выделяют туннели между двумя сегментами Сети (GRE, IPSec, PPTP,

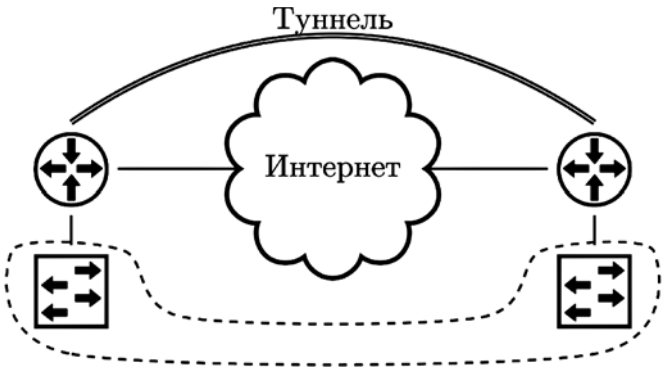

Рис. 1. Туннель канального уровня

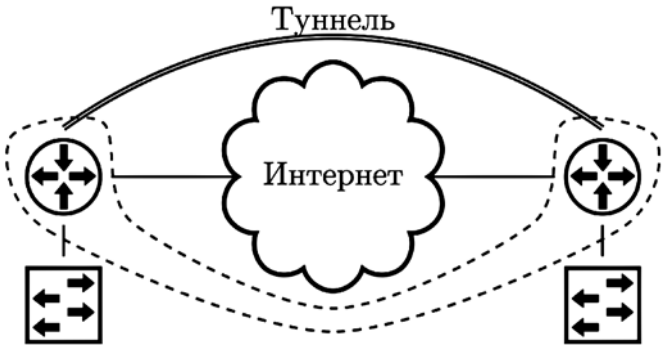

Рис. 2. Туннель сетевого уровня

L2TP и др.) и несколькими сегментами (mGRE, OpenVPN и др.).

Следует отметить, что туннели с несущим протоколом прикладного уровня и клиент-серверные варианты туннелирования на сеансовом уровне применяют только для передачи данных между отдельными приложениями, поэтому они не оказывают влияния на общую структуру Сети. При этом уровень несущего протокола не зависит от структуры Сети, необходима лишь поддержка этого протокола связываемым оборудованием. Технологии соединения нескольких сторон автоматизируют создание и управление двусторонними туннелями и имеют в своей основе одну или несколько технологий двусторонних туннелей (например, в основе технологии mGRE лежат GRE и IPSec). В обоих классах туннелей в классификации по назначению связь строится между двумя соединяемыми устройствами. Построение всех типов туннелей требует наличия на соединяемых устройствах точек инкапсуляции/декапсуляции пакетов, в качестве которых используются виртуальные интерфейсы. Для построения туннелей и канального, и сетевого уровней возникает необходимость в наличии механизмов коммуникации между соединяемыми интерфейсами на уровне не ниже сетевого через реальные интерфейсы.

Таким образом, с точки зрения структуры любая ВЧС основывается на сетевом туннеле, который состоит из двух сегментов Сети, подлежащих объединению, двух устройств в обоих сегментах и интерфей-

сов этих устройств, одной Сети-посредника, поверх которой осуществляется соединение. При этом туннели могут объединять либо широковещательные домены устройств, либо их подсети.

Модель структуры ВЧС

Исходя из отмеченных выше особенностей технологий ВЧС и туннелирования приведем требования к расширению модели из работы [12] для учета ВЧС в рамках графа структуры Сети.

Рассматриваемая модель предназначена для описания Сети единого лПСУ. Однако Сети чаще всего имеют соединения с Сетями других ПСУ, подробное описание которых не требуется для задач сетевого управления. Такие Сети будем называть внешними для описываемого лПСУ. Сетью-посредником в рамках туннелирования может быть внешняя Сеть, к которой подключены оба соединяемых туннелем сегмента. Таким образом, модель должна предоставлять возможность описания соединений с сегментами внешних Сетей. Наличие точек инкапсуляции/декапсуляции пакетов требует, чтобы модель предоставляла возможность описания интерфейсов, осуществляющих туннелирование, в том числе виртуальных.

Модель должна предоставлять возможность описания связей между элементами Сети в рамках туннелей сетевого и канального уровней, а также учитывать существенные различия структуры туннелей этих классов.

Далее при описании расширения модели будет использована нотация, введенная в работе [12]. Рассмотрим неориентированный граф структуры Сети G.

Введем множество NS сегментов внешних Сетей и переопределим отношения ассоциации так, что в них кроме устройств множества D могут участвовать внешние Сети. Определим отношение ассоциации физического уровня A^1 на множестве $P \cup D \cup NS$, где P — множество физических портов; отношение ассоциации канального уровня A^2 на $I^2 \cup D \cup NS$, где I^2 — множество канальных интерфейсов; отношение ассоциации сетевого уровня A^3 на $I^3 \cup D \cup NS$, где I^3 — множество сетевых интерфейсов. Интерфейсы множеств I^2 и I^3 или порты множества P ассоциированы с сегментом Сети из множества NS, если они ассоциированы с пограничными устройствами сегмента. Интерфейсы канального уровня, ассоциированные с сегментом из NS, находятся в отношении коммутации F^2, если между ними есть путь канального уровня в этом сегменте. Интерфейсы сетевого уровня, ассоциированные с сегментом из NS, находятся в отношении маршрутизации F^3, если между ними есть путь сетевого уровня в этом сегменте.

Пример описания соединения с внешней Сетью изображен на рис. 3, на котором два устройства d_1, $d_2 \in D$ соединены с сегментом $ns_1 \in NS$. Здесь p_{11}, p_{21}, p_{31}, p_{41} — это порты; (p_{11}, i_1), (p_{21}, i_2), (p_{31}, i_1), (p_{41}, i_2) — интерфейсы канального уровня; (n_1, h_1), (n_1, h_2), (n_2, h_1), (n_2, h_2) — интерфейсы сетевого уровня. На рис. 3 и далее приняты следующие изображения

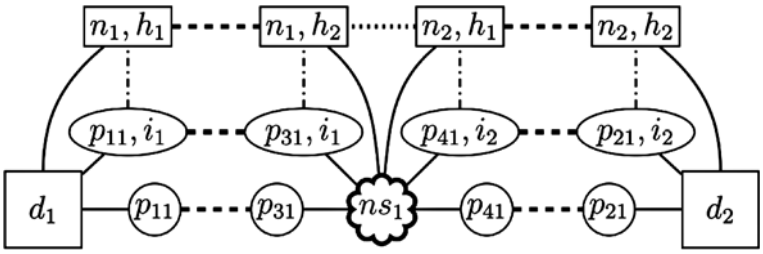

Рис. 3. Соединение с внешней Сетью

для вершин: квадрат — для устройств, облако — для внешних Сетей, круг — для портов, овал — для канальных и прямоугольник — для сетевых интерфейсов. Ребра ассоциации показаны сплошными линиями, ребра коммутации и маршрутизации — полужирными пунктирными линиями. Ребра соединения на физическом уровне L^1, канальном уровне L^2, сетевом уровне L^3 показаны штриховыми линиями. Вхождение канального интерфейса в сетевой интерфейс показано штрихпунктирными линиями.

Назовем порт $p \in P$ виртуальным, если о нем известно, что он имитируется программными средствами ассоциированного с ним устройства. Определим $VP \subset P$ как множество всех виртуальных портов. Введем множество $VI^2 \subset I^2$ интерфейсов канального уровня, построенных на основе только виртуальных портов из множества VP, и множество $VI^3 \subset I^3$ интерфейсов сетевого уровня, построенных на основе только канальных интерфейсов из множества VI^2. Обозначим множество $I^3 \backslash VI^3$ канальных интерфейсов, построенных на основе только реальных физических портов, как RI^3.

Введем на множестве $VI^2 \cup RI^3$ бинарное симметричное отношение туннелирования канального уровня T^2 со следующей интерпретацией: если канальный интерфейс $li \in VI^2_d$ и сетевой интерфейс $ni \in RI^3_d$, ассоциированные с одним устройством $d \in D$, находятся в отношении T^2, то исходящие из li пакеты данных инкапсулируются с помощью туннельного протокола и передаются через ni, а входящие пакеты принимаются на интерфейс ni и декапсулируются. Туннельным соединением назовем отношение соединения между двумя интерфейсами одного уровня, каждый из которых находится в отношении туннелирования.

Основываясь на перечисленных особенностях коммуникации между соединяемыми туннелем интерфейсами, сформулируем условие наличия туннельного соединения на канальном уровне.

Условие 1. Рассмотрим интерфейсы канального уровня li_1, $li_2 \in VI^2$ и сетевого уровня ni_1, $ni_2 \in RI^3$. Если li_1 и li_2 состоят в отношении T^2 с ni_1 и ni_2 соответственно, а li_1 состоит в отношении L^2 с li_2, то должен существовать путь сетевого уровня между ni_1 и ni_2.

На рис. 4 изображен пример описания структуры туннеля канального уровня с помощью отношения T^2: устройства d_1 и d_2 соединяются туннелем через сегмент Сети ns_1 с помощью виртуальных портов vp_{11}, vp_{21} и интерфейсов (vp_{11}, i_3), (vp_{21}, i_3). Последние

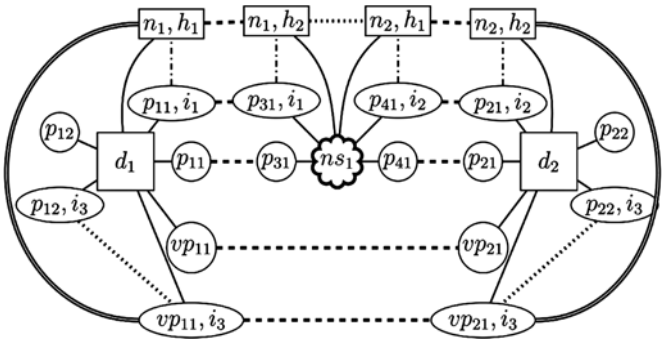

Рис. 4. Структура туннеля канального уровня

состоят в отношении T^2 с сетевыми интерфейсами (n_1, h_1) и (n_2, h_2) соответственно. Ребра, соответствующие связям туннелирования, изображены двойной сплошной линией.

Введем на множестве I^3 бинарное симметричное отношение туннелирования сетевого уровня T^3 со следующей интерпретацией: если сетевые интерфейсы $ni_1 \in VI_d^3$ и $ni_2 \in RI_d^3$, ассоциированные с устройством $d \in D$, находятся в отношении T^3, то исходящие из ni_1 пакеты данных инкапсулируются с помощью туннельного протокола и передаются через ni_2, а входящие пакеты принимаются на интерфейс ni_2 и декапсулируются.

Основываясь на приведенных особенностях коммуникации между соединяемыми туннелем интерфейсами, сформулируем условие наличия туннельного соединения на сетевом уровне.

Условие 2. Рассмотрим пары интерфейсов сетевого уровня ni_1, $ni_2 \in VI^3$ и ni_3, $ni_4 \in RI^3$. Если ni_1 и ni_2 состоят в отношении T^3 с ni_3 и ni_4 соответственно, а ni_1 находится в отношении L^3 с ni_2, то должен существовать путь сетевого уровня между ni_3 и ni_4.

На рис. 5 изображен пример описания структуры туннеля сетевого уровня с помощью отношения T^3: устройства d_1 и d_2 соединяются туннелем через сегмент Сети ns_1 с помощью виртуальных портов vp_{11}, vp_{21}, канальных интерфейсов (vp_{11}, i_3), (vp_{21}, i_3) и сетевых интерфейсов (n_3, h_1), (n_3, h_2). Последние

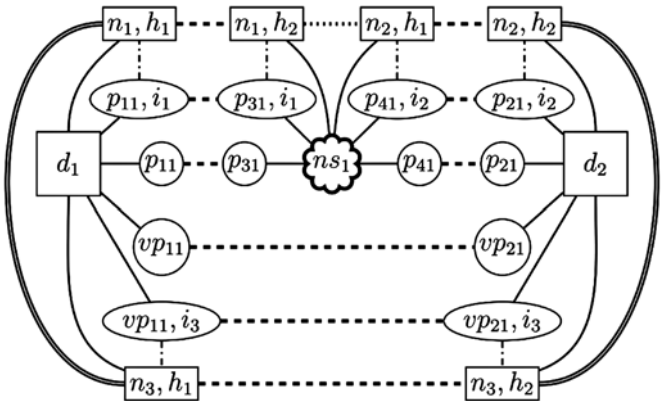

Рис. 5. Структура туннеля сетевого уровня

состоят в отношении T^3 с сетевыми интерфейсами (n_1, h_1) и (n_2, h_2) соответственно.

Основываясь на приведенных выше особенностях туннелей, сформулируем условия наличия соединения между виртуальными портами и интерфейсами.

Условие 3. Если порты $p_1 \in VP_{d1}$ и $p_2 \in VP_{d2}$ устройств d_1, $d_2 \in D$ состоят в отношении L^1, то у этих устройств должны существовать состоящие в отношении L^2 интерфейсы канального уровня $li_1 \in VI_{d1}^2$ и $li_2 \in VI_{d2}^2$ такие, что $p_1 \in li_1$, $p_2 \in li_2$.

Условие 4. Если интерфейсы $li_1 \in VI_{d1}^2$ и $li_2 \in VI_{d2}^2$ устройств d_1, $d_2 \in D$ состоят в отношении L^2 и нет ни одного основанного на них интерфейса сетевого уровня, то должны существовать такие интерфейсы сетевого уровня $ni_1 \in RI_{d1}^3$ и $ni_2 \in RI_{d2}^3$, что (li_1, ni_1) $\in$ $\in T^2$ и (li_2, ni_2) $\in T^2$.

Условие 5. Если интерфейсы $ni_1 \in VI_{d1}^3$ и $ni_2 \in VI_{d2}^3$ устройств d_1, $d_2 \in D$ состоят в отношении L^3, то должны существовать такие интерфейсы $ni_3 \in RI_{d1}^3$ и $ni_4 \in RI_{d2}^3$, что (ni_1, ni_3) $\in T^3$ и (ni_2, ni_4) $\in T^3$.

Представленные выше условия устанавливают наличие и порядок взаимодействия между портами и интерфейсами, участвующими в построении туннельного соединения.

Таким образом, структура Сети с ВЧС может быть описана с помощью графа структуры Сети с ВЧС $G' = \langle V \cup NS,\ E \cup T^2 \cup T^2 \rangle$, множество вершин которого включает сегменты внешних Сетей и виртуальные порты, а множество ребер — связи туннелирования канального и сетевого уровней.

Автоматизация построения структуры ВЧС в рамках графа структуры Сети

Процесс построения графа структуры Сети из работы [12] состоит из четырех этапов: сбор с сетевых устройств данных об элементах Сети; построение фрагментов графа, существование которых напрямую следует из анализа собранных данных; выявление элементов Сети, сведения о которых отсутствуют в собранных данных; построение ребер соединения на физическом, канальном и сетевом уровнях. Для отражения ВЧС в графе Сети структуры необходимо внести изменения в перечисленные этапы.

Для построения графа необходимо наличие сведений о виртуальных портах и связях туннелирования, которые могут быть получены на первом этапе процесса с помощью простого протокола сетевого управления (SNMP) из баз информации управления устройств. Данные о виртуальных портах присутствуют в базе IF-MIB, о связях туннелирования — в базе TUNNEL-MIB (для GRE, IPSec и др.), L2TP-MIB (для L2TP) и в других базах, специфичных для протокола или производителя оборудования.

На втором этапе необходимо проводить построение виртуальных портов (множество VP) и интерфейсов (множества VI^2, VI^3), а также связей

туннелирования $(T^2$ и $T^3)$. Данные о соединениях между виртуальными портами или интерфейсами для устройств не отличаются от обычных связей, поэтому могут быть обнаружены без изменения этапов процесса с помощью информации из баз LLDP-MIB, CDP-MIB, BRIDGE-MIB, IP-MIB.

В случаях, когда данные о туннелированнии отсутствуют во входных данных (см. описание в работе [12]), наличие ребер T^2 и T^3 можно установить с помощью следующих утверждений.

Утверждение 1. Пусть d_1, $d_2 \in D$, интерфейс $li_1 \in VI^2_{d1}$ находится в отношении L^2 с интерфейсом $li_2 \in VI^2_{d2}$ и нет ни одного основанного на одном из них интерфейса сетевого уровня. Если существует единственный интерфейс $ni_1 \in RI^3_{d1}$ такой, что от него существует путь сетевого уровня до какого-либо интерфейса $ni_2 \in RI^3_{d2}$, то li_1 и ni_1 состоят в отношении T^2.

Доказательство. Исходя из условия 4, так как интерфейсы li_1 и li_2 не входят ни в один сетевой интерфейс, то должны существовать интерфейсы $ni_3 \in RI^3_{d1}$ и $ni_4 \in RI^3_{d2}$, для которых $(li_1, ni_3) \in T^2$ и $(li_1, ni_3) \in T^2$. Допустим, что $ni_3 \neq ni_1$. Так как ni_1 — это единственный интерфейс устройства d_1, от которого существует путь сетевого уровня до интерфейса устройства d_2, то возникает противоречие с условием 1. Значит, $ni_3 = ni_1$ и $(li_1, ni_1) \in T^2$.

Приведенное утверждение 1 позволяет обнаружить связь туннелирования канального уровня при наличии данных о соединениях канального и сетевого уровней и связях маршрутизации.

Утверждение 2. Пусть d_1, $d_2 \in D$ и интерфейс $ni_1 \in VI^3_{d1}$ находится в отношении L^3 с интерфейсом $ni_2 \in VI^3_{d2}$. Если существует единственный интерфейс $ni_3 \in RI^3_{d1}$ такой, что от него существует путь сетевого уровня до какого-либо интерфейса $ni_4 \in RI^3_{d2}$, то ni_1 и ni_3 находятся в отношении T^3.

Доказательство. Исходя из условия 5 должны существовать интерфейсы $ni_5 \in RI^3_{d1}$ и $ni_6 \in RI^3_{d2}$, для которых $(ni_1, ni_5) \in T^3$ и $(ni_2, ni_6) \in T^3$. Допустим, что $ni_5 \neq ni_3$. Так как ni_3 — это единственный интерфейс устройства d_1, от которого существует путь сетевого уровня до интерфейса устройства d_2, то возникает противоречие с условием 2. Значит, $ni_5 = ni_3$ и $(ni_1, ni_3) \in T^3$.

Приведенное утверждение 2 позволяет обнаружить связь туннелирования сетевого уровня при наличии данных о соединениях сетевого уровня и связях маршрутизации.

Отметим, что руководствами по настройке оборудования (например, [16]) рекомендуется, чтобы интерфейс, через который происходит туннелирование, имел соединение с интерфейсами Сети-посредника. Поэтому, если имеющиеся данные не удовлетворяют условиям утверждений 1 и 2, можно воспользоваться следующими эмпирическими правилами разрешения возникшей неопределенности. Ситуации в следующих правилах соответствуют таковым в условиях утверждений 1 и 2.

Правило 1. Если существует единственный интерфейс $ni_1 \in RI^3_{d1}$ такой, что от него существует путь сетевого уровня до какого-либо интерфейса $ni_2 \in RI^3_{d2}$, начинающийся с ребра L^3, то li_1 и ni_1 находятся в отношении T^2.

Правило 2. Если существует единственный интерфейс $ni_3 \in RI^3_{d1}$ такой, что от него существует путь сетевого уровня до какого-либо интерфейса $ni_4 \in RI^3_{d2}$, начинающийся с ребра L^3, то ni_1 и ni_3 находятся в отношении T^3.

Построение ребер туннелирования необходимо проводить на четвертом этапе процесса построения графа структуры Сети после обнаружения всех ребер соединения.

Заключение

Эффективное внедрение и обслуживание виртуальных частных сетей в коммуникационных инфраструктурах лПСУ требует использования формальных методов сетевого управления, которые, в свою очередь, требуют наличия моделей структуры Сети с ВЧС. Расширение обобщенной графовой модели структуры Сети, предложенное в данной работе, позволяет описывать структуру ВЧС, раскрывая при этом подробности лежащей в ее основе структуры физического, канального и сетевого уровней. Многоуровневая архитектура модели позволила описать большинство технологий построения ВЧС и отразить нарушение этими технологиями стандартного взаимодействия в рамках модели OSI. Данные возможности позволяют эффективно использовать модель в методах сетевого управления, связанных с ВЧС.

Расширение алгоритма построения графа структуры Сети и формально обоснованные методы поиска отношений внутри ВЧС при отсутствии части информации во входных данных алгоритма, которые предложены в данной статье, предоставляют возможность автоматизированного отражения структуры ВЧС.

В будущем планируется разработка дополнения к модели, позволяющего отразить технологию MPLS (*Multiprotocol Label Switching*), которая используется для альтернативного способа организации маршрутизации и ВЧС.

Автор выражает благодарность научным руководителям Ю. А. Богоявленскому и А. С. Колосову за полезные идеи, обсуждения и поддержку данной работы.

Список литературы

1. **Таненбаум Э., Уэзеролл Д.** Компьютерные сети. 5-е изд. СПб.: Издательский дом Питер, 2012. 960 с.

2. **Олифер В., Олифер Н.** Компьютерные сети: Принципы, технологии, протоколы. 5-е изд. СПб.: Питер, 2016. 992 с.

3. **Clemm A.** Network management fundamentals. Cisco Press, 2006. 552 p.

4. **Росляков А. В., Нуштаев А. В.** Модели и методы реализации отказоустойчивых VPN // Электросвязь. 2007. № 7. С. 47—50.

5. **Ravindran R. S., Huang C., Thulasiraman K.** A dynamic managed VPN service: Architecture and algorithms // 2006 IEEE International Conference on Communications. IEEE, 2006. Vol. 2. P. 664—669.

6. **Sivakumar L., Balabaskaran J., Thulasiraman K., Arumugam S.** Virtual topologies for abstraction service for IP-VPNs // 17th International Telecommunications Network Strategy and Planning Symposium (Networks). IEEE. 2016. P. 213—220.

7. **Gobjuka H., Breitbart Y.** Ethernet topology discovery for networks with incomplete information // Networking, IEEE/ACM Transactions on. 2010. Vol. 18, No. 4. P. 1220—1233.

8. **Zichao L., Ziwei H., Geng Z., Yan M.** Ethernet topology discovery for virtual local area networks with incomplete information // Network infrastructure and digital content (ic-nidc), 2014, 4th IEEE International conference on. 2014. P. 252—256.

9. **Xiaobo Ma, Tingting Yu.** An algorithm of physical network topology discovery in multi-VLANs // TELKOMNIKA. 2016. Vol. 14, No. 3A. P. 375—379.

10. **Богоявленский Ю. А.** Прототип экспериментальной платформы Nest для исследования моделей и методов управления ИКТ-инфраструктурами локальных поставщиков услуг Интернет // Программная инженерия. 2013. № 2. С. 11—20.

11. **Андреев А. А., Колосов А. С., Богоявленский Ю. А.** Автоматизация построения графа канального уровня ИКТ-инфраструктуры локального поставщика услуг интернета // Ученые записки Петрозаводского государственного университета. Серия: Естественные и технические науки. 2015. № 2 (147). С. 97—102.

12. **Андреев А. А., Колосов А. С., Богоявленский Ю. А., Воронин А. В.** Обобщенная графовая модель структуры физического, канального и сетевого уровней ИКТ-инфраструктуры локального поставщика сетевых услуг // Программная инженерия. 2016. Т. 7, № 9. С. 400—407.

13. **Andreev A., Kolosov A., Voronin A., Bogoiavlenskii I.** A graph model of the topology of physical, link and network layers of an enterprise network // Proceedings of the 19th Conference of Open Innovations Association FRUCT. Helsinki, Finland: FRUCT Oy, 2016. P. 3—9.

14. **Diab W. B., Tohme S., Bassil C.** VPN analysis and new perspective for securing voice over VPN networks // Proceedings of Fourth International Conference on Networking and Services. IEEE, 2008. P. 73—78.

15. **Pepelnjak I., Guichard J.** MPLS and VPN architectures, CCIP edition. Cisco Press, 2002. 512 p.

16. **Brocade Communications Systems.** FastIron ethernet switch layer 3 routing configuration guide. 2015. URL: http: // www.brocade.com/content/dam/common/documents/content-types/configuration-guide/fastiron-08030-l3guide.pdf

Generalized Graph Model of Virtual Private Networks in the Communication Infrastructure of a Local Network Service Provider

A. A. Andreev, andreev@cs.petrsu.ru, Petrozavodsk State University, 185910, Petrozavodsk, Russian Federation

Corresponding author:

Andreev Anton A., Undergraduate Student, Petrozavodsk State University, 185910, Petrozavodsk, Russian Federation, E-mail: andreev@cs.petrsu.ru

Received on 30 March, 2017
Accepted on 05 April, 2017

Implementation and maintenance of virtual private networks (VPN) in a local network service provider's communication infrastructure (network) requires solving a number of problems related to quality assurance, VPN topology design and scaling. Most of these problems require a topology graph of the logical and physical structure of the network with implemented VPN. Existing methods of network management use models for such graphs that are not able to represent some important things in modern networks such as VLAN and complex routing algorithms.

This paper proposes the graph model of the structure of tunnel connection of VPN implemented in network. This model is an extension of the generalized graph model of physical, link and network layers topology of a modern local network service provider's network. The model defines network elements involved in implementations of link and network layer VPNs. The paper also describes the methods for automated discovering and building of VPN in network topology graph basing on data from network devices.

The proposed model and VPN discovery methods could be used in a variety of network management tasks, in particular for automated network topology discovery.

Keywords: *network management, ICT-infrastructure graph, topology discovery, graph model, virtual private network*

For citation:

Andreev A. A. Generalized Graph Model of Virtual Private Networks in the Communication Infrastructure of a Local Network Service Provider, *Programmnaya Ingeneria*, 2017, vol. 8, no. 6, pp. 243—249.

DOI: 10.17587/prin.8.243-249

References

1. **Tanenbaum A., Wetherall D.** *Komp'juternye seti* (Computer Networks). 5-th ed., Saint-Peterburg, Piter, 2012, 960 p. (in Russian).
2. **Olifer V., Olifer N.** *Komp'juternye seti: Principy, tehnologii, protokoly* (Computer Networks: Principles, Technologies and Protocols for Network Design), 5-th ed., Saint-Peterburg, Piter, 2016, 992 p. (in Russian).
3. **Clemm A.** *Network management fundamentals*, Cisco Press, 2006, 552 p.
4. **Rosljakov A. V., Nushtaev A. V.** Modeli i metody realizacii otkazoustojchivyh VPN (Models and methods of implementation of a fault-tolerant VPN), *Jelektrosvjaz',* 2007, no. 7, pp. 47—50 (in Russian).
5. **Ravindran R. S., Huang C., Thulasiraman K.** A dynamic managed VPN service: Architecture and algorithms, *2006 IEEE International Conference on Communications*, IEEE, 2006, vol. 2, pp. 664—669.
6. **Sivakumar L., Balabaskaran J., Thulasiraman K., Arumugam S.** Virtual topologies for abstraction service for IP-VPNs, *17th International Telecommunications Network Strategy and Planning Symposium (Networks),* IEEE, 2016, pp. 213—220.
7. **Gobjuka H., Breitbart Y.** Ethernet topology discovery for networks with incomplete information, *Networking, IEEE/ACM Transactions on,* 2010, vol. 18, no. 4, pp. 1220—1233.
8. **Zichao L., Ziwei H., Geng Z., Yan M.** Ethernet topology discovery for virtual local area networks with incomplete information, *Network infrastructure and digital content (IC-NIDC), 2014 4th IEEE international conference on*, 2014, pp. 252—256.
9. **Xiaobo Ma, Tingting Yu.** An algorithm of physical network topology discovery in multi-VLANs*, TELKOMNIKA*, 2016, vol. 14, no. 3A, pp. 375—379.
10. **Bogoiavlenskii Iu. A.** Prototip jeksperimental'noj platformy Nest dlja issledovanija modelej i metodov upravlenija IKT-infrastrukturami lokal'nyh postavshhikov uslug Internet (Prototype of the Tested Nest for Research of Network Management Methods and Models at the Enterprise Network Level), *Programmnaya Ingeneria,* 2013, no. 2, pp. 11—20 (in Russian).
11. **Andreev A. A., Kolosov A. S., Bogoiavlenskii Iu. A.** Avtomatizacija postroenija grafa kanal'nogo urovnja IKT-infrastruktury lokal'nogo postavshhika uslug Interneta (Automation of ICT-infrastructure link layer graph discovery for local internet service providers), *Uchenye zapiski Petrozavodskogo gosudarstvennogo universiteta. Serija: Estestvennye i tehnicheskie nauki*, 2015, no. 2 (147), pp. 97—102 (in Russian).
12. **Andreev A. A., Kolosov A. S., Voronin A. V., Bogoiavlenskii Iu. A.** Generalized graph model of the physical, link and network layers structure of the ICT-infrastructure of a local network service provider, *Programmnaya Ingeneria,* 2016, vol. 7, no. 9, pp. 400—407 (in Russian).
13. **Andreev A., Kolosov A., Voronin A., Bogoiavlenskii I.** A graph model of the topology of physical, link and network layers of an enterprise network, *Proceedings of the 19th Conference of Open Innovations Association FRUCT*, Helsinki, Finland: FRUCT Oy, 2016, pp. 3—9.
14. **Diab W. B., Tohme S., Bassil C.** VPN analysis and new perspective for securing voice over VPN networks, *Proceedings of Fourth International Conference on Networking and Services*, IEEE, 2008, pp. 73—78.
15. **Pepelnjak I., Guichard J.** *MPLS and VPN architectures*, CCIP edition, Cisco Press, 2002, 512 p.
16. **Brocade Communications Systems.** FastIron ethernet switch layer 3 routing configuration guide, 2015, available at: http: // www.brocade.com/content/dam/common/documents/content-types/configuration-guide/fastiron-08030-l3guide.pdf

ИНФОРМАЦИЯ

УДК 004.4'22 DOI: 10.17587/prin.8.250-257

В. Ю. Трифанов, канд. техн. наук, инженер-программист, e-mail: vitaly.trifanov@gmail.com,
Д. И. Цителов, руководитель группы внутренних разработок, Devexperts LLC, г. Санкт-Петербург

Язык описания синхронизационных контрактов для задачи поиска гонок в многопоточных приложениях[1]

Состояние гонки приложения (data race) — это одновременное обращение двух потоков к одной и той же памяти, причем одно из обращений является записью. Гонки являются одним из самых частых и трудно обнаружимых типов ошибок параллельного программирования. Существует много методов поиска гонок, однако ни один из них не дает полной гарантии. Широко используется динамический поиск гонок, который, однако, обладает высокими накладными расходами. Для решения этой задачи авторами была разработана концепция обнаружения гонок на основе синхронизационных контрактов. Для описания контрактов применялся XML-язык, однако при практическом использовании этого языка был выявлен ряд ограничений, затрудняющих создание и повторное использование контрактов. В данной работе представлен новый язык описания контрактов, близкий по синтаксису к языку Java, приведены рекомендации по его применению и примеры описания контрактов для классов пакета java.util.concurrent.

Ключевые слова: *параллельное программирование, состояние гонки, динамическое обнаружение гонок, контракты, Java-приложения*

Введение

Параллельное программирование активно применяется в различных областях — при разработке бизнес-приложений, систем реального времени и т. д. [1]. Во многих индустриальных языках программирования используется модель разделяемой памяти, в которой существуют несколько независимых потоков управления и общая память, с помощью которой потоки обмениваются данными. В программах, разработанных на таких языках, могут возникать состояния гонки — несинхронизированные обращения к общим данным из различных потоков, среди которых хотя бы одно обращение — запись данных [2]. Состояния гонки сложно обнаружить на стадии тестирования, потому что их возникновение зависит от чередования операций в потоках, и эти состояния, фактически, невоспроизводимы. Автоматическое обнаружение гонок возможно путем статического и динамического анализа. В первом случае программа анализируется без запуска, во втором анализируется запущенная (работающая) программа. Динамический анализ позволяет добиться высокой точности и поддержки неблокирующих механизмов синхронизации, но обладает высокими накладными расходами.

Задача статического анализа программы NP-полна для конечных графов выполнения и алгоритмически неразрешима в общем случае [2, 3]. Поэтому статическим детекторам приходится существенно ограничивать область анализа, что приводит к ложным срабатываниям. Кроме того, статические детекторы способны поддерживать только механизмы синхронизации, основанные на захвате блокировок [4]. Ввиду значительных ограничений статического анализа получили развитие подходы, основанные на динамическом анализе. Ключевая задача этих подходов — сократить накладные расходы на работу системы без потери точности анализа. Так, в работе [5] предложена эффективная оптимизация алгоритма, основанного на векторных часах, а в работе [6] — его гибридизация с другим известным динамическим алгоритмом lockset [7]. Авторы работ [8, 9] применили семплиирование (выборочный анализ) для уменьшения числа обрабатываемых операций и получили практически приемлемое число ложных срабатываний на нескольких промышленных приложениях. Тем не менее, общая задача получения производительного высокоточного динамического детектора далека от решения. В работах [10, 11] нами был предложен подход к сокращению накладных расходов динамического обнаружения гонок в Java-программах путем разработки и использования *синхронизационных контрактов*, позволяющих ограничить анализ многопоточного поведения в стандартных библиотеках, что существенно сокращает накладные расходы. Данная концепция была реализована в детекторе jDRD и успешно использована на ряде промышленных проектов. В процессе апробации было выявлено, что язык описания синхронизационных контрактов

[1] Работа выполнена при поддержке гранта РФФИ 15-01-05431-а.

имеет ряд недостатков, затрудняющих создание и повторное использование контрактов. Кроме того, известно, что использование в одном проекте различных языков спецификаций существенно усложняет разработку с психологической точки зрения.

С целью упрощения промышленного использования динамического детектора гонок jDRD, а также для унификации используемых языковых средств, в данной работе предложен новый язык спецификации контрактов с близкой к языку Java грамматикой. В статье представлены примеры описания ряда контрактов для классов пакета java.util.concurrent [12] и приведены рекомендации по применению языка.

1. Обзор существующих исследований

Понятие контракта происходит из концепции контрактного программирования — метода проектирования программной системы, в котором описываются декларативные верифицируемые спецификации ее частей [13]. Традиционно в основе описания контракта метода лежит тройка Хоара: предусловие, постусловие и инвариант. Концепция контракта была предложена Бертраном Мейером и реализована в языке программирования Eiffel [13]. К сожалению, парадигма контрактного программирования, хорошо применимая в детерминированных условиях однопоточного выполнения, сталкивается с серьезными трудностями в многопоточной среде. По этой причине в Eiffel разработана отдельная модель параллельного исполнения SCOOP, значительно отличающаяся от существующей модели разделяемой памяти, используемой в индустриальных языках типа Java и C#. Среди библиотек, предоставляющих возможность описания контрактов методов в Java, например, [14, 15], также нет конструкций для специфицирования поведения метода в многопоточной среде. Таким образом, задача разработки языка описания синхронизационных контрактов является новой и актуальной.

2. Динамическое обнаружение гонок с помощью синхронизационных контрактов

Состояние гонки возникает, если два обращения к общему участку разделяемой памяти из различных потоков не были упорядочены с помощью операции синхронизации, которая является частичным отношением порядка, называемым *happens-before* [16]. Детектор jDRD отслеживает это отношение с помощью логических (векторных) часов Лампорта [17]. Для сокращения области анализа без потери точности jDRD предоставляет возможность описать с помощью контракта частичную спецификацию замкнутой части программы (например, стандартной библиотеки) с точки зрения ее поведения в многопоточной среде и исключить эту часть из анализа. Детектор jDRD динамически обрабатывает контракты, поэтому при полном описании контрактов такой подход исключает ложные срабатывания [10]. На практике все контракты описать не удается, но нам удалось выделить основные часто встречающиеся контракты, классифицировать их и предложить механизм описания. Этот подход был апробирован на нескольких промышленных приложениях и показал практическую полезность и высокую производительность [11].

2.1. Синхронизационные контракты

Синхронизационные контракты бывают трех типов — happens-before-контракт, контракт потокобезопасного метода, контракт потоконебезопасного метода.

Happens-before-контракт описывает пару явно связанных методов, вызовы которых из различных потоков гарантируют синхронизацию потоков. Детектор jDRD отслеживает такие пары вызовов во время работы программы и обрабатывает их как синтетическую высокоуровневую передачу отношения happens-before. Явная связь является суперпозицией конечного числа простых связей типа "владелец—владелец", "владелец — параметр" и "параметр — параметр" [10]. При описании такого контракта, указываются вызовы методов, вовлеченных в контракт и все примитивные связи, образующие связь между ними. Ниже приведен пример контракта для методов put и get класса ConcurrentHashMap, декларирующего, что вызов метода put по некоторому ключу из одного потока синхронизирован с последующим вызовом метода get того же объекта по тому же ключу из другого потока:

```
<Sync>
  <Links>
    <Link send="owner" receive="owner"/>
    <Link send="param" send-number="0" receive="param" receive-number="0"/>
  </Links>
<Send>
    <MethodCall owner="java.util.concurrent.ConcurrentMap" name="put"
       descriptor = "(Ljava/lang/Object;Ljava/lang/Object;)Ljava/lang/Object;"/>
</Send>
<Receive>
    <MethodCall owner = "java.util.concurrent.ConcurrentMap" name="get"
       descriptor="(Ljava/lang/Object;)Ljava/lang/Object;"/>
  </Receive>
</Sync>
```

Контракт потокобезопасного метода указывает, что данный метод не может быть вовлечен в гонку. Детектор jDRD игнорирует такие методы. При описании такого контракта указывается класс-владелец и название метода.

Контракт потоконебезопасного метода указывает тип метода по отношению к его объекту-владельцу — является метод модифицирующим или немодифицирующим. Детектор jDRD обрабатывает такие методы, как изменение их объекта-владельца на запись или чтение соответственно. Ниже приведен набор таких контрактов для коллекций Java:

```
<Contracts>
   <Contract clazz = "java.util.Map" read=
     "keySet,values,entrySet"/>
   <Contract clazz = "java.util.List" read=
     "listIterator"/>
   <Contract clazz="*" read="get*,toString,
     hashCode,equals,is*,contains*,iter*,has*,
     size"/>
</Contracts>
```

2.2. Трудности описания контрактов с помощью XML

По мере эксплуатации детектора jDRD были выявлены следующие ограничения и неудобства XML-языка описания контрактов:

- описания вызовов методов далеки от Java-синтаксиса, что затрудняет использование подхода в Java-разработках;
- описания потокобезопасных, модифицирующих и немодифицирующих методов имеют такую же синтаксическую природу, как и описания синхронизационных контрактов, в то время как в XML-языке им соответствуют разные конструкции;
- XML-язык не предусматривает включение (импорт, include) других файлов с описанием контрактов;
- необходимость указывать полное имя класса и отсутствие сокращений (aliases) существенно затрудняют чтение и поддержку спецификаций контрактов;
- XML-язык не предоставляет возможность использовать названия (переменные) или номера параметров при описании простых связей.

Также было сделано важное наблюдение, касающееся разработки контрактов. Контракты, как правило, разрабатываются структурированно — сразу для класса или группы классов. Таким образом, создателю контрактов приходится переводить фрагменты кода Java-классов на XML. Гораздо удобнее было бы иметь в качестве языка описания контрактов Java-подобный язык, что ускорило бы процесс разработки контрактов. Кроме того, при использовании языка с формальной грамматикой существенно проще проверять контракты на непротиворечивость, потому что значительная часть проверки будет сделана автоматически на этапе синтаксического разбора.

3. Язык описания контрактов

Для устранения перечисленных выше сложностей и ограничений было решено разработать новый язык описания контрактов, в основе которого лежит простая, максимально похожая на синтаксис Java, грамматика. Существенным дополнением является возможность использования мета-символов (например, '*') в описании названий пакетов, классов, методов и их параметров. Конструкции с подобным синтаксисом часто возникают при описании аспектов, поэтому целесообразно было взять за основу соответствующие элементы языка AspectJ [18]. Этот язык входит в состав одноименной библиотеки, являющейся самой известной аспектно-ориентированной библиотекой. Выразительная сила этих конструкций достаточна для реализации требований к разрабатываемому языку, а сама библиотека знакома большинству индустриальных программистов.

При описании сущности контракта необходимо указать ее тип и идентификатор. Под идентификатором понимается любой корректный Java-идентификатор.

Понятие типов также полностью взято из языка Java: поддерживаются базовые типы, ссылочные типы и массивы. Часто в качестве типа указывается java.lang.Object — это корневой класс в иерархии классов Java. Вместо этого для упрощения читаемости спецификаций можно указать "?" по аналогии с конструкцией Java под названием generic:

```
referenceType : qualifiedName | '?'; // '?' stands for java.lang.Object;
type : (referenceType | BasicType) ('[]')*;
return_type : type | 'void';
```

Описание happens-before-контракта требует указания совокупности простых связей между методами, входящими в контракт. Каждая такая связь состоит из двух концов и ключа, с которым она связана. Декларация ключа схожа с объявлением переменной в Java:

```
key_spec : 'key' key_component (',' key_component) * ';';
key_component : type name? ('=' mapping)?;
```

Каждый конец связи указывает на объект-владелец или параметр метода в зависимости от ее типа:

```
mapping : instance _ mapping|param _ mapping|class _ mapping;
instance _ mapping: 'o';
param _ mapping : PPlusDigit;
class _ mapping : 'c';
PPlusDigit : 'p' Digit+;
```

Наконец, в описании самой связи указываются оба метода, которые она связывает. Описания методов соответствуют описанию методов в Java:

```
hb _ link : hb _ link _ side hb _ flags? method _ pattern';';
hb _ link _ side : 'send' | 'receive' | 'full';
hb _ flags : '('hb _ flag (','hb _ flag)* ')';
hb _ flag : 'shouldReturnTrue';
method _ pattern : return _ type _ pattern? member _ name _ pattern '(' signature _ pattern ')';
```

Предложенная грамматика позволяет как просто указать связь между двумя методами, так и описать связи между несколькими методами одного класса, объединив их с помощью ключевого слова class. Таким образом, получается конструкция, очень близкая к объявлению интерфейса или класса в Java. Кроме того, для длинных имен можно вводить сокращения (aliases). Соответствующий пример представлен на рис. 1.

```
method_pattern : return_type_pattern? member_name_pattern '(' signature_pattern ')';
type_pattern : '*' | type;
return_type_pattern : '*' | return_type;
member_name_pattern : name_pattern | member_class_pattern '.' name_pattern;
member_class_pattern : member_class_name_pattern sig_param_mapping?;
member_class_name_pattern : '%' ('.'qualifiedName)? | qualifiedName;
signature_pattern : params_pattern? | '..';
params_pattern : param_pattern (','param_pattern )*;
param_pattern : type name? sig_param_mapping?;
name_pattern : '*' | name'*'?;
name : Identifier;
sig_param_mapping : '=' Digit+;
class_pattern : '*' | qualifiedName (('.'| '..') '*')?;
```

Рис. 1.

Все описания happens-before-контрактов объединяются в общую секцию, например:

```
sync _ section : 'synchronization' '{' (sync _ class _ block | sync _ block | alias _ spec)* '}';
sync _ block : 'sync' '{' alias _ spec* key _ spec? (hb _ link* | sync _ class _ block) '}';
sync _ class _ block : 'class' class _ pattern '{' (sync _ block | alias _ spec | hb _ link)* '}';
```

Контракты потокобезопасных и потоконебезопасных методов описываются в отдельной секции `contracts`, например:

```
contracts _ section : 'contracts' '{' contracts _ class* '}';
contracts _ class : 'class' class _ pattern '{' contracts _ member* '}';
contracts _ member : contract _ mode method _ pattern ';';
contract _ mode : 'skip' | 'read' | 'write';
```

Важным преимуществом синхронизационных контрактов является возможность их повторного использования: будучи единожды описанными, например, для некоторой стандартной библиотеки, они могут быть

использованы во всех системах, использующих данную библиотеку. В языке предусмотрена возможность подключения других файлов с контрактами, а также описание областей отслеживания операций синхронизации и обнаружения гонок, например:

```
include _ statement ::= 'include' StringLiteral ';'
instrumentation _ section ::=...
configuration ::=(instrumentation _ section | sync _ section | include _ statement)*
```

Сходство предложенного языка с Java позволяет в едином стиле описывать как синхронизационные контракты, так и контракты потокобезопасных и потоконебезопасных методов.

4. Примеры

Рассмотрим пример конфигурации, разработанной на новом языке описания контрактов при подготовке jDRD к анализу простейшего Java-приложения. Операции синхронизации отслеживаются во всем программном коде, а гонки ищутся во всех классах целевого приложения, кроме одного. Ниже представлена начальная часть спецификации контрактов для этого примера.

```
instrumentation {
   interceptSyncOperations {
     include*;
   }

   raceDetection {
     include com.myapp..*;
     exclude com.myapp.ClassToIgnore;
   }
}
```

В секции `contracts` находятся описания контрактов потокобезопасных и потоконебезопасных методов. Здесь указано, что все методы класса `java.lang.System` потокобезопасны и их вызовы обрабатывать не следует. Вызовы методов `keySet`, `values` и `entrySet` объектов типа `Map` стоит трактовать как обращения к этому объекту на чтение. Кроме того, у любых классов методы `toString`, `equals`, `hashcode`, а также методы, начинающиеся на `get`, `iter`, `is`, `contains` и `has` тоже следует трактовать как немодифицирующие. Все остальные методы будут рассматриваться, как модифицирующие. Все указанное выше представлено в листинге на рис. 2.

Далее, в разделе happens-before-контрактов описан контракт класса `AbstractQueuedSynchronizer`, который является основой для большинства синхронизационных механизмов пакета `java.util.concurrent`. Все пары методов в нем связаны через объект-владелец, что указано с помощью ключевого слова `key`, а также перечислены методы и их тип — являются ли они передающими или принимающими отношение happens-before. Всё сказанное выше иллюстрирует листинг на рис. 3.

```
contracts {
  class java.lang.System {
    skip * * (..);// skip foreign calls
  }
  class java.util.Map {
    read * keySet (..),
    read * values (..),
    read * entrySet (..);
  }
  class * {
    read * get* (..);
    read * toString ();
    read * hashCode (..);
    read * equals (..);
    read * is* (..);
    read * contains * (..);
    read * iter * (..);
    read * has * (..);
  }
…
}
```

Рис. 2.

```
synchronization {
  class juc.locks.AbstracrQueuedSynchronizer {
    sync {
      key ? = o;

      send boolean tryRelease (int);
      receive (shouldReturnTrue) boolean tryAcquireShared (int);
      send boolean tryReleaseShared (int);
      receive void acquire (int);
      receive void acquireInterruptibly (int);
      receive (shouldReturnTrue) boolean tryAcquireNanos (int,long);
      send boolean release (int);
      receive void acquireShared (int);
      receive void acquireSharedInterruptibly (int);
      receive (shouldReturnTrue) boolean tryAcquireSharedNanos (int,long);
      send boolean releaseShared (int);
      send boolean setState (int);
      receive int getState ();
      full (shouldReturnTrue) boolean compareAndSetState (int,int)
    }
  }
}
```

Рис. 3.

5. Измерения и оценки

В настоящем разделе приведены данные экспериментов с контрактным подходом к динамическому поиску гонок с помощью детектора jDRD и нового языка описания контрактов. Эксперименты отвечают на следующие вопросы:

- каково количество контрактов, которое нужно создавать для типового Java-приложения;
- каково замедление скорости работы приложения при использовании динамического детектора jDRD;
- как изменился объем спецификаций контрактов на новом языке по сравнению со старым.

Количество контрактов, которое нужно создавать для типового Java-приложения. Авторами были составлены контракты для трех проектов среднего размера (400...2000 классов, 10...30 потоков) и одного крупного индустриального проекта (несколько десятков тысяч строк кода, более 100 потоков). Были составлены три следующих вида контрактов.

- Контракты ядра Java — около 15 happens-before-контрактов и 40 описаний потокобезопасных и потоконебезопасных методов. Эти контракты необходимы при анализе любого Java-приложения. Несмотря на то, что полученный набор контрактов является неполным, уже можно оценить, что общее число контрактов ядра Java не превосходит 100...150. Контракты данного вида можно будет повторно использовать в других Java-приложениях.
- Контракты для часто используемых стандартных библиотек и компонентов: log4j (логирование), Swing (графическая подсистема), некоторые части Spring (каркас для построения приложений), XStream (сериализация в XML). Как правило, такие библиотеки обладают замкнутой функциональностью и для них требуется лишь небольшое число контрактов потокобезопасных и потоконебезопасных методов. В нашем случае было создано 3...5 контрактов на библиотеку, всего было создано 17 таких контрактов.
- Контракты, специфичные для целевой системы. В эту категорию попадают контракты подсистем и нетиповых библиотек, использующихся целевой системой. Число таких контрактов растет пропорционально сложности системы и степени интеграции исключаемых подсистем. В крупном проекте, который анализировался авторами, потребовалось создать 22 таких контракта, в проектах среднего размера их число не превысило 10. В целом было составлено 30 контрактов такого вида.

Таким образом, для систем широкого класса может быть создано от 50 (небольшие системы с типовыми зависимостями) до нескольких сотен контрактов (масштабные и узкоспециализированные системы). При этом наблюдается процесс "насыщения", т. е. повторно используются контракты, созданные для стандартных библиотек, причем контрактные спецификации для этих библиотек могут дополняться в части используемых в данном проекте возможностей, для которых ранее не были созданы контракты. Если подход начнет активно применяться в индустрии и описанные контракты будут открытыми (*open source*), то с определенного момента можно будет ограничиться лишь описанием контрактов, специфичных для целевых систем (третий вид). Это существенно понизит дальнейшую трудоемкость применения подхода.

Замедление скорости работы системы при использовании динамического детектора jDRD. Детектор jDRD компилирует описание контрактов во внутренние структуры данных. Скорость компиляции контрактов пренебрежимо мала по сравнению с затратами на их отслеживание во время динамического анализа, поскольку компиляция проводится лишь один раз при запуске приложения и не занимает больше одной секунды. Трудоемкость динамического отслеживания выполнения контрактов пропорцио-

нальна частоте обнаружения описанных в контракте ситуаций. Для happens-before-контрактов она более высока, поскольку в этом случае требуется выполнение ресурсоемкой операции слияния векторных часов. Проверка контрактов потоконебезопасных методов нагружает систему в меньшей степени.

Сравнение объемов спецификаций на старом и новом языках. Для крупной Java-системы было выполнено два вида спецификаций контрактов — на прежнем XML-языке и с помощью нового языка, предложенного в данной статье. Объем контрактов на новом языке в символах на 45 % меньше, чем на прежнем, в строках же меньше лишь на 5 %. Число строк спецификации осталось почти на прежнем уровне, потому что новый язык сохраняет старый подход к описанию. Число же символов уменьшилось вследствие того, что в новом языке появились так называемые *wildcards* — символы "*" и "?", которые можно использовать для обозначения понятия "произвольный объект", а фактическая проверка типов перенесена в фазу исполнения программы. Таким образом, например, громоздкое описание сигнатуры метода с четырьмя параметрами `run(Ljava.lang.Object; Ljava.lang.Object; Ljava.lang.Object; J)` превращается в следующее описание: `run(?,?,?,?)`. Также добавлена возможность именовать простую связь между вызовами методов (по сути, декларировать переменную) и далее ссылаться на нее по имени. Все это позволяет опускать ненужные подробности при описании контракта, оставляя лишь существенную информацию и повышая удобство разработки и читаемость контрактов. С увеличением числа контрактов и количества пользователей детектора этот фактор становится все более значимым.

Заключение

Динамический способ обнаружения гонок обладает рядом значительных преимуществ по сравнению с другими подходами, но привносит существенные накладные расходы на производительность целевой системы. Для повышения производительности в динамическом детекторе jDRD была реализована концепция синхронизационных контрактов и создан язык их описания на основе XML. После промышленной эксплуатации jDRD был выявлен ряд ограничений и практических неудобств этого языка — непохожий на Java синтаксис, слабая структурированность, отсутствие возможности импортировать другие файлы конфигурации и т. д. Для устранения этих трудностей был создан язык с синтаксисом, максимально близким к Java и AspectJ. Существующие контракты были переписаны с использованием этого языка.

В качестве направления дальнейшего развития средств спецификации контрактов можно указать визуальное моделирование, причем интерес представляет как структурное описание контрактов с помощью диаграмм классов, компонентов UML или с помощью предметно-ориентированных визуальных языков [19—21], так и спецификация их поведения с помощью динамических моделей [1]. Также интересны подходы по автоматизированному извлечению описания контрактов из Java-документации и интеграции повторного использования контрактов с повторным использованием документации [23, 24].

Список литературы

1. **Терехов А. Н.** Технология программирования. Учебное пособие. М.: Интернет-ун-т информ. технологий: БИНОМ. Лаб. знаний, 2007. 148 с.
2. **Netzer R., Miller B.** What Are Race Conditions? Some Issues and Formalizations // ACM Letters On Programming Languages and Systems. 1992. Vol. 1, No. 1. P. 74—88.
3. **Netzer R.** Race Condition Detection for Debugging Shared-Memory Parallel Programs. Madison, PhD Thesis. 1991. 187 p.
4. **Elmas T., Qadeer S., Tasiran S.** Goldilocks: A Race and Transaction-Aware Java Runtime // Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'07), 2007. P. 245—255.
5. **Flanagan C., Freund S.** FastTrack: Efficient and Precise Dynamic Race Detection // ACM Conference on Programming Language Design and Implementation, 2009. P. 121—133.
6. **O'Callahan R., Choi J.-D.** Hybrid Dynamic Data Race Detection // Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming, 2003. P. 167—178.
7. **Savage S., Burrows M., Nelson G., Sobalvarro P., Anderson T.** Eraser: A Dynamic Data Race Detector for Multithreaded Programs // ACM Transactions on Computer Systems. 1997. Vol. 15, Issue 4. P. 391—411.
8. **Bond M., Coons K., McKinley K.** Pacer: Proportional Detection of Data Races // Proceedings of 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2010). Toronto, June 2010. P. 255—268.
9. **Marino D., Musuvathi M., Narayanasamy S.** LiteRace: Effective Sampling for Lightweight Data Race Detection // PLDI'09 Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation. 2009. Vol. 44, Issue 6. P. 134—143.
10. **Трифанов В. Ю.** Обнаружение состояний гонки в Java-программах на основе синхронизационных контрактов // Компьютерные инструменты в образовании. 2012. № 4. С. 16—29.
11. **Трифанов В.Ю., Цителов Д. И.** Динамический поиск гонок в Java-программах на основе синхронизационных контрактов // Материалы конференции "Инструменты и методы анализа программ (TMPA-2013)". Кострома, 2013. С. 273—285.
12. **Documentation** of java.util.concurrent. URL: http: // download.oracle.com/javase/6/docs/api/java/util/concurrent/package-summary.html
13. **Meyer B.** Object-Oriented Software Construction. 2nd edition. Prentice Hall, 2000. 1296 p.
14. **Проект C4J.** URL: http://c4j-team.github.io/C4J/
15. **JContractor.** URL: http://jcontractor.sourceforge.net/
16. **Java** Language Specification, Third Edition. Threads and Locks. URL: http://java.sun.com/docs/books/jls/third_edition/html/memory.html
17. **Lamport L.** Time, Clocks and the Ordering of Events in a Distributed System // Communications of the ACM. 1978. Vol. 21, Issue 7. P. 558—565.
18. **Проект** AspectJ. URL: https://eclipse.org/aspectj/
19. **Гаврилова Т. А., Лещева И. А., Кудрявцев Д. В.** Использование моделей инженерии знаний для подготовки специалистов в области информационных технологий // Системное программирование. 2012. Т. 7, № 1. С. 90—105.
20. **Кознов Д. В.** Визуальное моделирование компонентного программного обеспечения: дис. ... канд. физ.-мат. наук. СПб: 2000. 82 с.
21. **Кознов Д. В.** Разработка и сопровождение DSM-решений на основе MSF // Системное программирование. 2008. Т. 3, № 1. С. 80—96.
22. **Иванов А. Н., Кознов Д. В., Мурашева Т. В.** Поведенческая модель RTST++ // Записки семинара Кафедры системного программирования "Case-средства RTST++". 1998. № 1. С. 37—52.
23. **Луцив Д. В., Кознов Д. В., Басит Х. А., Терехов А. Н.** Задачи поиска нечетких повторов при организации повторного использования документации // Программирование. 2016. № 4. С. 39—49.
24. **Романовский К. Ю., Кознов Д. В.** Язык DRL для проектирования и разработки документации семейства программных продуктов // Вестник Санкт-Петербургского университета. Серия 10. Прикладная математика. Информатика. Процессы управления. 2007. № 4. С. 110—122.

A Language for Specification of Synchronization Contracts for Detecting Data Race in Parallel Applications

V. Yu. Trifanov, e-mail: vitaly.trifanov@gmail.com, **D. I. Tsitelov,** tsitelov@acm.org, Devexperts LLC, Saint-Petersburg, 197110, Russian Federation

Corresponding author:

Trifanov Vitaly Yu., Senior Developer, Devexperts LLC, Saint-Petersburg, 197110, Russian Federation, E-mail: vytaly.trifanov@gmail.com

Received on March 8, 2017
Accepted om March 29, 2017

Today parallel programming is actively used in many areas — business applications, telecommunications, etc. However it is not easy to develop parallel programs. One of the most serious errors of parallel programing is data races. These are situations when two or more parallel program entities (process, threads, etc.) read/write the same variable simultaneously, and one of them writes to it. Such errors are really hard to found, they lead to unpredictable corruption of application data and strange program behavior. There are a lot of approaches to cope with date races, however there is no absolutely reliable one. Dynamic paradigm of data race detection is widely used, but it leads to high overheads. We suggested a dynamic approach in our previous papers, that used synchronization contracts to cope with the overhands problem. Also we presented a toolkit for data race detection in Java-applications. Here we suggest a Java-like language to specify contracts. We also present a number of contract examples for java.util.concurrent library, and guidelines for industrial using the language and the approach.

Keywords: *parallel programming, data race, dynamic approach to date race detection, design contracts, Java-applications*

Acknowledgements: *This work was supported by the Russian Foundation for Basic Research, project no. 15-01-05431-a.*

For citation:

Trifanov V. Yu., Tsitelov D. I. A Language for Specification of Synchronization Contracts for Detecting Data Race in Parallel Applications**,** *Programmnaya Ingeneria,* 2017, vol. 8, no. 6, pp. 250—257.

DOI: 10.17587/prin.8.250-257

References

1. **Terekhov A. N.** *Tehnologija programmirovanija. Uchebnoe posobie* (Software engineering), Moscow, Internet-un-t inform. tehnologij: BINOM. Lab. znanij, 2007, 148 p. (in Russian).
2. **Netzer R., Miller B.** What Are Race Conditions? Some Issues and Formalizations, *ACM Letters On Programming Languages and Systems*, 1992, vol. 1, no. 1, pp. 74—88.
3. **Netzer R.** *Race Condition Detection for Debugging Shared-Memory Parallel Programs*, PhD Thesis, Madison, 1991. 187 p.
4. **Elmas T., Qadeer S., Tasiran S.** Goldilocks: A Race and Transaction-Aware Java Runtime, *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'07)*, 2007, pp. 245—255.
5. **Flanagan C., Freund S.** FastTrack: Efficient and Precise Dynamic Race Detection, *ACM Conference on Programming Language Design and Implementation*, 2009, pp. 121—133.
6. **O'Callahan R., Choi J.-D.** Hybrid Dynamic Data Race Detection, *Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, 2003, pp. 167—178.
7. **Savage S., Burrows M., Nelson G., Sobalvarro P., Anderson T.** Eraser: A Dynamic Data Race Detector for Multithreaded Programs, *ACM Transactions on Computer Systems*, 1997, vol. 15, issue 4, pp. 391—411.
8. **Bond M., Coons K., McKinley K.** Pacer: Proportional Detection of Data Races, *Proceedings of 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2010),* Toronto, June 2010, pp. 255—268.
9. **Marino D., Musuvathi M., Narayanasamy S.** LiteRace: Effective Sampling for Lightweight Data Race Detection. *PLDI '09 Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation*, 2009, vol. 44, issue 6, pp. 134—143.
10. **Trifanov V. Y.** Obnaruzhenie sostojanij gonki v Java-programmah na osnove sinhronizacionnyh kontraktov (Date race detection basing on synchronization contracts), *Komp'juternye instrumenty v obrazovanii*, 2012, no. 4, pp. 16—29 (in Russian).
11. **Trifanov V. Y., Tsitelov D. I.** Dinamicheskij poisk gonok v Java-programmah na osnove sinhronizacionnyh kontraktov (Dynamic rate race detection in Java-applications basing on synchronization contracts), *Materialy konferencii "Instrumenty i metody analiza programm (TMPA-2013)",* Kostroma, 2013, pp. 273—285 (in Russian).
12. **Documentation** of java.util.concurrent, available at: http: // download.oracle.com/javase/6/docs/api/java/util/concurrent/package-summary.html
13. **Meyer B.** *Object-Oriented Software Construction,* 2nd edition, Prentice Hall, 2000, 1296 p.
14. **C4J,** available at: http://c4j-team.github.io/C4J/
15. **JContractor**, available at: http://jcontractor.sourceforge.net/
16. **Java** Language Specification, Third Edition. Threads and Locks, available at: http://java.sun.com/docs/books/jls/third_edition/html/memory.html
17. **Lamport L.** Time, Clocks and the Ordering of Events in a Distributed System. Communications of the ACM, vol. 21, issue 7, 1978, pp. 558—565.
18. **AspectJ,** available at: https://eclipse.org/aspectj/
19. **Gavrilova T. A., Leshheva I. A., Kudrjavcev D. V.** Ispol'zovanie modelej inzhenerii znanij dlja podgotovki specialistov v oblasti informacionnyh tehnologij (Knowledge engineering models in information technology education), *Sistemnoe programmirovanie*, 2012, vol. 7, no. 1, pp. 90—105 (in Russian).
20. **Koznov D. V.** *Vizual'noe modelirovanie komponentnogo programmnogo obespechenija* (Visual modeling in component software development), PhD thesis, Sankt-Peterburgskij gos. universitet. Saint-Petersburg, 2000 (in Russian).
21. **Koznov D. V.** Razrabotka i soprovozhdenie DSM-reshenij na osnove MSF (Development and evolution of DSM-solutions basing MSF), *Sistemnoe programmirovanie*, 2008, vol. 3, no. 1, pp. 80—96 (in Russian).
22. **Ivanov A. N., Koznov D. V., Murasheva T. V.** Povedencheskaja model' RTST++ (Behavior model RTST + +), *Zapiski seminara Kafedry sistemnogo programmirovanija "Case-sredstva RTST++",* 1998, no. 1, pp. 37—52 (in Russian).
23. **Luciv D. V., Koznov D. V., Basit H. A., Terehov A. N.** Zadachi poiska nechjotkih povtorov pri organizacii povtornogo ispol'zovanija dokumentacii (Near duplicate search in documentation reuse), *Programmirovanie*, 2016, no. 4, pp. 39—49 (in Russian).
24. **Romanovsky K. Ju., Koznov D. V.** Jazyk DRL dlja proektirovanija i razrabotki dokumentacii semejstva programmnyh produktov (DRL language for design for software documentation development of product lines), *Vestnik Sankt-Peterburgskogo universiteta. Serija 10. Prikladnaja matematika. Informatika. Processy upravlenija*, 2007, no. 4, pp. 110—122 (in Russian).

УДК 004.457 DOI: 10.17587/prin.8.258-263

М. С. Аристов, мл. науч. сотр., e-mail: maristov@niisi.ras.ru, **Я. А. Зотов,** инженер, **Д. В. Яриков,** инженер, Федеральное государственное учреждение "Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук", Москва

Непрерывная интеграция программного обеспечения реального времени

Описан опыт внедрения инструментального средства для контроля версий GitLab и использование непрерывной интеграции программного обеспечения для автоматизации процесса тестирования. При разработке программного обеспечения реального времени используется подход, в котором его разработка и запуск выполняются на ЭВМ с разными архитектурами (кроссплатформенность). Такую особенность приходится учитывать при автоматизации тестирования программного обеспечения для многопроцессорных систем реального времени.

Ключевые слова: *непрерывная интеграция программного обеспечения, git, GitLab, программа реального времени, система контроля версий*

Введение

Федеральное государственное учреждение "Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук" (далее НИИСИ РАН) является разработчиком оборудования и базового программного обеспечения для различных прикладных систем. Результатом работы являются процессоры, микросхемы, процессорные модули, ЭВМ, операционные системы, целевые общесистемные средства (общесистемные программы). Разрабатываются также законченные фрагменты функционального программного обеспечения многомашинных вычислительных систем.

Трудоемкие процессы разработки и отладки общесистемного программного обеспечения требуют создания средств их автоматизации. Без таких средств разработка и отладка программ может занять недопустимо большое время или вообще оказаться незаконченной в силу необходимости использования для этого большого количества ресурсов.

Настоящая статья резюмирует опыт, полученный авторами при разработке и отладке общесистемного программного обеспечения, предназначенного для вычислительных систем реального времени. Специфика программных комплексов реального времени в контексте настоящей статьи проявляется в особенностях архитектуры системы и номенклатуры применяемых для ее создания программных средств.

Требования к среде разработки и тестирования

Среда разработки и тестирования должна обеспечивать многопользовательский режим разработки—отладки, включать средства слежения за ошибками (*bug-tracking*), иметь в составе модуль Code Review, обеспечивающий совместный анализ и редактирование исходных кодов, средства автоматизированного документирования и реализацию ряда других функций. Используемые в среде разработки и тестирования программного обеспечения современные системы контроля версий (*Concurrent Versions Systems, CVS*), кроме основных функций, поддерживают выполнение перечисленных выше требований.

Наибольшую популярность среди операционных систем (ОС) на инструментальных ЭВМ имеют семейства ОС Linux, Windows и macOS. Большие организации, разрабатывающие различное программное обеспечение, могут иметь в штате сотрудников, работающих на разных инструментальных ОС. По этой причине система контроля версий должна предоставлять необходимый для разработки набор функций независимо от используемой ОС.

Для средних и больших групп разработчиков программного обеспечения его актуальность как возможность *многопользовательской работы* с исходными текстами в режиме online приобретает важное значение. Системы, предоставляющие такую возможность, кроме всего прочего, должны иметь систему управления уровнями доступа. При росте компании увеличиваются число и сложность проектов. Как следствие, чтобы процесс разработки шел непрерывно, современная CVS должна поддерживать механизмы *масштабирования.*

Современное программное обеспечение имеет сложную структуру и большой объем исходных текстов, что приводит к появлению ошибок. Значительное упрощение работы над исправлением ошибок обеспечивает *система отслеживания ошибок bug-tracking.* Данная система должна предоставлять разработчикам функциональные возможности создания записей с описанием ошибок, а также возможности коммуникации команды для понимания ошибки и выбора исполнителя для ее исправления.

Кроме описанных выше систем, призванных улучшить качество разрабатываемого программного

обеспечения, существует подход Code Review. Он подразумевает, что программисты просматривают код друг друга и существует некоторая формальная процедура принятия кода. Такой подход позволяет устранить ошибки в программном обеспечении еще до процедур тестирования и поддерживать единый стиль программирования.

Одним из этапов разработки в жизненном цикле программного обеспечения является тестирование. Современная CVS должна поддерживать возможность *автоматизированной компиляции и тестирования разрабатываемого программного обеспечения*, а также хранения и оповещения разработчиков о результате работы. Это позволяет существенно оптимизировать деятельность группы разработчиков и отслеживать качество программного обеспечения.

При разработке программ, работающих в режиме реального времени, для тестирования необходимо запускать программное обеспечение на специализированном оборудовании. Одно и то же программное обеспечение нужно проверять на различных версиях ОС реального времени, различных составах аппаратного обеспечения и пакетов поддержки модулей. Однако возможность предоставить каждому разработчику отдельный испытательный стенд со всей возможной номенклатурой аппаратных средств не всегда реализуема. В связи с этим система автоматизированного тестирования должна предоставлять механизмы, позволяющие разграничить доступ тестировщиков к одному или нескольким испытательным стендам, а также автоматически, в пакетном режиме запускать тесты и сохранять результаты их работы.

К программному обеспечению, работающему в режиме реального времени, предъявляются не только требования по корректной работе, но и жесткие временные рамки, отведенные для каждого этапа вычислений. Это предъявляет дополнительное требование к автоматизированному тестированию, а именно — условие успешного прохождения теста не только с позиции корректного результата, но и в плане времени, затраченного на вычисления, которое должно укладываться в заранее определенный диапазон.

Разработка программного обеспечения подразумевает создание не только исходного кода, но и документации. Поэтому одним из требований к современным CVS является наличие *возможности ведения документации*.

Использование системы контроля версий и непрерывной интеграции при разработке и тестировании программного обеспечения

В НИИСИ РАН создана и активно эксплуатируется среда разработки, предназначенная для создания общесистемного программного обеспечения. В качестве системы контроля версий в этой среде используется программа GitLab [1, 2]. Она представляет собой веб-приложение, которое далее именуется системой, основанное на программе контроля версий Git, предоставляющей дополнительные возможности для разработчиков и менеджеров проекта.

Для загрузки и выгрузки кода можно пользоваться как веб-интерфейсом, так и кроссплатформенным приложением Git [3, 4], которое доступно для основных популярных ОС, таких как Linux, Windows и macOS. Веб-приложение предоставляет одинаковый интерфейс для современных браузеров, что позволяет удобно работать с GitLab, используя различные ОС, в том числе и мобильные, и тем самым обеспечивается кроссплатформенность процессов разработки.

Система контроля версий GitLab обладает всеми преимуществами многопользовательского режима. Каждый разработчик использует собственную учетную запись при работе в этом приложении. При управлении проектом можно указать уровень доступа для каждой учетной записи, что позволяет использовать предоставляемые приложением средства разработки в многопользовательском режиме. В приложении GitLab существует также групповая политика: пользователи могут создавать группы, объединяющие несколько проектов схожей тематики, добавлять в них других участников и разграничивать для них уровни доступа (рис. 1).

Веб-приложение GitLab позволяет эффективно организовать работу в больших командах разработчиков, может быть настроено для работы с несколькими крупными проектами одновременно, успешно обрабатывает файлы большого размера.

В целях учета и контроля ошибок, выявления некорректной работы разрабатываемого программного обеспечения, для получения обратной связи от пользователей, а также для слежения за процессом устранения ошибок и выполнения пожеланий в GitLab присутствуют механизмы отслеживания ошибок (*bug-tracking*) (рис. 2). Эти механизмы позволяют пользователям, имеющим доступ к репозиторию проекта, оставлять заметки, где они могут детально описать возникающие у них сложности или пожелания. В целях облегчения поиска по разделу к заметкам можно добавить пользовательские метки. Разработчики при необходимости могут назначать ответственного исполнителя для каждой ошибки и этапа проекта, а также дату, когда данная ошибка должна быть устранена. К каждой заметке можно оставлять комментарии, что облегчает взаимодействие и ускоряет разрешение тех или иных вопросов.

Планировщик задач предоставляет возможности для разработчика и других участников проекта следить за исполнением тех или иных задач, устанавливать соответствующие им подзадачи и сроки их решения, а также возможности комментировать действия.

Работа с Git, как правило, предполагает наличие основной ветки разработки (*master*), а также веток, либо индивидуальных для каждого разработчика, либо предназначенных для решения конкретных задач, таких как исправление ошибок или добавление новых возможностей в программу. При совмещении кода от разных разработчиков из разных веток используется механизм Merge Requests (рис. 3). Разработчик выбирает ветки, которые должны быть объединены, и отправляет запрос на их слияние. Разработчики могут участвовать в обсуждении вопросов, возникающих в процессе создания кода, оставляя комментарии непосредственно к строкам кода с изменениями. Уже прошедшие обсуждение вопросы можно пометить как разрешенные. Разработ-

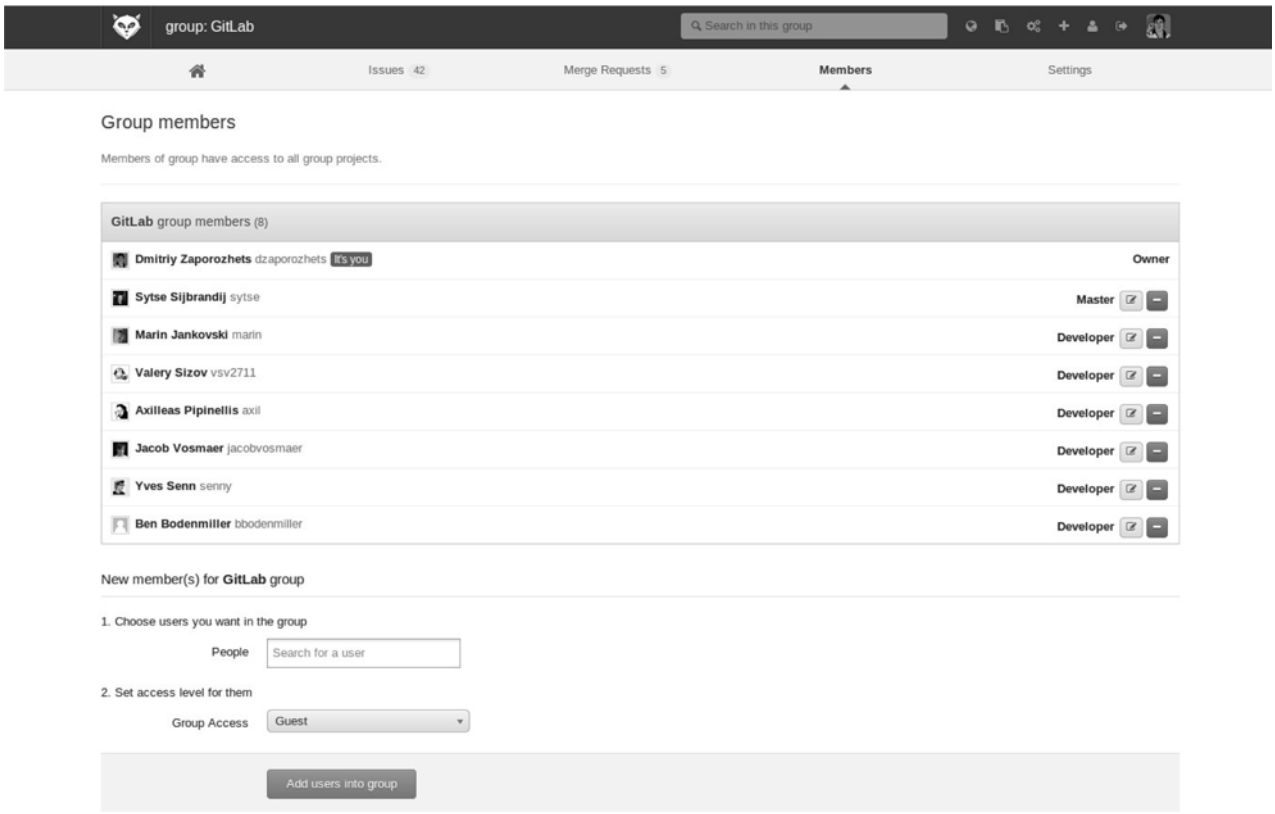

Рис. 1. Система контроля версий GitLab: групповая политика

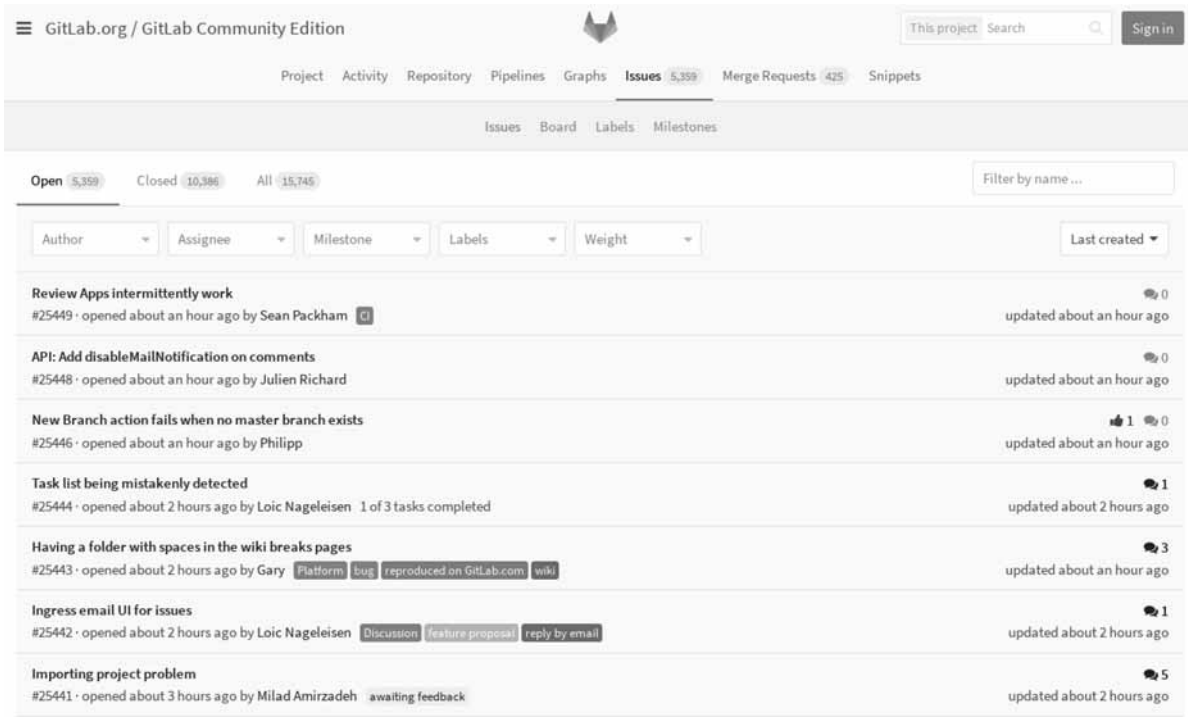

Рис. 2. Система контроля версий GitLab: механизмы отслеживания ошибок

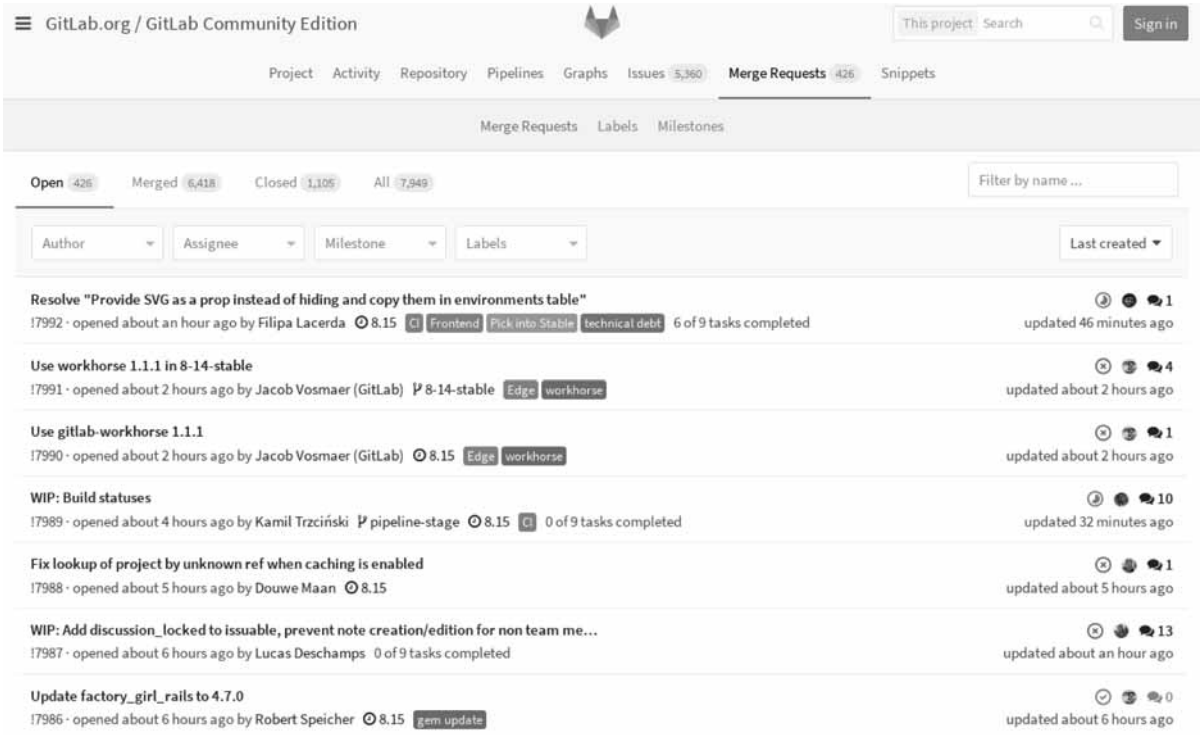

Рис. 3. Система контроля версий GitLab: механизм Merge Requests

чик, имеющий соответствующие права доступа, может принять запрос, после чего система GitLab автоматически объединит ветки. Таким образом, механизм Merge Requests предоставляет возможности оценки и обсуждения кода с участием других участников проекта, которые реализует механизм Code Review.

Для удобства работы группы из нескольких разработчиков над одним проектом в GitLab используется подход на основе ведения и контроля веток. В проекте кроме основной ветки может быть несколько побочных веток. Каждая ветка содержит отдельную версию файлов и позволяет вносить изменения, не затрагивая другие ветки проекта. Такой подход может быть полезен не только команде, но и отдельному разработчику. Например, разработчик может создать ветку для исправления ошибки или добавления экспериментальной возможности. Пока эти изменения не будут отлажены, их можно держать в данной ветке, и уже после того, как они будут работать корректно, "слить" в основную ветку. При объединении веток изменения, которые не вызывают конфликтов (а именно — не затрагивают одинаковые участки кода), объединяются в автоматическом режиме. В ином случае разработчику предлагается разрешить конфликты, т. е. исправить участки кода, которые не могут быть объединены автоматически (рис. 4).

Кроме работы командой над одним проектом представляется возможность создать ответвление (*fork*) для конкретного разработчика. В таком случае у программиста будет своя копия проекта, содержащая код на момент своего создания. После этого каждая из копий проекта будет существовать независимо, но сохранять при этом возможность внесения изменений из созданной копии в основной проект. Этот механизм может быть удобен

```
266
267 <<<<<<< HEAD
268         $("#new").click(function(){
269 =======
270         $(el).find("#new").click(functio
271 >>>>>>> Convert list to use presenter
272           $("tr input[type=text]:first")
```

Рис. 4. Система контроля версий GitLab: механизмы разрешения конфликтов

в том случае, когда есть сторонние разработчики, которым нежелательно предоставлять возможность изменять основной проект. Он позволяет таким разработчикам отдельно добавлять улучшения или исправлять ошибки, предложив после этого головному разработчику добавить изменения или создать собственную версию проекта, принимая из основной версии исправления и улучшения.

Для создания и ведения документации в удобном для чтения и редактирования формате, включая создание справочных и информационных страниц, GitLab использует облегченный язык разметки Markdown и wiki-разметку.

В состав GitLab входит GitLab Continuous Integration (GitLab CI) — средство автоматизации тестирования и сборки разрабатываемого программного обеспечения. Его механизмы предоставляют возможности запуска процесса автоматического тестирования и сборки проекта при наступлении определенных событий. Как правило, такими событиями являются изменения в исходном коде. Если тестирование или сборка выполнились с ошибкой, предусмотрена отправка e-mail-сообщений по указанному списку адресов.

Инструментальное средство GitLab Runner — это отдельное приложение, осуществляющее выполнение скриптов сборки. Один или несколько экземпляров Runner могут быть установлены на одном или нескольких серверах. Эти экземпляры взаимодействуют с помощью интерфейса API. Средство GitLab Runner написано на языке Go и может быть установлено в среду ОС, для которой можно собрать приложение, написанное на языке Go, например, в среду ОС Linux, macOS, Windows и др. Для наиболее популярных ОС предоставляются готовые пакеты установки, а для тех ОС, для которых пакеты не предоставляются, Runner возможно собрать из исходных кодов. С помощью Runner можно выполнять тестирование программного обеспечения, написанного на различных языках программирования, включая .Net, Java, Python, C, C++, PHP и др. Скрипты, выполняемые при автоматизированном тестировании, могут быть написаны как на языке Shell(Bash) Script, так и с использованием таких приложений, как Make, CMake и подобных.

После установки и конфигурирования запущенный Runner ожидает от GitLab команд на сборку и тестирование проекта. Конфигурирование процесса сборки и тестирования проекта выполняется в конфигурационном файле **gitlab-ci.yml**, расположенном в корне проекта. Конфигурационный файл имеет формат YAML [5, 6]. В этом файле можно задать команды для выполнения тестирования, а также команды, выполняемые до запуска тестирования и после завершения тестирования, как успешного, так и окончившегося ошибкой.

Пример конфигурационного файла:

```
stages:
- build
- cleanup _ build
build _ job:
  stage: build
  script:
  — make build
cleanup _ build _ job:
  stage: cleanup _ build
  script:
  — make cleanup
when: on _ failure
```

В данном примере определены две стадии — **build** и **cleanup _ build**. Стадия **build** выполняет команду **make build**, стадия **cleanup _ build** — команду **make cleanup**. Стадия **cleanup _ build** запускается, если стадия **build** завершилась с ошибкой.

Для автоматизации тестирования программного обеспечения, предназначенного для работы в условиях жесткого реального времени, были созданы в **Make**-файлах следующие цели (*target*):

1) сборка исходных кодов под архитектуру x86;

2) сборка и запуск тестов на тестирующей машине;

3) сборка исходных кодов под целевую архитектуру;

4) сборка и запуск тестов на целевой машине.

Первая цель выполняет сборку исходных кодов и, при успешном ее выполнении, создает программное обеспечение в загрузочном виде для архитектуры x86. Вторая цель выполняет сборку и запуск набора тестов на архитектуре x86. Третья цель выполняет сборку исходных кодов и, при успешном ее выполнении, создает программное обеспечение в загрузочном виде для целевой архитектуры. Четвертая цель запускает тестирование на испытательном стенде.

При тестировании программного обеспечения на целевой архитектуре успехом считается получение ожидаемого результата за заранее определенный промежуток времени. Реализация каждой последующей цели происходит только в случае успешного выполнения предыдущей. Тестирование автоматически запускается в случае любого изменения в исходных текстах программы. Наличие таких механизмов освободило разработчиков от ручной сборки и запуска тестов, которые в сумме выполняются несколько часов. При возникновении ошибок на электронную почту разработчику в соответствии со списком указанных адресов отправляется информация об ошибке, полученной в ходе работы цели, как если бы она была запущена на локальном компьютере.

Автоматизированное тестирование библиотек цифровой обработки сигналов

В НИИСИ РАН при разработке библиотек цифровой обработки сигналов используют инструментальные средства их автоматизированного тестирования. Библиотеки, предназначенные для цифровой обработки изображений и сигналов, представляют собой программные продукты, которые написаны на языке программирования Си. Это библиотеки базовых математических функций для работы с векторами и матрицами целых, вещественных и комплексных чисел (быстрое преобразование Фурье, скалярные и векторные произведения матриц и т. п.) и типовых алгоритмов, предназначенные для решения задач обработки сигналов. Каждая библиотека предоставляет интерфейс, состоящий из нескольких десятков функций.

Библиотеки включают в себя наборы тестов — файлов исходного кода на языке Си. Эти тесты содержат контрольные задачи и предназначены для проверки работоспособности каждой функции библиотеки по отдельности в искусственной среде.

При выходе новой версии ОС реального времени или при обновлении библиотек появляется необходимость проверить совместимость ОС реального времени с указанными выше библиотеками, а также их работоспособность.

Применяют следующие приемы автоматизации тестирования.

1. Сборка и запуск на инструментальной ЭВМ. Библиотеки написаны таким образом, что их можно скомпилировать, собрать и запустить на инструментальной ЭВМ. При данном подходе в результате сборки библиотек получается исполняемый файл, содержащий программу поочередного запуска всех

доступных тестов. В случае неудачного завершения одного из тестов механизмы рассматриваемого инструментального средства автоматически прерывают тестирование и уведомляют разработчиков об ошибке. Это позволяет провести быстрое тестирование программы без необходимости запуска на целевом стенде.

2. Сборка и запуск на целевой ЭВМ. Библиотеки компилируются и собираются под целевую архитектуру 1890ВМ7Я с использованием ОС реального времени семейства Багет 3.х и всех доступных тестов, написанных для этой платформы. Полученный таким способом образ ОС реального времени запускается на целевом стендовом оборудовании и поочередно запускает тесты для каждой функции библиотек. В случае их успешного выполнения обновляется статус проекта в GitLab на "успех". В случае ошибки устанавливается статус "ошибка", а на электронную почту разработчиков отправляются письма с уведомлением о возникновении ошибки. Таким образом программное обеспечение проверяется на работоспособность на целевой архитектуре и на соответствие предъявляемым к нему требованиям.

Заключение

Использование GitLab и механизмов непрерывной интеграции позволило существенно оптимизировать трудозатраты на тестирование программ, повысить качество программных продуктов за счет частичного исключения человеческого фактора в процессах тестирования. На проектах, которые разрабатываются без использования подхода непрерывной интеграции, время, затрачиваемое на тестирование программного обеспечения, составляет 25...30 % от общего рабочего времени. При выполнении проектов, сконфигурированных с использованием автоматизированного тестирования, экономится время разработчиков на тестирование промежуточного кода, а общее время, затраченное на тестирование проекта в целом, составляет не более 5 % от общего рабочего времени.

Список литературы

1. **Baarsen J.** GitLab Cookbook. Packt Publishing, 2014. 172 p.
2. **Gitlab** Documentation — Gitlab-CI Quick Start, URL: http://docs.gitlab.com/ce/ci/quick_start/README.html
3. **Olsson A., Voss R.** Git Version Control Cookbook, Packt Publishing, 2014. 340 p.
4. **Git** Documentation, URL: https://git-scm.com/doc
5. **Ben-Kiki O., Evans C., Ingerson B.** Yaml Ain't Markup Language (YAML^{TM}) version 1.2. YAML.org, Tech. Rep., September 2009.
6. **YAML** Ain't Markup Language (YAML^{TM}) Version 1.2, URL: http://www.yaml.org/spec/1.2/spec.html

Real-Time Software Continuous Integration

M. S. Aristov, maristov@niisi.ras.ru, **Ya. A. Zotov, D. V. Yarikov,** Federal State Institution "Scientific Research Institute of System Analysis of the Russian Academy of Science", Moscow, 117218, Russian Federation

Corresponding author:

Aristov Mikhail S., Researcher Fellow, Scientific Research Institute of System Development, Moscow, 117218, Russian Federation

E-mail: maristov@niisi.ras.ru

Received on February 02, 2017
Accepted on March 06, 2017

The article describes the experience of implementing the tools for project management system GitLab and the usage of continuous software integration for automatization of test process. In case of real-time systems, software development and execution are performed on hardware with different architectures, this approach is called cross-platform and its properties (such as software development and execution on different devices using specified software tools and hardware resources) should be taken into account during automatization of testing of software for multiprocessor real-time systems. In addition special aspects of GitLab components usage (such as Bug-tracking, Code Review, Continuous Integration) are considered in the context of cross-platform development. Those components are basically designed for software development teams, but it does not mean that a project created by a single software developer can't be controlled by the software tool considered. Quite the opposite, a single software developer can take advantage of all the features provided by the version control system and gain the same benefit as a big development team.

Keywords: *continuous integration, git, GitLab, real time software, version control system*

For citation:

Aristov M. S., Zotov Ya. A., Yarikov D. V. Real-Time Software Continuous Integration, *Programmnaya Ingeneria,* 2017, vol. 8, no. 6, pp. 258—263.

DOI: 10.17587/prin.8.258-263

References

1. **Baarsen J.** *GitLab Cookbook*, Packt Publishing, 2014, 172 p.
2. **Gitlab** Documentation — Gitlab-CI Quick Start, available at: http://docs.gitlab.com/ce/ci/quick_start/README.html
3. **Olsson A., Voss R.** *Git Version Control Cookbook*, Packt Publishing, 2014, 340 p.
4. **Git** Documentation, available at: https://git-scm.com/doc
5. **Ben-Kiki O., Evans C., Ingerson B.** Yaml Ain't Markup Language (YAML^{TM}) version 1.2. YAML.org, Tech. Rep., September 2009.
6. **YAML** Ain't Markup Language (YAML^{TM}) Version 1.2, available at: http://www.yaml.org/spec/1.2/spec.html

УДК 004.9 DOI: 10.17587/prin.8.264-275

В. А. Васенин, д-р физ.-мат. наук, проф., **А. Э. Гаспарянц**, аспирант,
e-mail: artem.gaspariants@gmail.com, МГУ имени М. В. Ломоносова

Разрешение неоднозначности имен авторов: анализ публикаций

Статья содержит краткий обзор и анализ публикаций, которые рассматривают задачу разрешения неоднозначности имен авторов. В работе проведена классификация методов решения этой задачи, отмечены преимущества и недостатки подходов, выделены открытые вопросы в данной области.

Ключевые слова: *показатели цитирования, библиографическая запись, машинное обучение, связывание записей, связывание именованных сущностей*

Введение

Публикационная активность является одним из системообразующих факторов, характеризующих эффективность деятельности как отдельного ученого, так и коллектива исследователей, а в конечном итоге и научный потенциал отдельной организации и страны в целом. Как следствие, в последние годы исследования, направленные на определение темпов роста числа научных изданий и публикуемых в них статей, а также на оценки уровня цитируемости их авторов приобретают практическую значимость. Количество издаваемой научной литературы растет, поэтому сделать это с использованием традиционных методов библиотечной каталогизации и ручного анализа, без помощи современных средств автоматизации подобных процессов в настоящее время не представляется возможным. По этой причине особую значимость приобретают модели, методы и средства такой автоматизации с использованием наукометрических показателей (индикаторов).

Наукометрические показатели являются инструментальными средствами для анализа эффективности научной деятельности, актуальности и востребованности направлений исследований по той или иной тематике. Поскольку они прямо или косвенно зависят от числа цитирований и числа публикаций в научных журналах, возможность эффективно вычислять их значения с минимальным участием человека является немаловажным фактором. С учетом изложенных выше причин количественные оценки результатов научной деятельности являются одним из основных аргументов для принятия управленческих решений в организации науки и высших учебных заведений. К числу таких решений относятся: предоставление отдельным ученым, научным группам и организациям дополнительных финансовых средств, стимулирование работ на отдельных направлениях и т. д. Например, к числу наукометрических индикаторов относится индекс Хирша, который определяется как максимальное число h, такое, что h самых цитируемых статей ученого цитируются как минимум h раз каждая, в то время как оставшиеся статьи цитируются не более чем h раз каждая. Следует отметить, что число цитирований научных публикаций сильно зависит от области знаний. Тем не менее в пределах одной научной области эти показатели являются наиболее востребованными и приняты в настоящее время в качестве стандартных индикаторов для оценки эффективности научного исследования.

Сложность построения автоматизированных средств вычисления показателей цитирования состоит в отсутствии возможности собирать достоверную информацию о публикации без вмешательства человека. Для крупных информационно-аналитических систем (ИАС), таких как ИСТИНА — Интеллектуальная Система Тематического Исследования НАукометрических данных [1, 2], эта сложность является одной из определяющих. В крупных ИАС подобного назначения хранятся записи о сотнях тысяч публикаций и об их авторах. Естественно, каждый автор, чья публикация представлена в крупной ИАС, должен получить некоторую запись автора в ИАС (или профиль), содержащую все опубликованные результаты его научной деятельности. Очевидно, что каждая публикация должна быть представлена в единственном экземпляре. Информацией о том, кто является автором этой статьи, кроме него изначально владеет некоторая группа лиц (авторы, издатели и т. д.), но не собственно ИАС. Эти люди способны указать для каждого соавтора публикации, какая запись в ИАС среди множества записей однофамильцев является достоверной для данной работы. Тем не менее требование вручную "находить" все профили соавторов и "привязывать" публикацию к ним очевидно скажется на темпах пополнения базы данных публикаций и на желании пользователей работать с ИАС. Поэтому, в крупных системах, ведущих учет научных публикаций, либо отсутствует требование "привязывать" статью к кому-либо из соавторов, оставляя это на автоматические средства, либо требуется привязать хотя бы одного из соавторов. Наличие однофамильцев

и, зачастую, неполное написание имени автора публикации приводит к тому, что в ИАС появляется много публикаций, которым приписан автор с одним именем. В этом случае необходимо определить, какие из данного набора публикаций относятся к тем или иным авторам с данным именем.

Описанная задача изначально строится на предположении, что существует некоторое имя автора, которое встречается в большом числе работ. По этому имени необходимо построить множество записей авторов для реализации на их основе механизмов автоматического вычисления показателей цитирования. Предположим, что из всего многообразия результатов научной деятельности в отдельной ИАС некоторым способом получен набор документов, к которым относятся не только статьи, но и книги, доклады и т. д. Предположим также, что этим документам поставлено в соответствие некоторое имя, которое назовем *первичным*. Следует отметить, что доступная информация о каждом документе зачастую минимальна: название работы; журнал, в котором она опубликована; имена (возможно, не полные) соавторов. Результатом обработки этого множества документов должно стать разбиение множества на группы документов, в каждой из которых документы принадлежат одному человеку под первичным именем.

В настоящем обзоре представлены различные подходы к решению описанной выше задачи, которая рассматривается в основном для следующих двух сфер: во-первых, разрешение неоднозначности имен авторов с целью определения множества людей с первичным именем и нахождение результатов их научной деятельности, во-вторых, определение дублетных записей (дубликатов), относящихся к одному человеку. Во втором случае каждая из дублетных записей может соответствовать некоторой части работ одного автора. Требуется объединить все дублетные записи, чтобы получить полное представление обо всех результатах работы данного автора. Поскольку данная задача естественно описывается в терминах кластеризации данных, множество методов ее решения основано на этом подходе. Рассмотрев множество методов, можно сделать вывод, что в целом публикации по данной теме поддаются классификации по методам решения. В задаче кластеризации требуется определить функцию близости точек в пространстве, в котором необходимо построить кластеры. Основные различия исследований, основанных на кластеризации, состоят в выборе функции близости документов и выборе способа кластеризации. Для оценки близости пары документов используется множество атрибутов, таких как место работы автора, издательства, электронные адреса и т. д. К одной группе работ можно отнести те, которые используют методы обучения с учителем, непосредственно обучая функцию близости по размеченным данным, другие используют обучение без учителя. Еще одну группу составляют работы, использующие частичное обучение. Среди оставшихся работ можно выделить те, которые определяют функцию близости с использованием вероятностных методов, оценивая распределения различных атрибутов статей и сравнивая полученные распределения между собой.

Целью представленного в настоящей публикации краткого обзора является классификация применявшихся в этой области методов и подходов к решению задачи, их сравнение и выделение открытых вопросов, которые еще необходимо решить. Описанные далее подходы помогут получить более полную картину текущей ситуации по данной тематике и могут сигнализировать о преимуществах или недостатках отдельных методов. В связи с этим обстоятельством практическая польза от рассмотрения результатов, представленных в обзоре, состоит в рекомендации по выбору наиболее эффективных и перспективных методов для реализации при решении практических задач.

1. Формулировка задачи

Формально опишем постановку задачи разрешения неоднозначности имен авторов. Пусть дано множество документов $D = \{d_1, ..., d_N\}$. Среди авторов этих документов есть авторы с *неоднозначным* именем, т. е. одно или более имен встречается во всех документах. Предполагается, что никакой дополнительной или вспомогательной информации (например, о том, что некоторые из этих одинаковых имен действительно принадлежат одному автору) не предоставлено. Кроме того, не представлена история публикаций авторов с неоднозначным именем. Требуется разбить это множество документов на непересекающиеся подмножества таким образом, чтобы документы, принадлежащие одному кластеру, были написаны одним и тем же автором. Обзор составлен следующим образом: результаты исследований, представленных в публикациях, сгруппированы по методам решения; в начале обзора каждой группы отмечается, какими методами пользовались авторы этих публикаций, чем эти методы различаются. Описаны различия в постановке задачи, если они имеются.

2. Обзор методов решения поставленной задачи

Описанная выше задача является частным случаем более общей задачи под названием *задача связывания записей* (*problem of record linkage*). В 1969 г. Ivan P. Fellegi и Alan B. Sunter в работе [3] одними из первых предложили математическую модель для получения ответа на вопрос, представляют ли записи в двух файлах идентичную сущность. Каждая запись в файле представляется набором характеристик, которые можно сравнивать друг с другом для принятия одного из вариантов решений: положительное решение (записи являются дублетными), отрицательное решение, возможное решение — данных характеристик недостаточно для принятия решения. Предлагается построение правила, которое будет принимать решения по парам записей с заранее определенной вероятностью ошибок. Для каждой пары значений вероятностей ошибок строится класс решающих правил. Необходимо найти представителя этого класса, который минимизирует вероятность того, что решение не будет принято, т. е. признает, что недостаточно данных для принятия решения. Такое правило называется *опти-*

мальным. В работе [3] представлена теорема, показывающая способ построения такого *оптимального* правила. Некоторые методы, предлагаемые в этой работе, встречаются и в современных статьях по этой теме.

Отдельной подзадачей является сбор документов для обучения и проведения экспериментов. Отметим, что получение текстовой информации о публикациях не является затруднительным, потому что существует немало электронных библиотек, предоставляющих доступ к своим данным. Например, в работе [4] описано построение поискового робота, просматривающего электронные библиотеки ACM, DBLP, IEEE. Поисковая система Google Scholar использовалась в работе [5] для выборки небольшого множества имен авторов и их публикаций, а в работе [6] рассматривали записи из Web of Science (WoS). Большинство работ использовали данные, собранные из различных библиотек и подготовленные для экспериментальной обработки. Однако разметка подмножества данных для обучения или тестирования является затруднительной. Для того чтобы удостовериться, относится ли имя автора в двух статьях к одному человеку, требуется немало усилий. Отсутствие общедоступной выборки, на которой можно было бы тестировать метод, также затрудняет не только процесс обучения, но и сравнение результатов из разных работ. Использование частичного обучения должно облегчить эту ситуацию и уже в настоящем показывает многообещающие результаты.

2.1. Показатели оценки результатов

Перед тем как приступить к описанию и анализу методов и подходов к решению поставленной выше задачи, опишем, какие показатели в основном используют при оценке результатов работы алгоритма, решающего данную задачу.

Наиболее используемыми величинами являются *точность* (*precision*) и *полнота* (*recall*), *F-мера* и *accuracy*, которые определяются следующим образом:

$$recall = \frac{tp}{fp+tp},$$

$$precision = \frac{tp}{fn+tp},$$

$$accuracy = \frac{tp+tn}{t}, \quad (1)$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}. \quad (2)$$

Здесь *tp* обозначает число пар документов, правильно классифицированных как относящиеся к одному автору; *fn* — число пар документов, неправильно классифицированных как относящиеся к разным авторам; *fp* — число пар документов, неправильно классифицированных как относящиеся к одному автору; *tn* — число пар документов, правильно классифицированных как относящиеся к разным авторам; *t* — общее число пар документов.

Помимо указанных оценок используется *k-метрика* [7] — индикатор определения правильности кластеризации:

$$k = \sqrt{ACP \cdot AAP}, \quad (3)$$

где ACP — средняя степень "чистоты" кластеров (*average cluster purity*), которая определяется как

$$ACP = \frac{1}{N}\sum_{i=1}^{q}\sum_{j=1}^{R}\frac{n_{ij}^2}{n_i};$$

AAP — средняя степень "чистоты" автора (*average author purity*), которая определяется по формуле

$$AAP = \frac{1}{N}\sum_{j=1}^{R}\sum_{i=1}^{q}\frac{n_{ij}^2}{n_i}.$$

Здесь N — общее число записей в одной группе; R — известное число кластеров; q — число кластеров, полученное в результате работы алгоритма; n_{ij} — число элементов в кластере i — эталонном (правильный ответ), которые также принадлежат кластеру j из ответа, полученного с помощью алгоритма; n_i — число элементов в кластере i из правильного ответа.

2.2. Методы, использующие обучение без учителя

Во всех рассмотренных публикациях, оценивающих близость пары документов, используются атрибуты, являющиеся метаданными документа. К их числу относятся: название публикации, имена соавторов, название журнала и другая доступная вспомогательная информация. Текст документа во внимание не принимается, поскольку, как правило, он недоступен для просмотра и анализа. Многие исследования делают акцент на том, что необходимо расширять множество метаданных и использовать для решения задачи всю доступную информацию. Некоторые исследования из этой группы проводят кластеризацию в следующем виде: процесс состоит из нескольких шагов; на первом шаге каждый документ составляет отдельный кластер; на каждом последующем шаге к уже сформированным кластерам применяется функция оценки близости кластеров, сравнивающая их по новому атрибуту (например, имена соавторов). К таким работам относится работа [4], где для каждого первичного имени строится лес деревьев, в котором отдельное дерево содержит в корне первичное имя, а узлы дерева содержат имена соавторов (изначально известны некоторые подмножества работ авторов с указанным первичным именем). Дерево изображает иерархическую структуру соавторства. Перед началом работы алгоритма каждое дерево объявляется кластером, после чего деревья сравниваются на близость и объединяются, если содержат общие узлы (т. е. содержащие одинаковое имя) и эти узлы находятся недалеко от корня. Схожий метод применен в работе [5]. В этой работе использован дополнительный атрибут — адрес электронной почты соавторов. При этом вместо точного сравнения двух адресов предлагается использовать строковую метрику для оценки их схожести. Кластеризация строится следующим образом: на первом шаге с использованием близости адресов электронной почты авторов создаются кластеры, далее полу-

ченные кластеры сливаются на основании близости места работы соавторов, на третьем шаге используется оценка близости имен соавторов. В работе [8] также использованы дополнительные атрибуты: множество адресов электронной почты авторов; место публикации; год публикации; списки литературы. Как и в работах, отмеченных ранее, кластеризация состоит из нескольких шагов, на каждом шаге для сравнения используется один из атрибутов. При этом для сравнения строк используются меры *Jaro-Winkler* и *Cosine-Similarity*. Для сравнения адресов электронной почты применяется следующий алгоритм: строится матрица, у которой на позиции (*i, j*) стоит 0 или 1, в зависимости от того, указан ли одинаковый электронный адрес у авторов в данной паре публикаций. С помощью алгоритма Флойда-Варшалла эта битовая матрица может быть преобразована в матрицу *достижимости* (*reachibility*). Матрица далее применяется для определения близости адресов электронной почты соавторов в двух кластерах. Поскольку заведомо число кластеров, которое должно получиться, неизвестно, в качестве метода кластеризации в основном использовалась иерархическая кластеризация. Так, например, в работе [9] описан следующий алгоритм.

- Каждый документ назначают в отдельный кластер.
- Документы, у которых кроме первичного имени совпадает еще имя хотя бы одного соавтора, объявляют близкими, и кластеры, образованные ими, объединяют.
- Кластеры сравнивают на близость по названиям публикаций, по местам их издательств и научных организаций (в качестве места работы).
- Кластеры сравнивают по функции на графе имен. В этом графе каждое уникальное имя образует вершину графа. Вершины образуют ребро, если авторы с соответствующими вершинам именами являются соавторами некоторой публикации. Если между двумя вершинами, содержащими близкие в метрике *Jaccard* имена, существует путь длины менее трех, то их объявляют близкими.
- Проверяют слова в заглавии публикации и в указании места работы (с помощью TF-IDF-меры). Если они близки, то соответствующие кластеры объединяют.

Как было отмечено выше, сложность выборки тестовых данных является общей, и в этой работе эксперименты проводили на множестве документов, выбранных по 11 первичным именам. В среднем одному из имен соответствовало 20 разных авторов, среднее общее число их публикаций составило 388. Оценку качества кластеризации проводили по *k*-метрике (3). В результате эксперимента среднее значение *k*-метрики на тестах оказалось равным 0,77.

Иерархическую кластеризацию применяли и в работе [6]. Сначала определяют функцию s_{ij} оценки близости двух документов, зависящую от следующих параметров: список соавторов; списки литературы; список работ, цитирующих данную. Кроме этого, в этом подходе используют несколько числовых значений, которые потом оптимизируются. На первом шаге проводят кластеризацию документов по значениям данной функции близости. Пара документов добавляется в один кластер, если степень их близости выше некоторой границы θ_1. В результате получают граф, в котором вершина — пара статей, а ребро между парой вершин существует, если в каждой из двух вершин находится одинаковый документ. На втором шаге каждую связную компоненту графа объявляют отдельным кластером. Определяют значение θ_2 и кластеры сравнивают на степень близости по функции $\sum_{s_{ij}>\theta_2} s_{ij}$. Если значение этой функции выше величины θ_3, то данные кластеры объединяют в один. После этого рассматривают документы, не попавшие ни в один кластер. Работу, не добавленную ни в один кластер, добавляют к кластеру, если s_{ij} между данной работой и хотя бы одной работой из этого кластера выше границы θ_4. Все определенные величины оптимизируют, минимизируя следующие ошибки: документы, написанные одним автором с первичным именем, оказались в разных кластерах; документы, написанные разными авторами с первичным именем, попали в один кластер. Кроме того, в этой работе параметры оптимизируют на основании вычисления индекса Хирша, т. е. документы, участвующие в определении индекса Хирша, имеют больший "приоритет".

Следует также отметить работу [10], в которой рассматривали задачу добавления новой статьи и ее привязки к базе данных. Данная задача рассматривается как частный случай поставленной выше. Вместо того чтобы решать задачу неоднозначности имен для всех работ в базе данных, предложено разрешить ее только для новой статьи. Для каждого из имен авторов новой статьи выбирают ссылки на известных (базе данных) авторов с похожим именем, имеющим хотя бы одного общего по имени соавтора и похожее название работы или место издательства. Если ни одно из этих сравнений не срабатывает, считают, что соответствующее имя автора принадлежит новому автору. Для оценки результатов были собраны два множества публикаций. Одно из них сгенерировано искусственно с использованием средства SyGAR, второе состояло из 363 публикаций, выбранных из электронной библиотеки BDBComp. Предложенный алгоритм сравнивали с методом HHC (эвристическая иерархическая кластеризация), предложенным в работе [7]. Оценки проводились по *k*-метрике (3), предлагаемый метод показал более высокую точность с наилучшим средним результатом $k = 0{,}83$ благодаря более высоким значениям *ACP*, чем полученные в работе [7]. При этом оценка точности по величине *AAP* оказалась достаточно невелика, это свидетельствует о том, что алгоритм имеет тенденцию к фрагментации кластеров. В работе представлены распространенные случаи ошибок работы алгоритма, например, ошибочное отнесение автора статьи к новым авторам, вследствие того, что он впервые публикуется с данным соавтором. При этом некоторые другие атрибуты (например, издательство) могут совпадать. Возможно, эта ошибка возникает в силу вручную подобранных функций сравнения кластеров на близость, которые не всегда соотносятся с данными.

В описанных работах проводили эксперименты разного масштаба. Например, в работе [4] результаты

представлены для множества работ с единственным первичным именем *Ken Barker*. В работах [5, 8] результаты оценивали по значениям показателей (1), (2). При этом, поскольку число реальных кластеров и число разных имен, на которых проводился эксперимент, слишком малы, данный эксперимент нельзя относить к чистым. В работе [7] оценка точности кластеризации включала в себя ручную проверку правильности кластеризации: использование в качестве подтверждения того факта, что два имени относятся к одному человеку, если их вторые инициалы совпадают, а также использование адресов электронной почты в качестве подтверждения правильности кластеризации. Достоверно оценить точность алгоритма сложно ввиду отсутствия эталонного теста.

2.3. Методы, использующие обучение с учителем

Общей чертой работ, представленных в этой группе, является использование методов обучения с учителем для нахождения функции, задающей близость между парой документов. Если определение близости пары документов не является самостоятельной задачей, значения этой функции на парах документов задают расстояния между документами и используются в алгоритме кластеризации.

В исследованиях этой группы наиболее распространено использование классификатора *SVM*: в публикациях [9, 11—13] использовали классификатор *SVM*, в работе [14] использовали бинарный классификатор *C-SVC* с *RBF*-функцией. Среди остальных методов в работе [15] использовали логистическую регрессию, в работе [16] — классификатор *Random Forest*, в работе [13] рассматривали еще один подход с использованием *Naive Bayes*. В работе [17] применяли глубинные нейронные сети.

Основные различия перечисленных методов состоят в определении пространства признаков, в использовании различных атрибутов в качестве признаков. Так, например, в работе [14] отмечено, что два документа более вероятно написаны одним автором, если они относятся к близкой тематике и представлены на одной веб-странице. Поэтому из документов извлекают темы, к которым они относятся. По этим темам создают сеть связей научных тематик, используя правила (определенные заранее), например, "машинное обучение" => "искусственный интеллект". Полученный граф является сильно связным, так как две тематики обычно оказываются связанными, даже если они не близки. В работе применяли алгоритм *hMETIS* [18], алгоритм разбиения гиперграфа, чтобы разбить гиперграф на несколько кластеров, в которых находятся только тесно связанные друг с другом научные темы. Далее, оценка тематик работ с использованием данного графа служит в качестве одного из признаков близости документов, как и название документов, имена соавторов, место публикации, близость в веб-странице. На полученном наборе признаков обучается классификатор *C-SVC*. Следует отметить, что при подходе к определению близости документов через классификацию возможно появление отсутствия транзитивности. Такая сложность возникает, когда есть три документа, которые сравнивают на близость. Три документа разбивают на три пары, две пары из которых считают близкими по значению классификатора, тогда как для третьей пары публикаций классификатор сигнализирует, что эти работы не могут считаться близкими. Для преодоления этой сложности в работе [16] применен алгоритм кластеризации DBSCAN [19]. Классификатор в этой работе строится по следующим признакам: адрес электронной почты (*edit distance*); место работы (по мере *token based Jaccard*), имена соавторов (гибридный *Soft-TFIDF*).

Решаемую задачу можно рассматривать и в контексте определения дубликатов записей в патентной базе данных. В работе [16] описан метод решения такой задачи. Его основное отличие состоит в том, что в патентных записях доступно больше информации и она лучше структурирована, например, известны полное имя владельца (включая суффикс, должность); правопреемник; группа. Оценивая разные алгоритмы классификации, авторы этой работы получили наибольшую точность, используя алгоритм *Random Forest*. В некоторых случаях, когда в ИАС хранятся записи о множестве авторов с первичным именем, можно исключить этап кластеризации и использовать классификацию с несколькими классами, с записями об авторах в качестве классов, чтобы относить каждый из документов к одному из классов. В работе [11] выбирали данные из базы данных Zentralblatt MATH, в которой у каждого автора есть персональный идентификатор. Для экспериментов выбирали документы, относящиеся к каждому из авторов с выбранным именем, после чего документы классифицировали во множество идентификаторов. Признаки содержали: число общих фамилий соавторов; число общих адресов электронной почты соавторов; общие коды, используемые при классификации работ по области науки (в статье использовали MSC-коды); число общих фраз в списках ключевых слов, число общих слов в списках ключевых слов; индикатор того, совпадают ли ISSN.

Если доступен полный текст документов, то можно рассмотреть данную задачу еще с одной стороны. В тексте документов явно или неявно могут присутствовать ссылки на другие работы, при этом информация об этих работах может быть неполной или вообще отсутствовать. Последнее справедливо для электронных энциклопедий, где документ является страницей энциклопедии, а в тексте документов выделяются отсылки к другим страницам. Для автоматического определения ссылок на другие страницы, которые нужно сопоставить с отрывком текста в первоначальной странице, могут использоваться схожие методы. Так, в работе [12] поставлена близкая задача: *викификация* — задача определения и связывания выражений из текста с их страницами в Википедии. Пусть дан текст с явно определенными подстроками (называемые упоминаниями). Упоминания необходимо отобразить на соответствующие им страницы Википедии. Разделяют два подхода к решению этой задачи: *локальный* подход разрешает

неоднозначность каждого упоминания отдельно; *глобальный подход* проводит одновременно операции над всеми упоминаниями, чтобы выявить близость в их значениях и, тем самым, установить их согласованность. В работе [12] решают глобальную задачу. Согласованность выбранных страниц описывается значениями функции, сравнивающей две страницы Википедии на близость. Функция оценки близости множества страниц: $\sum_i \varphi(m_i, t_i) + \sum_j \psi(t_i, t_j)$, где φ — функция, сравнивающая упоминание m_i и страницу t_i на близость, а ψ — сравнивает на близость две страницы. Функции φ и ψ обучаются алгоритмом *SVM*.

В работе [13] исследованы два подхода к решению задачи, а именно — с использованием алгоритмов *Naive Bayes* и *SVM*. В первом подходе вероятность того, что из определенного множества авторов $\{X_i\}$ представитель $X \in \{X_i\}$ является автором документа C, оценивают величиной $\max_i P(X_i|C)$. Используя правило Байеса, это выражение можно переписать в виде $\max_i P(C|X_i)P(X_i)/P(C)$.

Соавторов, названия работ и журналов используют в качестве *независимых* атрибутов. Соответственно, $\max_i P(X_i \mid C) = \max_i \prod_j P(A_j \mid X_i)$, где A_j — соответствующий атрибут. Пусть A_1 обозначает атрибут *соавторы* документа. Предложено разбить вероятность $P(A_1 \mid X_i)$ на следующие:

$P(N \mid X_i)$ — условная вероятность написания автором в одиночку при условии X_i;

$P(Co \mid X_i)$ — условная вероятность написания автором X_i с соавторами;

$P(Seen \mid Co,\ X_i)$ — вероятность написания автором работы с соавторами, которые уже встречались ранее в обучающей выборке. При этом $P(Unseen \mid Co,\ X_i) = 1 - P(Seen \mid Co, X_i)$.

$P(A_{1k} \mid Seen, Co,\ X_i)$ — вероятность того, что X_i написал статью с конкретным автором A_{1k} при условии, что X_i пишет статьи с авторами, которые уже встречались. При этом каждый автор имеет собственное распределение вероятностей на множестве соавторов. Второй подход предлагает использование алгоритма *SVM*. Каждого автора рассматривают как класс, и необходимо обучить классификатор для каждого класса авторов. Чтобы применить алгоритм *SVM* к задаче классификации с несколькими классами, используют подход "один класс против всех"; один класс — положительный, остальные — отрицательные.

Отдельно следует отметить подход к решению задачи, использующий глубинные нейронные сети. Причина в том, что общий недостаток многих работ в данной группе — использование в качестве метрик сравнения документов заранее предопределенных функций, эффективность которых может снижаться на произвольных (новых) данных. В работе [17] для устранения этого недостатка предложен метод автоматического обучения признаков по данным с использованием *глубинных нейронных сетей* (*Deep Neural Networks*). В этой работе использован многослойный перцептрон, число узлов на первом слое соответствует числу базовых используемых признаков. Последний слой состоит из двух узлов, которые изображают результат бинарной классификации. Структура сети содержит два параметра: число скрытых слоев (*hidden layers*) и число узлов на каждом слое. Эти параметры подобраны экспериментально с использованием *k*-кратной перекрестной проверки. На вход сети подают оценки сравнения документов по именам соавторов, ключевым словам и т. д.

Сложности извлечения размеченных данных для обучения в работе [20] предложено обойти с использованием обратной связи от пользователя. Предложен интерактивный подход, когда ответы пользователей используются классификатором вместе с обучающей выборкой. Сначала генерируются кластеры без обучения с использованием критерия, согласно которому два документа попадают в один кластер, если они имеют хотя бы одного совпадающего по имени соавтора. На втором шаге обучаются функции близости по сгенерированной выборке и с учетом обратной связи пользователей. Процесс обучения функций близости использует фреймворк генетического программирования (*Genetic Programming*). В этом методе представитель популяции — бинарное дерево, в узлах которого находятся функции близости двух ссылок. Сначала случайным образом генерируют начальную популяцию, полученным функциям назначают оценки их качества. В зависимости от этих оценок генерируют последующую популяцию, к которой применяют генетические операции. Этот итеративный процесс заканчивается, когда достигнут критерий остановки, после чего выбирают наилучшего представителя популяции, который определяет функции близости ссылок. По данным функциям близости обучается классификатор *Optimum-Path Forest (OPF)*. Этот классификатор не зависит от параметров, естественно поддерживает мультиклассовую классификацию и не подразумевает разделимость объектов классификации. Каждый *OPF*-класс представляет результирующий кластер документов. В некоторых случаях авторов интересует отображение множества не поставленных в соответствие авторам документов на множество известных авторов с первичным именем. Такую задачу рассматривали в работе [21]. Формальная постановка задачи использует граф попарных отношений: $G = (V, E, X, Y)$ — граф; V — множество узлов, каждый из которых соответствует паре (p_i, p_j) публикаций; E — множество неориентированных ребер; каждый узел $v_{ij} \in V$ и имеет вектор признаков $x_{ij} \in X$, который является конкатенацией вектора внутренних признаков x_{ij}^{int} и вектора относительных признаков x_{ij}^{rel}, а также неизвестную метку $y_{ij} \in Y$. Относительные признаки определяют как максимум среди значений признаков по узлам в окрестности. Между двумя узлами есть ребро, если узлы содержат общую публикацию. Задачу ставят следующим образом: по заданному графу G для неоднозначного имени для каждого узла необходимо определить значение метки $y_{ij} \in \{0, 1\}$, характеризующей, принадлежат ли

две публикации одному автору. Алгоритм состоит в следующем: на первом шаге для каждого узла, основываясь только на внутренних признаках, предсказывается его метка. Далее, для каждого узла из его окрестности выбирают те, для которых предсказана положительная метка, вычисляются относительные признаки применительно к узлам этой окрестности. Далее метка переназначается с использованием классификатора, основанного на обоих внутренних и относительных признаках. Этот процесс итеративный и заканчивается по истечении предопределенного числа итераций. В качестве классификатора брали линейный метод опорных векторов. Для оценки метода были собраны данные из системы SocialScholar. Для сравнения результатов использовали методы кластеризации (*HAC*, *k-means*) и классификации (*Pairwise Classification*). Была создана база данных SocialScholar, в которую из баз данных DBLP, IEEE, ACM, CiteSeer было извлечено свыше 8 млн статей. Для оценки были вручную проставлены метки по более чем 4000 статей для 75 имен авторов. Оценивали меры точности по формулам (1), (2). Сравнение показало превосходство данного метода, средние значения *precision* = 91 %, *recall* = 90,9 %.

Численные эксперименты в работах этой группы состояли либо из оценки точности предложенного метода в сравнении с другим методом, либо в оценке статистических показателей точности классификации. В публикации [11] оценки проводили по этим статистическим показателям. В работе [17] для экспериментальной проверки эффективности алгоритма были извлечены данные публикаций вьетнамских авторов из электронных библиотек ACM, IEEE Xplore, MAS. Данные были вручную проверены и размечены. Общий размер множества данных оказался свыше 30 000 примеров для 10 неоднозначных имен. Для определения параметров сети применяли пятикратную перекрестную проверку, в результате оптимальными параметрами оказались семь скрытых слоев сети и 50 узлов на каждом слое. Предложенный метод показал точность свыше 99 % на используемых данных. Отметим, однако, что более 70 % кластеров состояли из одного документа. В работе [13] предложенные методы применяли к двум выборкам. Первая состояла из множеств документов, написанных авторами с именами "J Anderson" и "J Smith". Разметку выборки проводили вручную. Вторая выборка состояла из данных, полученных из базы данных DBLP. Работы, написанные авторами с одинаковым именем, кластеризовали по совпадающим фамилии и первому инициалу. Из каждого полученного множества выбирались канонические авторы — полные имена, в которые необходимо классифицировать остальные имена. Множества разбивали случайным образом в отношении 1:1 на обучающую и тестовую выборки. Из отрицательных факторов — "ручная настройка" для отдельного автора. Описаны и проанализированы результаты экспериментов. При этом множества для тестов выбраны маленькие — в первом типе тестов только 2 имени, во втором — 9 имен. Очевидно, такая выборка является не репрезентативной. В работе [14] оценивали точность и полноту кластеризации в сравнении с алгоритмом из работы [22] на множестве, относящемся к 14 именам авторов. Предлагаемый алгоритм в целом показывал лучшие результаты, точность составила около 75 %. Отмечено, что корреляция тем увеличивает точность и уменьшает полноту. Также отмечено, что использование корреляции между тематиками работает хуже, поскольку у одной тематики могут быть две далекие друг от друга подтемы. Например, "*information process*" имеет две подтемы, "*medical informatics*" и "*public key cryptography*", которые не связаны между собой, однако близки в графе. В работе [20] для экспериментов использовали две коллекции данных, полученные из базы данных DBLP. Оценивали результаты по *k*-метрике (3) и попарной *F1*-мере (2). Параметры для GP (метод начальной популяции, размер начальной популяции, число поколений, операторы) были подобраны экспериментально. Во время экспериментов было смоделировано идеальное поведение пользователей, всегда отвечающих правильно. Наивысшие показатели *k*-метрики показал предложенный метод, в котором на каждой итерации пользователи определяют авторов пяти ссылок, и получено значение по точности выше 80 %.

2.4. Методы, использующие частичное обучение

Результаты исследований, опубликованных в статьях, представленных в этом разделе, объединяют подходы, основанные на автоматическом способе генерации качественной выборки для обучения. Для этого применяют методы частичного обучения, которые работают, как правило, с размеченной выборкой данных маленького размера и с большой выборкой неразмеченных данных.

В работе [23] метод решения задачи опирается на Марковские случайные поля (*Markov Random Fields*, *MRF*). *Атомными* кластерами называют те, в которых статьи очень близки, для них не требуется применения алгоритма. Публикации, написанные автором с именем *a*, и отношения между ними образуют *информативный* граф, в котором каждая вершина представляет публикацию, а ребро — отношение. Также была поставлена задача определения числа реальных авторов. Формализуется задача с помощью марковских полей следующим образом: пусть скрытые переменные Y — метки кластеров на статьях. Наблюдаемая переменная X относится к статьям, где случайная переменная x_i сгенерирована из условного распределения вероятностей $P(x_i|y_i)$. Распределение вероятностей значений y_i удовлетворяет марковскому свойству и зависит только от меток кластеров наблюдений, которые имеют отношение к x_i. Величина $P(Y)$ (определенный способ присвоения меток публикациям) определена таким образом, что зависит от признаков, определенных на ребрах графа, которые обозначают отношения между публикациями, а $P(X|Y)$ зависит от признаков, определенных на вершинах графа, которые представляют информацию об атрибутах публикации. Для оценки числа кластеров использован Байесовский информационный критерий (*Bayesian Information Criterion*).

В работе [24] предложено использовать признаки высокой точности для генерации обучающей выборки. Первый этап состоит в применении высокоточных правил для нахождения положительных примеров — пар документов, которые скорее всего написаны одним автором. К правилам относятся результаты сравнения имен соавторов и адресов электронной почты, самоцитирование, область науки. К отрицательным примерам относятся все пары документов, которые не были выбраны в качестве положительных примеров. Предложено также использовать *отрицательные правила*, которые могут сигнализировать о том, что две публикации написаны разными авторами. Признаками того, что две публикации написаны разными авторами, могут быть, например, различие языка, на котором написаны публикации, или отсутствие совпадающих по имени соавторов. На втором этапе эти примеры используют как обучающую выборку для классификатора. Результат классификации используется для кластеризации документов. Признаки классификатора содержат множество признаков, использующих информацию об авторах и публикациях, кроме того рассматривают всевозможные пары таких признаков и добавляют к признакам классификатора. Их вычисляют как произведение двух признаков. Для обучения модели использовали бинарный *L1-regularized logistic*-классификатор, обученный с помощью алгоритма *Orthant-Wise Limited-memory Quasi-Newton* (*OWL-QN*). К полученным с помощью такого классификатора данным применяли алгоритм иерархической кластеризации.

В работе [23] для оценки метода была создана коллекция данных, включающая 32 имени и более 2000 статей. Вручную были проставлены метки для каждой из статей. Для оценки использовали меры оценки статистических показателей. Алгоритм сравнивали с несколькими, используемыми ранее, основанными на методах кластеризации *k*-средних, *SOM* и *x*-средних. Предложенный метод показал точность выше данных методов со средним значением *F1*-меры 88 %. В работе [24] для экспериментов использовали данные Thomson Reuters базы данных Web of Knowledge, из которой было извлечено 14 млн статей по 253 областям науки и примерно 54 млн упоминаний авторов. Сложность с отбором данных состояла в отсутствии точно помеченных данных. Для построения выборки извлекали email-адреса авторов, которым рассылались запросы на подтверждение достоверности списка работ. Полученные ответы сопоставляли с данными Web of Knowledge для генерации выборки. Часть полученных данных использовали для обучения классификатора, другие — для тестирования. Далее проводили эксперименты с целью определения правил для использования на этапе предобучения для формирования обучающей и тестовой выборок для классификатора. Из выбранных правил выделяли комбинации, которые и оценивали на степень их эффективности. Кроме правила совпадения email-адресов, наилучший результат показало правило самоцитирования. Оказалось также, что правила, сравнивающие области науки и место издательства, менее точны. Правило соавторства также оказалось менее точным, что идет вразрез с результатами исследований, представленных в других статьях, в которых это правило использовали. В качестве комбинации положительных правил были выбраны email-адреса и самоцитирование, отрицательных — имя и язык работы. На следующем далее этапе исследовали, какие признаки и комбинации признаков наиболее эффективны для обучения классификатора. Для оценки метода решения в работе [24] предложенный алгоритм сравнивали с иерархической кластеризацией документов. В качестве функции близости документов использовали среднее значение результатов применения определенных признаков к паре документов. Иерархическая кластеризация показала худшие результаты по *F1*-мере, чем предложенный метод. Анализ предложенного метода показал, что он более эффективен для маленьких (<100 авторов) и средних (100...1000 авторов) групп неоднозначных имен. Рассматривали следующие типы ошибок.

- Неправильно кластеризованная статья в силу того, что добавленная в базу данных статья является другой версией уже добавленной ранее (ошибка высокоточных правил).
- Очень похожие статьи по присутствующим метаданным, но многие метаданные пропущены.
- Статьи неправильно кластеризуются в том случае, когда мало доступных метаданных, если при этом отличаются тематики статей и их названия непохожи.

Из результатов можно сделать вывод, что основные сложности возникают в ситуации, когда доступна малая часть атрибутов.

2.5. Методы, использующие вероятностный подход к решению задачи

Исследования в публикациях данной группы основаны на оценках распределения различных атрибутов либо на оценках условной вероятности написания одним автором двух заданных статей. Вероятностный подход к задаче выявления дублетных записей авторов, использующий оценку категориального распределения, представлен в работе [25]. У каждого документа в рамках такого подхода рассматривают следующие атрибуты: имена соавторов; название издательства; название работы. Для сравнения издательств и соавторов используют метод отношения правдоподобий (*categorical sampling likelihood ratio*). У каждого автора существуют предпочтения в выборе издательства, и эти предпочтения могут быть представлены как распределение, называемое *Preference Distribution*. Отмечено, что у разных авторов, вероятно, будут разные распределения, следовательно, можно оценить возможность того, что два разных кластера содержат работы одного и того же автора, сравнив два распределения издательств публикаций в этих кластерах. Это задача, известная под названием *two sampling problem*. При этом возникает сложность: как правило, кластеры малы по размеру, следовательно, доступна лишь часть всего распределения, и, кроме того, это лишь частичное наблюдение распределения. Для сравнения распределений проверяют две гипотезы: H_0 — то, что мульти-

множества A и B издательств из двух кластеров выбраны из различных распределений; H_1 — A и B выбраны из одного распределения. Оценив вероятности $P(H_0|B, A)$ и $P(H_1|B, A)$, близость двух кластеров по данному критерию определяют как $\Lambda = P(H_1|B, A)/P(H_0|B, A)$. Для кластеризации данных используется метод иерархической восходящей кластеризации. Изначально каждая статья образует кластер, на каждом шаге кластеризации находятся наиболее похожие кластеры и объединяются до тех пор, пока максимальное значение близости кластеров не опустится ниже определенной границы. Весь процесс кластеризации разбит на два этапа: кластеризация, основанная на сравнении общих соавторов; кластеризация, основанная на сравнении названий работ и издательств каждой пары кластеров. Предложен также статистический метод для оценки значения $k(e)$ — числа разных авторов с именем e. Предполагается, что имя состоит из различных частей и эти части выбраны независимо друг от друга. Оценивается вероятность каждого варианта для каждой части, а вероятность полного имени считается как совместная вероятность отдельных частей.

В работе [26] предложен метод, использующий сетевое представление задачи, алгоритм *Random Walk with Restart* (RWR [27]) и знания о предметной области. В модели данные используют для построения сети. Би-реляционные сети (*Bi-Relational Network*, *BRN*) — это $N = \{V, E, W\}$, $V = V_1 \cup V_2$, $E = E_1 \cap E_2 \cap E_3$, где V_1, V_2 — непересекающиеся множества вершин; E_1, E_2 — множества внутренних отношений в V_1 и V_2 соответственно; E_3 — множество смежных отношений вершин из V_1 и V_2. В данном случае вершины — это авторы и публикации.

Значение, получаемое в результате реализации *RWR*, определяется уравнением: $r = (1 - c)Wr + ce$, и интерпретируется как частица, которая случайно движется с определенной вероятностью $1 - c$ из некоторого узла в графе W в соседнее положение, а с вероятностью c она может вернуться обратно в стартовую позицию. С каждым движением энергия, передаваемая от стартовой позиции, распределена между смежными узлами с вероятностями, пропорциональными весам ребер. Цель *RWR* — вычислить оценку энергии (вектор r), распределенной между всеми узлами в спокойном состоянии частицы, с начальным распределением, представленным вектором e.

Метод решения задачи следующий: по входным данным (информация о публикациях) строят сеть. После этого используют алгоритм *RWR* для оценки близостей между конкретным упоминанием автора и всеми остальными. В результате получают матрицу близостей, которую используют для кластеризации. Поскольку заранее число кластеров неизвестно, используют *Affinity Propagation* — метод кластеризации. В работе [28] задача формализуется как задача разбиения графа. Наблюдаемой переменной является множество пар признаков документов. Пусть $y_{ij} \in Y$ — скрытая переменная (*hidden variable*), равная 1, если документы d_i и d_j написаны одним автором. Задача — максимизировать условную вероятность $P(Y/X)$, которая выражается через функции признаков пар документов. Близость пары векторов признаков определяется как среднее значение близости признаков. Как правило, вероятность того, что данный автор является автором данного документа, расписывается через правдоподобия отношения данного автора к набору атрибутов, относящихся к документу. Схожий метод можно применять и для оценки вероятности того, что два документа написаны одним автором. Для этого вычисляют оценки близости атрибутов этих документов, после чего оценивают правдоподобие того, что полученный вектор оценок представлял бы вектор сравнения документов, действительно написанных одним автором. Такой подход описан в работе [29]. Две публикации, которые предложено сравнить на то, что они написаны одним автором, сравнивают по девяти критериям. К их числу относят: сравнение инициалов; суффиксов; названий работ; журналов; наличие общих соавторов и т. д. Эти критерии образуют профиль близости работ. Далее вычисляют вероятность появления такого профиля при совпадении данных работ и не совпадении работ $r(x) = P(x|M)/P(x|N)$, где M означает совпадение авторов; N — несовпадение. Очевидно, чем выше это значение, тем достоверней можно утверждать, что две работы написаны одним автором. Вероятность совпадения авторов при заданном профиле схожести можно выразить по формуле Байеса через $r(x)$, $P(M \mid x) = 1 \Big/ \left(1 + \dfrac{1 - P(M)}{P(M)\, r(x)}\right)$, где $P(M)$ обозначает общую вероятность совпадения для конкретного имени. К дальнейшим относятся следующие действия: оценить значение $r(x)$ для всех возможных x; оценить $P(M)$ для отдельного имени автора. Далее рассматривают случай, когда есть три работы, для двух из которых вычисленная вероятность совпадения высока, а для третьей — мала. Поскольку такая ситуация невозможна, две большие вероятности уменьшаются, тогда как меньшая вероятность увеличивается. Авторы отмечают, что целью работы является не столько стремление получения высокой точности разрешения неоднозначности, сколько анализ особенностей публикации. К числу таких особенностей относятся, например, насколько сильна тенденция публиковаться в одном журнале и как часто ученые из разных научных дисциплин, так же как и из разных институтов, взаимодействуют между собой.

В работе [25] результаты оценивали по статистическим показателям и сравнивали с методами, предложенными в работе [28]. Оценивали точность на небольшом множестве имен, в среднем предложенный метод показал лучший результат, чем два других метода. Среднее значения *F1*-меры составило 0,86. В работе [28] для получения меток для пар работ использовали сервис Amazon Mechanical Turk. Результаты оценивали по *F1*-мере, максимальное значение точности в рамках данного исследования составило около 0,84.

2.6. Методы, не использующие описанные выше подходы

В работах [30—33] использовали методы, которые нельзя отнести к описанным выше группам.

В работе [31] рассматривали задачу, названную *Brazilian Ambiguity Problem*, имея в виду, что бра-

зильские имена содержат много первых имен и фамилий, которые появляются в разных комбинациях и формах. Сама статья описывает следующие действия: предложена вероятностная модель для построения неоднозначности имен в социальных графах и алгоритм для разрешения представленной неоднозначности. Задача ставится следующим образом: пусть дан граф $G = (O, E)$, O — множество объектов; E представляет попарные отношения между объектами. Пусть L_i — множество меток (имен), которые могут быть приписаны к объекту o_i. Рассматриваем процесс наблюдения графа. Пара объектов (ребро) может наблюдаться как пара соответствующих меток. Процесс наблюдения применяют к многим (может быть, ко всем) ребрам так, что получают граф $G' = (L', E')$, вершины которого — множество меток, ребра — все наблюдаемые отношения среди меток. Задача — восстановить граф G по графу G'. Предложена следующая модель введения неоднозначности в социальный граф. Пусть дан граф $G = (V, E)$, где V — авторы, а E — отношения между ними. В таком графе каждая вершина однозначно идентифицирует объект сети. Предложенная модель:

- вершина дублируется с вероятностью p;
- с вероятностью q создается ребро между соседом первоначальной вершины и ее дубликатом;
- с вероятностью r первоначальное ребро из прошлого пункта удаляется.

Далее предложен следующий простой алгоритм разрешения неоднозначности. Склеивают те вершины графа, которые лежат на расстоянии 2 друг от друга, и соседи одной вершины целиком содержатся во множестве соседей второй вершины. Предложено исключение неоднозначности с применением следующей эвристики: если вершина, находящаяся на расстоянии 2 от данной, имеет не меньше соседей и ее соседи содержат соседей данной, то она, вероятно, дублирующая, и их объединяем.

Для экспериментов были сгенерированы две социальные сети, в которые вводилась неоднозначность. К ним применяли алгоритм введения неоднозначности, после чего запускали алгоритм исключения неоднозначности, ответ сравнивали с изначальным графом по метрикам *precision* и *recall*. В качестве параметров здесь использованы значения вероятностей p, q, r. Были подобраны "оптимальные" значения этих параметров. Максимальное значение *precision* оказалось близко к 100 %, *recall* — к 60 %.

Заключение

Представленные в настоящем обзоре публикации и подходы к исследованиям отличаются по методам решения рассматриваемой целевой задачи, по полученным результатам, по глубине их анализа. Каждый из представленных подходов имеет свои достоинства и недостатки. Резюмируем некоторые недостатки.

В подходе, использующем методы обучения с учителем, реализуется попытка найти зависимость между атрибутами и близостью документов. Обученный на одной выборке (как правило, небольшого размера вследствие трудностей сбора данных) классификатор может быть гораздо менее эффективен на новых данных. Предполагается, что каждый автор имеет собственные распределения вероятности предпочтений научных тем, которым посвящены написанные им работы, вероятность публикации в определенных журналах и т. д., вероятность смены научной тематики, публикации с новыми соавторами.

Подходы, использующие вероятностные методы для определения близости между публикациями, основаны на оценке распределений по их малой части. Если в промежуточном кластере документов, относящихся к одному автору, присутствует некоторое множество издательств или научных тем, то это неполное множество оценивает все множества публикаций данного автора и распределения его предпочтений.

Подходы, основанные на заранее определенных функциях близости документов или кластеров документов, как правило, показывают худшую точность, чем те, которые предполагают обучение и вероятностное оценивание. Они не учитывают зависимости данных и зависят от вручную подобранных весов признаков, поэтому не могут достоверно отображать зависимости между признаками. Отказ от полного обучения в сторону *bootstrapping* или частичного обучения выглядит разумным, поскольку позволяет, во-первых, подстраивать обучаемые функции оценки близости под данные, изменяя зависимости между атрибутами, во-вторых, облегчить и уменьшить сбор данных для выполнения обучения. Подобные методы показывали наивысшую точность среди работ, представленных в обзоре. Однако для лучшего применения этого метода требуется либо репрезентативная выборка с правильно размеченными работами, либо набор высокоточных правил для генерации обучающей выборки. Эти данные могут отсутствовать, поскольку зачастую единственно доступной информацией о публикации является множество имен соавторов и название работы. Применять высокоточные правила к такому набору не представляется возможным.

Кроме перечисленных выше проблемных вопросов в представленных подходах следует отметить, что, как правило, тестовые испытания проводили на выборке данных малого объема. Практически все эти данные собраны вручную, а многие алгоритмы проверяются на небольшом множестве первичных имен, с заранее вручную проставленными ответами. Количественно эксперименты оценивают по статистическим показателям, которые не всегда точно соотносятся с тем обстоятельством, насколько хорош алгоритм в реальной ситуации при работе с постоянно обновляющейся базой данных публикаций.

Результаты представленного в настоящей статье анализа могут быть полезны исследователям и специалистам-практикам, занимающимся вопросами проектирования и построения информационно-аналитических систем в области библиометрии и наукометрии.

Список литературы

1. **Васенин В. А., Афонин С. А., Голомазов Д. Д., Козицын А. С.** Интеллектуальная Система Тематического Исследования НАучно-технической информации (ИСТИНА) // Информационное общество. 2013. № 1—2. С. 21—36.
2. **Афонин С. А., Гаспарянц А. Э.** Разрешение неоднозначности авторства публикаций при автоматической обработке библиографических данных // Программная инженерия. 2014. № 1. С. 25—29.
3. **Fellegi I. P., Sunter A. B.** A Theory for Record Linkage // Journal of the American Statistical Association. 1969. Vol. 64, No. 328. P. 1183—1210.

4. **Dau N., Russo M., Bouwsema E., Özyer T., Alhajj R.** Web Crawler System for Distinct Author Identification in Bibliographic Databases // ICIT 2015 The 7th International Conference on Information Technology, 2015. P. 295—302.
5. **Arif T., Ali R., Asger M.** Author name disambiguation using vector space model and hybrid similarity measures // Contemporary Computing (IC3), 2014 Seventh International Conference on, 7—9 August 2014, Noida, India. IEEE. 2014. P. 135—140.
6. **Schulz C., Mazloumian A., Petersen A. M., Penner O., Helbing D.** Exploiting citation networks for large-scale author name disambiguation // EPJ Data Science. 2014. 3 (1):1—14. doi: 10.1140/epjds/s13688-014-0011-3.
7. **Cota R. G., Ferreira A. A., Nascimento C., Gonçalves M. A., Laender A. H. F.** An Unsupervised Heuristic-based Hierarchical Method for Name Disambiguation in Bibliographic Citations // J. Am. Soc. Inf. Sci. Technol. 2010. Vol. 61, No. 9. P. 1853—1870.
8. **Arifa T., Alib R., Asgerc M.** A Multistage Hierarchical Method for Author Name Disambiguation // International Journal of Information Processing. 2015. Vol. 9. P. 92—105.
9. **Nadimi M., Mosakhani M.** A more Accurate Clustering Method by using Co-author Social Networks for Author Name Disambiguation // Journal of Computing and Security. 2014. Vol. 1, No. 4. P. 307—315.
10. **Carvalho A. P., Ferreira A. A., Laender A. H. F., Gonçalves M. A.**, Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries // Journal of Information and Data Management. 2011. Vol. 2, No. 3. P. 289—304.
11. **Dendek P. J., Bolikowski L., Lukasik M.** Evaluation of Features for Author Name Disambiguation Using Linear Support Vector Machines // Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on. 2012. P. 440—444.
12. **Ratinov L., Roth D., Downey D., Anderson M.** Local and Global Algorithms for Disambiguation to Wikipedia // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon. 2011. P. 1375—1384.
13. **Han H., Giles L., Zha H., Li C., Tsioutsiouliklis K.** Two supervised learning approaches for name disambiguation in author citations // JCDL'04: Proceedings of the 4th ACM/IEEE joint conference on Digital libraries. 2004. P. 296—305.
14. **Yang K.-H., Peng H.-T., Jiang J.-Y., Lee H.-M., Ho J.-M.** Author Name Disambiguation for Citations Using Topic and Web Correlation // Springer Berlin Heidelberg. 2008. Vol. 5173, P. 185—196.
15. **Cen L., Dragut E. C., Si L., Ouzzani M.** Author Disambiguation by Hierarchical Agglomerative Clustering with Adaptive Stopping Criterion // Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, 2013. URL: https://www.cs.purdue.edu/homes/lsi/p741-cen.pdf
16. **Kim K., Khabsa M., Giles C. L.** Random Forest DBSCAN for USPTO Inventor Name Disambiguation // IJCAI-16 Workshop on Scholarly Big Data: AI Perspectives, Challenges, and Ideas, 2016. https: // arxiv.org/abs/1602.01792
17. **Tran H. N., Huynh T., Do T.** Author Name Disambiguation by Using Deep Neural Network // ACIIDS 2014, LNCS. Vol. 8397. P. 123—132.
18. **Karypis G., Aggarwal R., Kumar V., Shekhar S.** Multilevel hypergraph partitioning: applications in VLSI domain // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 1999. Vol. 7, No. 1. P. 69—79.
19. **Ester M., Kriegel H.-P., Sander J., Xu X.** A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise // Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996. URL: http: // www.dbs.informatik.uni-muenchen.de/Publikationen/Papers/KDD-96.final.frame.pdf
20. **Godoi T. A., Torres R. d. S., Carvalho A. M.** et al. A Relevance Feedback Approach for the Author Name Disambiguation Problem // Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries. Indianapolis, Indiana, USA. 2013. P. 209—218.
21. **Chen Z., Guo J., Lan Y., Cao L., Cheng X.** Name Disambiguation by Collective Classification // Information Retrieval Technology — 10th Asia Information Retrieval. Kuching, Malaysia, 2014. P. 406—417.
22. **Han H., Zha H., Giles C. L.** Name disambiguation in author citations using a K-way spectral clustering method // Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on. IEEE. 2005. P. 334—343.
23. **Tang J., Fong A. C. M., Wang B., Zhang J.** A Unified Probabilistic Framework for Name Disambiguation in Digital Library // IEEE Transactions on Knowledge and Data Engineering. 2012. Vol. 24, No. 6. P. 975—987.
24. **Levin M., Krawczyk S., Bethard S., Jurafsky D.** Citation-based bootstrapping for large-scale author disambiguation // Journal of the Association for Information Science and Technology. 2012. Vol. 63, No. 5. P. 1030—1047.
25. **Li S., Cong G., Miao C.** Author Name Disambiguation Using a New Categorical Distribution Similarity // Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012. Bristol, UK, September 24—28. 2012. Proceedings, Berlin, Heidelberg. 2012. P. 569—584.
26. **Yuechang Liu, Yong Tang.** Network based Framework for Author Name Disambiguation Applications // International Journal of u- and e- Service, Science and Technology. 2015. Vol. 8, No. 9. P. 75—82.
27. **Tong H., Faloutsos C., Pan J.-Yu.** Fast Random Walk with Restart and Its Applications // Proceedings of the Sixth International Conference on Data Mining. Washington, DC, USA. 2006. P. 613—622.
28. **Cheng Yu, Chen Z., Wang J., Agrawal A., Choudhary A.** Bootstrapping Active Name Disambiguation with Crowdsourcing // Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management. San Francisco, California, USA. 2013. P. 1213—1216.
29. **Torvik V. I., Weeber M., Swanson D. R., Smalheiser N. R.** A probabilistic similarity metric for medline records: A model for author name disambiguation // Journal of the American Society for Information Science and Technology. 2005. Vol. 56. P. 40—158.
30. **Shen W., Han J., Wang J.** A Probabilistic Model for Linking Named Entities in Web Text with Heterogeneous Information Networks // in Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, Snowbird, Utah, USA. 2014. P. 1199—1210.
31. **Gomide J., Kling H., Figueiredo D.** A Model for Ambiguation and an Algorithm for Disambiguation in Social Networks // Complex Networks VI: Proceedings of the 6th Workshop on Complex Networks CompleNet 2015. 2015. P. 37—44.
32. **Shoaib M., Daud A., Khiyal M.** Role of references in similarity estimation of publications // Int. Arab J. Inf. Technol. 2016. Vol. 13. P. 256—263.
33. **Arif T.** Exploring The Use Of Hybrid Similarity Measure For Author Name Disambiguation // International Journal of Scientific & Technology Research. 2015. Vol. 4, Issue 12. P. 171—175.

Author Name Disambiguation: Analysis of Publications

V. A. Vasenin, vasenin@msu.ru, **A. E. Gaspariants,** artem.gaspariants@gmail.com,
Mechanics and Mathematics Department, Lomonosov Moscow State University, Moscow, 119192, Russian Federation

Corresponding author:

Gaspariants Artem E., Postgraduate Student, Mechanics and Mathematics Department, Lomonosov Moscow State University, Moscow, 119192, Russian Federation,
E-mail: artem.gaspariants@gmail.com

Received on March 08, 2017
Accepted on April 03, 2017

This article presents an overview of the methods and models of solving the authors' names disambiguation problem. This problem occurs in bibliographic databases and in digital libraries when several different authors share a common name. Inability

to distinguish between publications written by authors with the same name may affect determination of their true citation index, index of their scientific group or organization and eventually lead to misunderstanding and false judgment of current trends and potentialities in scientific world. Due to the scale of the problem, only automatic methods of solving are considered. The main contributions of this article are classification of the methods and models for solving this problem for the last several years, analysis of these methods and specification of their possible advantages and disadvantages, presenting the open and unresolved challenges in this field.

The results of the analysis presented in this article can be useful for researchers and practitioners involved in the design and construction of information-analytical systems in the field of bibliometry and scientometrics.

Keywords: *citation metrics, bibliographic record, machine learning, record linkage, named entities linkage*

For citation:

Vasenin V. A., Gaspariants A. E. Author Name Disambiguation: Analysis of Publications, *Programmnaya Ingeneria,* 2017, vol. 8, no. 6, pp. 264—275.

DOI: 10.17587/prin.8.264-275

References

1. **Vasenin V. A., Afonin S. A., Golomazov D. D., Kozitsyn A. S.** Intellektual'naja Sistema Tematicheskogo Issledovanija NAuchno-tehnicheskoj informacii (ISTINA) (The Intellectual System of Thematic Analysis of Scientific Information (ISTINA)), *Informacionnoe obshhestvo,* 2013, no. 1—2, pp. 21—36 (in Russian).

2. **Afonin S. A., Gaspariants A. E.** Razreshenie neodnoznachnosti avtorstva publikacij pri avtomaticheskoj obrabotke bibliograficheskih dannyh (Scientific Article Authorship Disambiguation for Automated Bibliographic Records Processing), *Programmnaya Ingeneria,* 2014, no. 1, pp. 25—29 (in Russian).

3. **Fellegi I. P., Sunter A. B.** A Theory for Record Linkage, *Journal of the American Statistical Association*, 1969, vol. 64, no. 328, pp. 1183—1210.

4. **Dau N., Russo M., Bouwsema E., Ozyer T., Alhajj R.** Web Crawler System for Distinct Author Identification in Bibliographic Databases, *ICIT 2015 The 7th International Conference on Information Technology*, 2015, pp. 295—302.

5. **Arif T., Ali R., Asger M.** Author name disambiguation using vector space model and hybrid similarity measures, *Contemporary Computing (IC3), 2014 Seventh International Conference on*, 7—9 August 2014, Noida, India, IEEE, 2014, pp. 135—140.

6. **Schulz C., Mazloumian A., Petersen A. M., Penner O., Helbing D.** Exploiting citation networks for large-scale author name disambiguation, *EPJ Data Science*. 2014. 3(1):1—14. doi: 10.1140/epjds/s13688-014-0011-3.

7. **Cota R. G., Ferreira A. A., Nascimento C., Gonçalves M. A., Laender A. H. F.** An Unsupervised Heuristic-based Hierarchical Method for Name Disambiguation in Bibliographic Citations, *J. Am. Soc. Inf. Sci. Technol*. 2010, vol. 61, no. 9, pp. 1853—1870.

8. **Arifa T., Alib R., Asgerc M.** A Multistage Hierarchical Method for Author Name Disambiguation, *International Journal of Information Processing,* 2015, vol. 9, pp. 92—105.

9. **Nadimi M., Mosakhani M.** A more Accurate Clustering Method by using Co-author Social Networks for Author Name Disambiguation, *Journal of Computing and Security*, 2014, vol. 1, no. 4, pp. 307—315.

10. **Carvalho A. P., Ferreira A. A., Laender A. H. F., Gonçalves M. A**. Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries, *Journal of Information and Data Management,* 2011, vol. 2, no. 3, pp. 289—304.

11. **Dendek P. J., Bolikowski L., Lukasik M.** Evaluation of Features for Author Name Disambiguation Using Linear Support Vector Machines, *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, 2012, pp. 440—444.

12. **Ratinov L., Roth D., Downey D., Anderson M.** Local and Global Algorithms for Disambiguation to Wikipedia, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* Portland, Oregon, 2011, pp. 1375—1384.

13. **Han H., Giles L., Zha H., Li C., Tsioutsiouliklis K.** Two supervised learning approaches for name disambiguation in author citations, *JCDL'04: Proceedings of the 4th ACM/IEEE joint conference on Digital libraries,* 2004, pp. 296—305.

14. **Yang K.-H., Peng H.-T., Jiang J.-Y., Lee H.-M., Ho J.-M.** Author Name Disambiguation for Citations Using Topic and Web Correlation, *Springer Berlin Heidelberg,* 2008, vol. 5173, pp. 185—196.

15. **Cen L., Dragut E. C., Si L., Ouzzani M.** Author Disambiguation by Hierarchical Agglomerative Clustering with Adaptive Stopping Criterion, *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland, 2013, available at: https://www.cs.purdue.edu/homes/lsi/p741-cen.pdf

16. **Kim K., Khabsa M., Giles C. L.** Random Forest DBSCAN for USPTO Inventor Name Disambiguation, *IJCAI-16 Workshop on Scholarly Big Data: AI Perspectives, Challenges, and Ideas*, 2016, available at: https://arxiv.org/abs/1602.01792

17. **Tran H. N., Huynh T., Do T.** Author Name Disambiguation by Using Deep Neural Network, *ACIIDS*, 2014, LNCS, vol. 8397, pp. 123—132.

18. **Karypis G., Aggarwal R., Kumar V., Shekhar S.** Multilevel hypergraph partitioning: applications in VLSI domain, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* 1999, vol. 7, no. 1, pp. 69—79.

19. **Ester M., Kriegel H.-P., Sander J., Xu X.** A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, 1996, available at: http: // www.dbs.informatik.uni-muenchen.de/Publikationen/Papers/KDD-96.final.frame.pdf

20. **Godoi T. A., Torres R. d. S., Carvalho A. M., Gonsalves A. A. M., Ferreura A., Fan W., Fox E. A.** A Relevance Feedback Approach for the Author Name Disambiguation Problem, *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries,* Indianapolis, Indiana, USA, 2013, pp. 209—218.

21. **Chen Z., Guo J., Lan Y., Cao L., Cheng X.** Name Disambiguation by Collective Classification, *Information Retrieval Technology — 10th Asia Information Retrieval,* Kuching, Malaysia, 2014, pp. 406—417.

22. **Han H., Zha H., Giles C. L.** Name disambiguation in author citations using a K-way spectral clustering method, *Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on,* IEEE, 2005, pp. 334—343.

23. **Tang J., Fong A. C. M., Wang B., Zhang J.** A Unified Probabilistic Framework for Name Disambiguation in Digital Library, *IEEE Transactions on Knowledge and Data Engineering,* 2012, vol. 24, no. 6, pp. 975—987.

24. **Levin M., Krawczyk S., Bethard S., Jurafsky D.** Citation-based bootstrapping for large-scale author disambiguation, *Journal of the Association for Information Science and Technology*, 2012, vol. 63, no. 5, pp. 1030—1047.

25. **Li S., Cong G., Miao C.** Author Name Disambiguation Using a New Categorical Distribution Similarity, *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012,* Bristol, UK, September 24—28, 2012, Proceedings, Berlin, Heidelberg, 2012, pp. 569—584.

26. **Yuechang Liu, Yong Tang** Network based Framework for Author Name Disambiguation Applications, *International Journal of u-and e- Service, Science and Technology,* 2015, vol. 8, no. 9, pp. 75—82.

27. **Tong H., Faloutsos C., Pan, J.-Yu.** Fast Random Walk with Restart and Its Applications, *Proceedings of the Sixth International Conference on Data Mining,* Washington, DC, USA, 2006, pp. 613—622.

28. **Cheng Yu, Chen Z., Wang J., Agrawal A., Choudhary A.** Bootstrapping Active Name Disambiguation with Crowdsourcing, *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management,* San Francisco, California, USA, 2013, pp. 1213—1216.

29. **Torvik V. I., Weeber M., Swanson D. R., Smalheiser N. R.** A probabilistic similarity metric for medline records: A model for author name disambiguation, *Journal of the American Society for Information Science and Technology,* 2005, vol. 56, pp. 40—158.

30. **Shen W., Han J., Wang J.** A Probabilistic Model for Linking Named Entities in Web Text with Heterogeneous Information Networks, *in Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Snowbird, Utah, USA, 2014, pp. 1199—1210.

31. **Gomide J., Kling H., Figueiredo D.** A Model for Ambiguation and an Algorithm for Disambiguation in Social Networks, *Complex Networks VI: Proceedings of the 6th Workshop on Complex Networks CompleNet 2015,* 2015, pp. 37—44.

32. **Shoaib M., Daud A., Khiyal M.** Role of references in similarity estimation of publications, *Int. Arab J. Inf. Technol,* 2016, vol. 13, pp. 256—263.

33. **Arif T.** Exploring The Use Of Hybrid Similarity Measure For Author Name Disambiguation, *International journal of scientific & technology research,* 2015, vol. 4, issue 12, pp. 171—175.

УДК 519.612 DOI: 10.17587/prin.8.276-281

К. Ф. Иванова, канд. техн. наук, докторант, e-mail: klara.i2010@yandex.ru,
Санкт-Петербургский государственный университет

Прогноз валового объема продукции и трудовых ресурсов межотраслевой балансовой модели

Предложен унифицированный способ для оценки погрешности решения выходных характеристик классической межотраслевой балансовой модели В. Леонтьева, отличный от известных интервальных подходов. Показана тождественность граничных значений исходной интервальной технологической матрицы и формализованных оптимальных угловых матриц, на базе которых строится оценка решения интервальной системы. Алгоритм расчета направлен на поиски допустимой области максимальных отклонений компонент вектора решения в положительном и отрицательном направлениях, возникающих под воздействием неопределенности входных параметров. Полученные результаты являются основанием для принятия оптимальных решений и прогноза в экономическом, производственном или социальном планировании при решении статических задач межотраслевой балансовой модели.

Ключевые слова: *балансовая модель, неопределенность параметров, унифицированная методика, интервальный подход, алгоритм расчета, оптимальные угловые матрицы, покомпонентная оценка, принятие решений*

Введение

Линейная балансовая статическая модель В. Леонтьева устанавливает взаимосвязь между процессами, выраженными математическими приближениями межотраслевых потоков, участвующих в производственной или иных материальных сферах [1]. Постановка задачи балансовой модели относится к задачам функциональной оптимизации конечного распределения продукции, ориентированной на прогнозирование планового хозяйства. Расчет модели основан на численном решении систем матричных уравнений с неопределенными параметрами. Неопределенность входных параметров возникает в связи с неточностью измерений и их флуктуаций в пределах некоторых границ в статической модели, что приводит к необходимости двухсторонней оценки погрешности решения системы. Разработанный алгоритм двухсторонней оценки векторов решения матричных уравнений базируется на использовании известных алгебраических методов линейной алгебры [2] и синтезе двух подходов к решению поставленной задачи: оценке чувствительности решения систем линейных алгебраических уравнений (СЛАУ) [3] и внешнем оценивании множества решений интервальных систем линейных алгебраических уравнений (ИСЛАУ), представленных в работе [4].

В тексте настоящей статьи приняты обозначения множества интервалов $\boldsymbol{a} \in \mathbb{IR}$, векторов $\mathbf{Y} \in \mathbb{IR}^n$ и матриц $\mathbf{A} \in \mathbb{IR}^{n\times n}$, их вещественные аналоги (множества чисел, векторов и матриц) обозначаются $a \in R$, $\mathrm{Y} \in R^n$ и $\mathrm{A} \in R^{n\times n}$ соответственно. Таким образом, интервальные числа обозначаются полужирным шрифтом в отличие от вещественных. Векторы и матрицы обозначаются прямым шрифтом, элементы матрицы и компоненты векторов — курсивом.

Принципиальная разница между ИСЛАУ и СЛАУ состоит как в задании интервальной неопределенности входных данных, так и в способах оценки реакции системы на эту неопределенность. С позиции классической алгебры приемы оценки решений этих систем аналитически различаются. Однако в соответствии с принципами системного анализа оказывается возможным единообразно организовать постановку задач СЛАУ и ИСЛАУ. Исходя из их концептуальной общности, к ним могут быть применены идентичные подходы к поиску решений и алгоритмов расчета, подчиненные общей методологии и аппаратной реализации.

В работах [5, 6] предложены аспекты унифицированной методики, доказывающей, что интервальный определитель может служить критерием выбора оптимальных угловых точечных матриц, ответственных за двухстороннюю оценку решения. Под угловыми понимаются матрицы с относительным приращением элементов выбранного направления. Показано, что вектор решения СЛАУ с максимальными отклонениями, полученными под влиянием внешних возмущений в положительном и отрицательном направлениях, оказывается идентичным известному в интервальном анализе минимальному интервальному вектору внешнего оценивания, объемлющему множество решений интервальной системы. Такой подход позволяет объединить обе эти оценки и отождествить задачу выявления максимальных отклонений возмущенной СЛАУ с интервальным вектором внешней оценки.

В унифицированной методике алгебраических оценок рассматривается процедура формализации угловых оптимальных матриц и доказывается обоснованность общей методологии поиска оптимальных угловых матриц не только как образующих узкоинтервальные системы, но и как основы нахождения решения интервальных систем с неточными входными данными. В работе [5] показано, что неособенная квадратная интервальная матрица размера $n\times n\ \mathbf{A} \in \mathbb{IR}^{n\times n}$ с узкими интервалами для элементов, не превышающими долей процентов, может быть представлена двумя точечными матрицами $\mathrm{A}^-, \mathrm{A}^+ \in \mathrm{R}^{n\times n}$, детерминант каждой из которых определяет левую и правую границы интервального детерминанта $|\mathbf{A}| = \left[|\mathrm{A}^-|, |\mathrm{A}^+|\right]$, так, что

$$|\mathbf{A}| = \left[\underline{A}^{u},\ \overline{A}^{u}\right] = \left[\left|A^{-}\right|, \left|A^{+}\right|\right], \qquad (1)$$

где индекс "*u*" означает "интервальный". В данном контексте — величина, полученная методами интервальной алгебры.

Подробное вычисление интервального определителя $|\mathbf{A}|$ показано в работе [7] через произведение элементов матриц на их интервальные алгебраические дополнения.

Необходимыми предпосылками для этого, кроме соблюдения неособенности квадратной интервальной матрицы, являются одинаковые знаки алгебраических дополнений к обеим границам каждого элемента. Так как этим условиям отвечают М-матрицы и их модификации с широкими интервалами, то и выражения для интервального определителя для них также отвечают формуле (1).

В работе [6] получены дополнительные алгебраические системы, участвующие в выборе оптимальных решений, необходимых для нахождения двухсторонних отклонений компонент вектора x.

Межотраслевая балансовая модель "затраты—выпуск"

Межотраслевой баланс представляет собой экономико-математическую модель народного хозяйства, позволяющую провести расчет звеньев общественного производства по заданному объему и структуре конечного продукта. Расчеты различных вариантов структуры экономических связей являются необходимой предварительной стадией планирования сбалансированности отраслей и предпосылок развития методологии оптимального планирования. Результаты расчетов, полученные по модели межотраслевого баланса, дают представление о тенденциях развития технического прогресса, о насыщении экономики производственными фондами, капитальными вложениями, трудовыми ресурсами и т. д. Межотраслевой баланс, представленный в трудовых единицах, дает информацию, необходимую для построения рациональной системы цен. Условием реализации балансовой модели является вычисление векторов, характеризующих межотраслевые потоки и конечный объемный продуктов в различных экономических сферах. Примеров решения статической модели Леонтьева достаточно много как при вычислении распределения валового объема производственной продукции по отраслям [1, 2, 8, 9], так и в социально-культурной сфере [10]. Подход к решению балансовых уравнений известными приемами интервальной алгебры описан в работах [4, 11]. Решение интервальной балансовой задачи с использованием оптимальных точечных уравнений дано в работе [12]. Отличие результатов данной работы от результатов, представленных в работе [12], заключается в постановке и решении не точечной, а интервальной задачи межотраслевого баланса, что стало возможным благодаря обоснованию тождественности граничных матриц интервальной М-матрицы и формализованных оптимальных угловых матриц.

Основной принцип, принятый при решении обычной неинтервальной балансовой модели, характеризуется следующими этапами вычислений: 1) формирование матрицы прямых затрат; 2) проверка ее продуктивности; 3) определение методом Гаусса матрицы полных затрат; 4) запрос точности нахождения запаса продуктивности; 5) введение вектора конечного продукта и нахождение вектора валового выпуска. Однако неопределенность входных параметров балансовой модели, основанной на ограниченных измерениях, порождает неточное решение, которое необходимо оценивать. В соответствии с этим интервальную постановку задачи и получаемое решение можно считать наиболее обоснованным. Слово "интервальный" употребляется согласно терминологии, применяющейся в новом направлении вычислительной математики — интервальном анализе, где под словом "интервал", в частности, правильный интервал величины *a*, понимается замкнутый промежуток $[\underline{a}, \overline{a}] \in \mathrm{IR}$, где $\underline{a} \leqslant \overline{a}$; $\underline{a}, \overline{a} \in R$.

Поэтапное создание линейной балансовой модели в соответствии с концепцией В. Леонтьева состоит в следующем. Процесс производства обычно рассматривают за некоторый период времени. Вводятся обозначения: X_i — общий объем продукции *i*-й отрасли (ее валовой выпуск); X_{ij} — объем продукции *i*-й отрасли, потребляемый *j*-й отраслью при производстве объема продукции X_j; Y_i — объем продукции *i*-й отрасли, предназначенный для реализации в непроизводственной сфере, — объем потребления (этот объем обычно составляет более 75 % всей производственной продукции).

Установлено, что в результате длительного применения одних и тех же технологий значения $a_{ij} = X_{ij} / X_j$, $i, j \in \{1, 2, \ldots, n\}$, называемые коэффициентами прямых затрат, меняются очень слабо и могут быть приняты константами. Однако значения коэффициентов a_{ij}, рассматриваемые за более длительный интервал времени (например, год), могут изменяться внутри замкнутого интервала. Эти коэффициенты приобретают любое значение из заданного промежутка $\boldsymbol{a}_{ij} = \left[\underline{a}_{ij}, \overline{a}_{ij}\right]$, и коэффициент a_{ij} становится интервальным числом. В интервальном варианте материальные издержки по-прежнему пропорциональны объему производимой продукции. Принцип интервальной линейности распространяется и на векторы конечного продукта, добавленной стоимости, трудовых затрат и другие компоненты балансовой статической модели, искомые векторы которых, в силу вариабельности исходных данных, неминуемо оказываются принадлежащими некоторому диапазону изменения.

Прямая и обратная задачи балансовой модели

На примере балансовой модели рассмотрим интервальное решение как прямой, так и обратной задач. Моделируемый экономический объект состоит из *n* взаимосвязанных процессов (отраслей) и может быть описан в соответствии с объектами и взаимосвязями классической модели, где отраслями экономического процесса могут выступать объекты индустрии кино и видео. В работе [10] рассматривается модель балансового распределения многопрофильного предоставления услуг по рекламе новых фильмов рекламной отраслью киновидео индустрии, и наоборот, выполнения кино-видео индустрией заказов рекламных агентств. При этом значительные сторонние заказы составляют объемы конечного потребления. Балансовый принцип связи различных отраслей промышленности состоит в том, что валовой выпуск *i*-й отрасли должен быть равным сумме объемов потребления в производственной и непроизводственной сферах. Вместе с конечным потреблением это дает полный объем X_j:

$$\boldsymbol{Y}_j + \sum_{i=1}^{n} \boldsymbol{a}_{ji} X_i = X_j, \quad j = \overline{1, n},$$

что представляет собой первую группу балансовых уравнений.

В матричном виде эту систему можно записать следующим образом:

$$\mathrm{X} = \mathbf{A}\mathrm{X} + \mathbf{Y}, \quad \mathbf{Y} \in \mathrm{IR}^n, \quad \mathbf{A} \in \mathrm{IR}^{n \times n}, \tag{2}$$

где **A**, **Y** — технологическая матрица и вектор конечного продукта соответственно, а X — валовой выпуск продукта. Как правило, решение интервальной системы линейных алгебраических уравнений такого вида проводится методами интервальной алгебры. В работе [11] анализ интервального решения проведен методом простых итераций на примере тестовой задачи региональной экономики и проверен наблюдаемыми на практике значениями валового выпуска продукции. Эта же модель задачи рассмотрена в работе [12], где впервые по унифицированной методике выявлена интервально-алгебраическая аналогия.

Вторая группа уравнений, определяющая полный объем трудозатрат суммированием полных объемов трудозатрат $Z \geqslant 0$ на каждый вид продукции (произведений $\boldsymbol{K}_i X_i$) по всем видам продукции, выражается суммой $Z = \sum_{i=1}^{n} \boldsymbol{K}_i X_i$.

Третьей группой из n балансовых уравнений являются уравнения для цен на единицу продукта, которые получаются на базе трудовой теории стоимости. Цена единицы j-й продукции P_i складывается из издержек производства (цены ресурсов i-х продуктов, израсходованных на производство $\boldsymbol{a}_{ij} P_i$) и прибавленной стоимости $\omega \boldsymbol{K}_j$: $P_j + \sum_{i=1}^{n} \boldsymbol{a}_{ij} P_i + \omega \boldsymbol{K}_j$, где ω — стоимость человеко-часа, одинаковая для всех отраслей. Получается полная система из $2n + 1$ уравнений баланса в натуральном выражении:

$$\begin{cases} X_j + \sum_{i=1}^{n} \boldsymbol{a}_{ji} X_i = \boldsymbol{Y}_j, \quad j = 1, n \\ Z = \sum_{i=1}^{n} \boldsymbol{K}_i X_i \\ P_j + \sum_{i=1}^{n} \boldsymbol{a}_{ij} P_i + \varpi \cdot \boldsymbol{K}_j, \quad j = 1, n \end{cases}. \tag{3}$$

В векторно-матричной форме эту систему можно записать как ИСЛАУ:

$$\begin{cases} \mathrm{X} - \mathbf{A}\mathrm{X} = \mathbf{Y}, \\ \mathrm{Z} = \mathbf{K}\mathrm{X}, \\ \mathrm{P} - \mathbf{A}^{\mathrm{T}}\mathrm{P} = \omega\mathbf{K}, \end{cases} \tag{4}$$

где $\mathbf{A} \in \mathrm{IR}^{n \times n}$; $\mathbf{Y}, \mathbf{K} \in \mathrm{IR}^n$; $\mathrm{X}, \mathrm{Z}, \mathrm{P} \in \mathrm{R}^{2n}$; $\mathbf{A}^{\mathrm{T}} = \left(\boldsymbol{a}_{ji}\right)^{\mathrm{T}} = \left(\boldsymbol{a}_{ij}\right)$ — операция транспонирования.

Уравнения баланса в форме (3), (4) позволяют по известным полным объемам продукции и ценам на нее определить объемы конечного потребления, значения трудозатрат на единицу продукции и полный объем трудозатрат, связанный с количеством занятых в производстве. Начальными векторами при анализе экономической системы являются два вектора — **Y** и **K**. Задачей, обратной к решенной, является задача определения полных объемов продукции X, цен на нее P и числа занятых Z в производстве по известным технологическим характеристикам системы. Эти характеристики прямо связаны с объемами конечного потребления **Y** и значениями **K** трудозатрат на единицу продукции, т. е. X, P и Z определяются через **Y** и **K**. Их значения формально можно найти из системы интервальных матричных уравнений с единичной матрицей E:

$$\begin{cases} \mathrm{X} = (\mathrm{E} - \mathbf{A})^{-1} \mathbf{Y}, \\ \mathrm{Z} = \mathbf{K}^{\mathrm{T}}\mathrm{X}, \\ \mathrm{P} = \left\{(\mathrm{E} - \mathbf{A})^{-1}\right\}^{\mathrm{T}} \mathbf{K}. \end{cases} \tag{5}$$

Структура производства и цен, как следует из уравнения (4), может измениться только в случае изменения технологии производства, принятой в обществе, т. е. изменения **A**, **Y** и **K**. Изменение же величины почасовой оплаты ω, принятой во всем экономическом объекте, приведет лишь к пропорциональному изменению всех цен без изменения соотношения между ними.

Система (5) записана в общем виде для интервальных величин, и все обратные технологические характеристики вычисляются через обратные матрицы $(\mathrm{E} - \mathbf{A})^{-1}$ по законом интервальной алгебры. В случае формализованных оптимальных угловых векторов и матриц вычисления проводятся для их точечных значений.

Унифицированная методика оценки решения интервальной модели межотраслевого баланса

Разработанный алгебраический принцип решения интервальных матричных уравнений (3), (4) основан на формализации точечных матриц и векторов, которые принадлежат их интервальным аналогам $\mathbf{A} \in \mathrm{IR}^{n \times n}$, $\mathbf{Y}, \mathbf{K} \in \mathrm{IR}^n$:

$$\left[\mathrm{A}^-, \mathrm{A}^+\right] = \left[\underline{\mathrm{A}}, \overline{\mathrm{A}}\right], \ \mathrm{A}^-, \ \mathrm{A}^+ \in \mathbf{A}, \ \mathrm{A}^-, \ \mathrm{A}^+ \in \mathrm{R}^{n \times n},$$

$$\left[\mathrm{Y}^-, \mathrm{Y}^+\right] = \left[\underline{\mathrm{Y}}, \overline{\mathrm{Y}}\right], \ \mathrm{Y}^-, \ \mathrm{Y}^+ \in \mathbf{Y}, \ \mathrm{Y}^-, \ \mathrm{Y}^+ \in \mathrm{R}^n,$$

$$[\mathrm{K}^-, \mathrm{K}^+] = [\underline{\mathrm{K}}, \overline{\mathrm{K}}], \ \mathrm{K}^-, \ \mathrm{K}^+ \in \mathbf{K}, \ \mathrm{K}^-, \mathrm{K}^+ \in \mathrm{R}^n.$$

Искомые векторы X, P, Z являются вещественными векторами двойной размерности:

$$\mathrm{X} = (\mathrm{X}^-, \mathrm{X}^+), \ \mathrm{P} = (\mathrm{P}^-, \mathrm{P}^+), \ \mathrm{Z} = (\mathrm{Z}^-, \mathrm{Z}^+), \ \mathrm{X}, \mathrm{P}, \mathrm{Z} \in \mathrm{R}^{2n}.$$

Считается, что неотрицательная матрица **A** продуктивная, если существует положительный вектор $\mathrm{v} > 0$, такой, что $(\mathrm{E} - \mathbf{A})\mathrm{v} > 0$, что означает: матрица прямых производственных затрат продуктивна тогда и только тогда, когда существует такой план X, что каждая отрасль может произвести некоторое количество продукции для конечного потребления **Y**.

Если технологическая матрица **A** — произвольная неотрицательная квадратная матрица, существуют перечисленные далее эквивалентные условия.

1. Если матрица $(\mathrm{E} - \mathbf{A})^{-1}$ существует, то **A** продуктивна.

2. Если матрица $(\mathrm{E} - \mathbf{A})^{-1}$ существует и неотрицательна, то матрицу $(\mathrm{E} - \mathbf{A})^{-1}$ называют матрицей полных затрат.

3. Ряд $\mathrm{E} + \mathbf{A} + \mathbf{A}^2 + \ldots \mathbf{A}^k + \ldots$ сходится, матрицы $\mathbf{A}^2$, ..., $\mathbf{A}^k$ называются матрицами косвенных затрат 2-го, ..., k-го порядков соответственно (каждая из матриц 2-го и большего порядков характеризует вторичные распределения продукции).

4) Спектральный радиус λ_A матрицы **A** удовлетворяет неравенству $\lambda_A < 1$. Спектральный радиус матрицы **A** определяется соотношением $\lambda_A = \max_j |\lambda_j|$, где $1 \leqslant j \leqslant n$, λ_1, ..., λ_n — собственные значения матрицы **A**.

Анализируя неравенство $\lambda_A < 1$, можно получить достаточное условие продуктивности матрицы **A**, интерпретируемое в экономических терминах. Им служит неравенство $\sum_{i-1}^{n} a_{ij} < 1$, означающее, что при любом $j = \overline{n, 1}$ суммарный вклад всех отраслей в выпуск 1 руб. продукции *j*-й отрасли меньше 1. Выполнение условия $\sum_{i-1}^{n} a_{ij} < 1$ означает рентабельность *j*-й отрасли. Как следствие, если все отрасли являются рентабельными, то задача планирования разрешима, причем единственным образом.

Если $\mathbf{A} \geqslant 0$ — продуктивная матрица, то запасом ее продуктивности называют положительное число α, обладающее следующим свойством: все матрицы $k\mathbf{A}$ при $1 < k < 1 + \alpha$ являются продуктивными, а матрица $(1 + \alpha)\mathbf{A}$ — нет. Технологическая матрица **A** представляет собой компактное количественное описание структурных свойств некоторой экономической системы, характеризуя результативность ее производства [11]. Интервальная матрица **A** является продуктивной, если все ее точечные матрицы продуктивны. Тогда предложенная методика позволяет легко адаптировать имеющиеся алгоритмы для точечных систем на интервальные объекты. Матрица $(\mathrm{E} - \mathbf{A})$, входящая в матричное уравнение (2), является М-матрицей, и ее границами являются $(\underline{\mathrm{E} - \mathrm{A}})$ и $(\overline{E - \mathrm{A}})$. На основании доказательств, представленных в работах [5, 6], границы матрицы $(\mathrm{E} - \mathbf{A})$ совпадают с оптимальными угловыми матрицами: $(\mathrm{E} - \mathrm{A})^- = (\underline{\mathrm{E} - \mathrm{A}})$ и $(\mathrm{E} - \mathrm{A})^+ = (\overline{E - \mathrm{A}})$. Обратная матрица $(\mathrm{E} - \mathrm{A})^{-1}$ имеет две границы $(\underline{\mathrm{E} - \mathrm{A}})^{-1}$ и $(\overline{\mathrm{E} - \mathrm{A}})^{-1}$, которые соответствуют обратным угловым матрицам $\left((\mathrm{E} - \mathrm{A})^-\right)^{-1}$ и $\left((\mathrm{E} - \mathrm{A})^+\right)^{-1}$. Неособенность М-матрицы означает неособенность всех ее точечных матриц, а также ее положительную обратимость. Таким же образом обратная матрица представляется через угловые матрицы $(\mathrm{E} - \mathbf{A})^{-1} = \left[\left((\mathrm{E} - \mathrm{A})^-\right)^{-1}, \left((\mathrm{E} - \mathrm{A})^+\right)^{-1}\right]$.

Окончательный результат представляется двумя точечными векторами с компонентами, соответствующими левой и правой границам компонент $2n$-мерных векторов $\mathrm{X} = \left[\mathrm{X}^-, \mathrm{X}^+\right]$, $\mathrm{P} = \left[\mathrm{P}^-, \mathrm{P}^+\right]$ и $\mathrm{Z} = \left[Z^-, Z^+\right]$.

На основании уравнения (2) можно сказать, что внешнее оценивание ИСЛАУ методами интервальной алгебры состоит из покоординатных оценок множества решений интервальной системы линейных алгебраических уравнений (2)

$$\Xi_{uni} = \left\{\mathrm{x} \in \mathrm{R}^n \,\middle|\, \left(\exists(\mathrm{E} - \mathrm{A}) \in (\mathrm{E} - \mathbf{A}),\ \exists \mathrm{b} \in \mathbf{b}\right)(\mathrm{E} - \mathrm{A})\mathrm{x} = \mathrm{b}\right\},$$

образованного всеми решениями точечных систем $(\mathrm{E} - \mathrm{Ax}) = \mathrm{b}$, с $(\mathrm{E} - \mathrm{A}) \in (\mathrm{E} - \mathbf{A})$, $\mathrm{b} \in \mathbf{b}$, $(\mathrm{E} - \mathrm{A}) \in \mathrm{R}^{n \times n}$, $\mathrm{b} \in \mathrm{R}^n$. Искомый вектор внешней оценки соответствует значениям объединенного множества решений интервальной системы $\min\left\{\mathrm{x}_\upsilon \in \Xi_{uni}\left((\mathrm{E} - \mathbf{A}), \mathbf{b}\right)\right\}$ и $\max\left\{\mathrm{x}_\upsilon \in \Xi_{uni}\left((\mathrm{E} - \mathbf{A}), \mathbf{b}\right)\right\}$, $\upsilon = \overline{1, n}$. Найденный интервальный вектор x_υ или гипербрус является прямым декартовым произведением *n*-мерных векторов, гарантированно содержащих множество решений рассматриваемой ИСЛАУ, и является наименьшим интервальным вектором, соответствующим проекциям внешней оболочки множества решений системы на координатные оси — $E0 \supseteq \sqcup \Xi_{uni}$, где $E0$ — прямые декартовы произведения вещественных интервалов. Если интервальная матрица $(\mathrm{E} - \mathbf{A})$ неособенная, то объединенное множество решений $\Xi_{uni}\left((\mathrm{E} - \mathbf{A}), \mathbf{b}\right)$ связно и компактно, т. е. ограничено [4].

Предлагаемая методика определения компонент вектора, которые определяют наибольшие отклонения, может быть применена для решения систем с М-матрицами. Как показано в работах [5, 6], из исходной интервальной системы с М-матрицами $\mathrm{A}^- = (\mathrm{E} - \mathrm{A})^-$ и $\mathrm{A}^+ = (\mathrm{E} - \mathrm{A})^+$ и компонент правых частей b^- и b^+ формируются матричные уравнения, позволяющие получить все множество решения СЛАУ $\mathrm{X} = \left\{\mathrm{X}^-, \mathrm{X}^+\right\}$, претендующее на максимальные покомпонентные отклонения:

$$\begin{cases}(\mathrm{E} - \mathrm{A})^+ \mathrm{x} = \mathrm{b}^- \\ (\mathrm{E} - \mathrm{A})^- \mathrm{x} = \mathrm{b}^+\end{cases}. \qquad (6)$$

Здесь под матрицей А понимается технологическая матрица, векторами b^- и b^+ являются Y^-, Y^+, K^-, K^+ из систем (4) и (5), а решением x являются векторы X и P.

Решения системы (6) $\mathrm{X} = \left(\mathrm{X}^-, \mathrm{X}^+\right)$ принадлежат линейному Евклидову пространству с удвоенной размерностью $2n$: $\mathrm{X} \in \mathrm{R}^{2n}$, $\mathrm{X}^- \in \mathrm{R}^n$, $\mathrm{X}^+ \in \mathrm{R}^n$. В качестве x понимаются валовой объем продукции X и цена продукции P.

Благодаря разработанной методике по найденным значениям x^- и x^+ модели можно проследить, при каких вариациях коэффициентов систем (4) и (5) могут возникнуть большие отклонения решений, а также заведомо неправильные решения. Неучтенные отклонения при практической реализации могут привести к нежелательным результатам, не отвечающим ожидаемым прогнозам.

Известные интервальные решения, предлагаемые в различных исследованиях, ограничиваются интервальными и переборными подходами [5, 6]. Точечный (вещественный) вариант решения рассматриваемой задачи приведен в работе [10].

Целью проводимого исследования является оценка искомых векторов в том случае, когда все элементы матриц и компоненты векторов представляются интервалами. Известны средние значения технологической матрицы А, вектора конечного продукта Y и вектора трудозатрат K:

$$\mathrm{A} = \begin{pmatrix} 0{,}095 & 0{,}05 & 0{,}025 & 0{,}08 \\ 0{,}09 & 0{,}08 & 0{,}015 & 0{,}05 \\ 0{,}085 & 0{,}095 & 0{,}050 & 0{,}25 \\ 0{,}08 & 0{,}09 & 0{,}08 & 0{,}015 \end{pmatrix};$$

$$Y = (1\,900\,000;\ 100\,000;\ 50\,000; 16\,000\,000)^{\mathrm{T}};$$

$$K = (100;\ 10;\ 500;\ 3)^{\mathrm{T}}.$$

Кроме того, считаются известными стоимость человекочаса $\varpi = 5$ и годовой объем рабочего времени одного занятого в социально-культурной сфере (СКС): $\mathrm{СКС} = 40 \cdot (52 - 4)$.

Задаются исходные относительные погрешности $\varepsilon = 0{,}01$, $\delta = 0{,}01$ элементов матрицы **A**, вектора конечного потребления **Y** и вектора человеческих трудозатрат **K**, т. е. отклонения границ коэффициентов, равные 1 % от среднего

значения. Интервальные элементы матрицы и компоненты векторов можно представить в следующем виде:

$$\begin{aligned} \boldsymbol{a}_{ij} &= \left[a_{ij} - \varepsilon a_{ij},\ a_{ij} + \varepsilon a_{ij}\right] = \left[\underline{a}_{ij},\ \overline{a}_{ij}\right], \\ \boldsymbol{Y}_i &= \left[Y_i - \delta Y_i,\ Y_i + \delta Y_i\right] = \left[\underline{Y}_i,\ \overline{Y}_i\right], \\ \boldsymbol{K}_i &= \left[K_i - \delta K_i,\ K_i + \delta K_i\right] = \left[\underline{K}_i,\ \overline{K}_i\right], \end{aligned} \qquad (7)$$

где a_{ij} — заданное среднее значение элемента технологической матрицы; Y_i и K_i — средние значения компонент векторов. Тогда, исходя из точечных матричных уравнений (6), полученных для систем (4), (5), найдем решения для переменных X, Y, P двойной размерности через обратные матрицы:

$$\underline{\mathrm{X}} = \left(\overline{\mathrm{E} - \mathrm{A}}\right)^{-1} \underline{\mathrm{Y}};\ \ \overline{\mathrm{X}} = \left(\underline{\mathrm{E} - \mathrm{A}}\right)^{-1} \overline{\mathrm{Y}};$$

$$\mathrm{P} = \left\{\left(\overline{\mathrm{E} - \mathrm{A}}\right)^{-1}\right\}^{\mathrm{T}} \underline{\mathrm{K}};\ \ \mathrm{P} = \left\{\left(\underline{\mathrm{E} - \mathrm{A}}\right)^{-1}\right\}^{\mathrm{T}} \overline{\mathrm{K}},$$

$$\mathrm{x} = \left(\mathrm{x}^{-},\ \mathrm{x}^{+}\right) = \left(\underline{\mathrm{x}},\ \overline{\mathrm{x}}\right),$$

$$\mathrm{P} = \left(\mathrm{P}^{-},\ \mathrm{P}^{+}\right) = \left(\underline{\mathrm{P}},\ \overline{\mathrm{P}}\right),\ \ \mathrm{x},\ \mathrm{P} \in \mathrm{R}^{2n}:$$

$$\mathrm{x}_c = 10^{+7} \begin{pmatrix} 0,3684 \\ 0,1395 \\ 0,0962 \\ 1,6748 \end{pmatrix},\ \ \mathrm{x} = 10^{+7} \begin{pmatrix} 0,3625, & 0,3743 \\ 0,1364, & 0,1426 \\ 0,0939, & 0,0986 \\ 1,6569, & 1,6928 \end{pmatrix}.$$

$$\mathrm{P}_c = 10^{+3} \begin{pmatrix} 0,8582 \\ 0,3941 \\ 2,6749 \\ 0,1728 \end{pmatrix},\ \ \mathrm{P} = 10^{+3} \begin{pmatrix} 0,8496, & 0,8668 \\ 0,3902, & 0,3981 \\ 2,6482, & 2,7017 \\ 0,1711, & 1,1746 \end{pmatrix},$$

где x_c, P_c — точечные решения задачи.

Полный объем трудозатрат $Z = \mathbf{K}X$, связанный с общим количеством занятых во всех отраслях, равен $Z = 10^8 \cdot (8{,}8660,\ 9{,}4160)$ чел.-ч при среднем значении Z_c, равном $10^8 \cdot 9{,}1379$ чел.-ч. Число человек, занятых в СКС, соответственно будет варьироваться в пределах $N = [461\ 770,\ 490\ 420]$ чел., при средней занятости $N_c = 475\ 930$ чел. Рассчитанные средние значения искомых векторов и матриц полностью совпадают со значениями, ранее полученными в работе [10].

В таблице представлены пределы количества людей, занятых в СКС, при заданных начальных погрешностях коэффициентов.

В первых двух столбцах таблицы заданы относительные погрешности ε и δ, в третьем и четвертом столбцах даны интервальные пределы $N = \left[\underline{N},\ \overline{N}\right]$, полученные при разной ширине задаваемых коэффициентов **A** и векторов **Y** и **K** в зависимости от ε и δ (7). В двух последних столбцах показаны погрешности границ интервальных отклонений N к их среднему значению $otn1 = \left(N_c - \underline{N}\right)/N_c \cdot 100\,\%$, $otn2 = \left(\overline{N} - N_c\right)/N_c \cdot 100\,\%$. Из данных таблицы видно, что унифицированная методика позволяет получить максимально возможные погрешности отклонений, составляющие 0,3...16 % в зависимости от задаваемой погрешности коэффициентов ε и δ, равных 0,1...5 %.

ε	δ	$\underline{N}$	$\overline{N}$	$otn1$, %	$otn1$, %
0,001	0,001	474 500	477 370	0,3	0,3
0,001	0,005	470 710	481 190	1,1	1,1
0,01	0,01	461 770	490 420	2,98	3,04
0,01	0,05	425 210	530 030	10,66	11,37
0,05	0,05	408 270	551 720	14,22	15,92

Из этих вычислений следует, что на практике возможное изменение количества людей, занятых в СКС, в действительности нельзя определять средним числом N_c, так как отклонение от среднего количества людей, занятых в СКС, может составлять сотни тысяч человек в зависимости от первоначальных погрешностей параметров. Значение этих отклонений должно быть заложено в интервальную модель расчета.

Заключение

Впервые получены интервальные решения прямой и обратной задач открытой линейной статической балансовой модели, где отраслями экономического процесса выступают объекты кино-видео индустрии. Алгебраический подход к интервальным системам позволил осуществить расчет с оптимальными угловыми матрицами, отождествив их с граничными М-матрицами интервальной системы, моделирующей экономический объект.

Определены интервалы изменения величины полного объема продукции X, вектора цен P и числа людей N, занятых в социально-культурной сфере, при интервально заданных векторах конечного объема продукции **Y** и трудовых затрат **K**.

Список литературы

1. **Гильмутдинов Р. З.** Курс лекций: "Математические методы в экономике". Уфа: УИКиП, 2006. 53 с.
2. **Шандра И. Г.** Математические аспекты макро- и микроэкономики. Тексты лекций спец. курса. М.: Финансовая академия при правительстве РФ, 1998. 40 с.
3. **Петров Ю. П.** Как получать надежные решения систем уравнений. СПб.: БХВ — Петербург, 2009. 176 с.
4. **Шарый С. П.** Конечномерный интервальный анализ. Новосибирск: Институт вычислительных технологий СО РАН, 2009. 569 с.
5. **Иванова К. Ф.** Унификация точечных алгебраических методик внешней оценки решений интервальных систем // Программная инженерия. 2016. Т. 7, № 4. С. 181—189.
6. **Иванова К. Ф.** Оценка решения линейных алгебраических уравнений с неопределенными входными параметрами // Материалы X международной научно-практической конференции "Фундаментальные и прикладные науки сегодня". 26—27 декабря 2016 г. Noth Charlston, USA. 2016. Vol. 3. P. 101—113.
7. **Nirmala T., Datta D., Kushwaha H. S., Ganesan K.** Inverse Interval Matrix: A New Approach // Applied Mathematical Sciences. 2011. Vol. 5, No. 13. P. 607—624.
8. **Малугин В. А.** Математика для экономистов. Линейная алгебра. М.: ЭКСМО, 2006. 211 с.
9. **Шананин А. А.** Математические модели в экономике. Москва, 1999. 58 с.
10. **Махов А. М.** Модель В. Леонтьева. Линейное программирование. Уч.-метод. пособие по дисциплине "Математическое моделирование в экономике". СПб., 1999. 54 с.
11. **Горемыкина Г. И., Ляшко М. А.** Избранные разделы линейной алгебры с элементами экономической алгоритмики. Уч.-метод. пособие для студентов экономич. и физ.-мат. ф-тов. Балашов: Николаев, 2003. 96 с.
12. **Иванова К. Ф.** Оценка объединенного множества решений задачи на основе интервальной модели Леонтьева // Программная инженерия. 2014. № 1. С. 40—47.

The Forecast of Gross Output and Manpower of Intersectoral Balance Model

K. F. Ivanova, Klara.I2010@yandex.ru, Saint Petersburg State University, Saint Petersburg, 195252, Russian Federation

Corresponding author:

Ivanova Klara F., Doctorant, Saint Petersburg State University, Saint Petersburg, 195252, Russian Federation, E-mail: Klara.I2010@yandex.ru

Received on March 23, 2017
Accepted on March 30, 2017

In article the unified technique for solution's estimation of V. Leont'ev's model, in the conditions of uncertainty of model's factors is offered. Statement of a problem of balance model concerns problems of functional optimization of gross output and other indicators oriented on forecasting of a planned economy.

The functioning of multisectoral macroeconomic requires the balance between separate sectors. Each sector, on the one hand, is the producer, and on the other hand — the consumer of products issued by other sectors. These conditions set the connection between sectors based on the output and consumption of different types of products. Uncertainty of the input data arising at an error of measurements and their fluctuations is limited by some boundaries (intervals) where there can be their true value and makes the considered balance model more difficult for evaluation. The developed technique of a two-sided estimation of equations is based on use of known algebraic methods of linear algebra, and synthesis of two approaches to a solution of an estimation of sensitivity of a solution of the linear algebraic equations and exterior estimations of solution's interval set. The suggested approach for an estimation of the solution of interval model is constructed on two algebraic equations received on the basis of identity of boundary interval technological matrices, and two formalized point optimal angular matrices. This approach differs from a earlier known one, received by methods of interval algebra. The algorithm of calculation is directed at searches of admissible areas of a solution of the balance equations of definition gross output, its prices, and amounts of the people occupied in an industry or in the social ratio-cultural areas. Definition of interval deviations of these indicators leads to a two-sided estimation of a solution in comparison with its mean values. The received results of calculations are the basis for acceptance of optimum decisions and the forecast in economic, industrial or social planning of static multisectoral balance model.

Keywords: *balance model, uncertainty of the parameters, the unified technique, the interval approach, algorithm of calculation, optimal angular matrices, two-sided estimation*

For citation:

Ivanova K. F. The Forecast of Gross Output and Manpower of Intersectoral Balance Model, *Programmnaya Ingeneria,* 2017, vol. 8, no. 6, pp. 276—281.

DOI: 10.17587/prin.8.276-281

References

1. **Gilmutdinov R. Z.** *Kurs lekcij "Matematicheskie metody v ekonomike"* (Course of lectures: "Mathematical methods in economy), Ufa: YIKiP. 2006, 53 p. (in Russian).

2. **Shandra I. G.** *Matematicheskie aspecty macro i mikroekonomiki. Teksty lekcij spec. kursa* (Mathematical aspects macro and micro-economics. Texts of special course lectures), Moscow, Finansovaya akademiya pri pravitelstve RF, 1998, 40 p. (in Russian).

3. **Petrov U. P.** *Kak poluchat' nadezhnye resheniya system uravnenij* (How to receive reliable decisions of systems of the equations), Saint Petersburg, BHV-Peterburg, 2009, 176 p. (in Russian).

4. **Sharij S. P.** *Konechnomernyj interval'nyj analiz* (Finite interval analysis), Novosibirsk, Institut vychislitel'nyh tehnologij SO RAN, 2009, 569 p. (in Russian).

5. **Ivanova K. F.** Unifikaciya tochechnyh algebraicheskih metodic vneshnej ocenki reshenij interval'nyh system (Unification of Point Algebraic Techniques of an Exterior Estimation of Set Solutions Interval Systems), *Programmnaya Ingeneria*, 2016, vol. 7, no. 4, pp. 181—189 (in Russian).

6. **Ivanova K. F.** Ocenka resheniya linejnyh tochechnyh uravnenij s neopredelennymi vhodnymi parametrami (Estimation of the solution of the linear algebraic equations with uncertain input parameters), *Materialy X mezhdunarodnoj nauchno-practicheskoj konferencii*, 26—27 December, 2016, Noth Charlston, USA, 2016, vol. 3, pp. 101—113 (in Russian).

7. **Nirmala T., Datta D., Kushwaha H. S., Ganesan K.** Inverse Interval Matrix: A New Approach, *Applied Mathematical Sciences*, 2011, vol. 5, no. 13, pp. 607—624.

8. **Malugin V. A.** *Matematika dlya ekonomistov. Linejnaya algebra* (Mathematics for economists. Linear algebra), 2006, 211 p. (in Russian).

9. **Shananin A. A.** *Matematicheskie modeli v ekonomike* (Mathematical model in economy), Moscow, 1999, 98 p. (in Russian).

10. **Mahov A. M.** *Model' Leont'eva. Linejnoe Programmirovanie. Uchebno-metodicheskoe posobie po discipline "Matematicheskoe modelirovanie v ekonomike"* (Leont'ev's model. The Linear programming. Study- methodical tutorial on discipline "Mathematical modelling in economy"), Saint Petersburg, 1999, 54 p. (in Russian).

11. **Goremykina G. A., Lyashko M. A.** *Izbrannye razdely linejnoj algebry s elementami ekonomicheskoj algoritmiki. Uchebno-metodicheskoe posobie dlya studentov ekonomicheskih fakul'tetov* (The selected sections of linear algebra with elements economic algoritmic. A study-method. The tutorial for students of economic and physics-mathematics faculties), Balashov, Nikolaev, 2003, 96 p. (in Russian).

12. **Ivanova K. F.** Ocenka ob'edinennogo mnozhestva reshenij zadachi linejnoj algebry na osnove inteval'noj modeli Leont'eva (Estimation of the United Solution Set of an Interval Leontjev's Model), *Programmnaya Ingeneria*, 2014, no. 1, pp. 40-47 (in Russian).

УДК 004.92 DOI: 10.17587/prin.8.282-288

В. В. Бурлов, канд. техн. наук, проф., e-mail: vladimir-burlov@yandex.ru,
Л. В. Ремонтова, доц., e-mail: remontova@mail.ru, Пензенский государственный технологический университет, г. Пенза, **В. В. Косолапов,** канд. техн. наук, доц., e-mail: vladimir.kosolapov@mail.ru, **Е. В. Косолапова,** ст. преподаватель, e-mail: K-art-inka@yandex.ru, Нижегородский государственный инженерно-экономический университет, г. Княгинино

Инструментальные средства 3D-моделирования поверхностей второго порядка

Представлены функциональные возможности программного продукта КОМПАС-3D. Рассматриваются различные способы построения кривой гиперболы и алгоритмы создания производных от нее поверхностей в системе трехмерного моделирования.

Статья направлена на продвижение отечественного IT-продукта в сфере образования, на более глубокое изучение дисциплин, связанных с моделированием, и развитие интереса к личной геометрической и графической подготовке, без которой невозможно качественное инженерное творчество.

Ключевые слова: *ассоциативный чертеж, библиотека, график функциональной зависимости, гипербола, конус, однополостный и двуполостный гиперболоид, канонические и параметрические уравнения, сечение, система КОМПАС-3D, цилиндр*

Введение

При проектировании изделий разного уровня сложности используются различные кривые. К ним можно отнести как кривые второго порядка, так и кривые, которые задаются различного рода уравнениями. Конструктор, использующий ту или иную систему САПР, должен иметь возможность не только строить, но и редактировать кривую в любой момент.

Значение как кривых, так и поверхностей второго порядка в нашей жизни достаточно велико. Эллипс, гипербола и парабола имеют многочисленные геометрические свойства и физические приложения. Например, орбиты планет, вращающихся вокруг своей звезды — эллипсы, причем звезда находится в фокусе этого эллипса. Кометы движутся в пределах солнечной системы почти по гиперболе. Оптическое свойство параболы широко применяется сегодня в самых различных сферах жизни — карманный фонарик, автомобильные фары, прожекторы и т. д. Траекторией подброшенного вверх тела также является парабола.

Благодаря техническому развитию решение простых и сложных пространственных задач, связанных с моделированием поверхностей, осуществляется с помощью различных программных продуктов. В России, благодаря мощным функциональным возможностям твердотельного и поверхностного моделирования, стандартом для тысяч предприятий стала система трехмерного моделирования КОМПАС-3D. Однако в графическом редакторе КОМПАС-3D есть команда **Эллипс**, но нет команды **Гипербола**, без которой невозможно создать 3D-модели соответствующих поверхностей второго порядка — однополостного или двуполостного гиперболоидов.

Цель настоящей статьи — продемонстрировать алгоритмы решения пространственных задач по созданию кривых и поверхностей второго порядка в программе КОМПАС-3D на основе законов начертательной геометрии с применением конических сечений кругового конуса, канонических и параметрических уравнений с построением 3D-моделей соответствующих поверхностей.

1. Использование конических сечений

Система КОМПАС-3D позволяет выполнить ассоциативный чертеж детали любой сложности с необходимыми разрезами и сечениями. Если в качестве 3D-модели использовать конус вращения, то на ассоциативном чертеже можно получить в виде конических сечений все кривые второго порядка. Сечения в виде гипербол можно использовать для 3D-моделирования таких поверхностей, как, например, однополостный или двуполостный гиперболоид.

Напомним, при каких условиях можно получить необходимые кривые второго порядка [1, 2].

Если плоскость параллельна двум образующим конуса, то коническое сечение представляет собой гиперболу (рис. 1).

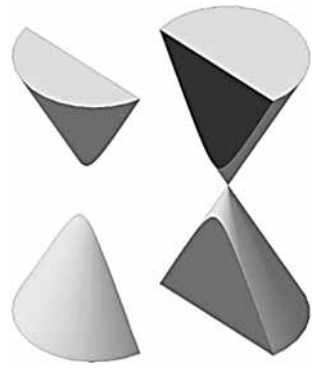

Рис. 1. Образование сечений конической поверхности

Для получения в системе КОМПАС-3D нужных для 3D-моделирования кривых создается двуполостный конус вращения с помощью операции **Вращения**.

Порядок создания конуса и конических сечений

Создаем новый документ ***Деталь***, в дереве модели присваиваем детали имя ***Конус*** и сохраняем файл под этим именем.

1. За плоскость эскиза выбираем фронтальную плоскость проекций (*XY*).
2. Выбираем команду **Эскиз** (для создания эскиза).
3. Создаем эскиз конуса, для этого (рис. 2):

- строим ось вращения вертикально через начало координат командой **Отрезок** со стилем линии *Осевая* (рис. 2, *а*);
- строим прямую через начало координат под углом 60° к оси *X* командой **Вспомогательная прямая** (рис. 2, *а*);
- строим прямую на расстоянии, например, 100 мм над началом координат командой **Горизонтальная прямая**;
- образующую конуса проводим по вспомогательной прямой от верхней границы до нижней границы командой **Отрезок** со стилем линии *Основная* (рис. 2, *б*);
- у зоны начала координат делаем разрыв образующей конуса командой **Усечь кривую 2 точками** на 0,2 мм (рис. 2, *б*), чтобы образующая не имела точки пересечения с осью вращения. ***В противном случае 3D-модель двуполостного конуса создаваться не будет***.

4. Выходим из режима **Эскиз**.
5. Выбираем команду **Операция вращения** 3D-моделирования и командой **Создать объект** получаем результат (рис. 3).

Чтобы получить изображение гиперболы, создадим ассоциативный чертеж конуса. Командами **Создать чертеж** и **Менеджер документа** заготавливаем формат А2 чертежа и сохраняем его под именем ***Конус***. В панели инструментальных средств ***Виды*** выбираем команду **Стандартные виды**. В опциях этой команды выбираем изображения ***Главного вида*** и ***Вида сверху***. Размещаем данные проекции в левой части чертежа (рис. 4). Теперь на ассоциативном чертеже можно построить любые сечения.

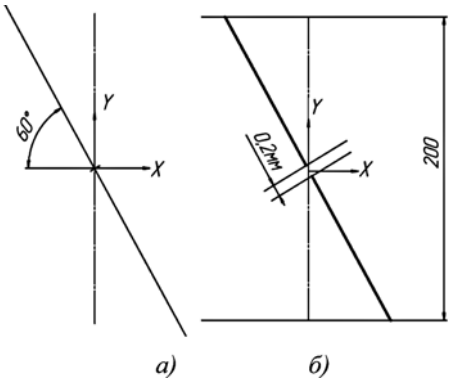

Рис. 2. Построение эскиза конуса в программе КОМПАС-3D

Рис. 3. Поверхность кругового конуса

Построение гиперболы. Для этого проведем через вершину *S* две образующие: $(S-2) \equiv (S-3)$. Эти образующие лежат во фронтально-проецирующей плоскости, поэтому их проекции на фронтальной плоскости совпадают. След секущей плоскости А-А проводим параллельно $(S''-2'')$. В результате получаем в сечении А-А две ветви гиперболы (рис. 4). Чтобы иметь возможность перемещать, вращать или масштабировать эти кривые, проведем в них оси.

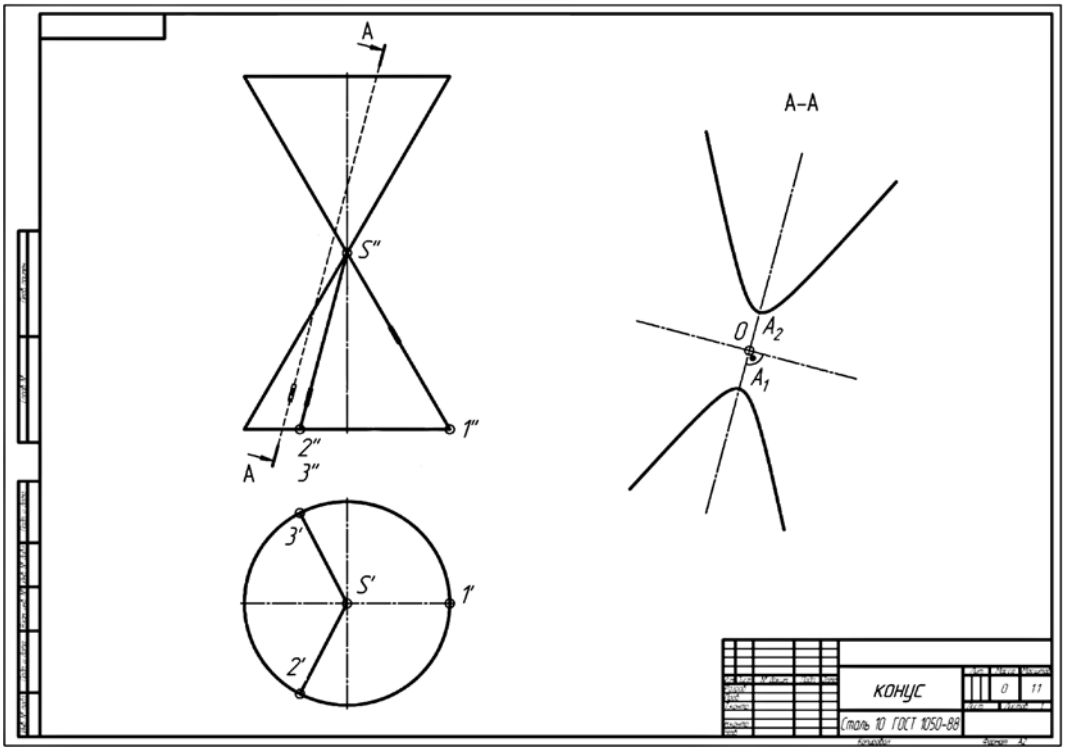

Рис. 4. Сечения поверхности конуса на ассоциативном чертеже

Каждая ось симметрии проводится через вершину кривой и параллельно следу секущей плоскости. Для определения мнимой оси гиперболы необходимо через середину 0 отрезка между вершинами A_1 и A_2 гиперболы провести прямую (ось), перпендикулярную к действительной оси гиперболы (рис. 4).

Рассмотрим примеры использования гиперболы при 3D-моделировании.

Алгоритм создания двуполостного гиперболоида вращения

Создаем новый документ ***Деталь***, в ***Дереве модели*** присваиваем детали имя ***Двуполостный гиперболоид***.

1. За плоскость эскиза выбираем фронтальную плоскость проекций *XY*.

2. Выбираем команду **Эскиз** (и создаем эскиз гиперболы):

- открываем чертеж конуса с сечениями (рис. 4) и командой **Выделить объект** выделяем гиперболу с ее осями;
- командой **Копировать** из меню ***Редактор*** копируем выделенные объекты в буфер системы с привязкой к центру 0 гиперболы;
- возвращаемся к документу ***Деталь*** и командой **Вставить** вставляем эскиз гиперболы с привязкой к началу координат (рис. 5, *а*);
- выделяем ось и гиперболу;
- командой **Поворот** доворачиваем фигуру до вертикального положения оси и удаляем правую часть гиперболы (рис. 5, *б*).

3. Выходим из режима **Эскиз**.

4. Выбираем команду **Операция вращения** 3D-моделирования и командой **Создать объект** завершаем работу (рис. 5, *в*).

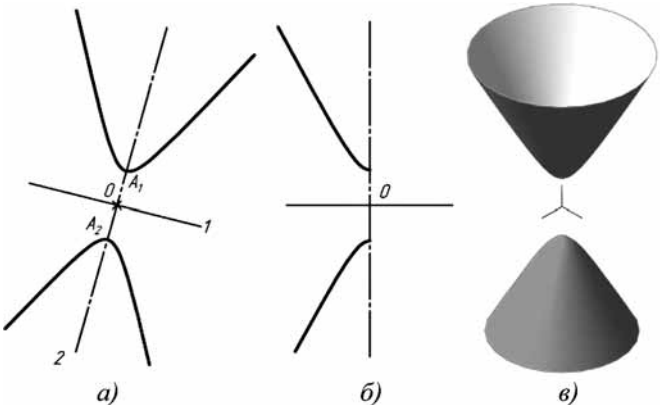

Рис. 5. Построение двуполостного гиперболоида вращения

2. Использование канонических уравнений гиперболы

В программе КОМПАС-3D среди множества библиотек имеется библиотека ***FTDraw***, входящая в меню библиотек ***Прочие***. С помощью этой библиотеки по каноническому уравнению кривой можно построить график функции, т. е. изображение кривой гиперболы или параболы по заданным условиям. Затем полученное изображение можно использовать для 3D-моделирования поверхностей второго порядка или четвертого порядка.

Геометрия и каноническое уравнение гиперболы

Элементы гиперболы (рис. 6): *ветви* — левая и правая; *центр* гиперболы 0; F_1 и F_2 — *фокусы*; A и A' — *вершины* гиперболы; отрезок AA' — *действительная ось* гиперболы; B и B' — концы *мнимой оси* гиперболы.

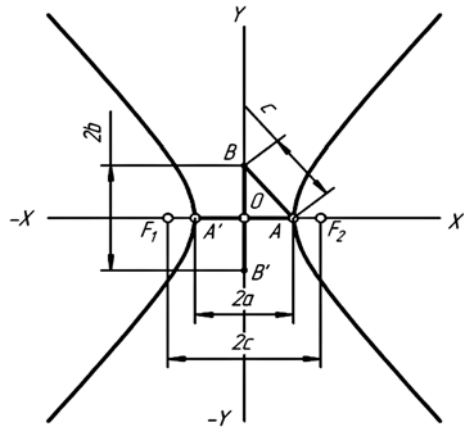

Рис. 6. Гипербола

Параметры гиперболы**:** $2a$ — размер действительной оси; $2b$ — размер мнимой оси; $2c$ — расстояние между фокусами.

Каноническое уравнение гиперболы:

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1. \qquad (1)$$

При этом известна зависимость: $c^2 = a^2 + b^2$, т. е. зная значения полуосей гиперболы, можно узнать значение межфокусного расстояния c.

Мнимая ось $2b$ может быть меньше, равна или больше действительной оси $2a$. Если оси равны ($a = b$), то гипербола называется равносторонней или равнобокой. Чем больше действительная полуось a по отношению к мнимой полуоси b, тем уже раствор ветви гиперболы.

Выразим для гиперболы зависимость координаты y от x из формулы (1):

$$\frac{y^2}{b^2} = \frac{x^2}{a^2} - 1 \rightarrow y^2 = b^2\left(\frac{x^2}{a^2} - 1\right) \rightarrow y = b\sqrt{\frac{x^2}{a^2} - 1}. \qquad (2)$$

Теперь, используя формулу (2) и задав нужные параметры кривой, можно получить ее изображение на экране монитора с помощью команд из библиотеки ***FTDraw***.

Алгоритм создания гиперболы

Открываем документ ***Фрагмент*** и присваиваем ему имя ***Гипербола***. Назначим для гиперболы, например, следующие параметры: размер действительной полуоси $a = 10$, размер мнимой полуоси $b = 12$, тогда по формуле (2):

$$y = 12\sqrt{\frac{x^2}{100} - 1}.$$

Приступаем к работе с библиотекой ***FTDraw*** (рис. 7). Обращаемся к командам: ***Менеджер библиотек*** (▤) → → ***Прочие*** (Прочие). В списке библиотек ***Прочие*** выбираем ***Библиотека FTDraw*** (рис. 7, *а*).

В диалоговом окне (ДО) ***Библиотека FTDraw1.1*** (рис. 7, *б*) выполняем двойной щелчок мышью на кнопке (*Построение графиков функциональных зависимостей по уравнению в декартовых координатах*). Появляется новое ДО — ***Построение графика функциональных зависимостей*** (рис. 7, *в*).

Используя форму записи формул, принятую в системе КОМПАС-3D, записываем в этом ДО уравнения гиперболы — 12*Sqrt(x^2/10^2-1) (рис. 7, *в*).

Судя по параметрам уравнения, вершина гиперболы удалена от оси *Y* на 10 мм. Поэтому заказываем пределы изменения *x* от 10 до, например, 40 мм, и число точек кривой 20 (для обеспечения более высокой точности графика).

В ДО (рис. 7, *в*) имеется пять команд для управления процессом создания графиков. В ДО ***Построение графиков функциональных зависимостей*** выбираем пункт ***Просмотр расчета*** (). Если расчет выполнен по заданной формуле и при заданных границах (рис. 7, *г*), то просто закрываем ДО ***Просмотр результатов расчета***.

В ДО ***Построение графиков функциональных зависимостей*** (рис. 7, *в*) выбираем пункт ***Указать положение базовой точки графика*** ().

На экране появляется контекстное меню Ввод базовой точки графика. Курсором привязываемся к началу координат документа ***Фрагмент***. Затем в ДО (рис. 7, *в*) выбираем пункт ***Построить график*** **()** и нажимаем кнопку **OK**. В результате получаем изображение верхней половины правой ветви гиперболы (рис. 7, *д*). Закрываем диалоговые окна.

Это изображение части гиперболы создано системой в виде ***Макроэлемента***. Чтобы его использовать для 3D моделирования, кривую выделяем и командой **Разрушить** из меню ***Редактор*** преобразуем в кривую Безье. Проводим оси гиперболы через начало координат: действительную ось линией *Осевая*, а мнимую ось — *Тонкой линией*. **Выделяем** все и **копируем** в буфер обмена с привязкой к центру гиперболы.

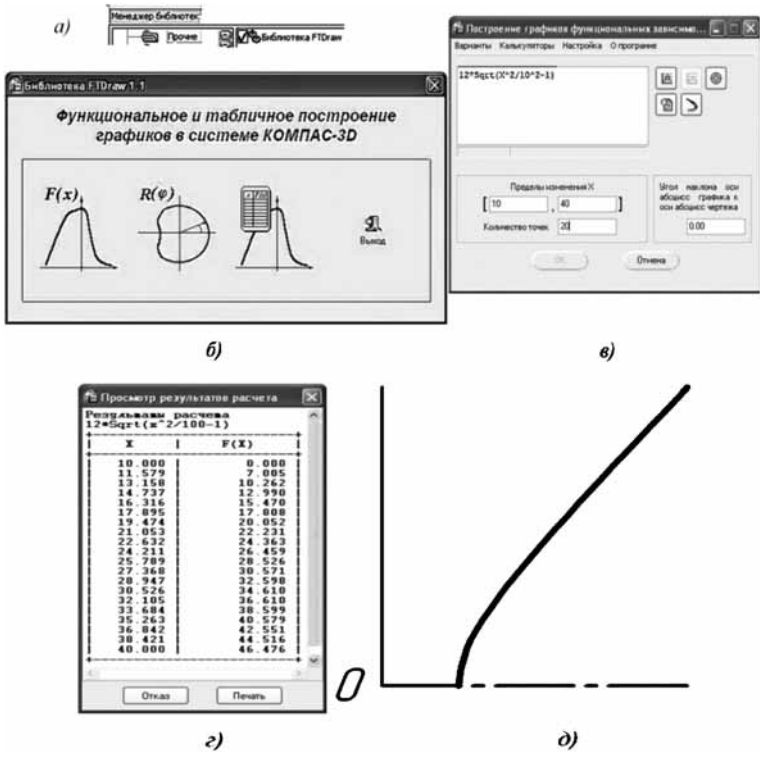

Рис. 7. Построение гиперболы в программе КОМПАС-3D

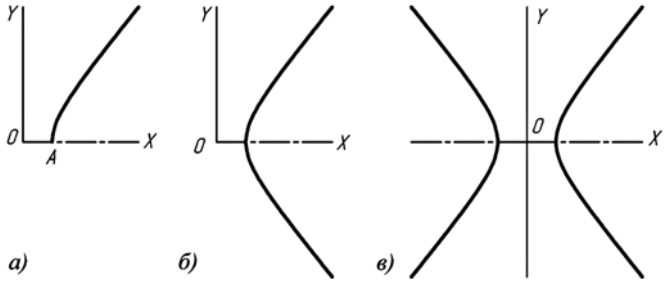

Рис. 8. Построение гиперболы

Алгоритм создания гиперболического цилиндра

Верхняя половина правой ветви гиперболы создана (рис. 8, *а*). Однако для моделирования гиперболического цилиндра необходимо иметь полное изображение гиперболы.

Командой **Симметрия** относительно оси *X* из меню ***Редактор*** создаем нижнюю часть правой ветви гиперболы (рис. 8, *б*), затем командой **Симметрия** относительно оси *Y* из меню ***Редактор*** создаем левую ветвь гиперболы (рис. 8, *в*). **Выделяем** полное изображение гиперболы. Командой **Копировать** из меню ***Редактор*** копируем гиперболу в буфер обмена с привязкой к центру 0 гиперболы.

Создаем новый документ ***Деталь***, в дереве модели присваиваем детали имя ***Гиперболический цилиндр*** и сохраняем файл под этим именем.

1. За плоскость эскиза выбираем горизонтальную плоскость проекций *ZX*.
2. Выбираем команду **Эскиз** (для создания эскиза гиперболы) → командой **Вставить** вставляем эскиз гиперболы с привязкой к началу координат.
3. Выходим из режима **Эскиз**.
4. Обращаемся к команде **Операция выдавливания.** Назначаем необходимые параметры, например, с тонкой стенкой толщиной 0,01 мм, и опцией **Создать объект** создаем 3D-модель гиперболического цилиндра (рис. 9).

Рис. 9. Поверхность гиперболического цилиндра

3. Использование параметрических уравнений гиперболы

В графическом редакторе КОМПАС-3D при работе с документом ***Деталь*** в меню ***Пространственные кривые*** () доступна команда **Кривая по закону** (). Эта команда позволяет получить изображение кривой второго порядка, заданной в параметрической форме в пространственной прямоугольной системе () координат. Так как имеем дело с плоской кривой,

то при задании координат X и Y в параметрической форме координате Z необходимо задать значение, равное нулю.

Чтобы использовать эту кривую для моделирования поверхности второго порядка, необходимо спроецировать ее на плоскость XY эскиза. Созданные в эскизе проекции **Кривой по закону** ассоциативно связаны со своими исходными объектами. Например, при редактировании параметров кривой будет изменяться и эскиз. Это проекционная связь не дает возможности редактировать эскиз на месте. Ее удаление возможно двумя путями:

- вырезать созданный эскиз в буфер и обратно вернуть плоскость эскиза;
- после создания эскиза удалить кривую по закону.

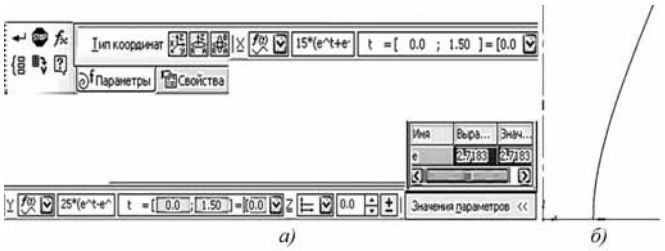

Рис. 10. Внесение исходных данных гиперболы

Параметрическое уравнение гиперболы

Используем форму записи параметрического уравнения гиперболы с помощью гиперболических функций:

$$x = a \cdot \mathrm{sh}(t), \quad y = b \cdot \mathrm{ch}(t), \qquad (3)$$

при форме записи (3) действительная ось гиперболы совпадает с осью X.

Здесь t — параметр уравнений; a — размер действительной полуоси гиперболы; b — размер мнимой полуоси гиперболы, а гиперболические функции определены следующими зависимостями:

$$\mathrm{sh}(t) = \frac{\mathbf{e}^t + \mathbf{e}^{-t}}{2}, \quad \mathrm{ch}(t) = \frac{\mathbf{e}^t - \mathbf{e}^{-t}}{2},$$

где $\mathbf{e} = 2{,}718281828$ — число Эйлера.

Существует также форма записи параметрического уравнения гиперболы и в тригонометрических функциях.

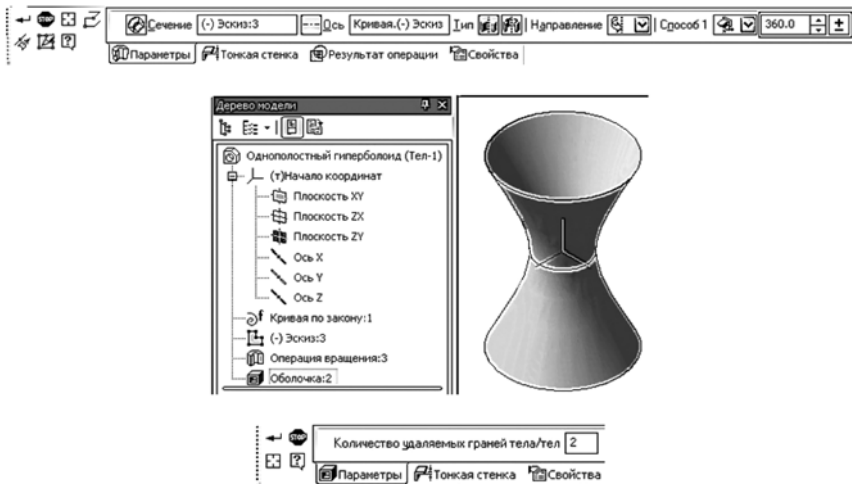

Рис. 11. Создание однополостного гиперболоида

Алгоритм создания однополостного гиперболоида

Открываем документ ***Деталь***. В ***Дереве модели*** с помощью команды **Свойства модели** присваиваем файлу имя **Однополостный гиперболоид** и сохраняем файл под этим именем.

Создадим эскиз гиперболы в плоскости XY по параметрическому уравнению (3), например, при следующих данных: $a = 15$; $b = 25$; $t = 1{,}5$.

Для этого активизируем инструментальную панель ***Пространственные кривые***. В этой панели выбираем команду **Кривая по закону**.

В панели **Свойств** команды записываем исходные данные гиперболы в прямоугольной системе координат (рис. 10, *а*). Завершаем работу опцией **Создать объект**.

В результате создана часть гиперболы в плоскости XY системы (рис. 10, *б*).

Создаем эскиз гиперболы:

- выделяем плоскость XY в дереве модели и выбираем команду **Эскиз**;
- в инструментальной панели ***Геометрия*** выбираем команду **Спроецировать объект**. Выделяем созданную кривую; завершаем работу кнопкой **Stop**;
- выделяем этот эскиз и командой **Вырезать** из меню ***Редактор*** помещаем эскиз в буфер обмена;
- командой **Вставить** из меню ***Редактор*** возвращаем изображение в эскиз плоскости XY; в результате связь с **Кривой по закону** разорвана;
- через начало координат проводим горизонтальную прямую стилем *Вспомогательная* и вертикальную прямую стилем *Осевая*;
- выделяем кривую и командой **Симметрия** создаем полное изображение правой ветви гиперболы;
- командой **Эскиз** завершаем создание изображения правой ветви гиперболы.

Для моделирования однополостного гиперболоида вращения используем команду **Операция вращения** (рис. 11, *а*) — вращение выполнено вокруг мнимой оси гиперболы. В результате получаем тело, ограниченное поверхностью гиперболоида. Для получения тонкостенного изображения гиперболоида использована команда Оболочка (рис. 11, *в*). В результате получено 3D-изображение тонкостенного однополостного гиперболоида вращения с толщиной стенки 0,01 мм (рис. 11, *б*).

Алгоритм создания двуполостного гиперболоида вращения

Открываем документ ***Деталь***. В ***Дереве модели*** с помощью команды **Свойства модели** присваиваем файлу имя **Двуполостный гиперболоид** и сохраняем файл под этим именем.

Кривая по закону создана с параметрами $a = 25$, $b = 15$ и $t = 2$ (рис. 12).

С помощью этой кривой создан эскиз гиперболы с поворотом на 90°. Затем командой **Операция вращения** (рис. 13, *а*) создана 3D-модель гиперболоида и операцией **Оболочка** (рис. 13, *в*) получено изображение гиперболоида с толщиной стенки, равной 0,01 мм (рис. 13, *б*). Для наглядности изображена его действительная ось.

Рис. 12. Параметры кривой

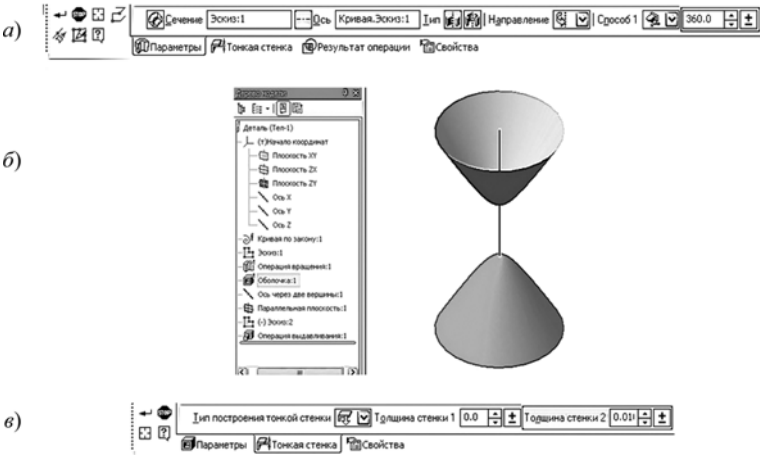

Рис. 13. Создание двухполостного гиперболоида

Заключение

В статье описаны функциональные возможности программного продукта КОМПАС-3D. Подробно изложено применение сечения кругового конуса для построения гиперболы и ее канонических и параметрических уравнений для моделирования 3D-поверхностей второго порядка в системе КОМПАС-3D.

Представленные методы позволяют создавать наглядные модели, которые можно показать обучающимся на экране с разных сторон путем вращения или выбором ориентации модели относительно плоскостей проекций. Это способствует развитию пространственного воображения.

Данная работа направлена на продвижение отечественного IT-продукта в сфере образования, на более глубокое изучение геометро-графических дисциплин. Представленные в статье инструментальные средства могут быть полезны студентам вузов, преподавателям и специалистам в области геометрического моделирования.

Список литературы

1. **КОМПАС-3D** V13: руководство пользователя в 3-х т. М.: ИТАР-ТАСС, 2012. Т. 3. 424 с.

2. **Иванов Г. С.** Теоретические основы начертательной геометрии: учеб. пособие. М.: Машиностроение, 1998. 160 с.

3. **Тактаров Н. Г.** Справочник по высшей математике. М.: ЛИБРОКОМ, 2014. 880 с.

4. **Выгодский М. Я.** Справочник по высшей математике. М.: Астрель, 2002. 992 с.

5. **Фролов С. А.** Начертательная геометрия: учеб. для втузов. 3-е изд., перераб. и доп. М.: Машиностроение, 2008. 240 с.

6. **Иванов Г. С.** Начертательная геометрия: учеб. для вузов. М.: Машиностроение, 1995. 224 с.

7. **Четверухин Н. Ф.** Проективная геометрия: учеб. для пед. вузов. 8-е изд. М.: Просвещение, 1969. 368 с.

8. **Нестеренко Л. А.** Компьютерная графика в начертательной геометрии: учеб. пособие. Пенза: Изд-во Пензенского гос. технолог. ун-та, 2013. 151 с.

9. **Хейфец А. Л.** Реорганизация курса начертательной геометрии как актуальная задача развития кафедр графики // Геометрия и графика. 2013. Т. 1, № 2 (2). С. 21—23.

10. **Серегин В. И., Иванов Г. С., Боровиков И. Ф., Сенченкова Л. С.** Геометрические преобразования в начертательной геометрии и инженерной графике // Геометрия и графика. 2015. Т. 3, № 2. С. 23—28.

11. **Арциховская-Кузнецова Л. В.** О головоломности в начертательной геометрии // Геометрия и графика. 2014. Т. 2, № 3. С. 31—35.

12. **Суфляева Н. Е.** Современные аспекты преподавания графических дисциплин в технических вузах // Геометрия и графика. 2014. Т. 2, № 4. С. 28—33.

13. **Сальков Н. А.** Параметрическая геометрия в геометрическом моделировании // Геометрия и графика. 2014. Т. 2, № 3. С. 7—13.

14. **Волошинов Д. В., Соломонов К. Н.** Конструктивное геометрическое моделирование как перспектива преподавания графических дисциплин // Геометрия и графика. 2013. Т. 1, № 2 (2). С. 10—13.

15. **Гирш А. Г.** Мнимости в геометрии // Геометрия и графика. 2014. Т. 2, № 2. С. 3—8.

16. **Короткий В. А., Хмарова Л. И.** Начертательная геометрия на экране компьютера // Геометрия и графика. 2013. Т. 1, № 1. С. 32—34.

17. **Бойков А. А.** О круговых орбитах планет // Геометрия и графика. 2013. Т. 1, № 2 (2). С. 66—67.

18. **Бойков А. А.** Компьютерные средства поддержки учебных курсов графических дисциплин // Геометрия и графика. 2013. Т. 1, № 2 (2). С. 29—30.

19. **Козлова И. А., Славин Б. М., Харах М. М., Гусева Т. В.** Построение линий пересечения некоторых сложных поверхностей // Геометрия и графика. 2015. Т. 3, № 2. С. 38—45.

20. **Сальков Н. А.** Свойства циклид Дюпена и их применение. Часть 1 // Геометрия и графика. 2015. Т. 3, № 1. С. 16—25.

3D Modeling Tools for Second-Order Surfaces

V. V. Burlov, vladimir-burlov@yandex.ru, **L. V. Remontova,** remontova@mail.ru, Penza State Technological University, Penza, 440039, Russian Federation, **V. V. Kosolapov,** vladimir.kosolapov@mail.ru, **E. V. Kosolapova,** K-art-inka@ya.ru, Nizhny Novgorod State University of Engineering and Economics, Knyaginino, 606340, Russian Federation

Corresponding author:

Burlov Vladimir V., Professor, Penza State Technological University, 440039, Penza, Russian Federation, E-mail: vladimir-burlov@yandex.ru

Received on March 23, 2017
Accepted on March 30, 2017

The article is intended for graduate students and professors whose professional activity is connected with problems of deep study descriptive geometry and engineering graphics on the basis of modern computer technology. The article will be useful to university students with the technical direction of training in order to better understand the subject of descriptive geometry and instilling interest in personal geometric and graphic training, without which it is impossible implement to quality engineering creativity.

The article focuses on harnessing the power of Kompas-3D software product with which it is possible to solve virtually any educational and professional engineering and graphics tasks. Along with this, the promotion of domestic IT-product in education is an urgent task, arising from the problem of technical education students at the present stage of development of the system of public education.

A number of examples of spatial solutions of tasks connected with modeling quadratics are considered. On the basis of laws applicable to descriptive geometry, surface tasks, the article illustrates a feature of their solution and display on your computer screen. For example, in the graphic editor Kompas-3D there is the ellipse command, but no commands Hyperbole. And without this curve, it is impossible to create a 3D model of a single-sheeted or two-sheeted hyperboloid. To create a hyperbola, the program suggests using: sections of a circular cone, canonical or parametric equations. For each of these options examples of creating 3D models of 2-surfaces are considered.

Keywords: *associative drawing, library, functional dependency graph, hyperbole, cone, tapered section, single cavity and dvupolostnyj hyperboloid, canonical and parametric equations, cross-section, Kompas-3D system, cylinder*

For citation:

Burlov V. V., Remontova L. V., Kosolapov V. V., Kosolapova E. V. 3D Modeling Tools for Second-Order Surfaces, *Programmnaya Ingeneria,* 2017, vol. 8, no. 6, pp. 282—288.

DOI: 10.17587/prin.8.282-288

References

1. ***KOMPAS-3D V13****: rukovodstvo pol'zovatelya* (KOMPAS-3D V13: User's manual), Moscow, ITAR-TASS, 2012, vol. 3. 424 p. (in Russian).

2. **Ivanov G. S.** *Teoreticheskie osnovy nachertatel'noi geometrii* (Theoretical Foundations of Descriptive Geometry), Moscow, Mashinostroenie, 1998, 160 p. (in Russian).

3. **Taktarov N. G.** *Spravochnik po vysshei matematike* (Handbook of Higher Mathematics), Moscow, LIBROKOM, 2014, 880 p. (in Russian).

4. **Vygodskii M. Ya.** *Spravochnik po vysshei matematike* (Handbook of Higher Mathematics), Moscow, Astrel', 2002, 992 p. (in Russian).

5. **Frolov S. A.** *Nachertatel'naya geometriya* (Descriptive geometry), Moscow, Mashinostroenie, 2008, 240 p. (in Russian).

6. **Ivanov G. S.** *Nachertatel'naya geometriya* (Descriptive geometry), Moscow, Mashinostroenie, 1995, 224 p. (in Russian).

7. **Chetverukhin N. F.** *Proektivnaya geometriya* (Proektive geometry), Moscow, Prosveshchenie, 1969, 368 p. (in Russian).

8. **Nesterenko L. A.** *Komp'yuternaya grafika v nachertatel'noi geometrii* (Computer graphics in descriptive geometry), Penza, Izd-vo Penzenskogo gos. tekhnolog. un-ta, 2013, 151 p. (in Russian).

9. **Heyfets A. L.** Reorganizatsiya kursa nachertatelnoy geometrii kak aktualnaya zadacha razvitiya kafedr grafiki (Reorganization of the course of descriptive geometry as an actual task of the development of the departments of graphics), *Geometriya i grafika*, 2013, vol. 1, no. 2 (2), pp. 21—23 (in Russian).

10. **Seregin V. I., Ivanov G. S., Borovikov I. F., Senchenkova L. S.** Geometricheskie preobrazovaniya v nachertatelnoy geometrii i inzhenernoy grafike (Geometric transformations in descriptive geometry and engineering graphics), *Geometriya i grafika*, 2015, vol. 3, no. 2, pp. 23—28 (in Russian).

11. **Artsihovskaya-Kuznetsova L. V.** O golovolomnosti v nachertatelnoy geometrii (About puzzling in descriptive geometry), *Geometriya i grafika*, 2014, vol. 2, no. 3, pp. 31—35 (in Russian).

12. **Suflyaeva N. E.** Sovremennyie aspektyi prepodavaniya graficheskih distsiplin v tehnicheskih vuzah (Modern aspects of teaching graphic disciplines in technical universities), *Geometriya i grafika*, 2014, vol. 2, no. 4, pp. 28—33 (in Russian).

13. **Salkov N. A.** Parametricheskaya geometriya v geometricheskom modelirovanii (Parametric geometry in geometric modeling), *Geometriya i grafika*, 2014, vol. 2, no. 3, pp. 7—13 (in Russian).

14. **Voloshinov D. V., Solomonov K. N.** Konstruktivnoe geometricheskoe modelirovanie kak perspektiva prepodavaniya graficheskih distsiplin (Constructive geometric modeling as the perspective of teaching graphical disciplines), *Geometriya i grafika*, 2013, vol. 1, no. 2 (2), pp. 10—13 (in Russian).

15. **Girsh A. G.** Mnimosti v geometrii (Imaginary in geometry), *Geometriya i grafika*, 2014, vol. 2, no. 2, pp. 3—8 (in Russian).

16. **Korotkiy V. A., Hmarova L. I.** Nachertatelnaya geometriya na ekrane kompyutera (Descriptive geometry on the computer screen), *Geometriya i grafika*, 2013, vol. 1, no. 1, pp. 33—34 (in Russian).

17. **Boykov A. A.** O krugovyih orbitah planet (O krugovyih orbitah planet), *Geometriya i grafika*, 2013, vol. 1, no. 2 (2), pp. 66—67 (in Russian).

18. **Boykov A. A.** Kompyuternyie sredstva podderzhki uchebnyih kursov graficheskih distsiplin (Computer tools to support training courses in graphic disciplines), *Geometriya i grafika*, 2013, vol. 1, no. 2 (2), pp. 29—30 (in Russian).

19. **Kozlova I. A., Slavin B. M., Harah M. M., Guseva T. V.** Postroenie liniy peresecheniya nekotoryih slozhnyih poverhnostey (Construction of the intersection line of some complex surfaces), *Geometriya i grafika*, 2015, vol. 3, no. 2, pp. 38—45 (in Russian).

20. **Salkov N. A.** Svoystva tsiklid Dyupena i ih primenenie. Chast 1 (Properties of Dupin cyclides and their application. Part 1), *Geometriya i grafika*, 2015, vol. 3, no. 1, pp. 16—25 (in Russian).

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *И. Е. Назарова*

Сдано в набор 07.04.2017 г. Подписано в печать 23.05.2017 г. Формат 60×88 1/8. Заказ PI617
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**