ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П. Г. Демидова

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

Том 23 № 2(62) 2016

Основан в 1999 году
Выходит 6 раз в год

Адрес редакции: ЯрГУ, ул. Советская, 14, г. Ярославль, 150000, Россия
Website: http://mais-journal.ru, e-mail: mais@uniyar.ac.ru; телефон (4852) 79-77-73

Научные статьи в журнал принимаются по электронной почте. Статьи должны содержать УДК, аннотации на русском и английском языках и сопровождаться набором текста в редакторе LaTeX. Плата с аспирантов за публикацию рукописей не взимается.

12+

СОДЕРЖАНИЕ

Моделирование и анализ информационных систем. Т. 23, №2. 2016

Построение хранилища данных с динамической структурой
Артамонов Ю. Н. 93

Метод сбалансированного выбора механизмов обеспечения отказоустойчивости для распределённых вычислительных систем
Волканов Д. Ю. 119

Криптосистема на индуцированных групповых кодах
Деундяк В. М., Косолапов Ю. В. 137

Об эффективности минимизирующего подхода к оптимизации запросов
Мендкович Н. А. 153

О группе Брауэра арифметической модели многообразия над глобальным полем положительной характеристики
Прохорова Т. В. 164

Построение CFC-программ ПЛК по LTL-спецификации
Рябухин Д. А., Кузьмин Е. В., Соколов В. А. 173

Asymptotic formula for the moments of Bernoulli convolutions
Timofeev E. A. 185

Коллективные потоковые вычисления: реляционные модели и алгоритмы
Усталов Д. А. 195

Задача маршрутизации, осложненная зависимостью функций стоимости и "текущих" ограничений от списка заданий
Ченцов А. Г., Ченцов А. А. 211

Свидетельство о регистрации СМИ ПИ №ФС77-49724 от 11.05.2012 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Учредитель – Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Ярославский государственный университет им. П. Г. Демидова". Подписной индекс – 31907 в Объединенном каталоге "Пресса России". Редактор, корректор А.А. Аладьева. Редактор перевода Э.И. Соколова. Подписано в печать 17.04.2016. Дата выхода в свет 28.04.2016. Формат 60x84$^1/_8$. Усл. печ. л. 16,27. Уч.-изд. л. 14,3. Объем 139 с. Тираж 50 экз. Свободная цена. Заказ 016/016. Адрес типографии: ул. Советская, 14, оф. 109, г. Ярославль, 150000 Россия. Адрес издателя: Ярославский государственный университет им. П. Г. Демидова, ул. Советская, 14, г. Ярославль, 150000 Россия.

ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

P.G. Demidov Yaroslavl State University

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

Volume 23 No 2(62) 2016

Founded in 1999
6 issues per year

Editorial Office Address: P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia
Website: http://mais-journal.ru, e-mail: mais@uniyar.ac.ru

Contents

Modeling and Analysis of Information Systems. Vol. 23, No 2. 2016

Building a data store with the dynamic structure
Artamonov Yu. N. 93

Method for choosing a balanced set of fault tolerance techniques for distributed computer systems
Volkanov D. Yu. 119

Cryptosystem based on induced group codes
Deundyak V. M., Kosolapov Y. V. 137

On the effectiveness of the minimization approach to the query optimization
Mendkovich N. 153

On the Brauer group of an arithmetic model of a variety over a global field of positive characteristic
Prokhorova T. V. 164

Construction of CFC-programs by LTL-specification
Ryabukhin D. A., Kuzmin E. V., Sokolov V. A. 173

Asymptotic formula for the moments of Bernoulli convolutions
Timofeev E. A. 185

Dataflow-driven crowdsourcing: relational models and algorithms
Ustalov D. A. 195

Routization problem complicated by the dependence of costs functions and "current" restrictions from the tasks list
Chentsov A. G., Chentsov A. A. 211

Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 93–118
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 93–118

DOI: 10.18255/1818-1015-2016-2-93-118

УДК 004.652.6

Построение хранилища данных с динамической структурой

Артамонов Ю. Н.

получена 24 февраля 2016

Аннотация. В данной работе проведен анализ подходов к построению хранилищ данных на основе реляционных и NoSQL решений, указаны ограничения реляционного подхода для интеллектуального анализа данных. Выявлено противоречие между представлением данных в реальной предметной области и моделями представления данных в реляционном и NoSQL подходах. Выявленное противоречие связано с темпоральностью не только значений отдельных атрибутов данных, но и изменчивостью состава этих атрибутов, а также структуры связей между ними. Предложена новая логическая модель хранилища данных с динамической структурой. В основу модели положено понятие объекта как своеобразного контейнера для хранения свойств. Каждое свойство объекта включает в себя имя свойства, а также два типа значений свойства – бессылочное и ссылочное, актуальных на заданный момент времени. Ссылочное значение свойства указывает на объект, имя которого интерпретируется как значение этого свойства на заданный момент времени. Дано формальное описание модели с выделением необходимого функционала по манипулированию объектами и их свойствами (селекторы, предикаты, конструкторы), введены необходимые управляющие конструкции. Дано обоснование предложенной модели, названной OP-model, на основе проведения соответствия с логической ER моделью данных. Доказано, что любая ER модель данных может быть реализована в OP-model. В то же время указаны преимущества OP-model, связанные с возможностью изменения связей между сущностями за счет изменения ссылочных значений на определенный момент времени, отмечены потенциальные возможности по масштабируемости хранилища данных за счет уникальной идентификации каждого объекта.

Ключевые слова: NoSQL, Big Data, ER модель, базы данных, СУБД

Для цитирования: Артамонов Ю. Н., "Построение хранилища данных с динамической структурой", *Моделирование и анализ информационных систем*, **23**:2 (2016), 93–118.

Об авторах:
Артамонов Юрий Николаевич, orcid.org/0002-0007-1896-0951, канд. техн. наук,
Федеральное государственное бюждетное научное учреждение «Госметодцентр»,
ул. Люсиновская, 51, г. Москва, 117997 Россия, e-mail: junaart@mail.ru

Благодарности:
Работа выполнена в рамках государственного задания Министерства образования и науки России по проекту №2.87.2016/НМ

1. Постановка проблемы

В настоящее время разработаны и используются самые разнообразные подходы к построению хранилищ данных [1–4]. Самое широкое распространение получил реляционный подход. Однако в ряде случаев приходится сталкиваться с ограничениями

реляционных баз данных. Особенно остро эти ограничения дают о себе знать при обработке очень больших объемов данных или при очень большом потоке обращений к базе данных. В этом случае приходится решать сложные вопросы масштабирования реляционных БД, а в наиболее проблемных ситуациях вообще отказываться от реляционных решений в пользу NoSQL [5, 9]. В NoSQL за счет отсутствия строгой структуры данных и минимума связей между неделимыми частями данных (кортежами ключ-значение) существенно проще добиться их распределения по разным кластерам (масштабируемость – partition tolerance), а также существенно проще получить доступ к этим данным в любой момент времени (доступность – availability). Для объемов информации, а также интенсивности запросов к ней, характерных для современных поисковых систем, NoSQL решения оказываются основной альтернативой реляционному подходу. Однако, как неоднократно отмечено [6, 7, 10], такой подход не обеспечивает целостности данных (консистентность – consictency), поскольку здесь мы попадаем под действие CAP теоремы [6–8]: одновременно можно выполнить только любые два из трех требований (partition tolerance, availability, consictency). В условиях этой теоремы подход NoSQL в большинстве случаев – это AP-системы (Cassandra, CouchDB) [13], реляционный подход – CA-системы. Остается также альтернатива строить CP-системы – такие системы беззатратно масштабируются, а в случае потери согласованности данных становятся недоступными до момента их согласования (MongoDB, HBase, Redis).

Кроме отмеченных ограничений, следует также указать на противоречие между требованиями к доступности в любой момент времени и запросами на интеллектуальную обработку этих данных (Data Mining). Основные приоритеты AP, CA-систем – это высокая производительность (минимальное время отклика) с отсутствием избыточности данных для CA-систем (данные нормализованы) или минимумом этой избыточности для AP-систем (избыточность лишь для обеспечения доступности). В то же время приоритетными требованиями интеллектуального анализа данных являются: гибкость представления и обработки данных, оперативное получение агрегированных данных, накопление данных за продолжительный интервал времени с возможностью их одновременного анализа, допустимость избыточных данных (хранение как детализированных, так и агрегированных данных), высокая пиковая нагрузка при использовании сложных алгоритмов анализа, допустимость среднего времени отклика. Действительно пользователь системы Data Mining вынужден мириться с приемлемыми затратами времени на получение необходимого результата, но останется не удовлетворен ограниченными возможностями анализа или ограниченного доступа лишь к сегменту данных. Таким образом, системы хранения данных для их интеллектуальной обработки следует отнести к CP-системам: масштабирование необходимо для хранения и обработки больших данных (Big Data), консистентность – для получения непротиворечивых результатов анализа. Однако на текущий момент достаточно отработанными следует признать лишь технологии интеллектуального анализа данных для структурированных данных (реляционный подход), выделяя следующие классы систем: OLTP-системы (структурированная информация в рамках реляционной модели); OLAP-системы (надстройка над OLTP [1], расширяющая возможности агрегирования данных построением витрин данных, многомерных кубов данных), Data-Mining-системы (как правило, некоторая надстройка над OLAP, реализующая алгоритмы кластеризации, классификации

и т.д.). Показательным примером в данном случае являются продукты компании Pentaho: СУБД (MySQL, PostGreSQL, MS SQL и др.) – Mondrian (OLAP-система) – Weka (Data-Mining-система) [20, 21]. В то же время предпринимаются попытки распространения OLAP-технологий и для NoSQL систем. Основным ограничением в этих направлениях, по мнению автора, становится оторванность структуры представления данных в реляционном подходе от реальной модели данных, используемой в предметной области, а вслед за ней и в программных приложениях (Impedance Mismatch) [23,24,26], причем данное ограничение наследуется OLAP технологиями. Как следствие этого явления, в OLAP приходится работать с разряженными кубами, сами кубы носят временный характер (не живут долго), формируются под каждую задачу, имеются сложности с сохранением агрегированных результатов обратно в базу данных и их повторного использования. Отметим, что для NoSQL решений также можно увидеть несогласованность модели представления данных в предметной области (где данные вступают во всевозможные отношения) с упрощениями модели ключ-значение, колонка и т.д., которые, однако, отчасти могут компенсироваться гибкостью и динамичностью создаваемой структуры.

Проведем детальный анализ потери соответствия между моделью представления данных в предметной области и моделями в реляционном и NoSQL подходах. Вначале выделим основные свойства данных в реальной предметной области, а затем рассмотрим, как эти свойства отображаются в реляционном и NoSQL подходах. Априори на достаточно абстрактном уровне можно выделить следующие наиболее важные в рамках нашего анализа свойства данных в большинстве предметных областей: структурированность данных (данные образуют различные структуры в соответствии с теми отношениями, в которые они вступают), темпоральность данных (данные меняются со временем), темпоральность структуры данных (меняются не только значения, но и связи между данными). Рассмотрим эти свойства на примере данных, представленных в таблице 1. Даны множества: A – множество различных федеральных округов; B – множество различных регионов; C – множество различных городов; D – множество различных типов организаций; E – множество различных вузов. Между элементами этих множеств введем бинарное отношение R – «включает». Например, имеем: $R(\text{ФО}_1, \text{Регион}_1), R(\text{Бюджетное}, \text{ВУЗ}_1)$ и т.д. Ясно, что любое отношение между элементами указанных множеств можно представить в виде отношения «включает».

Таблица 1. Объединение данных в таблицу
Table 1. Uniting data into the table

Федеральный округ Federal District	**Регион Region**	**Город City**	**Тип организации Type of organization**	**Вуз University**
ФО1	Регион1	Город1	Бюджетное	Вуз1
ФО1	Регион1	Город1	Внебюджетное	Вуз2
ФО2	Регион2	Город2	Внебюджетное	Вуз3
ФО3	Регион3	Город3	Бюджетное	Вуз4

Поскольку отношение «включает» транзитивно, порождаются цепочки таких отношений, образующие древовидные структуры:

$$\text{Федеральный\_округ} \to \text{Регион} \to \text{Город} \to \text{Вуз}; \text{Тип\_организации} \to \text{Вуз}.$$

Каждый элемент такой цепочки в большинстве случаев можно рассматривать как некоторый объект в соответствующей предметной области. Иногда вместо цепочек предпочитают иметь дело со свойствами объектов. Например, говорят: объект «Вуз» имеет свойство с названием «Тип организации», свойство с названием «Город». Таким образом, любое свойство объекта потенциально может выступать самостоятельным объектом в древовидной структуре, полученной из различных значений этого свойства, при условии, что значение свойства измеряется не в количественных шкалах (номинальных, порядковых). При построении структуры базы данных приходится иметь дело с теми же цепочками отношений «включает». Пример подобной структуры представлен на рисунке 1.

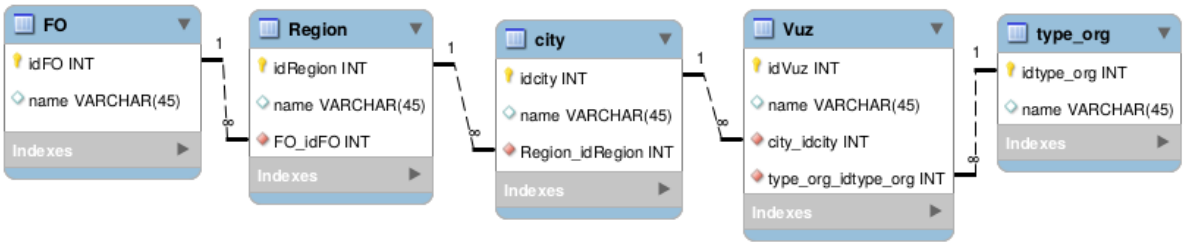

Рис. 1. Представление сущностей в реляционной базе данных
Fig 1. Presentation of the entities in a relational database

Как видно из рисунка 1, возникающие связи таблиц воспроизводят указанные выше цепочки отношений. Однако в реляционной БД после этого структура оказывается зафиксированной. В то же время, задачи, возникающие в предметных областях, приводят к постоянному дрейфу этой структуры: появляется необходимость добавления новых свойств у объектов (новых полей в соответствующие таблицы БД); выделения новых объектов (для реляционного подхода – это добавление новых таблиц БД) из значений свойств существующих объектов, которые уже получили необходимую степень уточнения. Например, в таблице «Вуз» может быть введено поле «Эффективность Вуза», которое по результатам мониторинга заполняется некоторыми значениями. После этого может появиться необходимость исследовать поле «Эффективность Вуза» как самостоятельную сущность, это потребует добавления новой таблицы в БД и соответствующей перестройки структуры. Любое действие со структурой БД усугубляется необходимостью переработки программных приложений анализа данных из БД, поскольку доработки БД автоматически не приводят к изменению функционирования программных приложений по работе с ней. Приведенные примеры требуют непротиворечивого наращивания структуры БД в согласовании с функционированием обслуживающих БД программных приложений. Однако на этом проблемы не заканчиваются. В ряде случаев в предметных областях приходится иметь дело с перестройкой структуры, т.е. изменением отношения «входит». Для реляционной базы данных такой подход означает изменение

инфологической структуры, что невозможно без существенных затрат времени на переработку как со стороны базы данных, так и со стороны обслуживающих ее программных приложений.

В NoSQL подходе следуют принципу хранить данные, как они приходят, перенося полностью или частично задачу формирования структуры данных на этап их обработки. Здесь традиционно выделяют хранилища по типу ключ-значение (key-value store), семейство колонок (bigtable store), а также документо-ориентированные БД (document store).

Системы хранения по типу ключ-значение (Redis, Reak, MemcacheDB) ориентированы в основном на производительность, доступ к данным осуществляется по ключу, часть данных, как правило, хранится в оперативной памяти [11]. Системы хранения по типу семейство колонок (Cassandra, HBase и др.) фактически индексируют строки и столбцы входной таблицы, структура фиксируется в виде совокупности столбцов, добавление новых столбцов приводит к разряженным таблицам. Анализ хранящейся информации, а также обеспечение ее целостности возлагается в основном на программные приложения, сама система реализует лишь базовые функции доступа. Поэтому говорить о соответствии модели данных в предметной области с моделью их представления в хранилище следовало бы именно для программных приложений (Hadoop, различные SQL-подобные решения под Hadoop: Hive, Impala, Shark).

Документо-ориентированные БД обычно хранят коллекции документов в некотором сериализованном формате (например, формат документов в MongoDB – это BSON), который позволяет гибко подстраивать структуру хранения документов под предметную область, изменять эту структуру в случае необходимости [12]. Однако за счет хранения в общем случае некоторой древовидной структуры осложняется доступ к необходимой информации, опять фиксируется структура данных. И хотя изменение структуры данных не становится в этом случае столь критичным для хранилища и программных приложений по работе с ним, как в реляционном подходе, все же требуется соответствующая настройка.

Следует отметить, что на потерю соответствия между моделью представления данных в предметной области и моделью в реляционном подходе указывает ряд авторов [14, 23, 24, 26]. В частности для описания предметной области в [14] рассматривается дискретная детерминированная модель (OD-модель) и отмечается, что традиционное описание такой модели осуществляется в терминах объектов и их поведения. Все это приводит авторов [14] к идее использования для моделирования нового объектно-ориентированного подхода при построении баз данных, учитывающего темпоральную природу как объектов, так и связей между ними. В цикле работ [14–19] последовательно развиваются идеи динамической информационной модели DIM, в основу которой положено представление предметной области в виде множеств объектов, свойств-атрибутов этих объектов, свойств связей объектов с другими объектами, причем на множестве объектов вводятся четыре базовых отношения: наследования, включения, истории и взаимодействия. В работах [18,19] показана как статическая полнота DIM, означающая возможность описания всех действий с фиксированной структурой OD-модели, так и динамическая полнота DIM, означающая возможность представления в DIM всех изменений во времени OD-модели. Для корректного использования DIM модели требуется приведение ее

к нормальной форме, в которой исключаются возможные противоречия, связанные с одновременным использованием на объектах отношений включения и наследования. Для исключения подобных противоречий авторы [14] вынуждены вводить ограничения определенности, однозначности, выбора, выделяя из всего множества DIM моделей DIM модели в нормальной форме (для которых соблюдаются все указанные ограничения). Отметим, что в самом описании OD-модели используются лишь понятия объекта, свойства объекта и взаимодействия объектов по времени. Понятие класса используется как один из возможных подходов (ООСУБД) к корректному представлению OD-моделей. Причем, если в реляционном подходе можно обеспечить целостность данных за счет спецификации вида связей, то в общем случае для обеспечения целостности, консистентности данных в ООСУБД возникают сложности. Кроме этого, широкий функционал по взаимодействию объектов, наследованию свойств и методов взаимодействия объектов в ООСУБД становится ограничением в масштабировании такой системы. Действительно, практика реализации NoSQL подходов в направлении масштабирования данных приводит к выводу о целесообразности наиболее простой структуры хранения данных. В то же время для ООСУБД, например, информация о структуре классов, их наследовании требует централизованного хранения. Поэтому возможные технические реализации систем, построенных на основе DIM, при успешном решении вопросов целостности данных целесообразно отнести к классу CA-систем.

На концептуальном уровне описания OD-модели можно придерживаться и другой позиции, считая первичной, наблюдаемой сущностью в предметной области признак (свойство). Именно признаки (свойства) меняются во времени. Объект или класс объектов – это производная сущность, получаемая как выделение часто встречающегося в предметной области подмножества признаков, привязанных к одному моменту времени (наблюдаемых одновременно). Объект становится лишь контейнером для хранения свойств. Каждое свойство характеризуется значением, привязанным к моменту времени. Выделяется специальное значение свойства – «пусто», также привязанное к моменту времени и означающее, что данное свойство отсутствует в этот момент времени. Объект, у которого все свойства на данный момент времени принимают значение «пусто», можно считать несуществующим в этот момент времени. Таким образом, следует говорить не о времени жизни объекта, а о динамике свойств. Как было показано, о свойствах на определенном уровне развития информационной системы часто начинают говорить как об объектах, или, наоборот, детальные характеристики некоторых объектов утрачивают интерес и их сворачивают до одного именованного свойства – имени объекта. В этом дуализме, по мнению автора, и отражается динамика связей объектов друг с другом. Для реализации идеи о наиболее простой структуре хранения данных, функционал которой минимально ограничивается заранее принятым подходом, важно моделировать именно первичные сущности OD-модели. Остальной функционал следует переносить в область надстроек над базовым функционалом.

Таким образом, на текущий момент времени сохраняется актуальность решения следующей проблемы: необходимо построить хранилище данных с динамической расширяемой структурой, которая позволяла бы проводить свое масштабирование.

2. Модель хранилища данных с динамической структурой

2.1. Концептуальное описание модели

Рассмотренные особенности реализации хранилищ данных и их ограничения, а также описанная последовательность представления данных в абстрактной предметной области позволяют предложить некоторый альтернативный подход к организации хранилищ данных, ориентированный на их интеллектуальную обработку. Данный подход сочетает в себе достоинства реляционных баз данных с их возможностью обеспечения консистентности данных в любой момент времени, а также особенности бесструктурного хранения данных, характерных в целом для NoSQL систем.

Вначале дадим концептуальное описание предлагаемого подхода. Для этого введем ряд понятий, а также опишем основные требования к функциональности.

Объект (object) – наиболее общая сущность, имеющая уникальный идентификатор (uuid_object), тип (type_object), а также имя объекта (name_object). В функциональном отношении объект представляет собой контейнер для хранения свойств.

Свойство (property) – совокупность трех атрибутов: имя свойства (name_property), значение свойства (value_property), ссылка на объект (link_object = uuid_object), чьё имя (name_object) является значением свойства. Для значения свойства (value_property) и ссылки на объект (link_object) имеется специальное значение «nil», соответствующее отсутствию значения у данного свойства.

Время (time) – имеется упорядоченное множество, каждый элемент этого множества – некоторый момент времени.

Требования к функциональности и ограничения:

1. У каждого объекта имеется множество пар, состоящих из имени объекта и момента времени, начиная с которого это имя становится актуальным. Время начала существования нового имени эквивалентно завершению существования старого имени.

2. Каждое свойство является слотом связи объекта, которому принадлежит это свойство, с другим объектом, который при необходимости может быть выделен из свойства. При выделении нового объекта B из свойства объекта A, name_property объекта A становится типом объекта B (type_object), а value_property объекта A – именем объекта B, причем в link_object объекта A заносится uuid_object объекта B.

3. Каждое значение свойства value_property – v_i, link_object – l_i относится к некоторому моменту времени t_i (time_value), который помечает эти свойства. В кортеже $\langle v_i, l_i, t_i \rangle$ каждый момент времени t_i соответствует началу актуальности указанных значений свойства v_i, l_i.

4. Свойство, для которого на данный момент времени соответствующее значение равно nil, считается несуществующим у объекта на данный момент времени.

5. Любое свойство имеет уникальный идентификатор uuid_property. У любого объекта может быть только одно одноименное название свойства.

6. Под переименованием имени свойства будем понимать одновременное прекращение существования свойства со старым именем и начало существование нового свойства с новым именем.

Базовый функционал должен быть расширен в соответствии с потребностями интеллектуального анализа данных. Однако здесь мы подробно остановимся именно на базовом функционале предложенной модели.

2.2. Формальное описание модели (базовый функционал)

Проведем формализацию предложенной концептуальной модели.

Определение 1. *Каждый объект представим упорядоченной четверкой:*

$$O = \langle i_o, t_o, P, N \rangle,$$

где i_o – уникальный идентификатор объекта O (uuid_ object); t_o – тип объекта O (type_ object); P – множество свойств объекта O: $P = \{p_i\}$; $\forall k : N = \{\langle n_{ok}, d_{ok} \rangle\}$ – множество кортежей из двух элементов, n_{ok} – имя объекта («name_ object»), d_{ok} – момент времени, который помечает имя объекта, на множестве N введено отношение линейного порядка $\prec$, такое, что

$$\forall k_1, k_2 : d_{ok_1} < d_{ok_2} \Leftrightarrow \langle n_{ok_1}, d_{ok_1} \rangle \prec \langle n_{ok_2}, d_{ok_2} \rangle$$

В дальнейшем, кроме математической нотации, будем предлагать также обозначения, используемые при технической реализации языка по манипулированию объектами и их свойствами. Так объект будем представлять списком:

$$[\text{uuid}, \text{type}, \text{Property}, \text{Name}]$$

Множество имен объекта будем также представлять списком пар:

$$\text{Name} = [[name_1, time_1], \ldots [name_n, time_n]]$$

Определение 2. *Каждое свойство объекта (каждый элемент множества P) представляет собой упорядоченный набор элементов:*

$$\forall i : p_i = \langle i_p, n, V \rangle,$$

где i_p – уникальный идентификатор свойства объекта (uuid_ property); n – название свойства (name_ property);$V = \{\langle v, l, d \rangle\}$ – множество кортежей из значений свойства для разных моментов времени, v – бессылочное значение свойства (value_ property); l – ссылка на объект, имя которого является значением свойства (link_ object), $l \in \{i_o\}$, v, l могут принимать пустое значение nil; d – момент времени (time_ value), который помечает набор $\langle v, l \rangle$; на множестве V введено отношение линейного порядка $\prec$, такое, что

$$\forall k_1, k_2 : d_{k_1} < d_{k_2} \Leftrightarrow \langle v_{k_1}, l_{k_1}, d_{k_1} \rangle \prec \langle v_{k_2}, l_{k_2}, d_{k_2} \rangle$$

Для технической реализации также введем списковую нотацию:

$$\text{Property} = [\text{Property}[0], \ldots, \text{Property}[k]]$$

$$\text{Property}[i] = [\text{uuid}_i, \text{name}_i, \text{Value}_i]$$

$$\text{Value} = [[value_1, link_1, time_1] \ldots [value_m, link_m, time_m]]$$

Пусть j-й объект представлен в виде:

$$O_j = \langle i_{oj}, t_{oj}, P_j, N_j \rangle,$$

где $P_j = \{p_k | \forall k : p_k = \langle i_{pk}, n_k, \{\langle v_n, l_n, d_n \rangle\}_k \rangle\}$, $N_j = \{\langle n_{ok}, d_{ok} \rangle\}$.

Введем функции локализации характеристик объекта, а также функции проверки свойств объекта (*селекторы и предикаты*).

1. Функция получения уникального идентификатора объекта:

   $$U_o(O_j) = i_{oj},$$

   а также обратная функция получения объекта по уникальному идентификатору:

   $$\overline{U}_o(i_{oj}) = O_j,$$

   если по данному идентификатору объект не найден, то $\overline{U}_o(i_{oj}) = nil$, т.е. данная функция является одновременно предикатом проверки наличия заданного объекта.

   В технической реализации используем точечную нотацию: $O.Uuid = O[0]$ – выдает уникальный идентификатор объекта; $Object.Uuid(value)$ – выдает объект по уникальному идентификатору value.

2. Функция получения типа объекта:

   $$T_o(O_j) = t_{oj},$$

   а также обратная функция получения множества объектов заданного типа:

   $$\overline{T}_o(t_{oj}) = \{O_k | \forall k : T_o(O_k) = t_{oj}\},$$

   если объекты данного типа отсутствуют, то $\overline{T}_o(t_{oj}) = nil$, т.е. данная функция является одновременно предикатом проверки наличия объектов заданного типа.

   Техническая реализация: $O.Type = O[1]$ – выдает тип объекта; $Object.Type(name)$ – выдает список объектов заданного типа с именем name.

3. Функция получения множества кортежей имен объекта:

   $$N_o(Q_j) = N_j.$$

   Техническая реализация: $O.Name = O[3]$ – выдает список имен объекта.

4. Функция получения имени объекта, актуального на заданный момент времени t:
$$N_{ot}(O_j, t) = n_o \Leftrightarrow \langle n_o, \max(\{d_{ok} | \forall k : d_{ok} \leq t\})\rangle \in N_j,$$
а также функция получения моментов времени для заданного имени объекта n_o:
$$\overline{N}_{ot}(O_j, n_o) = \{d_{ok} | \forall k : \langle n_o, d_{ok}\rangle \in N_j\}.$$
Техническая реализация: $O.TimeToName(t)$ - выдает имя объекта, актуального на момент времени t; $O.NameToTime(name)$ - выдает список моментов времени для заданного имени.

5. Функция получения множества свойств объекта:
$$P_o(Q_j) = P_j.$$
Техническая реализация: $O.Property = O[2]$ – выдает список свойств объекта.

6. Функция получения множества уникальных идентификаторов свойств объекта:
$$P_{ip}(Q_j) = \{i_{pk} | \forall k : p_k = \langle i_{pk}, n_{pk}, V\rangle \in P_j\},$$
если у объекта нет свойств, то $P_{ip}(Q_j) = nil$, т.е. данная функция одновременно является предикатом проверки наличия у объекта свойств.
Техническая реализация: $O.Property.Uuid$ – выдает список всех уникальных идентификаторов свойств объекта.

7. Функция получения множества имен свойств объекта:
$$P_{np}(Q_j) = \{n_k | \forall k : p_k = \langle i_{pk}, n_k, V\rangle \in P_j\}.$$
Техническая реализация: $O.Property.Names$ - выдает для всех свойств объекта список их имен.

8. Функция получения имени свойства по уникальному идентификатору свойства:
$$N_p(i_p) = n_{pk} \Leftrightarrow \langle i_p, n_{pk}, V\rangle \in P_j.$$
Техническая реализация: $O.Property.Name(uuid)$.

9. Функция получения множества значений свойства по уникальному идентификатору свойства:
$$V(i_p) = V.$$
Техническая реализация: $Property.Value(uuid)$.

10. Функция получения множества уникальных идентификаторов свойств с заданным именем:
$$I_p(n_p) = \{i_{pk} | \forall k : P_k = \langle i_{pk}, n_p, V\rangle\},$$
если свойств с таким именем не найдено, то $I_p(n_p) = nil$, т.е. данная функция одновременно является предикатом наличия свойств с заданным именем.
Техническая реализация: $Property.Uuid(name)$.

11. Функция получения уникального идентификатора объекта по уникальному идентификатору его свойства:

$$I_{op}(i_p) = i_o \Leftrightarrow i_p \in P_{ip}(\overline{U}_o(i_o)).$$

Техническая реализация: $Property.object(uuid)$.

12. Функция получения уникальных идентификаторов свойств объекта, актуальных на заданный момент времени:

$$P_{ot}(O_j, t) = \{i_{pk} | \forall k : \langle i_{pk}, n_k, \{\langle v_n, l_n, \max(\{d_l | \forall l : d_l \leq t\})\rangle |$$
$$v_n \neq \text{nil} \vee l_n \neq \text{nil}\}\rangle \in P_j\}.$$

Техническая реализация: $O.Property.Uuid(t)$.

13. Функция получения множества всех моментов времени для заданного свойства:

$$D_p(i_p) = \{d_l | \forall l : p_k = \langle i_p, n_{pk}, \{\langle v_{pk}, l_{pk}, d_l\rangle\}\rangle\}.$$

Техническая реализация: $Property.Time(uuid)$.

14. Функция получения множества всех бессылочных значений заданного свойства:

$$V_p(i_p) = \{v_l | \forall l : p_k = \langle i_p, n_{pk}, \{\langle v_l, l_{pk}, d_{pk}\rangle\}\rangle\}.$$

Техническая реализация: $Property.Value.Values(uuid)$.

15. Функция получения множества всех ссылочных значений заданного свойства:

$$L_p(i_p) = \{l_l | \forall l : p_k = \langle i_p, n_{pk}, \{\langle v_{pk}, l_l, d_{pk}\rangle\}\rangle\}.$$

Техническая реализация: $Property.Value.Links(uuid)$.

16. Функция получения бессылочного значения свойства на заданный момент времени:

$$V_{pt}(i_p, t) = v \Leftrightarrow p_k = \langle i_p, n_{pk}, \{\langle v, l_{pk}, \max(\{d_l | \forall l : d_l \leq t\})\rangle\}\rangle,$$

если $V_{pt}(i_p, t) = nil$, то это означает, что у данного свойства на текущий момент времени отсутствует бессылочное значение, таким образом, данная функция одновременно является предикатом проверки наличия бессылочного значения у заданного свойства на заданный момент времени.

Техническая реализация: $Property.Value.Values.Time(uuid, t)$.

17. Функция получения ссылочного значения свойства на заданный момент времени:

$$L_{pt}(i_p, t) = l \Leftrightarrow p_k = \langle i_p, n_{pk}, \{\langle v_{pk}, l, \max(\{d_l | \forall l : d_l \leq t\})\rangle\}\rangle,$$

если $L_{pt}(i_p, t) = nil$, то это означает, что у данного свойства на текущий момент времени отсутствует ссылочное значение, таким образом, данная функция одновременно является предикатом проверки наличия ссылочного значения у заданного свойства на заданный момент времени.

Техническая реализация: $Property.Value.Links.Time(uuid, t)$.

Введем функции добавления и изменения объектов (*конструкторы*). Заметим, что изменение объектов приводит к изменению их контекста, которое невозможно без побочных эффектов – операторов присваивания (обозначим такой оператор символом <-).

1. Функция создания нового объекта Q_k:

$$A_o(i_o, t_o, n_o, d_o) = \langle i_o, t_o, \varnothing, \{\langle n_o, d_o\rangle\}\rangle$$

$$Q_j\text{<-}A_o(i_o, t_o, n_o, d_o).$$

Техническая реализация: $Object.Create(uuid, type, name, time)$.

2. Функция добавления нового свойства в заданный объект:

$$A_p(O_j, i_p, n_p) = \langle U_o(Q_j), T_o(Q_j), P_o(Q_j) \cup \{\langle i_p, n_p, \varnothing\rangle\}, N_o(Q_j)\rangle$$

$$Q_j\text{<-}A_p(O_j, i_p, n_p).$$

Техническая реализация: $O.Property.Add(uuid, name)$.

3. Функция изменения имени объекта на определенный момент времени:

$$CN_o(Q_j, n_o, d) = \langle U_o(Q_j), T_o(Q_j), P_o(Q_j), N_o(Q_j) \cup \{\langle n_o, d\rangle\}\rangle$$

$$Q_j\text{<-}CN_o(Q_j, n_o, d)$$

Техническая реализация: $O.Name.Change(name, time)$.

4. Функция изменения значения бессылочного свойства на определенный момент времени:

$$CPV(Q_j, i_p, v_p, d_p) = \langle U_o(Q_j), T_o(Q_j), (P_o(Q_j) \setminus \{\langle i_p, N_p(i_p), V(i_p)\rangle\})$$

$$\cup\{\langle i_p, N_p(i_p), V(i_p) \cup \{\langle v_p, L_{pt}(i_p, d_p), d_p\rangle\}\rangle\}, N_o(Q_j)\rangle$$

$$Q_j\text{<-}CPV(Q_j, i_p, v_p, d_p).$$

Техническая реализация: $Property.Value.Change(uuid, value, time)$.

5. Функция изменения значения ссылочного свойства на определенный момент времени:

$$CPL(Q_j, i_p, l_p, d_p) = \langle U_o(Q_j), T_o(Q_j), (P_o(Q_j) \setminus \{\langle i_p, N_p(i_p), V(i_p)\rangle\})$$

$$\cup\{\langle i_p, N_p(i_p), V(i_p) \cup \{\langle V_{pt}(i_p, d_p), l_p, d_p\rangle\}\rangle\}, N_o(Q_j)\rangle$$

$$Q_j\text{<-}CPL(Q_j, i_p, v_p, d_p).$$

Техническая реализация: $Property.Link.Change(object, uuid, value, time)$.

6. Функция выделения нового объекта Q_k из заданного на определенный момент времени t свойства i_p:

$$E_o(Q_j, i_p, t, d_o) = A_o(i_o, N_p(i_p), V_{pt}(i_p, t), d_o).$$

$$Q_k\text{<-}E_o(Q_j, i_p, t, d_o)$$

Техническая реализация: $Allocation(object, uuid, time1, time2)$.

Заметим, что использование предложенных функций без операторов присваивания не создает побочных эффектов, что соответствует функциональной парадигме программирования. Это может оказаться важным достоинством при построении объектов на лету в интеллектуальном анализе данных. Для манипулирования объектами и свойствами базовый функционал целесообразно расширить рядом полезных общепринятых управляющих конструкций [22], сохраняя при этом функциональный подход. Вместе с введенными селекторами, предикатами и конструкторами весь этот функционал можно рассматривать как язык по манипулированию данными хранилища на нижнем уровне. В дальнейшем можно создавать высокоуровневые абстракции над операторами данного языка, расширяя базовый функционал предложенной модели.

1. Предикат проверки наличия элемента в множестве:

$$a \notin A \Rightarrow mem(a, A) = nil,$$

$$a \in A \Rightarrow mem(a, A) = t$$

Техническая реализация: $mem(x, Set)$.

2. Функция редукции множества:

$$red(L) = \{l_i | \forall l_i \in L, l_i \neq nil\}$$

Техническая реализация: $red(Set)$.

3. Отображающий функционал:

$$map(f(x_1, x_2, \ldots, x_n), A_1 = \{a_{11}, a_{12}, \ldots, a_{1m}\}, A_2 = \{a_{21}, a_{22}, \ldots, a_{2m}\}, \ldots$$
$$A_n = \{a_{n1}, a_{n2}, \ldots, a_{nm}\}) = \{f(a_{11}, a_{21}, \ldots, a_{n1}), f(a_{12}, a_{22}, \ldots, a_{n2}), \ldots,$$
$$f(a_{1m}, a_{2m}, \ldots, a_{nm})\}$$

Техническая реализация: $map([x1, .., xn], f(x1, ..., xn), A1, ..., An)$ – указываем имена переменных, используемых в отображении, отображаемую функцию, а также множества, откуда переменные принимают значения.

4. Блочный оператор:

$$block(command_1; command_2; \ldots; command_n),$$

где $command_1, command_2, \ldots, command_n$ – блок команд (в дальнейшем $block$), выполняемых последовательно. Оператор возвращает результат последней команды $command_n$.

Техническая реализация: $block(command1; ...; command_n)$.

5. Оператор локального контекста:

$$let(command_1; command_2; \ldots; command_n),$$

аналогично блочному оператору – перечень команд, выполняемых последовательно с возможным присваиванием переменным локальных значений, видимых только внутри $let()$. Оператор возвращает результат последней команды $command_n$.

Техническая реализация: $let(command1; ...; command_n)$.

6. Условный оператор:

$$condition \neq nil \Rightarrow if(condition, block_1, block_2) = block_1$$

$$condition = nil \Rightarrow if(condition, block_1, block_2) = block_2$$

Техническая реализация: $if(condition, \{block1\}, \{block2\})$.

Продемонстрируем использование введенных управляющих конструкций.

1. Проверка наличия объекта заданного типа «type» с заданным именем «name» на данный момент времени «d»:

$$Test_o(type, name, d) = mem(name, map(N_{ot}(x, d), \overline{T}_o(type))).$$

2. Получение уникального идентификатора свойства объекта Q по имени его свойства «name»:

$$Get_{name}(Q, name) = red(map(if(mem(name, \{N_p(x)\}), x, nil), P_{ip}(Q))).$$

3. Выделение объектов из некоторого свойства для всех объектов, которые имеют это свойство, также, кроме создания объектов, обновляется ссылочное значение свойства объектов, от которых были созданы новые объекты (предполагаем, что заданное свойство имеет название «name», также осуществляется проверка - если объект такого типа с заданным на данный момент времени «d» именем уже существует, то он не создается):

$$G(name, d) = map(if(Test_o(name, V_{pt}(Get_{name}(x, name), d), d), nil,$$
$$x\text{<-}CPL(x, Get_{name}(x, name), U_o(E_o(x, Get_{name}(x, name), d, d)), d)).$$
$$map(\overline{U}_o(x), (map(I_{op}(x), I_p(name)))))$$

Прокомментируем составляющие функции $G(name, d)$:

$I_p(name)$ – получаем множество уникальных идентификаторов свойств с заданным именем;

$map(I_{op}(x), I_p(name))$ – получаем множество уникальных идентификаторов объектов, у которых имеется свойство с именем «name»;

$A = map(\overline{U}_o(x), (map(I_{op}(x), I_p(name))))$ – получаем множество объектов, у которых имеется свойство с именем «name»;

$V_{pt}(Get_{name}(x, name), d)$ – получаем значение свойства с именем «name» у объекта $x \in A$ на заданный момент времени d;

$Test_o(name, V_{pt}(Get_{name}(x, name), d), d)$ – проверяем, есть ли объекты типа «name» с именем, соответствующим значению свойства «name» у объекта $x \in A$ на заданный момент времени d, если это условие не выполнено, то выполняется блок кода по созданию нового объекта и модификации ссылочного значения у объекта x:

$$x\text{<-}CPL(x, Get_{name}(x, name), U_o(E_o(x, Get_{name}(x, name), d, d)), d),$$

где $E_o(x, Get_{name}(x, name), d, d)$ – выделяем новый объект из свойства с именем «name» объекта $x \in A$ на момент времени d;

$x\text{<-}CPL(x, Get_{name}(x, name), U_o(E_o(x, Get_{name}(x, name), d, d)), d)$ – создаем новый контекст у объекта x, в котором модифицируем ссылочное значение свойства с именем «name» объекта $x \in A$ на uuid созданного с помощью $E_o(x, Get_{name}(x, name), d, d)$ объекта.

Для удобства в дальнейшем будем ссылаться на описанную модель хранилища данных как на OP-model (object-property-model).

3. Обоснование некоторых свойств модели

Для обоснования свойств предложенной модели покажем соответствие и имеющиеся различия OP-model и логической ER-модели данных [2, 3, 25].

Лемма 1. *Любая сущность ER-модели данных может быть реализована в OP-model.*

Доказательство. В данном случае возможны два варианта, представленные на рисунке 2:

1. Первичный ключ сущности (Entity_1) является простым. Подобный пример представлен на рисунке 2 (а). Такая сущность соответствует множеству объектов в OP-model. Каждый экземпляр сущности отображается в соответствующий объект OP-model. Все объекты этого множества имеют одинаковый тип – имя сущности. Атрибут (Attribute_1) – первичный ключ задает имя объекта, все остальные атрибуты задают свойства объекта.

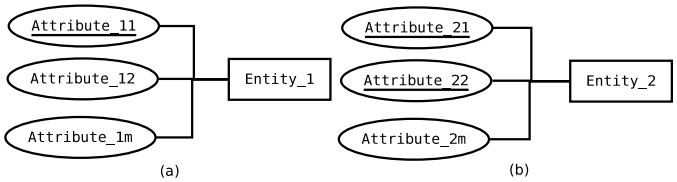

Рис. 2. Одна сущность
Fig.2 One entity

Обозначим:

$Entity_j.Name$ – имя j-й сущности (Entity_j, j=1,2), соответствует типу объекта;

$Attribute_{jk}.Name$ – имя k-го атрибута j-й сущности;

$Attribute_{jk}.Value$ – значение k-го атрибута j-й сущности;

d_1 – момент времени начала существования сущности и ее свойств;

m – количество атрибутов сущности, а также количество свойств объектов соответствующего типа;

n – количество экземпляров сущности, а также количество объектов соответствующего типа;

$uuid_{oi}$ – уникальный идентификатор i-го объекта;

$uuid_{pik}$ – уникальный идентификатор k-го свойства i-го объекта.

Тогда множество объектов OP-model строится следующим образом:

$$\{\langle uuid_{oi}, Entity_1.Name, P_i, \langle Attribute_{11}.Value, d_1\rangle\rangle_{i=1..n}\}$$

$$\forall i : P_i = \{\langle uuid_{pik}, Attribute_{1k}.Name, \{\langle Attribute_{1k}.Value, nil, d_1\rangle\}\rangle_{k=1..m,k\neq 1}\}$$

2. Имеется одна сущность (Entity_2), первичный ключ которой является составным – например, состоит из атрибутов Attribut_21,Attribute_22 (рисунок 2 (b)). В этом случае отображение строится аналогично предыдущему случаю, за исключением имени объекта. Имя объекта можно построить соединением частей составного ключа, кроме этого, каждая часть составного первичного ключа добавляется также как отдельное свойство объекта.

$$\{\langle uuid_{oi}, Entity_2.Name, P_i, \langle Attribute_{21}.Value : Attribute_{22}.Value, d_1\rangle\rangle_{i=1..n}\}$$

$$\forall i : P_i = \{\langle uuid_{pik}, Attribute_{2k}.Name, \{\langle Attribute_{2k}.Value, nil, d_1\rangle\}\rangle_{k=1..m}\}$$

Заметим, что если рассматривать статическую OP модель (значение d_1 не меняется у имени объектов и их свойств), то построенное отображение экземпляров сущностей ER модели в объекты OP модели является биективным, т.е. возможно взаимно однозначное отображение объектов OP модели в сущности и экземпляры сущностей ER модели. □

Лемма 2. *Рекурсивное отношение ER-модели данных может быть реализовано в OP-model.*

Доказательство. Возможные виды рекурсивных отношений в ER-модели показаны на рисунке 3.

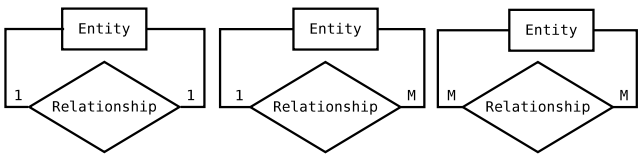

Рис. 3. Рекурсивные связи
Fig 3. Recursive links

Заметим, что в базовом функционале OP-model отсутствует механизм фиксации вида отношения – один к одному, один ко многим, многие ко многим. Данное разграничение вида отношений может быть перенесено на программные приложения по взаимодействию с хранилищем. Поэтому для доказательства достаточно рассмотреть только один вид отношений: многие ко многим, считая остальные виды частными случаями, получаемыми из этого отношения наложением определенных ограничений (как правило, для обеспечения целостности данных).

Для реализации рекурсивного отношения многие ко многим в OP-model в объекты, соответствующие экземплярам сущности, кроме свойств – атрибутов сущности, добавляется свойство с именем типа объектов. Причем ссылочное значение этого свойства указывает на uuid объекта, отвечающего структуре отношения:

$$\{\langle uuid_{oi}, Entity.Name, P_i, \langle Attribute_1.Value, d_1\rangle\rangle_{i=1..n}\}$$

$$\forall i : P_i = \{\langle uuid_{pik}, Attribute_k.Name, \{\langle Attribute_k.Value, nil, d_1\rangle\}\rangle_{k=1..m}\} \cup$$

$$\{\langle uuid_{pi(m+1)}, Entity.Name, \{\langle nil, uuid_o, d_1\rangle | T_o(\overline{U}_o(uuid_o)) = Entity.Name\}\rangle\}.$$

□

Лемма 3. *Связи сущностей ER-модели данных могут отображаться в связи объектов в OP-model.*

Доказательство. На рисунках 4, 5, 6 показаны возможные виды связей между сущностями в ER модели данных. Кроме этого, на рисунках показана модальность связей: двойная линия связи – «должен» (каждый экземпляр сущности имеет связь), одинарная линия связи – «возможно» (некоторые экземпляры сущности имеют связь). Как уже было отмечено, в базовом функционале OP-model не предусмотрено механизмов контроля вида отношений. Также в OP-model не предусмотрено специальных механизмов контроля модальности связей. Однако введение ограничений на создание новых объектов определенного типа, добавления и изменения свойств объектов позволяют реализовать все рассмотренные виды связей ER модели данных (причем реализация этих ограничений может быть вынесена за рамки базового функционала OP-model).

Реализация связей один к одному.

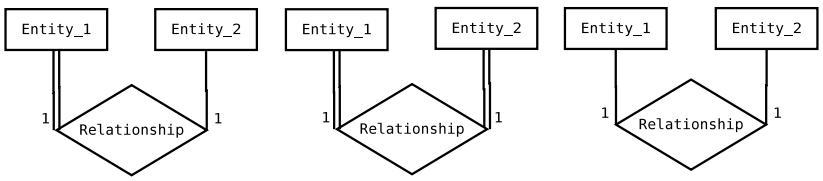

Рис. 4. Виды связей сущности один к одному
Fig 4. Types of links for Entity one to one

1. Пусть экземпляры сущности «Entity_1» при простом первичном ключе «Attribute_11» отображаются в объекты OP-model:

$$\{\langle uuid_{oi}, Entity_1.Name, P_i, \langle Attribute_{11}.Value, d_1\rangle\rangle_{i=1..n}\}$$

$$P_i = \{\langle uuid_{pk}, Attribute_{1k}.Name, \{\langle Attribute_{1k}.Value, nil, d_1\rangle\}\rangle_{k=1..m}\}.$$

Для доказательства примем соглашение, что $F(O_{ji})$ - внешняя функция, задающая наличие связи один к одному для объекта (принимает значение t при наличии связи и nil при ее отсутствии), соответствующего i-му экземпляру j-й сущности. В этом случае объекты, соответствующие экземплярам «Entity_2», можно выделить из значений первичного ключа экземпляров «Entity_1» следующим образом:

- выделить множество A объектов, типа «Entity_1.Name»: A<-$\overline{T}_0(Entity_1.Name)$;
- выделить подмножество B объектов из A, для которых внешняя функция $F(..)$ указывает наличие связи: B<-$red(map(if(F(x), x, nil), A))$;
- для каждого объекта из B создать объект типа «Entity_2.Name» со ссылкой на созданный объект в B:

$$C\text{<-}map(Get_{name}(x, Attribute_{11}.Name), B)$$

$$D\text{<-}map(A_o(gen_{uuid}, Attribute_{11}.Name, V_{pt}(x, d), d), C)$$

$$map(x_1\text{<-}x_2, B, map(CPL(x_1, x_2, x_3, d), B, C, map(U_o(x), D)))$$

Все это удобнее записать, используя введенные обозначения функций при технической реализации.

```
A <- Object.Type(Entity1.Name)
B <- red(map([x], if(F(x), x, nil), A))
C <- map([x],GetName(x, Attribute11.Name), B)
D <- map([x], Object.Create(GenUuid, Attribute11.Name,
                  Property.Value.Values.Time(x,d),d), C)
 map([x,y], block(x <- y), B,
                  map([x1,x2,x3],
                  Property.Link.Change(x1,x2,x3,d), B, C,
                        map([z], z.Uuid,D)))
```

2. Пусть экземпляры сущности «Entity_1» при составном первичном ключе «Attribute_11», «Attribute_12» отображаются в объекты OP-model:

$$\{\langle uuid_{oi}, Entity_1.Name, P_i, \langle Attribute_{11}.Value : Attribute_{12}.Value, d_1\rangle\rangle_{i=1..n}\}$$

$$P_i = \{\langle uuid_{pk}, Attribute_{1k}.Name, \{\langle Attribute_{1k}.Value, nil, d_1\rangle\}\rangle_{k=1..m}\}$$

В этом случае порядок действий может быть аналогичен предыдущему пункту, отличие состоит в том, что объекты выделяются из каждого атрибута составного первичного ключа «Entity_1»:

```
A <- Object.Type(Entity1.Name)
B <- red(map([x], if(F(x), x, nil), A))
C <- map([x],GetName(x, Attribute11.Name), B)
D <- map([x],GetName(x, Attribute12.Name), B)
E <- map([x], Object.Create(GenUuid, Attribute11.Name,
                  Property.Value.Values.Time(x,d),d), C)
F <- map([x],Object.Create(GenUuid, Attribute12.Name,
                  Property.Value.Values.Time(x,d),d), D)
```

```
map([x,y], block(x <- y), B,
                  map([x1,x2,x3],
                  Property.Link.Change(x1,x2,x3,d),
                  B, C, map([z], z.Uuid,E)))
map([x,y], block(x <- y), B,
                  map([x1,x2,x3],
                  Property.Link.Change(x1,x2,x3,d),
                  B, D, map([z], z.Uuid,F)))
```

Модальность связей: «Entity_1» – «должен»; «Entity_2» – «возможно». В дальнейшем разрешается добавлять новые объекты в множество объектов типа «Entity_2», а также запрещается добавлять объекты «Entity_1» без выделения нового объекта из значения первичного ключа и добавления его в множество «Entity_2».

Модальность связей: «Entity_1» – «должен»; «Entity_2» – «должен».

Порядок действий совпадает с предыдущим случаем. Однако после выделения объектов запрещается добавлять новые объекты в множество объектов типа «Entity_2», а также запрещается добавлять объекты «Entity_1» без выделения нового объекта из значения первичного ключа и добавления его в множество «Entity_2».

Модальность связей: «Entity_1» – «возможно»; «Entity_2» – «возможно».

Порядок действий совпадает с предыдущими случаями. После выполнения действий разрешается добавлять новые объекты в множество объектов типа «Entity_2» и «Entity_1».

Реализация связей один ко многим.

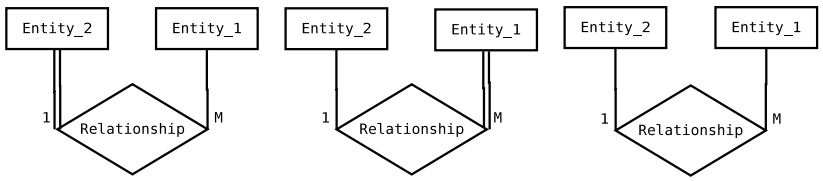

Рис. 5. Виды связей сущности один ко многим
Fig 5. Types of links for Entity one to many

Для доказательства аналогично предыдущему случаю примем соглашение о задании внешней функции $F(O_{ji})$, определяющей наличие связи для объекта (принимает значение t при наличии связи и nil при ее отсутствии). В отличие от связи один к одному в данном случае разные объекты сущности «Entity_1» могут быть связаны с одинаковым объектом сущности «Entity_2». Поэтому объекты сущности «Entity_2» должны создаваться в одном экземпляре.

1. В случае простого первичного ключа такая связь может быть реализована следующим образом (вводится вспомогательная функция $NameUuid$, которая для заданного множества объектов A и заданного имени $name$ на момент времени d формирует подмножество уникальных идентификаторов объектов, которые в данный момент времени имеют имя $name$):

```
A <- Object.Type(Entity1.Name)
B <- red(map([x], if(F(x),x,nil), A))
NameUuid(A,name,d) =
```

```
red(map([x],
        if(mem(name, x.TimeToName(d)),
        x.Uuid, nil), A))
C <- map([x], GetName(x,Attribute11.Name),B)
D <- Object.Type(Attribute11.Name)
map([x],
if(TestO(Attribute11.Name,
     Property.Value.Values.Time(x,d),d),
     map([y],
     block(Object.Uuid(Property.Object(x)) <-
               Property.Link.Change(
               Object.Uuid(Property.Object(x)),x,y,d)),
     NameUuid(D, Property.Value.Values.Time(x,d))),
     block(G <- Object.Create(GenUuid,Attribute11.Name,
               Property.Value.Values.Time(x,d),d),
               Object.Uuid(Property.Object(x)) <-
               Property.Link.Change(
               Object.Uuid(Property.Object(x)),
               x,G.Uuid,d))), C)
```

2. Для реализации отношения один ко многим с составным первичным ключом «Attribute_11.Name» и «Attribute_12.Name» порядок действий полностью совпадает с предыдущим пунктом, но указанная там процедура выполняется отдельно для каждого атрибута «Attribute_11.Name» и «Attribute_12.Name».

Для реализации модальности связей необходимо ввести следующие ограничения.

Модальность связей: «Entity_1» – «возможно»; «Entity_2» – «должен».

Разрешается добавлять новые объекты в множество объектов типа «Entity_1», а также запрещается непосредственно добавлять объекты типа «Entity_2». Объекты типа «Entity_2» можно добавлять лишь путем их выделения из объектов типа «Entity_1».

Модальность связей: «Entity_1» – «должен»; «Entity_2» – «возможно».

После выделения объектов типа «Entity_2» из свойства объектов «Entity_1» разрешается добавление новых объектов типа «Entity_2», запрещается добавление новых объектов типа «Entity_1» без выделения новых объектов типа «Entity_2» из соответствующих свойств.

Модальность связей: «Entity_1» – «возможно»; «Entity_2» – «возможно».

После выделения объектов типа «Entity_2» из свойства объектов «Entity_1» разрешается добавление как новых объектов типа «Entity_1», так и типа «Entity_2».

Реализация связей многие ко многим

Данный вид связи может быть сведен к двухстороннему использованию отношения один ко многим. Для этого в объекты типа «Entity_1» добавляются свойства, соответствующие компонентам составного первичного ключа сущности «Entity_2» (или одно свойство в случае простого первичного ключа). Аналогично в объекты типа «Entity_2» добавляются свойства, соответствующие компонентам составного первичного ключа сущности «Entity_1» (или одно свойство в случае простого первичного ключа). В дальнейшем в ссылочные значения добавленных свойств объектов типа «Entity_1», «Entity_2» в двухстороннем порядке добавляются ссылки

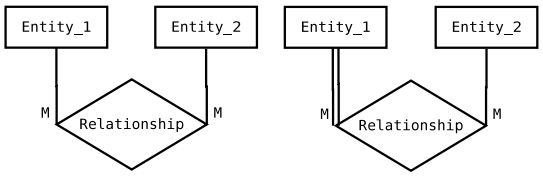

Рис. 6. Виды связей сущности многие ко многим
Fig 6. Types of links for Entity many to many

на объекты типа «Entity_2», «Entity_1» соответственно, согласно реализуемому отношению многие ко многим.

Модальность связей: «Entity_1» – «возможно»; «Entity_2» – «возможно».

Разрешается добавление новых объектов как типа «Entity_1», так и типа «Entity_2».

Модальность связей: «Entity_1» – «необходимо»; «Entity_2» – «возможно».

Разрешается добавление новых объектов типа «Entity_2», однако при добавлении новых объектов типа «Entity_1» требуется добавлять ссылочные значения соответствующих свойств в объекты типа «Entity_2» на добавленный объект типа «Entity_1».

□

Теорема 1. *Любая ER модель данных отображается в OP-model.*

Доказательство. Для доказательства используем принцип математической индукции: предположим, что n сущностей с возможными связями между ними уже реализованы в OP-model. Докажем, что, исходя из этого предположения, добавление еще одной сущности с возможными связями ее с другими сущностями из n уже существующих также реализуемо в OP-model. Возможны следующие варианты:

1. Добавление новой сущности без связей сводится к случаям, рассмотренным в лемме 1;

2. Добавление рекурсивной связи для некоторой сущности сводится к случаям, рассмотренным в лемме 2;

3. Добавление новой сущности A и ее связи с уже существующей сущностью B в отношении один к одному, один ко многим, многие ко многим сводятся к случаям, рассмотренным в лемме 3.

Таким образом, исходя из предположения реализуемости ER модели данных в OP-model для n сущностей, следует ее реализуемость для $n+1$ сущностей, а поскольку реализуемость ER модели данных в OP-model для одной или двух связанных сущностей была доказана в леммах 1, 2, 3, то тем самым доказана реализуемость ER модели данных в OP-model для любого количества сущностей и связей между ними. □

Заметим, что после отображения сущностей и связей этих сущностей в ER модели в объекты и связи между объектами в OP-model сохраняются виды связей по фактическим данным. Например, если таблица 1 связана с таблицей 2 в отношении один ко многим, но в текущем наполнении этих таблиц всем строкам таблицы 1 соответствует только одна строка таблицы 2 (фактическое наполнение пока отвечает отношению один к одному), то информация, что это была связь один ко многим

в OP-model нигде не хранится. Соответственно в дальнейшем контроль за выполнением данного ограничения в OP-model будет отсутствовать. Все это позволяет сделать вывод, что отображение связей ER модели в связи OP-model не является биективным за счет отсутствия в базовом функционале OP-model механизмов фиксации вида связей, модальности связей. По умолчанию в OP-model любая связь имеет минимальные ограничения и может превратиться со временем в связь многие ко многим с модальностью в оба направления «возможно». Аналогичная ситуация наблюдается в NoSQL подходе. Как было отмечено, возможность введения данных ограничений вынесена за рамки базового функционала OP-model и связана с решением вопросов целостности данных. Решение этих вопросов в свою очередь будет зависеть от принятого подхода к хранению данных: централизованное, децентрализованное.

Проследим также на различия между ER моделью и OP-model по возможности учета темпоральности: сущностей; атрибутов сущностей; связей между сущностями. Для этого введем требуемые определения.

Определение 3. *Темпоральность сущностей ER модели данных – это возможность изменения состава сущностей во времени.*

Определение 4. *Темпоральность атрибутов сущности ER модели данных – это возможность изменения состава атрибутов сущности во времени.*

Определение 5. *Темпоральность связей между сущностями ER модели данных – это возможность изменения вида, модальности связей между сущностями, а также возможность возникновения новых связей, исчезновения уже существующих связей во времени.*

Акцентируем внимание на требуемых сопоставлениях между ER моделью и OP-model:

- сущность ER модели (в реляционном подходе – это таблица) соответствует типу объекта в OP-model;

- атрибут сущности ER модели (в реляционном подходе – столбец таблицы) соответствует свойству объекта OP-model;

- экземпляр сущности Entity в ER модели (в реляционном подходе – строка таблицы) соответствует объекту OP-model типа Entity;

- связь между сущностями Entity 1, Entity 2 по атрибуту первичный ключ Attr1 ER модели (в реляционном подходе – связь между таблицами в отношении 1:1, 1:M, M:M) соответствует связи объектов типа Entity 1 с объектами типа Entity 2 по свойству с именем Attr1 у объектов типа Entity2.

Теорема 2. *В рамках сопоставления ER модели и OP-model в предложенной OP-model поддерживается темпоральность сущностей, атрибутов сущности ER модели данных, темпоральность связей между сущностями ER модели данных.*

Доказательство. Поддержка темпоральности на уровне сущностей ER модели данных связана с возможностью добавления новых таблиц в базу данных без связей их с уже существующими таблицами. Такая возможность может быть реализована в рамках реляционного подхода. Однако возникает рассогласование между SQL-запросами к базе данных и программными приложениями, обслуживающими ее

(Impedance Mismatch). Программные приложения приходится перерабатывать. В OP-model темпоральность на уровне сущностей ER модели поддерживается возможностью добавления объектов нового типа.

Поддержка темпоральности на уровне атрибутов сущности ER модели данных связана с возможностью создания таблиц с меняющимся во времени составом столбцов или поддержания для каждой строки таблицы своего перечня столбцов. Такая возможность очевидно отсутствует в рамках реляционного подхода. В OP-model такая возможность поддерживается:

- для исключения необходимого атрибута на заданный момент времени достаточно записать в ссылочное и бессылочное значения заданного свойства «пусто» (nil);

- для добавления необходимого атрибута на заданный момент времени d_p необходимо выполнить функцию добавления нового свойства у выбранного объекта $A_p(Q_j, i_p, n_p)$, а также добавить бессылочное значение этого свойства с помощью $CPV(Q_j, i_p, v_p, d_p)$;

- поскольку в OP-model строки таблицы хранятся отдельно в виде объектов, существует возможность поддержки для каждого объекта своего состава атрибутов-свойств.

Поддержка темпоральности на уровне связей между сущностями ER модели данных означает возможность менять тип и модальность связей между таблицами базы данных во времени, а также добавлять или исключать эти связи. Для реляционного подхода изменение типа или модальности связи приводит к нарушению целостности данных. Если такие изменения выполняются, то они приводят к необходимости существенной переработки программных приложений по работе с базой данных. Кроме того, такие изменения являются необратимыми, т.е. информация, что раньше эта связь имела другой тип или модальность, утрачивается. Добавление или исключение связей, как правило, также приводят к нарушению целостности данных и утрачиванию информации о структуре базы данных до момента изменения.

В рамках предложенной OP-model поддержка темпоральности на уровне связей между сущностями ER модели данных осуществляется за счет отсутствия ограничения на поддержания такой целостности данных в базовом функционале OP-model (как было показано, любая связь со временем может стать связью многие ко многим с модальностью «возможно» с обоих концов связи), причем за счет фиксации моментов времени в модели данных существует возможность сохранения истории о виде и модальности связи фактических данных в любой момент времени. □

Как было отмечено, реализация ER модели данных в рамках реляционного подхода приводит к проблемам масштабирования при кластерном, облачном хранении данных. В OP-model каждый объект имеет свой глобальный уникальный идентификатор (uuid). Это позволяет хранить объекты одного типа на разных компьютерах и повышает масштабируемость хранилища данных. Также следует отметить преднамеренную избыточность: значение каждого свойства может храниться в бессылочном и ссылочном виде. В соответствии с принятым протоколом можно реализовать разные схемы согласования этих значений, например, 1) всегда пытаться получать ссылочное значение, а при недоступности узла выдавать бессылочное значение, при

каждом успешном получении ссылочного значения синхронизировать бессылочное значение; 2) всегда выдавать бессылочное значение и с определенным регламентом проводить синхронизацию этого значения со ссылочным. Все это позволит повысить доступность системы.

4. Направления расширения базового функционала OP-model

Базовый функционал OP-model целесообразно расширить в направлении интеллектуального анализа данных. Для этого введем следующее понятие. *Макет* – совокупность свойств объектов, используемых для иерархического древовидного представления данных. Такое представление целесообразно строить на лету. Фактически макет реализует динамическое построение интересующих структур данных, существующих на указанный момент времени, его использование наряду с возможностью древовидной визуализации данных позволяет проводить сводную аналитику.

Для реализации макета на вход подается структура макета в виде последовательности свойств объектов в выбранном множестве. На примере таблицы 1 формируется последовательность: (<перечень свойств вуза, ссылочное/бессылочное свойство вуза – город>,<перечень свойств города, ссылочное/бессылочное свойство города – регион>, <перечень свойств региона, ссылочное/бессылочное свойство региона – федеральный округ>, <перечень свойств федерального округа>). В указанной последовательности каждый последний элемент каждого кортежа является свойством, по которому формируется структура. В этом случае итоговую древовидную структуру формируют путем последовательного уточнения – сериализации (например, в формате JSON): получают перечень всех вузов, отсортированных по значению последнего свойства – город, сериализуют объекты по последнему свойству (раскладывают вузы по городам), получают перечень свойств всех городов, отсортированных по значению последнего свойства – регион, сериализуют города по регионам и т.д. В итоге получается свернутая древовидная структура с сохранением в ней uuid соответствующих объектов и uuid их свойств. Полученная структура может быть передана внешним программным приложениям для дальнейшего анализа. Основные направления анализа здесь отчасти совпадают с задачами OLAP технологий – получение сводной аналитики, однако в отличие от OLAP технологий появляется возможность непротиворечивого развития этой структуры, в частности для каждого узла дерева, который представляет либо свойство, либо объект исходного хранилища, можно добавлять новые свойства. При реализации взаимодействия программных приложений анализа данных с сервером исходного хранилища по известным uuid объектов и их свойств новые свойства могут быть непротиворечиво добавлены в хранилище: если объект уже существовал, то новое свойство просто добавляется, если новое свойство добавлено в древовидную структуру для исходного uuid свойства, то в хранилище из этого свойства выделяется новый объект с добавлением к нему нового свойства.

В заключение отметим, что предложенную модель хранилища данных за счет ее свойства унификации структуры, учитывающей динамику данных и их связей,

можно развивать в разных направлениях, используя как традиционный реляционный подход, так и объектно-ориентированный подход, или NoSQL решения.

Список литературы / References

[1] Барсегян А. А., *Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP*, БХВ-Петербург, СПб, 2007; [Barsegjan A. A., *Tehnologii analiza dannyh: Data Mining, Visual Mining, Text Mining, OLAP*, BHV-Peterburg, SPb, 2007, 384 pp., (in Russian).]

[2] Дейт К.Дж., *Введение в системы баз данных*, Вильямс, М., 2001, 1072 с.; [Dejt K.Dzh., *Vvedenie v sistemy baz dannyh*, Viljams, M., 2001, (in Russian).]

[3] Мартин Дж., *Организация баз данных в вычислительных системах*, Мир, М., 1980, 665 с.; [Martin J., *Computer data-base organization*, IBM Systems Research Institute, New Jersey, 1977, 665 pp., (in Russian).]

[4] Коннолли Т., *Базы данных: проектирование, реализация и сопровождение: Теория и практика*, Вильямс, М., 2003, 1440 с.; [Konnolli T., *Bazy dannyh: proektirovanie, realizacija i soprovozhdenie: Teorija i praktika*, Viljams, M., 2003, 1440 pp., (in Russian).]

[5] *List Of NoSQL Databases*, http://nosql-database.org/.

[6] Marcos Kawazoe Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg, "Communication-efficient leader election and consensus with limited link synchrony", *The Proceedings of the International Symposium on Principles of Distributed Computing (PODC)*, 2004, 328–337.

[7] Herlihy M. Shavit N., "The topological structure of asynchronous computability", *Journal of the ACM*, **46**:6 (1999), 858–923.

[8] Haifeng Y. Amin V., "The costs and limits of availability for replicated services", *ACM Transactions on Computer Systems*, **24**:1 (2006), 70–113.

[9] Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, Ramana Yerneni, "Pnuts: Yahoo!'s hosted data serving platform", *PVLDB*, **1**:2 (2008), 1277–1288.

[10] Swati Ahirrao Rajesh Ingle, "Scalable transactions in cloud data stores", *Journal of Cloud Computing: Advances, Systems and Applications*, **4**:21 (2015), 1–14.

[11] *In-memory data structure store Redis*, http://redis.io/.

[12] *MongoDB Professional with Cloud Manager*, https://www.mongodb.org/.

[13] *A Database for the Web CouchDB*, http://couchdb.apache.org/.

[14] Писаренко Д.С., Рублев В.С., "Объектная СУБД Динамическая информационная модель DIM и ее основные концепции", *Моделирование и анализ информационных систем*, **16**:1 (2009), 62–91; [Pisarenko D.S., Rublev V.S., "Object DBMS DIM and its main concepts", *Modeling and Analysis of Information Systems*, **16**:1 (2009), 62–91, (in Russian).]

[15] Рублев В.С., "Язык объектных запросов динамической информационной модели DIM", *Моделирование и анализ информационных систем*, **17**:3 (2010), 144–161; [Rublev V.S., "The object query language of the dynamic information model DIM", *Modeling and Analysis of Information Systems*, **17**:3 (2010), 144–161, (in Russian).]

[16] Рублев В.С., "Отношение истории и динамика схем баз данных СУБД DIM", *Моделирование и анализ информационных систем*, **19**:2 (2012), 97–108; [Roublev V.S., "Evolution of DBMS DIM Database Schemes", *Modeling and Analysis of Information Systems*, **19**:2 (2012), 97–108, (in Russian).]

[17] Антонов Д.В., Рублев В.С., "Эффективность доступа к данным в СУБД DIM", *Моделирование и анализ информационных систем*, **22**:2 (2015), 158–175; [Antonov D.V., Roublev V.S., "Access Efficiency to Data in DIM DBMS", *Modeling and Analysis of Information Systems*, **22**:2 (2015), 158–175, (in Russian).]

[18] Петров А.Н., Рублев В.С., "Полнота динамики значений свойств данных в СУБД DIM", *Моделирование и анализ информационных систем*, **22**:2 (2015), 259–277. [Petrov A.N., Roublev V.S., "Completeness of the Dynamics of the Attributes Values of Data in the Database DIM", *Modeling and Analysis of Information Systems*, **22**:2 (2015), 259–277, (in Russian)].

[19] Roublev V.S., "Static completeness of the dynamic information model", *Automatic control and computer sciences*, **49**:3 (2015), 167–176.

[20] *A Comprehensive Data Integration and Business Analytics Platform*, http://www.pentaho.com/.

[21] *Data Mining Software in Java*, http://www.cs.waikato.ac.nz/ml/weka/.

[22] Doug H., *Let Over Lambda*, 2010, 384 pp.

[23] Alexandros B., "An Efficient Database Storage Structure for Large Dynamic Objects", *Proceedings, IEEE Data Engineering Conference, Phoenix, Arizona*, 1992, 301–308.

[24] Полтавцев А.А., "Динамические структуры в реляционных базах данных", *Программные продукты и системы*, **2**:110 (2015), 95–97; [Poltavtsev A.A., "Dynamic structures in relation databases", *Software & Systems*, **2**:110 (2015), 95–97, (in Russian).]

[25] Цикритзис Д., Лоховски Ф., *Модели данных*, Финансы и статистика, М., 1985, 168 с.; [Tsikritzis D., Lokhovski F., *Modeli dannykh*, Finansy i statistika, M., 1985, 168 pp., (in Russian).]

[26] Калиниченко Л.А., *Методы и средства интеграции неоднородных баз данных*, Наука, М., 1983, 424 с.; [Kalinichenko L.A., *Metody i sredstva integratsii neodnorodnykh baz dannykh*, Nauka, M., 1983, 424 pp., (in Russian).]

Artamonov Yu. N., "Building a Data Store with the Dynamic Structure", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 93–118.

DOI: 10.18255/1818-1015-2016-2-93-118

Abstract. This article presents the analysis of approaches to data warehouse construction based on relational and NoSQL solutions and lists the limitations of the relational approach to data mining. The contradiction between data presentation in the real subject domain and the model of data presentation in the relational and NoSQL approaches is revealed. The revealed contradiction is related to the temporality of the values of individual data attributes, the variability of the composition of these attributes, and structure of connections between them. A new logical model of the data warehouse with dynamic structure is proposed. The model is based on the concept of the object as a container for properties storage. Each property of the object includes the property name and two property values - without reference and with reference, that are relevant at a given time. The reference property value points to an object whose name is interpreted as the value of the property at a given time. A formal description of the model with allocation of the necessary functionality to manipulate objects and their properties (selectors, predicates, constructors) is given and the necessary control structures are introduced. Substantiation of the proposed model, called an OP-model is given on the basis of compliance with the logical ER data model. It is proved that any ER data model can be implemented in the OP-model. At the same time, the advantages of the OP-model are indicated, they are associated with the possibility of changing connections between entities due to changes in the reference value at a particular time. The potential for scalability of data warehouse due to the unique identification of each object is noted.

Keywords: NoSQL, Big Data, ER model, Databases, DBMS

On the authors:
Artamonov Yuri Nikolaevich, orcid.org/0000-0002-7246-8329, PhD,
Federal state budgetary scientific institution «Gosmetodcentr»,
Lyusinovskaya str., 51, Moscow, 117997 Russia, e-mail: junaart@mail.ru

Acknowledgments: This work was supported by the state task of the Ministry of Education and Science of the Russian under the project №2.87.2016/HM

Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 119–136
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 119–136

DOI: 10.18255/1818-1015-2016-2-119-136

УДК 517.9

Метод сбалансированного выбора механизмов обеспечения отказоустойчивости для распределённых вычислительных систем

Волканов Д. Ю.

получена 31 марта 2016

Аннотация. В статье рассматривается задача сбалансированного выбора набора механизмов обеспечения отказоустойчивости для распределённых вычислительных систем (РВС). В данной задаче требуется выбрать сбалансированный набор вариантов модулей РВС максимальный по надёжности при ограничениях на стоимость на множестве возможных вариантов РВС. В статье приводится описание рассматриваемых механизмов обеспечения отказоустойчивости, из которых происходит выбор, рассматривается математическая модель в рамках которой дана постановка задачи и метод её решения. Данная задача широко рассматривается в литературе. Приводится подробное описание метода выбора сбалансированного набора механизмов обеспечения отказоустойчивости для РВС. Предложенный метод представляет собой эволюционный алгоритм с использованием схемы нечёткой логики. Схема нечёткой логики в процессе работы алгоритма анализирует результаты его работы в каждом поколении и, исходя из этой информации, корректирует параметры эволюционного алгоритма. Метод позволяет получить эффективное решение, что показано в экспериментальном исследовании. Ключевой особенностью предлагаемого подхода является использование адаптивной схемы. Метод реализован в виде программного средства, интегрированного со средой моделирования ДИАНА. Заключение статьи содержит краткое описание будущих исследований.

Ключевые слова: надёжность, отказоустойчивость, вычислительные системы, генетический алгоритм, задача оптимизации надёжности, механизмы обеспечения отказоустойчивости, эволюционный алгоритм

Для цитирования: Волканов Д. Ю., "Метод сбалансированного выбора механизмов обеспечения отказоустойчивости для распределённых вычислительных систем", *Моделирование и анализ информационных систем*, **23**:2 (2016), 119–136.

Об авторах:
Волканов Дмитрий Юрьевич, orcid.org/0000-0001-9940-5822, ассистент,
Московский государственный университет имени М.В. Ломоносова,
119991, ГСП-1, Россия, Москва, Ленинские горы, МГУ имени М.В. Ломоносова,
2-й учебный корпус, факультет ВМК, комната 764,
e-mail: volkanov@lvk.cs.msu.su

Благодарности:
Исследование выполнено при поддержке Министерства образования и науки Российской Федерации, уникальный номер (ID) RFMEFI60714X0070, соглашение о предоставлении субсидии 14.607.21.0070.

Введение

В данной статье рассматривается задача сбалансированного выбора набора механизмов обеспечения отказоустойчивости (МОО) для распределённых вычислительных систем (РВС) в следующей постановке. Пусть нам задана РВС в виде набора модулей и структуры связей между модулями РВС. К каждому модулю может применяться один МОО. Каждый модуль содержит не менее одного аппаратного и программного компонента. Каждый компонент может иметь несколько версий. Количество аппаратных и программных компонентов в модуле зависит от МОО, используемого для модуля. Тем самым возникает несколько вариантов РВС. Требуется выбрать сбалансированный набор вариантов модулей РВС, эффективный по определенным критериям на множестве возможных вариантов РВС.

РВС оценивается по четырём показателям: функциональность, производительность, надёжность и стоимость [1]. При разработке РВС основным оптимизируемым показателем является надежность при фиксированных ограничениях на стоимость. Надёжность РВС можно повысить двумя основными способами [2, 3]:

1. Использование более надёжных отдельных модулей, из которых состоит система;

2. Использование МОО на уровне модулей РВС.

Оба этих подхода приводят к увеличению стоимости и/или уменьшению производительности РВС. Поэтому при проектировании РВС необходимо увеличивать надёжность сбалансированно, то есть набор МОО и версии аппаратных и программных компонентов модулей РВС должны быть выбраны так, чтобы обеспечить максимальную надёжность, при ограничениях на стоимость. Данная задача в литературе называется задачей выбора сбалансированного набора МОО для вычислительных систем с целью повышения их надёжности или более кратко задачей оптимизации надёжности [3].

Надо сразу отметить, что задача, в которой максимизируется надёжность вычислительной системы при ограничении на стоимость системы, рассматривалась разными исследователями начиная с 60-х годов прошлого века. В [4] было показано, что эта задача является NP-трудной. Обзоры работ различных временных периодов приведены в [3, 5–10].

В большинстве рассматриваемых работ в качестве МОО рассматривалось исключительно резервирование и оценивалась надёжность только для аппаратных или программных компонентов модуля РВС. Как было показано в работе [11], для РВС рассмотрение аппаратной и программной составляющих РВС по отдельности не позволяет адекватно описывать характеристики системы. Поэтому для оценки надёжности всей системы необходимо рассматривать надёжность аппаратуры и программ совместно. В статье [12] авторы допускают возможность использования различных МОО. В статье [13] те же авторы предлагают решение поставленной задачи с помощью классического генетического алгоритма. Эти работы также не лишены недостатков. В них в качестве алгоритмов, при помощи которых ищется решение, используются классические схемы генетического алгоритма и алгоритма имитации отжига, выбор которых обусловлен простотой реализации данных алгоритмов, а не возможностью нахождения эффективного решения.

Здесь будет рассмотрена задача обозначенная в [3] как задача P_1. Для увеличения вычислительной эффективности или для того, чтобы избегать преждевременного попадания в локальный экстремум, многие исследователи используют эффективные комбинации эволюционного алгоритма (ЭА) с другими эвристическими алгоритмами, нейронными сетями и методами локального поиска экстремума. Эти комбинации называются гибридными эволюционными алгоритмами и являются одними из наиболее перспективных направлений в разработке оптимизационных методов. Если при кодировании задачи в методе, использующем схему генетического алгоритма, используется кодировка отличная от бинарной, то такой алгоритм называется эволюционным алгоритмом. Для решения задачи в данной статье будет рассмотрен адаптивный гибридный эволюционный алгоритм (АГЭА). Этот алгоритм включает в себя процедуру автоматического изменения параметров и является развитием алгоритма, предложенного в статье [14].

Структура работы следующая. В следующем разделе вводятся необходимые понятия и определения. В разделе 2 описываются механизмы обеспечения отказоустойчивости, рассматриваемые в данной работе. Раздел 3 содержит модель функционирования РВС. В разделе 4 приводится постановка решаемой задачи. Раздел 5 содержит алгоритм решения поставленной задачи. Экспериментальное исследование предлагаемого алгоритма приводится в разделе 6. В заключении кратко перечислены полученные результаты и список перспективных направлений дальнейших исследований.

1. Основные понятия

У каждого модуля РВС есть *точки предоставления услуг* другим модулям. Модуль может иметь иерархическую структуру, то есть он может состоять из других модулей, каждый из которых имеет свои точки предоставления услуг. Один и тот же модуль не может входить в состав двух разных модулей и самого себя. Внешняя среда и человек-оператор РВС могут рассматриваться как специфичные модули РВС. *Состоянием* модуля назовем совокупность значений его элементов памяти (ячеек ОЗУ, ПЗУ, регистров и т.д.), доступных в точке предоставления услуг модуля. Состояния модуля в точках предоставления услуг этого модуля будем называть *внешними*, а остальные *внутренними*. То есть, если модуль состоит из нескольких модулей, то внешние состояния внутренних модулей являются внутренними состояниями для внешнего/объемлющего модуля. *Действием* модуля РВС будем называть смену состояния модуля. *Поведением* модуля РВС будем называть последовательность действий модуля [1].

Услугой, предоставляемой модулем, будем называть поведение модуля, влияющее на состояния, а может быть, и на поведение других модулей РВС. Услуга является *корректной*, если внешние состояния модуля соответствуют функциональной спецификации. Услуга может быть распределённой, то есть складываться из внешних состояний нескольких модулей, но мы для простоты изложения не будем рассматривать этот случай.

Будем говорить об *отказе услуги*, если предоставляемая услуга некорректна. Если в модуле произошёл отказ хотя бы одной из услуг, то будем говорить об *отказе*

модуля или просто *отказе*. *Ошибка* – это отклонение одной из частей внутреннего состояния от ожидаемого. Поскольку услуга – это последовательность состояний, то некоторые ошибки могут не изменять внешних состояний модуля и не приводить к отказу. Причину ошибки будем называть *неисправностью*.

Под *надёжностью* РВС будем понимать способность РВС сохранять доступность услуг, при определенных спецификацией условиях, в течение заданного периода времени. Для количественной оценки надёжности вводят *меру надёжности* как вероятность безотказной работы (то есть отсутствие отказов услуг) в течение интервала $[0, t_{end}]$, при условии, что система находится в работоспособном состоянии в момент времени t=0 [15].

Для обеспечения необходимого значения надёжности применяют МОО, предотвращающие отказ РВС при отказах её модулей. МОО работают на этапе эксплуатации РВС и предотвращают отказ при возникновении ошибки. В следующем разделе рассматриваемые МОО описаны более подробно.

2. Рассматриваемые механизмы обеспечения отказоустойчивости

В этой статье будем считать, что в каждом модуле РВС может использоваться один из следующих МОО: один из двух видов N-версионного программирования (NVP/0/1, NVP/1/1), восстановление блоками (RB/1/1) или не использоваться никакие МОО (None) [16].

Структурная схема N-версионного программирования (NVP) [17, 18] состоит из блока принятия решений (голосователя) и N независимо разработанных версий программы. N обычно является нечётным числом. В NVP все N версий программы для одной и той же задачи запускаются параллелельно, а результаты их работы собираются и обрабатываются голосователем. Тот результат, который получается в большинстве версий, принимается голосователем как финальный. Различные аппаратные неисправности могут вызвать ошибки в программе и недопустимый ответ в результате её работы. В этой статье рассматривается механизм NVP для N=3. NVP/0/1 [13] не позволяет предотвратить отказы, возникающие вследствие неисправностей аппаратуры, но позволяет предотвращать одну неисправность в программных версиях. Архитектура этого МОО включает голосователь и 3 версии программы, запускаемых параллельно на одном аппаратном компоненте. Главное отличие МОО NVP/1/1 [13] от NVP/0/1 заключается в том, что каждая программная версия выполняется на отдельном аппаратном компоненте.

МОО RB/1/1 [13, 17, 18] включает в себя две версии программного компонента. Если результат работы одной признан некорректным, то вычисления "откатывают" в начало и запускают вторую версию. Процесс продолжается, пока результат одной из версий не будет принят, либо после определённого числа попыток результат работы всех версий не будет отклонён.

3. Модель функционирования РВС

3.1. Основные предположения

В данной работе будем предполагать:

1. Все модули неремонтируемы, то есть после любого отказа модуль считается неработоспособным.

2. Моменты появления отказов для аппаратных компонентов модулей РВС являются статистически независимыми.

3. Вся аппаратные компоненты модулей РВС вычислительной системы являются активными.

4. Для всех модулей вычислительной системы количество доступных версий программных и аппаратных компонентов фиксировано и постоянно. Набор этих версий для каждого модуля свой.

5. Для каждого компонента модуля РВС интенсивность отказов постоянна, а, соответственно, распределение наработки на отказ является показательным, что позволяет работать с функциями надёжности компонентов системы как со скалярными величинами.

Эти предположения позволят нам ниже трактовать рассматриваемую задачу как задачу дискретной оптимизации.

3.2. Конфигурация РВС

Введем обозначения:

- n – количество модулей в РВС;
- U_i – модуль вычислительной системы, здесь и далее $i \in [1, n]$;
- H_i – аппаратный компонент модуля U_i;
- S_i – программный компонент модуля U_i;
- p_i – количество версий аппаратного компонента H_i;
- q_i – количество версий программного компонента S_i;
- $H_{i,j}$ – j-я версия аппаратного компонента H_i в модуле U_i, $i \in [1, n]$, $j \in [1, p_i]$
- $S_{i,j}$ – j-я версия аппаратного компонента S_i в модуле U_i, $i \in [1, n]$, $j \in [1, q_i]$
- FT_{avail} – множество доступных МОО для РВС. В данной работе $FT_{avail} = \{None, NVP/0/1, NVP/1/1, RB/1/1\}$
- $FT_i \subseteq FT_{avail}$ – для каждого модуля i определён свой набор доступных МОО;

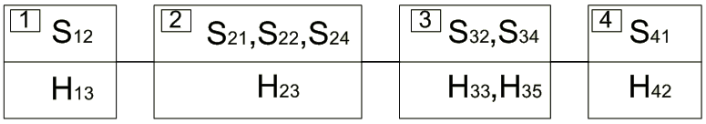

Рис. 1. Пример конфигурации РВС

- $H_i^{F_i}$ – множество версий $H_{i,j}$ аппаратного компонента H_i, фактически используемый в модуле U_i, зависит от типа применяемого МОО;
- $S_i^{F_i}$ – множество версий $S_{i,j}$ программного компонента S_i, фактически используемый в модуле U_i, зависит от типа применяемого МОО;

Конфигурация РВС $System_{RTES} = \{\{H_1^{F_1}, S_1^{F_1}, F_1\}, ..., \{H_i^{F_i}, S_i^{F_i}, F_i\}, ..., \{H_n^{F_n}, S_n^{F_n}, F_n\}\}$, $i \in [1, n]$, включает в себя число и версии используемых программных и аппаратных компонентов и набор МОО. Пример конфигурации РВС приведён на рисунке 1.

Множество $\{H_i^{F_i}, S_i^{F_i}, F_i\}$ является корректным в одном из следующих случаев:

- если $F_i = None$, то $H_i^{F_i} = \{H_{i,k}\}$ $(k \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j}\}$ $(j \in [1, q_i])$;
- если $F_i = NVP/0/1$, то $H_i^{F_i} = \{H_{i,k}\}$, $(k \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j1}, S_{i,j2}, S_{i,j3}\}$, причём $j1, j2, j3 \in [1, q_i], j1 \neq j2 \neq j3$;
- если $F_i = NVP/1/1$, то $H_i^{F_i}$ содержит ровно три элемента $H_i^{F_i} = \{H_{i,k1}, H_{i,k2}, H_{i,k3}\}$ $(k1, k2, k3 \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j1}, S_{i,j2}, S_{i,j3}\}$, причём $j1, j2, j3 \in [1, q_i]$, $j1 \neq j2 \neq j3$;
- если $F_i = RB/1/1$, то $H_i^{F_i} = \{H_{i,k1}, H_{i,k2}\}$ $(k1, k2 \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j1}, S_{i,j2}\}$, причём $j1, j2 \in [1, q_i], j1 \neq j2$.

Конфигурация РВС называется корректной, если $\forall i \in [1, n]$ множество $\{H_i^{F_i}, S_i^{F_i}, F_i\}$ является корректным.

3.3. Оценка надёжности и стоимости РВС

Будем оценивать:

1. Надёжность РВС
$$R_{system} = \prod_{i=1}^{n} R_i,$$
где R_i – надёжность i-го модуля.
2. Надёжность модуля будем рассматривать в соответствии с формулами, предложенными в [13].

3. Стоимость аппаратных компонентов модуля РВС

$$C_i^{hw} = \sum_{j=1}^{p_i} x_{i,j} * C_{i,j}^{hw},$$

где p_i – количество версий аппаратного компонента H_i $\forall i \in [1,n]$; $x_{i,j}$ – количество экземпляров $H_{i,j}$ в модуле i;$C_{i,j}^{hw}$ – стоимость j-й версии аппаратной компоненты H_i в модуле i, $i \in [1,n], j \in [1,p_i]$.

4. Стоимость программных компонентов

$$C_i^{sw} = \sum_{j=1}^{q} y_{i,j} * C_{i,j}^{sw},$$

где q_i – количество версий программного компонента S_i $\forall i \in [1,n]$; $y_{i,j}$ – количество экземпляров $S_{i,j}$ в модуле i;$C_{i,j}^{sw}$ – стоимость j-й версии программной компоненты S_i в модуле i, $i \in [1,n], j \in [1,q_i]$.

5. Стоимость модуля вычислительной системы:

$$C_i = C_i^{hw} + C_i^{sw}.$$

6. Стоимость РВС:

$$C_{system} = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \sum_{j=1}^{p_i} x_{i,j} * C_{i,j}^{hw} + \sum_{i=1}^{n} \sum_{j=1}^{q_i} y_{i,j} * C_{i,j}^{sw}.$$

Все значения стоимости являются натуральными величинами.

4. Задача выбора сбалансированного набора МОО для РВС

Задачу выбора сбалансированного набора МОО для РВС будем рассматривать в следующей постановке:
Дано:

1. n – количество модулей в РВС;

2. p_i,q_i – количество версий программных и аппаратных компонент для каждого модуля i, $i \in [1,n]$;

3. $C_{i,j}^{hw}$, $R_{i,j}^{hw}$ – стоимость и надёжность версий аппаратного компонента $\forall i \in [1,n], j \in [1,p_i]$;

4. $C_{i,j}^{sw}$, $R_{i,j}^{sw}$ – стоимость и надёжность версий программного компонента $\forall i \in [1,n], j \in [1,q_i]$;

5. FT_{avail} – множество доступных МОО для РВС.
   $FT_{avail} = \{None, NVP/0/1, NVP/1/1, RB/1/1\}$

6. $FT_i \subseteq FT_{avail} - \forall i \in [1, n]$ определён свой набор доступных МОО из множества FT_{avail};

7. $P_{v_{i,j}}, P_{rv}, P_d, P_{all}$ – вероятность отказа j-й версии i-го модуля, вероятность одновременного отказа нескольких версий программных компонентов, вероятность отказа схемы принятия решения, вероятность отказа сразу всех версий программного компонента;

8. C^{max}_{system} – максимально допустимая стоимость вычислительной системы.

Требуется определить:
Конфигурацию РВС $System_{RTES}$, такую что:

$$\begin{cases} R_{system_{RTES}} = \max\limits_{system} R_{system} \\ C_{system_{RTES}} \leq C^{max}_{system} \end{cases} \quad (1)$$

В работе [4] было показано, что эта задача является NP-трудной.

5. Эволюционный алгоритм с использованием схемы нечёткой логики

5.1. Общее описание метода

Метод, предлагаемый в данной работе для решения поставленной задачи (1), представляет собой ЭА с использованием схемы нечёткой логики, которая в процессе работы алгоритма анализирует результаты его работы в каждом поколении и, исходя из этой информации, корректирует параметры эволюционного алгоритма.

Предлагаемый метод решения задачи, поставленной в разделе 3, включает в себя следующие шаги:

1. **Кодирование решения.** Каждое решение задачи выбора сбалансированного набора МОО кодируется в виде строки. Строка состоит из блоков, которые соответствуют модулям РВС. Каждый блок (ген) представляет собой тройку вида $<H, S, F>$. H – это номер конфигурации аппаратной составляющей модуля, S – это номер конфигурации программной составляющей, а F – это порядковый номер МОО из множества FT_{avail}, используемого в данном модуле. По каждой строке может быть вычислена целевая функция – надёжность системы R_{system}, которая характеризует качество решения, и однозначно восстановлена соответствующая конфигурация РВС. Количество особей возьмём равным некоторому числу num.

2. **Подготовка популяции решений.** Генерация случайным образом популяции решений $System_k, k \in [1, num]$, для первой итерации алгоритма или популяция с предыдущей итерации запуска алгоритма.

3. **Начальная оценка популяции.** На этом этапе происходит вычисление целевой функции R_{system} и проверка ограничений стоимости $C_{system} \leq C_{system}^{max}$.

4. **Выполнение операции селекции.** В данном алгоритме используется пропорциональная схема селекции [21].

5. **Отбор особей для скрещивания в отдельную промежуточную популяцию.** Популяция сортируется, отбираются лучшие X% (изменяемый параметр) особей, наилучших по значению целевой функции, которые затем участвуют в операции скрещивания.

6. **Выполнение операции скрещивания.** В качестве операции скрещивания используется одноточечное скрещивание [21] модулей РВС. Скрещивание запускается столько раз, сколько особей в первоначальной популяции (num). В результате получается промежуточная популяция, которая состоит из num родителей и num потомков.

7. **Формирование новой популяции.** Промежуточная популяция из *2*num* особей сортируется по возрастанию целевой функции. В новую популяцию берётся доля (изменяемый параметр) лучших особей от исходной популяции, остальная часть популяции формируется из лучших особей промежуточной популяции, полученной после операции скрещивания. Таким образом, количество особей в популяции становится равным первоначальному значению.

8. **Выполнение операции мутации.** Некоторый процент лучших особей популяции (изменяемый параметр) не мутирует. Все остальные особи мутируют с некоторой степенью мутации (изменяемый параметр) и с некоторой вероятностью (изменяемый параметр). Оператор мутации является модификацией одноточечной мутации [21]. Он работает по следующему принципу: случайным образом выбирается ген (тройка *<H, S, F>*), соответствующий модулю системы. Пусть M_i – количество всевозможных вариантов этого гена для i-го модуля РВС. Далее случайным образом выбирается число n в диапазоне от *0* до $M_i - 1$. На это число n должен быть "увеличен" ген. Далее n раз случайным образом выбираем разряд в гене – либо *H*, либо *S*, либо *F* и увеличиваем соответствующий разряд на единицу, если это возможно. Если выпал разряд *H* и его уже нельзя увеличить, увеличиваем разряд *S* на единицу, а в разряде *H* устанавливаем минимальное значение (подобие операции переноса единицы в десятичной системе счисления). Если выбран разряд *S*, и его нельзя увеличить, увеличиваем разряд *F* на единицу, в разрядах *H* и *S* устанавливаем минимальные значения для данного *F*. Если выбран разряд *F* и его нельзя увеличить, все три разряда устанавливаем минимальные возможные значения. При такой операции может быть получен любой возможный ген, при этом некорректных генов получиться не может.

9. **Оценка популяции.** На этом этапе происходит проверка ограничений стоимости $C_{system} \leq C_{system}^{max}$ и вычисление целевой функции R_{system}. Если решение не удовлетворяет ограничениям, то оно штрафуется. Идея использования

штрафных функций заключается в том, чтобы искусственно понизить значение целевой функции (надежности) для тех решений, которые не удовлетворяют заданным ограничениям. Таким образом, решения, не удовлетворяющие ограничениям, будут взяты в следующую популяцию с меньшей вероятностью. Штрафная функция представляет собой коэффициент, на который умножается значение надёжности. Этот коэффициент должен быть равен 1, если решение удовлетворяет ограничениям, и должен принадлежать (0;1] иначе. Кроме того, логично выбирать его таким образом, чтобы большему значению стоимости соответствовало меньшее значение штрафного коэффициента. В соответствии с такими условиями был выбран следующий вид штрафных функций: функция равна 1, если решение удовлетворяет ограничениям, иначе – обратно пропорциональна стоимости.

Для стоимости:

$$E_{cost} = \begin{cases} 1 & \text{,если } C_{system} < C^{max}_{system} \\ \frac{C^{max}_{system}}{C_{system}} & \text{,если } C_{system} \geq C^{max}_{system} \end{cases}$$

$$R^{**}_{system} = R^{*}_{i} * E_{cost}$$

Также на этом шаге фиксируется лучшее, на текущий момент решения, вычисление среднего значения целевой функции.

10. **Проверка критерия останова.** Если он выполнен, то переход к п. 13, если нет, то переход к п. 12.

11. **Блок нечёткой логики.** Блок нечёткой логики осуществляет автоматическую подстройку параметров алгоритма и переход к п. 5. Подробно работа блока описаны в следующем разделе.

12. **Завершение алгоритма.** В качестве результата выбирается наилучшая из найденных конфигураций $System_{RTES}$.

Данный метод реализован в виде программного средства, сопряжённого с новой версией среды моделирования ДИАНА [19, 20].

5.2. Описание блока нечёткой логики

Блок нечёткой логики нужен для того, чтобы в автоматическом режиме корректировать настройки ЭА, управлять степенью влияния операций селекции, скрещивания и мутации на эволюционный процесс согласно некоторым правилам в зависимости от результатов работы алгоритма, которые получаются в каждом поколении. Правила нечёткой логики для АГЭА зависят непосредственно от значений этих параметров и сформулированы в таблице 1. Каждый из перечисленных в таблице параметров имеет три значения (большое, среднее и малое). Конкретное числовое значение определяется экспериментатором, исходя из параметров оптимизируемой системы.

Таблица 1. Правила работы схемы нечёткой логики

	$F_{av_2} < F_{av_1}$	$F_{av_2} \approx F_{av_1}$ (в пределах 3%)	$F_{av_2} > F_{av_1}$
$F_{max_2} < F_{max_1}$	Малый P_{mut} Большой N_{nmut} Малый S_{mut} Большой N_{sel} Большой P_{cross}	Малый P_{mut} Большой N_{nmut} Малый S_{mut} Большой N_{sel} Средний P_{cross}	Средний P_{mut} Средний N_{nmut} Средний S_{mut} Средний N_{sel} Малый P_{cross}
$F_{max_2} \approx F_{max_1}$ (в пределах 3%)	Средний P_{mut} Средний N_{nmut} Малый S_{mut} Средний N_{sel} Большой P_{cross}	Большой P_{mut} Малый N_{nmut} Средний S_{mut} Малый N_{sel} Средний P_{cross}	Большой P_{mut} Малый N_{nmut} Большой S_{mut} Малый N_{sel} Малый P_{cross}
F_{av_1} и F_{av_2} – среднее значение надёжности текущей и предыдущей популяции F_{max_1} и F_{max_2} – лучшее значение надёжности в текущей и предыдущей популяциях P_{mut} – вероятность мутации N_{nmut} – доля лучших особей текущей популяции, которые не мутируют S_{mut} – степень мутации гена N_{sel} – доля от популяции лучших особей, которые затем будут скрещиваться P_{cross} – вероятность скрещивания			

В зависимости от изменений среднего и лучшего значения надёжности популяции в процессе работы АГЭА происходит изменение значений параметров алгоритма в соответствии с таблицей 1.

Такая схема нечёткой логики имеет малую сложность по сравнению с вычислением целевых функций и проверки ограничений для всех особей популяции, но при этом позволяет производить автоматическую перенастройку ЭА в процессе его работы.

Теорема 1. *Предложенный алгоритм АГЭА обладает свойствами корректности и полноты.*

Доказательство. Идея доказательства данного утверждения состоит в рассмотрении всех операций алгоритма. По построению начальная популяция состоит из корректных решений. Во время работы алгоритма решения изменяются при помощи операций скрещивания и мутации. При скрещивании новое решение получается из старого заменой одного из модулей модулем из другого решения. Поскольку новых модулей не создаётся, то полученное решение корректно. При мутации изменяется один из модулей. По построению эта операция корректна. Полнота вытекает из того, что вероятность попадания корректного решения в следующую популяцию никогда не равна нулю. Для того, чтобы перебрать все решения, достаточно последовательно применять операцию мутации с параметром $M_i = 2$. □

Для сравнения предложенного метода и существующих алгоритмов было проведено экспериментальное исследование.

6. Экспериментальное исследование

6.1. Цель экспериментального исследования

В работе [13] был предложен классический ГА (КГА) для решения рассматриваемой задачи (1). Поэтому с ним и будем сравнивать предложенный АГЭА алгоритм. Сформулируем цели экспериментального исследования:

1. Сравнить эффективность работы АГЭА с классическим ГА (КГА), описанным в статье [13], по следующим критериям:
   - точность как отклонение значения целевой функции от оптимального значения;
   - сложность, вычислительные затраты на выполнение алгоритма (характеризуется количеством итераций);
   - стабильность работы алгоритма (стабильность значения целевой функции на полученном решении и вычислительных затрат на нахождение решения).
2. Исследовать зависимость работы АГЭА от числа элементов области решений, удовлетворяющих разным ограничениям на стоимость системы.

6.2. Методика проведения экспериментов

Для каждого запуска алгоритма фиксируются следующие параметры:

- конфигурация системы $System_{RTES}$, её надёжность $R_{system_{RTES}}$ и стоимость $C_{system_{RTES}}$;
- количество итераций алгоритма N;
- время работы алгоритма T_{alg}.

При заданных условиях эксперимента (значениях фиксированных параметров; распределениях параметров, значения которым присваиваются случайным образом), гипотеза H_0 формулируется следующим образом: «с вероятностью p_0 , значение $R_{system_{RTES}}$ находится в интервале $[R_{system_{min}}$; $R_{system_{max}}]$». В данной работе гипотеза H_0 рассматривалась для значения $p_0 = 0.9$. Определим количество запусков алгоритма для оценки его точности, сложности и стабильности при выбранном значении p_0. На основе метода проверки статистической гипотезы о числовом значении вероятности события в различных источниках даются разные оценки о минимальном количестве запусков алгоритма. В данной работе использована наиболее пессимистическая оценка [22], где объём выборки должен быть не менее 223 для $p_0 = 0.9$. Поэтому для исследования каждого алгоритма будем проводить серию из 225 экспериментов.

Для оценки надёжности по серии экспериментов для каждого алгоритма вычислим среднее значение надёжности, полученное по 225 запускам, отбросив крайние значения. Затем сравним его с оптимальным значением, полученным с помощью алгоритма полного перебора, и оценим отклонение от оптимального значения.

Для оценки сложности по серии экспериментов для каждого алгоритма вычислим среднее значение количества итераций, выполненных алгоритмом.

Для оценки стабильности работы алгоритма в каждой серии найдём минимальное и максимальное значения надёжности (за исключением отброшенных крайних значений) полученных конфигураций и определим интервал как разность этих значений.

Исследование зависимости работы алгоритмов от размеров области решений заключается в исследовании точности, сложности и стабильности алгоритмов при разных ограничениях на стоимость вычислительной системы.

6.3. Описание экспериментальной вычислительной системы

Экспериментальная вычислительная система имеет следующие параметры [13, 14]:

- количество модулей в системе: 6;
- количество версий аппаратных компонентов в каждом модуле: 3;
- количество версий программы в каждом модуле: 4;
- значения стоимости и надёжности версий программных и аппаратных компонентов для каждого модуля экспериментальной системы приведены в таблицах 2–5;
- набор доступных МОО: *None, NVP/0/1, NVP/1/1, RB/1/1*;
- ограничения на стоимость вычислительной системы: 180, 320, 460, нет ограничения (∞).

6.4. Результаты экспериментов

Результаты исследования алгоритма по точности и сравнение среднего значения надёжности с оптимальным решением приведены в таблице 6. В таблице 6 представлены средние значения надёжности для каждого алгоритма (кроме алгоритма полного перебора) по 225 запускам с разными ограничениями на стоимость вычислительной системы – 180, 320, 460 и без ограничения. В таблице 7 приведено отклонение среднего значения надёжности от оптимального в процентах. Необходимо отметить, что размер популяции в КГА и АГЭА был одинаковым.

Результаты исследования алгоритма по сложности приведены в таблице 8. В таблице указано среднее количество итераций алгоритмов при разных ограничениях на стоимость системы.

Результаты исследования алгоритмов по стабильности приведены в таблице 9. В таблице указано максимальное (max) и минимальное (min) значения надёжности для каждой серии экспериментов, а также длина доверительного интервала (len) при разных ограничениях на стоимость вычислительно системы.

На основании проведённого экспериментального исследования можно сделать следующие выводы:

Таблица 2. Стоимость аппаратных компонентов (АК) в системе

Модуль	Версия 1	Версия 2	Версия 3
АК 1	30,0	10,0	10,0
АК 2	30,0	20,0	10,0
АК 3	20,0	30,0	10,0
АК 4	30,0	10,0	10,0
АК 5	30,0	20,0	30,0
АК 6	30,0	20,0	20,0

Таблица 3. Надёжность аппаратных компонентов (АК) в системе

Модуль	Версия 1	Версия 2	Версия 3
АК 1	0,995	0,980	0,980
АК 2	0,995	0,995	0,970
АК 3	0,994	0,995	0,992
АК 4	0,990	0,980	0,985
АК 5	0,995	0,980	0,995
АК 6	0,998	0,995	0,994

Таблица 4. Стоимость версий программных компонент (ПК) в системе

Модуль	Версия 1	Версия 2	Версия 3	Версия 4
ПК 1	30,0	10,0	20,0	30,0
ПК 2	30,0	20,0	10,0	20,0
ПК 3	20,0	30,0	20,0	30,0
ПК 4	20,0	10,0	20,0	20,0
ПК 5	30,0	20,0	30,0	30,0
ПК 6	10,0	30,0	20,0	20,0

Таблица 5. Надёжность версий программных компонент (ПК) в системе

Модуль	Версия 1	Версия 2	Версия 3	Версия 4
ПК 1	0,950	0,908	0,908	0,950
ПК 2	0,965	0,908	0,887	0,908
ПК 3	0,978	0,954	0,860	0,954
ПК 4	0,950	0,908	0,910	0,950
ПК 5	0,905	0,967	0,967	0,905
ПК 6	0,908	0,968	0,968	0,955

Таблица 6. Среднее значение надёжности ВС

Алгоритм	Ограничение на стоимость системы			
	180	320	460	∞
КГА	0,611542	0,824016	0,840537	0,821873
АГЭА	0,625433	0,874313	0,922647	0,9252
Полный Перебор	0,659682	0,891419	0,924541	0,925244

Таблица 7. Отклонение решения от оптимального

Алгоритм	Ограничение на стоимость системы			
	180	320	460	∞
КГА	7,30	5,71	8,40	11,17
АГЭА	5,19	1,92	0,02	0,005

Таблица 8. Количество итераций алгоритмов

Алгоритм	Ограничение на стоимость системы			
	180	320	460	∞
КГА	909,2	109,7	23,6	21,3
АГЭА	75,4	76,29	82,95	97,27

Таблица 9. Стабильность алгоритмов

Алгоритм	Ограничение на стоимость системы					
	180			320		
	min	max	len	min	max	len
КГА	0,57314	0,625008	0,051868	0,748954	0,852204	0,10325
АГЭА	0,544715	0,667985	0,123269	0,807257	0,911711	0,104454
	460			∞		
	min	max	len	min	max	len
КГА	0,765034	0,883054	0,11802	0,712925	0,873224	0,160299
АГЭА	0,898389	0,925049	0,0266	0,924766	0,92544	0,000468

- точность алгоритма АГЭА оказалась выше алгоритма КГА, а отклонение АГЭА от оптимального решения получилось не превосходящим 5%;
- на малой области приемлемых решений (удовлетворяющих ограничению на стоимость конфигурации системы) предпочтительным оказывается АГЭА, поскольку он имеет наименьшую сложность;
- на большой области приемлемых решений точность, стабильность и сложность алгоритма АГЭА лучше, чем у алгоритма КГА.

Таким образом, можно сделать вывод, что предложенный в статье метод работает не хуже существующего, а на малой области приемлемых решений лучше существующего.

7. Заключение

В рамках данной работы были получены следующие результаты:

- Дана математическая постановка задачи выбора сбалансированного набора МОО для РВС.
- Разработан новый метод решения задачи выбора сбалансированного набора МОО для РВС. Ключевыми особенностями разработанного метода является учёт ограничений последовательно, что позволяет обобщить данный метод для вычислительных систем с дополнительными ограничениями, а также использование разработанного алгоритма АГЭА для поиска решения при ограничениях только на стоимость системы. Предложенный метод реализован в виде программного средства.
- В рамках проведённого экспериментального исследования определена область эффективного применения АГЭА, а именно большая область приемлемых решений. В остальных случаях предложенный в статье метод работает не хуже предложенного в статье [13].

В рамках дальнейших исследований можно выделить следующие направления:

- Исследовать поддержку в популяции большего количества различных решений. Это ухудшит скорость сходимости эволюционного алгоритма. Зато позволит в случае нахождения серии решений, в которых задачи не удовлетворяют ограничениям на стоимость, быстрее получить решения, в которых задачи удовлетворяют ограничениям на стоимость.
- Исследовать возможность применения других адаптивных схем. Это позволит выбрать наилучшую схему адаптации для АГЭА.
- Оценить применимость методов многокритериальной оптимизации для нахождения компромисса между надёжностью и производительностью РВС. На данный момент все характеристики системы, на которые не влияет использование МОО, считаются фиксированными.

- Оценить применимость предлагаемого метода для решения задачи оптимизации надёжности для распределённых систем реального времени, где помимо ограничений на стоимость появляется ограничение на директивное время работы каждого модуля системы.

Список литературы / References

[1] Avižienis A., Laprie J.-C., Randell B., "Dependability and itsthreats: a taxonomy", *Building the Information Society*, **156** (2004), 91–120.

[2] Лупанов О. Б., "Об одном методе синтеза схем", *Изв. ВУЗов, Радиофизика*, **1**:1 (1958), 120–140; [Lupanov O. B., "Ob odnom metode sinteza schem", *Izv. VUZov,Radiophizika*, **1**:1 (1958), 120–140, (in Russian).]

[3] Kuo W.,Wan R., "Recent Advances in Optimal Reliability Allocation", *Handbook of Military Industrial Engineering by Badiru A., Thomas M.*, 2009, 1–24.

[4] Chern M.S., "On the computational complexity of reliability redundancy allocation in a series system", *Building the Information Society*, **11**:5 (1992), 309–315.

[5] Tillman F.A., Hwang C.L., Kuo W., "Optimization techniques for systems reliability with redundancy – A review", *IEEE Transactions on Reliability*, **R-26**:3 (1977), 148–155.

[6] Misra K.B., "On optimal reliability design: A review", *System Science*, **12** (1986), 5–30.

[7] Kuo W., Prasad V.R., "An annotated overview of system-reliability optimization", *IEEE Transactions on Reliability*, **49**:2 (2000), 176–187.

[8] Gen M., Yun Y.S., "Soft computing approach for reliability optimization: State-of-the-art survey", *Reliability Engineering & System Safety*, **91**:9 (2006), 1008–1026.

[9] Aleti A. et al., "Software architecture optimization methods: A systematic literature review", *IEEE Transactions on Software*, **39**:5 (2013), 658–683.

[10] Soltani R., "Reliability optimization of binary state non-repairable systems: A state of the art survey", *International Journal of Industrial Engineering Computations*, **5**:3 (2014), 339–364.

[11] Смелянский Р. Л., "Модель функционирования распределенных вычислительных систем", *Вестник Московского Университета*, **3** (1990), 3–21; [Smeliansky R. L., "Model funkcionirovaniya raspredelennyh vychislitelnyh sistem", *Vestnik Moskovskogo Universiteta*, **3** (1990), 3–21, (in Russian).]

[12] Wattanapongsakorn N., Levitan S.P., "Reliability optimization models for embedded systems with multiple applications", *IEEE Transactions on Reliability*, **53**:3 (2004), 406–416.

[13] Wattanapongsakorn N., Coit D.W., "Fault-tolerant embedded system design and optimization considering reliability estimation uncertainly", *Reliability Engineering & System Safety*, **92**:4 (2007), 395–407.

[14] Bakhmurov A.G. et al., "Method For Choosing An Effective Set Of Fault Tolerance Mechanisms For Real-Time Embedded Systems, Based On Simulation Modeling", *Problems of dependability and modelling*, 2011, 13–26.

[15] Laprie J.C., Coste A., "Dependability: A Unifying Concept for Reliable Computing", *Proceedings of the 12th Fault Tolerant Computing Symposium*, 1982, 18–21.

[16] Xie Z., Sun H., Saluja K., "Survey of Software Fault Tolerance Techniques", *University of Wisconsin-Madison, Department of Electrical and Computer Engineering*, 2006.

[17] Laprie J.C. et al., "Definition and analysis of hardware and software-fault-tolerant architectures", *IEEE Computer*, **23**:7 (1990), 39–51.

[18] Lyu M.R. (Editor-in-Chief), *Handbook of software reliability engineering*, McGraw-Hill: IEEE Computer Society Press, 1996.

[19] Bakhmurov A.G., Kapitonova A.P., Smeliansky R.L., "DYANA: An Environment for Embedded System Design and Analysis", *Proceedings of 5-th International Conference TACAS'99, Amsterdam, The Netherlands*, **1579**, 1999, 390–404.

[20] Бахмуров А. Г. и др., "Интегрированная среда для анализа и разработки встроенных вычислительных систем реального времени", *Программирование*, **5** (2013), 35–52; [Bakhmurov A. G., "Integrated environment for the analysis and design of distributed real-time embedded computing systems", *Programming and Computer Software*, **39**:5 (2013), 242–254, (in Russian).]

[21] Гладков Л. А., Курейчик В. В., Курейчик В. М., *Генетические алгоритмы*, ФИЗМАТЛИТ, 2006; [Gladkov L. A., Kureychik V. V., Kureychik V. M., *Geneticheskie algoritmy*, PHIZMATLIT, 2006, (in Russian).]

[22] Чистяков В. П., *Курс теории вероятностей*, Наука, 1987; [Chistyakov V. P., *Kurs teorii veroyatnostey*, Nauka, 1987, (in Russian).]

Volkanov D. Yu., "Method for Choosing a Balanced Set of Fault Tolerance Techniques for Distributed Computer Cystems", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 119–136.

DOI: 10.18255/1818-1015-2016-2-119-136

Abstract. In the paper we consider a method for a reliability allocation problem (RAP) of distributed computer systems (DCS) under cost constraints. In this problem we maximize reliability of DCS under constraints of system cost. The article describes considered fault tolerance mechanisms. The mathematical formulation of RAP is provided. RAP is widely discussed in the literature. A detailed description of the method is ensured. The applied method is an evolutionary algorithm with an adaptive logic control procedure. The adaptive logic control procedure analyzes the results of evolutionary algorithm work in each generation and, based on this information, adjusts parameters. The key feature of the proposed method is the use of an adaptive hybrid genetic algorithm. The results of experiments with the implemented method are presented. This method was implemented as a pilot system which works in cooperation with DYANA simulation environment. Finally, future plans for the development of the presented method and tools are briefly described.

Keywords: dependability, fault tolerance techniques, genetic algorithm, computer systems, reliability, reliability allocation problem, reliability-redundancy allocation problem, evolutionary algorithm

On the authors:
Volkanov Dmitry Yurievich, orcid.org/0000-0001-9940-5822, assistant lecturer,
Lomonosov Moscow State University,
2nd Education Building, Faculty CMC, room 764, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation,
e-mail: volkanov@lvk.cs.msu.su

Acknowledgments:
This research is supported by the Ministry of education and science of the Russian Federation, Unique ID RFMEFI60714X0070, agreement No 14.607.21.0070 dated Sep 23, 2014.

Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 137–152
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 137–152

DOI: 10.18255/1818-1015-2016-2-137-152
УДК 517.9

Криптосистема на индуцированных групповых кодах

Деундяк В. М., Косолапов Ю. В.

получена 15 марта 2016

Аннотация. Код C на группе $\mathcal{G}$, индуцированный кодом N на подгруппе $\mathcal{H}$, обладает тем свойством, что для декодирования кода C может применяться декодер кода N. Поэтому, если для N имеется эффективный алгоритм декодирования, то по N с помощью конструкции индуцирования можно построить класс кодов с известными алгоритмами декодирования. Эта особенность используется в настоящей работе для построения кодовой криптосистемы с открытым ключом типа Мак-Элиса на индуцированных групповых кодах. Для этой криптосистемы описаны операции шифрования и расшифрования, приводится анализ стойкости к атаке на ключ, а также выделены слабые ключи, в случае использования которых взлом криптосистемы типа Мак-Элиса на индуцированном коде C сводится к взлому этой криптосистемы на коде N. Показано, что практически стойкая криптосистема на индуцированном коде C может быть построена на коде N малой длины. На основе предложенной криптосистемы разработан протокол выработки общего ключа по открытому каналу.

Ключевые слова: групповые коды, индуцированные групповые коды, криптосистема Мак-Элиса

Для цитирования: Деундяк В. М., Косолапов Ю. В., "Криптосистема на индуцированных групповых кодах", *Моделирование и анализ информационных систем*, **23**:2 (2016), 137–152.

Об авторах:
Деундяк Владимир Михайлович, orcid.org/0000-0001-8258-2419, канд. физ.-мат. наук, доцент,
ФГНУ НИИ "Спецвузавтоматика", пер. Газетный, 51, г. Ростов-на-Дону, 344002, Россия,
Южный Федеральный Университет, ул. Большая Садовая, 105/42, г. Ростов-на-Дону, 344006, Россия,
e-mail: vl.deundyak@gmail.com,
Косолапов Юрий Владимирович, orcid.org/0000-0002-1491-524X, канд. техн. наук,
Южный Федеральный Университет, ул. Большая Садовая, 105/42, г. Ростов-на-Дону, 344006, Россия,
e-mail: itaim@mail.ru

Введение

В [1] Р. Мак-Элисом была предложена криптосистема с открытым ключом, основанная на применении помехоустойчивых кодов Гоппы. В дальнейшем идея применения помехоустойчивых кодов в асимметричных криптосистемах получила развитие. Именно, были построены такие наиболее известные в настоящее время кодовые криптосистемы, как система Нидеррайтера на основе кодов Рида–Соломона [2], система Габидулина–Парамонова–Третьякова (ГПТ) на основе ранговых кодов [3], система Сидельникова на основе кодов Рида–Маллера [4]. Ниже такие криптосистемы

будем называть криптосистемами *типа* Мак-Элиса, так как правила построения открытого и секретного ключа в таких системах во многом схожи с соответствующими правилами из работы [1].

Отметим, что к настоящему моменту эффективные атаки на ключ классической криптосистемы Мак-Элиса не известны. В то же время для других перечисленных выше криптосистем такие атаки имеются. В частности, для криптосистемы Нидеррайтера, основанной на расширенных кодах Рида–Соломона, в [5] Сидельниковым В.М. и Шестаковым С.О. построен полиномиальный по времени алгоритм нахождения подходящего секретного ключа; в [6] и [7] построены редукции криптоаналитического алгоритма Сидельникова–Шестакова для классических кодов Рида–Соломона. В [8] показано, что классическая криптосистема ГПТ на ранговых кодах небольших параметров также не является стойкой к атакам на ключ; в [9] построены атаки на ключ для некоторых модификаций системы ГПТ. Для криптосистемы Сидельникова алгоритм успешной атаки на ключ построен в [10], а в [12] этот алгоритм существенно ускорен.

В настоящей работе с целью усиления стойкости криптосистемы к атакам на ключ строится новая кодовая асимметричная криптосистема на основе индуцированных групповыми кодами. Индуцированные групповые коды представляют интерес по той причине, что их декодирование может быть выполнено с помощью декодера для кода, на основе которого строятся индуцированные коды. В работе рассматривается задача усиления стойкости криптосистемы за счет использования индуцированных кодов.

Статья имеет следующую структуру. В первом разделе приводится определение групповых кодов и рассматривается конструкция индуцированных кодов; для последующих криптографических приложений уточняются операции кодирования и декодирования групповых и индуцированных кодов; формулируется ряд утверждений, необходимых при анализе стойкости криптосистемы типа Мак-Элиса на индуцированных кодах. В разделе 2 на индуцированных кодах конструируется криптосистема типа Мак-Элиса и проводится анализ ее криптостойкости. В третьем разделе на основе построенной криптосистемы строится протокол выработки общего секретного ключа.

1. Индуцированные групповые коды

1.1. Групповые коды

Пусть $\mathbb{F}$ — поле Галуа, $V = \mathbb{F}^m$ и (V, ρ) — метрическое пространство с метрикой Хэмминга ρ и стандартным базисом B. Для вектора $\mathbf{x}(\in V)$ множество базисных векторов, коэффициенты при которых в разложении $\mathbf{x} = \sum_{\mathbf{b}\in B} x_{\mathbf{b}}\mathbf{b}$ ненулевые, называется носителем вектора x относительно базиса B и обозначается $\mathrm{supp}_B(\mathbf{x})$. Тогда $w(\mathbf{x}) = |\mathrm{supp}_B(\mathbf{x})|$ – вес вектора $\mathbf{x}$. Носителем множества D ($D \subseteq V$) называется множество $\mathrm{supp}_B(D) := \cup_{\mathbf{x}\in D}\mathrm{supp}_B(\mathbf{x})$.

Всякое линейное подпространство C метрического пространства (V, ρ) называется линейным кодом. Размерность, длину (мощность множества $\mathrm{supp}(C)$) и минимальное расстояние (относительно метрики ρ) кода C будем обозначать соответственно $k(C)$, $n(C)$ и $d(C)$, а код C с такими параметрами будем называть

$[n(C), k(C), d(C)]$-кодом [13]. Двойственный код к коду C обозначим $C^{\perp}$. Если $G(C)$ – порождающая матрица кода C, то кодирование удобно представить в виде отображения: $\mathrm{Enc}_C : \mathbb{F}^{k(C)} \to \mathbb{F}^{n(C)}$, причем $\mathrm{Enc}_C(\mathbf{s}) = \mathbf{s}G(C)$ для любого информационного вектора $\mathbf{s} \in \mathbb{F}^k(C)$. Будем полагать, что для C имеется эффективный алгоритм декодирования Dec_C который позволяет исправить до $t := \lfloor (d(C) - 1)/2 \rfloor$ ошибок и реализует отображение $\mathrm{Dec}_C : \mathbb{F}^{n(C)} \to \mathbb{F}^{k(C)}$.

Пусть $\mathcal{G}$ — конечная группа, $\hat{1}(\in \mathcal{G})$ – нейтральный элемент. Рассмотрим групповую алгебру $\mathbb{F}\mathcal{G}$, элементами которой являются формальные суммы (функции)

$$\sum_{g \in \mathcal{G}} a_g g, a_g \in \mathbb{F}. \tag{1}$$

Группа $\mathcal{G}$ действует слева на $\mathbb{F}\mathcal{G}$ по правилу:

$$\mathcal{G} \times \mathbb{F}\mathcal{G} \ni (g, \phi = \sum_{h \in \mathcal{G}} \phi_h h) \mapsto \phi g^{-1} := \sum_{h \in \mathcal{G}} \phi_{hg^{-1}} h \in \mathbb{F}\mathcal{G}. \tag{2}$$

В $\mathbb{F}\mathcal{G}(\cong \mathbb{F}^{|\mathcal{G}|})$ зафиксируем базис $B = B^{\mathcal{G}} := \{\delta_g = 1g\}_{g \in \mathcal{G}}$, что позволяет рассматривать в этой алгебре метрику Хэмминга d_B. Групповым $\mathbb{F}\mathcal{G}$–кодом, следуя [14], будем называть левый идеал алгебры $\mathbb{F}\mathcal{G}$. Отметим, что длина группового кода равна $|\mathcal{G}|$.

Пусть C — групповой $[n(C), k(C), d(C)]$-код, $C \subseteq \mathbb{F}\mathcal{G}$, $n(C) = |\mathcal{G}|$, $B(C) := \{\epsilon_1; ...; \epsilon_{k(C)}\}$ – базис кода C, где для $i = 1, ..., k(C)$ имеет место представление:

$$\epsilon_i = \sum_{g \in \mathcal{G}} \alpha_{i,g} \delta_g, \quad \alpha_{i,g} \in \mathbb{F}.$$

Тогда при кодировании произвольному информационному вектору $\mathbf{s} = (s_1, ..., s_{k(C)})$ ставится в соответствие кодовый вектор $\mathbf{c} = \mathrm{Enc}_C(\mathbf{s})$ вида:

$$\mathbf{c} = \sum_{i=1}^{k(C)} s_i \epsilon_i = \sum_{i=1}^{k(C)} s_i \sum_{g \in \mathcal{G}} \alpha_{i,g} \delta_g = \sum_{g \in \mathcal{G}} (\sum_{i=1}^{k(C)} s_i \alpha_{i,g}) \delta_g.$$

Для определения порождающей матрицы $G(C)$ группового $\mathbb{F}\mathcal{G}$–кода C зададим линейный порядок $\{g_1 = \hat{1}; ...; g_{|\mathcal{G}|}\}$ на группе $\mathcal{G}$, который естественным образом однозначно определяет линейный порядок на базисе B. Зафиксированный на группе порядок будем обозначать $\mathrm{ord}(\mathcal{G})$. Порядок $\mathrm{ord}(\mathcal{G})$ определяет отображение $\nu_{\mathcal{G}} : \mathbb{F}\mathcal{G} \to \mathbb{F}^{|\mathcal{G}|}$, поэтому матрица $G(C)$ имеет вид:

$$G(C) = \begin{pmatrix} \nu_{\mathcal{G}}(\epsilon_1) \\ ... \\ \nu_{\mathcal{G}}(\epsilon_{k(C)}) \end{pmatrix}.$$

Другими словами, порождающая матрица $G(C)$ – это матричное представление базиса $B(C)$ через стандартный базис B алгебры $\mathbb{F}\mathcal{G}$.

В работе [15] разработаны алгоритмы для мажоритарного декодирования групповых кодов. В основе разработанных алгоритмов лежит мажоритарный декодер Дж. Мэсси [16], использующий декодирующие деревья [14], а представление кодов

в виде идеалов групповых алгебр позволяет эффективно строить декодирующие деревья. Для кода C декодирующим деревом $\mathrm{WB}_{\mathbf{b},r,L_{\mathbf{b}}} = \mathrm{WB}_{\mathbf{b},r,L_{\mathbf{b}}}[C]$ будем называть помеченное дерево с корнем $\mathbf{b}$, обладающее следующими свойствами:

• множество вершин этого дерева состоит из $L_{\mathbf{b}}+1$ уровня; корень $\mathbf{b}(\in B)$ находится на уровне 0, а листья – на уровне $L_{\mathbf{b}}$;

• каждая вершина, не являющаяся листом, имеет не менее $r(\in \mathbb{N})$ непосредственно следующих за ней вершин;

• листья дерева помечены элементами из $C^{\perp}$;

• метки $\mathbf{v}^{(1)},...,\mathbf{v}^{(r)}$ вершин, непосредственно следующих из произвольной вершины $\mathbf{v}$, находящейся на уровне $i(0 \le i < L_{\mathbf{b}})$, образуют в совокупности множество, M-ортогональное $\mathbf{v}$: $\mathrm{supp}_B(\mathbf{v}^{(i)}) \cap \mathrm{supp}_B(\mathbf{v}^{(j)}) = \mathrm{supp}_B(\mathbf{v})$ для всех $i \ne j$.

Далее полагается, что

$$r_{\mathbf{b}} := \max\{r \in \mathbb{N} : \exists \ \ \mathrm{WB}_{\mathbf{b},r,L_{\mathbf{b}}}[C] \text{ для некоторого натурального } L_{\mathbf{b}}\}. \tag{3}$$

Пусть $\mathcal{W}[C] = \{\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]\}_{\mathbf{b}\in B}$ — набор декодирующих деревьев кода C. Величина

$$\mathrm{dmaj}_B(C) := \min_{\mathbf{b}\in B}\{r_{\mathbf{b}}\} \tag{4}$$

называется MLD-расстоянием кода C [14], с. 53. Отметим, что если для $[n(C), k(C), d(C)]$-кода C выполняется равенство $d(C) = \mathrm{dmaj}_B(C)+1$, то он называется MLD-кодом [14]. Обозначим символом $\mathrm{MajDecoder}_C$ декодер, описанный в алгоритме Decoder3 работы [15], который по принятому из канала слову $\mathbf{v} = \mathbf{c}+\mathbf{e}$, соответствующему информационному вектору $\mathbf{s}(\in \mathbb{F}^{k(C)})$, и набору декодирующих деревьев $\mathcal{W}[C]$ возвращает вектор $\mathbf{s}'(\in \mathbb{F}^{k(C)})$, при этом $\mathbf{s}' = \mathbf{s}$, если $w(\mathbf{e}) \le \lfloor \mathrm{dmaj}_B(C)/2 \rfloor$.

1.2. Конструкция индуцированных групповых кодов

Пусть $\mathcal{H}$ — подгруппа группы $\mathcal{G}$, $|\mathcal{H}| = u$. Определенный выше порядок $\mathrm{ord}(\mathcal{G})$ на группе $\mathcal{G}$ индуцирует естественным образом линейный порядок $\mathrm{ord}(\mathcal{H}) = \{h_1 = \hat{1}; ....; h_u\}$ на подгруппе $\mathcal{H}$ и отображение $\nu_{\mathcal{H}} : \mathbb{F}\mathcal{H} \to \mathbb{F}^{|\mathcal{H}|}$. Группа $\mathcal{G}$ разбивается на множество непересекающихся правых классов смежности $\{\mathcal{H}y\}_{y\in Y}$ по $\mathcal{H}$, где $Y(\subset \mathcal{G})$ — правая трансверсаль $\mathcal{G}$ по $\mathcal{H}$. Отметим, что порядок $\mathrm{ord}(\mathcal{G})$ естественным образом индуцирует порядок $\mathrm{ord}(Y) = \{y_1; y_2; ...; y_\lambda\}$ на выбранной трансверсали Y, где $\lambda = [\mathcal{G} : \mathcal{H}]$. Так как $\mathbb{F}\mathcal{G}$ является свободным левым $\mathbb{F}\mathcal{H}$-модулем с базисом Y, т.е.

$$\mathbb{F}\mathcal{G} = \bigoplus_{i=1}^{\lambda} (\mathbb{F}\mathcal{H})\delta_{y_i}, \tag{5}$$

то базис $B^{\mathcal{G}}$ представим в виде: $B^{\mathcal{G}} = \cup_{i=1}^{\lambda} B^{\mathcal{H}}\delta_{y_i}$, где $B^{\mathcal{H}} = \{\delta_h\}_{h\in\mathcal{H}}$ — базис в линейном векторном пространстве $\mathbb{F}\mathcal{H}$. Индуцированный порядок на подгруппе $\mathcal{H}$ однозначно определяет линейный порядок на базисе $B^{\mathcal{H}}$.

Рассмотрим $\mathbb{F}\mathcal{H}$–код N размерности $k(N) = |B(N)|$ и длины $n(N) = |\mathcal{H}| = u$ с $\mathbb{F}$-базисом $B(N) = \{\eta_1; ...; \eta_{k(N)}\}$, $\eta_i = \sum_{h\in\mathcal{H}} \beta_{i,h}\delta_h$, $i = 1, ..., k(N)$. Порождающая

матрица $G(N)$ имеет вид:

$$G(N)=\begin{pmatrix}\nu_{\mathcal{H}}(\eta_1)\\ \dots\\ \nu_{\mathcal{H}}(\eta_{k(N)})\end{pmatrix}=\begin{pmatrix}\beta_{1,h_1} & \beta_{1,h_2} & \dots & \beta_{1,h_{n(N)}}\\ \beta_{2,h_1} & \beta_{2,h_2} & \dots & \beta_{2,h_{n(N)}}\\ \dots & \dots & \dots & \dots\\ \beta_{k(N),h_1} & \beta_{k(N),h_2} & \dots & \beta_{k(N),h_{n(N)}}\end{pmatrix}=\left(B_1|...|B_{n(N)}\right), \tag{6}$$

где $B_j=(\beta_{1,h_j},...,\beta_{k(N),h_j})^{\top}$ – вектор-столбец.

Тензорное произведение $\mathbb{F}\mathcal{G}\otimes_{\mathbb{F}\mathcal{H}}N(\subseteq\mathbb{F}\mathcal{G})$ является $\mathbb{F}\mathcal{G}$-кодом и называется кодом, индуцированным $\mathbb{F}\mathcal{H}$-кодом N [14], с. 41. (Теория тензорных произведений модулей, в рамках которой изучаются тензорные произведения групповых кодов, изложена, например, в [17].) Базис индуцированного кода может быть выражен через базис кода N и трансверсаль Y:

$$B(\mathbb{F}\mathcal{G}\otimes_{\mathbb{F}\mathcal{H}}N)=\cup_{i=1}^{\lambda}B(N)\delta_{y_i}. \tag{7}$$

Таким образом, размерность $k(\mathbb{F}\mathcal{G}\otimes_{\mathbb{F}\mathcal{H}}N)$ индуцированного кода $\mathbb{F}\mathcal{G}\otimes_{\mathbb{F}\mathcal{H}}N$ равна $|Y|k(N)$, а длина $n(\mathbb{F}\mathcal{G}\otimes_{\mathbb{F}\mathcal{H}}N)$ этого кода равна $|\mathcal{G}|=|Y|n(N)$. Отметим, что если $\mathrm{supp}_B(B(N))=B^{\mathcal{H}}$, то

$$\mathrm{supp}_B(B(N)\delta_x)\cap\mathrm{supp}_B(B(N)\delta_z)=\begin{cases}\mathrm{supp}_B(B(N)\delta_x), & \text{если } x-z\in\mathcal{H},\\ \emptyset, & \text{если } x-z\notin\mathcal{H}.\end{cases}$$

Опишем правило кодирования для индуцированного кода. Из (7) следует, что базис индуцированного кода состоит из векторов (функций) вида:

$$\kappa_{i,j}=\eta_i\delta_{y_j},\quad i=1,...,k(N),\quad j=1,...,\lambda.$$

Введем одинарную нумерацию на базисных векторах $\kappa_{i,j}$, полагая, что $\kappa_l:=\kappa_{i,j}$, если $l=i+(j-1)k(N)$. Рассмотрим две такие вспомогательные функции

$$\widehat{i}:\{1;...;k(N)\lambda\}\to\{1;...;k(N)\},\quad \widehat{j}:\{1;...;k(N)\lambda\}\to\{1;...;\lambda\},$$

что $\widehat{i}(l)+(\widehat{j}(l)-1)k(N)=l$. Эти две функции связывают одинарную нумерацию базисных векторов с двойной. Тогда кодирование произвольного вектора $\mathbf{s}=(s_1,...,s_{k(N)\lambda})(\in\mathbb{F}^{k(N)\lambda})$ можно представить в виде:

$$\begin{aligned}\mathbf{s}\to\mathbf{c}&=\sum_{l=1}^{k(N)\lambda}s_l\kappa_l=\sum_{l=1}^{k(N)\lambda}s_l\kappa_{\widehat{i}(l),\widehat{j}(l)}=\sum_{l=1}^{k(N)\lambda}s_l\eta_{\widehat{i}(l)}\delta_{y_{\widehat{j}(l)}}\\ &=\sum_{l=1}^{k(N)\lambda}s_l\sum_{h\in\mathcal{H}}\beta_{\widehat{i}(l),h}\delta_h\delta_{y_{\widehat{j}(l)}}=\sum_{h\in\mathcal{H}}\sum_{l=1}^{k(N)\lambda}s_l\beta_{\widehat{i}(l),h}\delta_{hy_{\widehat{j}(l)}}\\ &=\sum_{p=1}^{n(N)}\sum_{v=1}^{\lambda}\left(\sum_{i=1}^{k(N)}s_{i+(v-1)k(N)}\beta_{i,h_p}\right)\delta_{h_py_v}.\end{aligned}$$

Если $\mathbf{s} = (\mathbf{s}_1, ..., \mathbf{s}_\lambda)$ — представление информационного вектора $\mathbf{s}$ в виде конкатенации λ подвекторов длины $k(N)$ каждый, то получаем, что в ходе кодирования для фиксированного $h_j(\in \mathcal{H})$ вычисляются λ коэффициентов кодового вектора $\mathbf{c}$ при базисных элементах $\delta_{h_j y_1}, ..., \delta_{h_j y_\lambda}$:

$$c_{\delta_{h_j y_v}} = \left\langle \mathbf{s}_v, B_j^\top \right\rangle \quad v = 1, ..., \lambda, \tag{8}$$

где $\langle \mathbf{a}, \mathbf{b} \rangle$ — скалярное произведение векторов $\mathbf{a}$ и $\mathbf{b}$, B_j — j-ый столбец матрицы (6). Для выбранной трансверсали Y и подгруппы $\mathcal{H}$ определим на элементах группы $\mathcal{G}$ новый линейный порядок с помощью индуцированных на Y и $\mathcal{H}$ соответствующих порядков $\mathrm{ord}(Y)$ и $\mathrm{ord}(\mathcal{H})$:

$$\mathrm{ord}(\mathcal{H}, Y) = \{\underbrace{h_1 y_1; h_2 y_1; ...; h_{n(N)} y_1}_{\mathcal{H} y_1}; \underbrace{h_1 y_2; h_2 y_2; ...; h_{n(N)} y_2}_{\mathcal{H} y_2}; ...; \underbrace{h_1 y_\lambda; h_2 y_\lambda; ...; h_{n(N)} y_\lambda}_{\mathcal{H} y_\lambda}\}. \tag{9}$$

Порядок $\mathrm{ord}(\mathcal{H}, Y)$, вообще говоря, отличается от исходного порядка $\mathrm{ord}(\mathcal{G})$ и задает отображение $\nu_{\mathcal{H},Y} : \mathbb{F}\mathcal{G} \to \mathbb{F}^{|\mathcal{G}|}$. Отметим, что порядок выбранной трансверсали Y индуцирует порядок $\mathrm{ord}(Y, \mathcal{G}/\mathcal{H})$ на смежных классах фактор-группы $\mathcal{G}/\mathcal{H}$, из которых выбраны соответствующие элементы трансверсали.

Пусть $\omega_{(\mathcal{H},Y);\mathcal{G}} : \mathbb{F}^{|\mathcal{G}|} \to \mathbb{F}^{|\mathcal{G}|}$ — такое линейное отображение, что

$$\nu_{\mathcal{G}} = \nu_{\mathcal{H},Y} \circ \omega_{(\mathcal{H},Y);\mathcal{G}}, \tag{10}$$

где в композиции отображений $\nu_{\mathcal{H},Y} \circ \omega_{(\mathcal{H},Y);\mathcal{G}}$ сначала применяется отображение $\nu_{\mathcal{H},Y}$, а затем $\omega_{(\mathcal{H},Y);\mathcal{G}}$. Отметим, что отображению $\omega_{(\mathcal{H},Y);\mathcal{G}}$ соответствует перестановочная $(|\mathcal{G}| \times |\mathcal{G}|)$-матрица $W_{(\mathcal{H},Y);\mathcal{G}}$, которую назовем матрицей перехода от порядка $\mathrm{ord}(\mathcal{H}, Y)$ к порядку $\mathrm{ord}(\mathcal{G})$. В дальнейшем для $a(\in \mathbb{N})$ множество перестановочных $(a \times a)$-матриц будем обозначать $\mathrm{MP}(a)$.

Лемма 1. *Пусть $G(C)$ — порождающая матрица кода $C = \mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ в зафиксированном порядке $\mathrm{ord}(\mathcal{G})$; $\mathrm{ord}(\mathcal{H})$ и $\mathrm{ord}(Y)$ — индуцированные порядки соответственно на подгруппе $\mathcal{H}$ и выбранной трансверсали Y, тогда*

$$G(C) = G_0(C) W_{(\mathcal{H},Y);\mathcal{G}}, \tag{11}$$

где

$$G_0(C) = \left(\begin{array}{c|c|c|c} G(N) & O & ... & O \\ \hline O & G(N) & ... & O \\ \hline ... & ... & ... & ... \\ \hline O & ... & ... & G(N) \end{array}\right). \tag{12}$$

Доказательство. Учитывая вид (6) матрицы $G(N)$, правило вычисления коэффициентов кодового слова (8) и порядок $\mathrm{ord}(\mathcal{H}, Y)$ (см. (9)), получим, что порождающая матрица кода $C = \mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ имеет вид (12) (ср. [15], формула (16)). Для перехода к зафиксированному на $\mathcal{G}$ порядку необходимо, как следует из (10), матрицу $G_0(C)$ умножить на матрицу перехода $W_{(\mathcal{H},Y);\mathcal{G}}$. □

Очевидно, что если Y' — трансверсаль, отличная от Y, с индуцированным на ней порядком $\mathrm{ord}(Y)'$, то существуют такие линейные отображения $\omega_{(\mathcal{H},Y);(\mathcal{H},Y')} : \mathbb{F}^{|\mathcal{G}|} \to \mathbb{F}^{|\mathcal{G}|}$ и $\omega_{(\mathcal{H},Y');\mathcal{G}} : \mathbb{F}^{|\mathcal{G}|} \to \mathbb{F}^{|\mathcal{G}|}$, что

$$\omega_{(\mathcal{H},Y);\mathcal{G}} = \omega_{(\mathcal{H},Y);(\mathcal{H},Y')} \circ \omega_{(\mathcal{H},Y');\mathcal{G}}. \tag{13}$$

Перестановочные матрицы, соответствующие отображениям $\omega_{(\mathcal{H},Y);(\mathcal{H},Y')}$ и $\omega_{(\mathcal{H},Y');\mathcal{G}}$ в (13), обозначим $W_{(\mathcal{H},Y);(\mathcal{H},Y')}$ и $W_{(\mathcal{H},Y');\mathcal{G}}$. Далее для сокращения записи понадобится тензорное произведение $A \otimes B$ матрицы $A = (a_{i,j})$ размера $(r \times s)$ и матрицы B , под которым понимается матрица вида:

$$A \otimes B = \begin{pmatrix} a_{1,1}B & \dots & a_{1,s}B \\ a_{2,1}B & \dots & a_{2,s}B \\ \dots & \dots & \dots \\ a_{r,1}B & \dots & a_{r,s}B \end{pmatrix}.$$

Лемма 2. *Пусть $G(C)$ — порождающая матрица кода $C = \mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ в зафиксированном порядке $\mathrm{ord}(\mathcal{G})$; $\mathrm{ord}(\mathcal{H})$, $\mathrm{ord}(Y)$, $\mathrm{ord}(Y')$ — индуцированные порядки соответственно на подгруппе $\mathcal{H}$ и выбранных разных трансверсалях Y и Y'. Тогда*

$$W_{(\mathcal{H},Y);(\mathcal{H},Y')} = (M \otimes I_{n(N)})\mathrm{diag}(D_1, ..., D_\lambda), \tag{14}$$

где I_p — единичная матрица порядка p, $M \in \mathrm{MP}(\lambda)$, $D_i \in \mathrm{MP}(n(N))$, $i = 1, ..., \lambda$, $\mathrm{diag}(D_1, ..., D_\lambda)$ — блочно-диагональная $(n(N)\lambda \times n(N)\lambda)$-матрица.

Доказательство. Непосредственно из определения порядка (9) следует, что если трансверсаль Y' отличается от Y только представителями смежных классов, из которых выбраны соответствующие представители (то есть если $y'_i y^{-}1_i \in \mathcal{H}$ для всех $i = 1, ..., \lambda$), то найдутся такие перестановочные $(n(N) \times n(N))$–матрицы $D_1, ..., D_\lambda$, что $W_{(\mathcal{H},Y),\mathcal{G}} = \mathrm{diag}(D_1, ..., D_\lambda)W_{(\mathcal{H},Y'),\mathcal{G}}$. В этом случае порядки на смежных классах $\mathrm{ord}(Y, \mathcal{G}/\mathcal{H})$ и $\mathrm{ord}(Y', \mathcal{G}/\mathcal{H})$, индуцированные соответствующими порядками $\mathrm{ord}(Y, \mathcal{H})$ и $\mathrm{ord}(Y', \mathcal{H})$, не отличаются. Если же трансверсаль Y' выбирается произвольно, то порядки $\mathrm{ord}(Y, \mathcal{G}/\mathcal{H})$ и $\mathrm{ord}(Y', \mathcal{G}/\mathcal{H})$ совпадают с точностью до некоторой перестановки. Отсюда для произвольных Y и Y' следует (14). □

Из леммы 1 непосредственно следует, что $d(\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N) = d(N)$. В [14] показано также, что $\mathrm{dmaj}_{B^{\mathcal{G}}}(\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N) = \mathrm{dmaj}_{B^{\mathcal{H}}}(N)$, где $B^{\mathcal{G}}$ и $B^{\mathcal{H}}$ — канонические базисы групповых алгебр $\mathbb{F}\mathcal{G}$ и $\mathbb{F}\mathcal{H}$ соответственно. Декодирование индуцированного группового кода $\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ выполняется с помощью декодирующих деревьев группового кода N. В [14] показано, что для построения всех декодирующих деревьев индуцированного кода достаточно иметь одно дерево для кода N, например, $\mathrm{WB}_{\mathbf{1},r_1,L_1}[N]$, где $\mathbf{1} = 1\delta_{\hat{1}}$, $\hat{1}$ – нейтральный элемент группы $\mathcal{H}$; в [15] построен соответствующий алгоритм MakeGWBTree, который по дереву $\mathrm{WB}_{\mathbf{1},r_1,L_1}[N]$ для любого $g \in \mathcal{G}$ строит дерево $\mathrm{WB}_{1g,r_{1g},L_{1g}}[C]$, используя действие (2). Таким образом, по дереву $\mathrm{WB}_{\mathbf{1},r_1,L_1}[N]$ набор декодирующих деревьев для индуцированного кода определяется однозначно. Заметим, что правило декодирования индуцированного кода зависит от выбора трансверсали Y и зафиксированного линейного порядка на группе $\mathcal{G}$. Эта зависимость раскрывается в следующем разделе, где рассматривается стойкость криптосистемы типа Мак-Элиса на индуцированных групповых кодах.

Очевидно, что если на группе $\mathcal{G}$ зафиксирован порядок $\mathrm{ord}(\mathcal{H}, Y)$, то любое кодовое слово индуцированного кода $\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ представляет собой конкатенацию λ кодовых слов кода N. Поэтому если на группе $\mathcal{G}$ зафиксирован порядок (9), то из представления (12) порождающей матрицы кода $C = \mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ следует, что исправление ошибок в принятом векторе $\mathbf{z}(\in \mathbb{F}^{n(N)\lambda})$ можно выполнить путем применения λ раз декодера $\mathrm{Dec}_N(= \mathrm{MajDecoder}_N)$ для кода N к подвекторам $\mathbf{z}_i(\in \mathbb{F}^{n(N)})$ вектора $\mathbf{z} = (\mathbf{z}_1, ..., \mathbf{z}_{n(N)})$ и последующей конкатенации результатов декодирования. Обозначим такой декодер кода $C = \mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ символом Dec_C^0.

Таким образом, для зафиксированного порядка $\mathrm{ord}(\mathcal{G})$, если $\mathbf{z} = \mathbf{c} + \mathbf{e}(\in \mathbb{F}^{n(N)\lambda})$ – принятый вектор, соответствующий информационному вектору $\mathbf{s} \in \mathbb{F}^{k(N)\lambda}$, и $w(\mathbf{e}) \leq \mathrm{dmaj}(N)/2$, то правило декодирования вектора $\mathbf{z}(\in \mathbb{F}^{n(N)\lambda})$ имеет вид:

$$\mathrm{Dec}_{\mathbb{F}\mathcal{G}\otimes_{\mathbb{F}\mathcal{H}}N}(\mathbf{z}) = \mathrm{Dec}_C^0(\mathbf{z}W^{-1}_{(\mathcal{H},Y);\mathcal{G}}). \tag{15}$$

2. Криптосистемы типа Мак-Элиса на индуцированных групповых кодах

2.1. Криптосистема типа Мак-Элиса

Опишем криптосистему типа Мак-Элиса в случае произвольного $[n(C), k(C), d(C)]$-кода C; для такой криптосистемы будем использовать обозначение $\mathrm{McE}(C)$. Пусть $G(C)$ — порождающая матрица кода C. Для шифрования сообщений из пространства $\mathbb{F}^{k(C)}$ используется открытый ключ $\mathbf{k}_{\mathrm{pub}} = (\widetilde{G}, t = \lfloor (d(C)-1)/2 \rfloor)$, где $\widetilde{G} = SG(C)P$, S – случайная невырожденная $(k(C) \times k(C))$-матрица, случайная P – перестановочная $(n(C) \times n(C))$-матрица, а для расшифрования применяется секретный ключ $\mathbf{k}_{\mathrm{sec}} = (S, P)$. Правило шифрования произвольного сообщения $\mathbf{s}(\in \mathbb{F}^{k(C)})$ имеет вид: $\mathbf{z} = \mathbf{s}\widetilde{G} + \mathbf{e}$, где вес искусственно добавляемой ошибки $\mathbf{e}$ удовлетворяет неравенству $w(\mathbf{e}) \leq t$. Для расшифрования $\mathbf{c}$ секретный ключ $\mathbf{k}_{\mathrm{sec}}$ используется по правилу: $\mathbf{s} = \mathrm{Dec}_C(\mathbf{z}P^{-1})S^{-1}$.

Пусть теперь N – групповой $[n(N), k(N), d(N) = \mathrm{dmaj}(N)+1]$-код, $N \subseteq \mathbb{F}\mathcal{H}$, $\mathcal{H}$ — подгруппа группы $\mathcal{G}$, $C := \mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ —индуцированный $[\lambda n(N), \lambda k(N), d(N)]$-код, где $\lambda = (\mathcal{G} : \mathcal{H})$ – индекс подгруппы $\mathcal{H}$ в группе $\mathcal{G}$, $\mathrm{ord}(\mathcal{G})$ — зафиксированный на $\mathcal{G}$ порядок, не являющийся секретом. Рассмотрим два варианта криптосистемы типа Мак-Элиса: криптосистему $\mathrm{WeakMcE}(C)$, в которой трансверсаль является частью секретного ключа и $P = W_{(\mathcal{H},Y);\mathcal{G}}$, и криптосистему $\mathrm{InducedMcE}(C)$, в которой трансверсаль не является секретной, а P случайно и равновероятно выбирается из множества перестановочных матриц размера $(n(N)\lambda \times n(N)\lambda)$. Полное описание этих криптосистем приведено в Таблице 1.

Отметим, что длины открытых и секретных ключей криптосистем $\mathrm{WeakMcE}(C)$ и $\mathrm{InducedMcE}(C)$ возрастают не более чем в λ^2 раз по сравнению с размерами соответствующих ключей криптосистемы $\mathrm{McE}(N)$.

Таблица 1. Описание криптосистем WeakMcE(C) и InducedMcE(C)
Table 1. Description of WeakMcE(C) and InducedMcE(C) cryptosystems

	WeakMcE(C)	InducedMcE(C)
$\mathrm{ord}(\mathcal{G})$ и $\mathrm{ord}(\mathcal{H})$	не секретные	
трансверсаль Y	секретная	не секретная
матрица S	случайная и равновероятная	
матрица P	$W_{(\mathcal{H},Y);\mathcal{G}}$	случайная и равновероятная
матрица $\widetilde{G}$	$SG_0(C)P = SG_0(C)W_{(\mathcal{H},Y);\mathcal{G}}$	$SG(C)P = SG_0(C)W_{(\mathcal{H},Y);\mathcal{G}}P$
t	$\lfloor \mathrm{dmaj}(N)/2 \rfloor$	
шифрование $\mathbf{s}$	$\mathbf{z} = \mathbf{s}\widetilde{G} + \mathbf{e}, w(\mathbf{e}) \le t$	
расшифрование $\mathbf{z}$	$\mathrm{Dec}_C^0\left(\mathbf{z}W^{-1}_{(\mathcal{H},Y);\mathcal{G}}\right)S^{-1}$	$\mathrm{Dec}_C^0\left(\mathbf{z}P^{-1}W^{-1}_{(\mathcal{H},Y);\mathcal{G}}\right)S^{-1}$
длина $\mathbf{k}_{\mathrm{pub}}$, битов	$\lceil (k(N)n(N)\lambda^2)\log_2\lvert\mathbb{F}\rvert\rceil$	
длина $\mathbf{k}_{\mathrm{sec}}$, битов	$\left\lceil \log_2\left(\lvert\mathbb{F}\rvert^{(k(N)\lambda)^2} n(N)^\lambda \lambda!\right)\right\rceil$	$\left\lceil \log_2\left(\lvert\mathbb{F}\rvert^{(k(N)\lambda)^2} (n(N)\lambda)!\right)\right\rceil$

2.2. Анализ стойкости криптосистем типа Мак-Элиса на индуцированных групповых кодах к атакам на ключ

В качестве модели нарушителя рассмотрим наблюдателя, целью которого является нахождение подходящего секретного ключа для правильного расшифрования криптограмм. Предполагается, что наблюдатель имеет алгоритм Attack, с помощью которого может быть эффективно найден подходящий секретный ключ для криптосистемы McE(N).

2.2.1. Анализ криптосистемы WeakMcE(C)

Так как наблюдателю неизвестна трансверсаль Y, то он может зафиксировать произвольную трансверсаль Y', на которой зафиксированный порядок $\mathrm{ord}(\mathcal{G})$ индуцирует порядок $\mathrm{ord}(Y)$. В соответствии с леммой 2, существуют такие перестановочные матрицы $M(\in \mathrm{MP}(\lambda))$ и $D_1, ..., D_\lambda(\in \mathrm{MP}(n(N)))$, что

$$W_{(\mathcal{H},Y);\mathcal{G}} = (M \otimes I_{n(N)})\mathrm{diag}(D_1, ..., D_\lambda)W_{(\mathcal{H},Y');\mathcal{G}}. \tag{16}$$

Тогда

$$\widetilde{G} = SG_0(C)(M \otimes I_{n(N)})\mathrm{diag}(D_1, ..., D_\lambda)W_{(\mathcal{H},Y');\mathcal{G}}.$$

Так как матрицу $W_{(\mathcal{H},Y');\mathcal{G}}$ наблюдатель может построить по общеизвестному порядку $\mathrm{ord}(\mathcal{G})$ и порядку $\mathrm{ord}(\mathcal{H}, Y)$, то несложно видеть, что матрица $\widetilde{G}W^{-1}_{(\mathcal{H},Y');\mathcal{G}}$ может

быть представлена в виде:

$$\begin{aligned}\widetilde{G}W^{-1}_{(\mathcal{H},Y');\mathcal{G}} &= SG_0(C)(M \otimes I_{n(N)})\mathrm{diag}(D_1, ..., D_\lambda) \\ &= (M \otimes I_{k(N)})SG_0(C)\mathrm{diag}(D_1, ..., D_\lambda).\end{aligned} \tag{17}$$

Представим матрицу $S' = M \otimes I_{k(N)}S (\in \mathrm{GL}(k(N)\lambda, \mathbb{F}))$ в блочном виде:

$$S' = \left(\begin{array}{c|c|c|c} S'_{11} & S'_{12} & ... & S'_{1\lambda} \\ \hline S'_{21} & S'_{22} & ... & S'_{2\lambda} \\ \hline ... & ... & ... & ... \\ \hline S'_{\lambda 1} & S'_{\lambda 2} & ... & S'_{\lambda\lambda} \end{array}\right), \tag{18}$$

где S'_{ij} – $(k(N) \times k(N))$-матрица. Тогда

$$\widetilde{G}W^{-1}_{(\mathcal{H},Y');\mathcal{G}} = \left(\begin{array}{c|c|c|c} S'_{11}G(N)D_1 & S'_{12}G(N)D_2 & ... & S'_{1\lambda}G(N)D_\lambda \\ \hline S'_{21}G(N)D_1 & S'_{22}G(N)D_2 & ... & S'_{2\lambda}G(N)D_\lambda \\ \hline ... & ... & ... & ... \\ \hline S'_{\lambda 1}G(N)D_1 & S'_{\lambda 2}G(N)D_2 & ... & S'_{\lambda\lambda}G(N)D_\lambda \end{array}\right). \tag{19}$$

Заметим, что, несмотря на то, что матрица S' имеет полный ранг, каждая из подматриц S'_{ij} может быть неполного ранга, поэтому применение алгоритма Attack непосредственно к блокам матрицы (19) не представляется корректным. С другой стороны, каждый блочный столбец в матрице (18) имеет ранг $k(N)$, следовательно, для каждого $j = 1, ..., \lambda$ в матрице

$$\left(\begin{array}{c} S'_{j1}G(N)D_j \\ \hline S'_{j2}G(N)D_j \\ \hline ... \\ \hline S'_{j\lambda}G(N)D_j \end{array}\right) = \left(\begin{array}{c} S'_{j1} \\ \hline S'_{j2} \\ \hline ... \\ \hline S'_{j\lambda} \end{array}\right) G(N)D_j \tag{20}$$

найдутся $k(N)$ линейно независимых строк, которые образуют матрицу вида $G_j = \widehat{S}_j G(N)D_j$, $\mathrm{rank}(\widehat{S}_j) = k(N)$. Тогда к матрице G_j может быть применен алгоритм атаки Attack для нахождения подходящих матриц $(\widehat{S}'_j, D'_j)$ таких, что $G_j = \widehat{S}'_j G(N)D'_j$:

$$(\widehat{S}'_j, D'_j) = \mathrm{Attack}(G_j). \tag{21}$$

Таким образом, подходящий секретный ключ криптосистемы $\mathrm{WeakMcE}(C)$ может быть найден по алгоритму AttackWeakInduced.

Теорема 1. *Пусть Q — вычислительная сложность алгоритма* Attack*, применяемого для нахождения подходящего секретного ключа для криптосистемы* $\mathrm{McE}(N)$*, тогда $\mathcal{O}(\lambda Q + (k(N)\lambda)^3)$ – вычислительная сложность* AttackWeakInduced *нахождения подходящего секретного ключа криптосистемы* $\mathrm{WeakMcE}(C)$.

Доказательство. В алгоритме AttackWeakInduced алгоритм Attack применяется λ раз для нахождения перестановочной матрицы P', а для нахождения матрицы S' необходимо решить матричное уравнение. □

Таким образом, сложность взлома криптосистемы $\mathrm{WeakMcE}(C)$ линейно зависит от сложности взлома системы $\mathrm{McE}(N)$. Отметим также, что для каждого $i = 1, ..., \lambda$ матрица D_i выбирается не из множества мощности $|\mathcal{H}|!$ всех возможных перестановочных матриц, а из множества матриц, которое имеет мощность $|\mathcal{H}|$. Поэтому алгоритм Attack, возможно, может быть существенно упрощен.

Алгоритм 1 AttackWeakInduced

Вход: Матрицы $\widetilde{G}$ и $W_{(\mathcal{H},Y');\mathcal{G}}$
Выход: Подходящий секретный ключ (S', P')
Для $j \in \{1; ...; \lambda\}$ **выполнять**
  выбрать из матрицы $\widetilde{G}W^{-1}_{(\mathcal{H},Y');\mathcal{G}}$ столбцы с номерами $(j-1)n(N)+1, ..., jn(N)$;
  из выбранных столбцов составить матрицу $\widetilde{G}_j$;
  в $\widetilde{G}_j$ выбрать $k(N)$ линейно независимых строк;
  из выбранных строк составить матрицу G_j;
  $(\widehat{S}'_j, D'_j) = \mathrm{Attack}(G_j)$;// найти секретный ключ для McE(N), см. (21)
конец Для
$P' := \mathrm{diag}(D'_1, ..., D'_\lambda)W_{(\mathcal{H},Y');\mathcal{G}}$;
решить матричное уравнение $\widetilde{G}P'^{-1} = S'G_0(C)$ относительно S';
Вернуть (S', P')

2.2.2. Анализ криптосистемы InducedMcE(C)

Рассмотрим криптосистему InducedMcE(C), в которой перестановочная матрица P выбирается случайно и равновероятно из MP$(n(N)\lambda)$. Из (11) следует, что открытый ключ этой криптосистемы имеет вид: $\widetilde{G} = SG_0(C)W_{(\mathcal{H},Y);\mathcal{G}}P$. Отметим, что найдется такая матрица $P' \in \mathrm{MP}(n(N)\lambda)$, что $PW^{-1}_{(\mathcal{H},Y);\mathcal{G}} = W^{-1}_{(\mathcal{H},Y);\mathcal{G}}P'$, то имеет место представление

$$\widetilde{G}W^{-1}_{(\mathcal{H},Y);\mathcal{G}} = SG_0(C)P', \tag{22}$$

где P' – неизвестная (секретная) матрица. Следующая лемма очевидна.

Лемма 3. *Множество перестановочных* $(n(N)\lambda \times n(N)\lambda)$*-матриц*

$$\Theta = \{(M \otimes I_\lambda)\mathrm{diag}(D_1, ..., D_\lambda) | M \in \mathrm{MP}(\lambda), D_i \in \mathrm{MP}(n(N)), i = 1, ..., \lambda\}. \tag{23}$$

является подгруппой группы MP$(n(N)\lambda)$, $|\Theta| = (n(N)!)^\lambda \lambda!$.

Таким образом, секретная матрица P' (см. (22)) принадлежит одному из фактор-классов фактор-множества $\mathrm{MP}(n(N)\lambda)/\Theta = \{\Gamma_j\}_{j=1}^g$, где $g = \frac{|\mathrm{MP(n(N)\lambda)}|}{|\Theta|} = \frac{(n(N)\lambda)!}{(n(N)!)^\lambda \lambda!}$. Пусть $\Gamma \in \mathrm{MP}(n(N)\lambda)/\Theta$ — фактор-класс, которому принадлежит матрица P'. Тогда имеет место представление: $\Gamma = \{TP' | T \in \Theta\}$. Рассмотрим произвольную матрицу $\Pi = TP'$ из фактор-класса Γ. Из (22) следует, что

$$\begin{aligned} \widetilde{G}W^{-1}_{(\mathcal{H},Y);\mathcal{G}}\Pi^{-1} &= SG_0(C)P'\Pi^{-1} \\ &= SG_0(C)T^{-1}. \end{aligned} \tag{24}$$

В силу леммы 3, матрица T^{-1} принадлежит Θ. Следовательно, матрица (24) имеет вид (17), и поэтому к ней может быть применен алгоритм AttackWeakInduced. Алгоритм AttackInduced для нахождения подходящего ключа системы InducedMcE(C) основан на переборе представителей смежных классов из $\mathrm{MP}(n(N)\lambda)/\Theta$.

Алгоритм 2 AttackInduced

Вход: Матрицы $\widetilde{G}$ и $W_{(\mathcal{H},Y);\mathcal{G}}$
Выход: (S', P')
1: **Для** $\Gamma \in \mathrm{MP}(n(N)\lambda)/\Theta$ **выполнять**
2: выбрать любую матрицу Π из Γ;
3: $(S', P') = \mathrm{AttackWeakInduced}(\widetilde{G}\Pi^{-1}, W_{(\mathcal{H},Y);\mathcal{G}})$;
4: **Если** $\mathrm{rank}(S') = k(N)\lambda$ **и** $P' \in \mathrm{MP}(n(N)\lambda)$ **и** $S'G_0(C)P' = \widetilde{G}\Pi^{-1}$ **тогда**
5: выйти из цикла;// подходящий ключ найден
6: **конец Если**
7: **конец Для**
8: **Вернуть** (S', P');

Теорема 2. *Пусть Q — вычислительная сложность алгоритма* Attack, *применяемого для нахождения подходящего ключа для криптосистемы* $\mathrm{McE}(N)$, *тогда*

$$\mathcal{O}\left(\left(\frac{\lambda^{n(N)-1}}{e}\right)^{\lambda}(\lambda Q + (k(N)\lambda)^3)\right)$$

– вычислительная сложность алгоритма AttackInduced *нахождения подходящего секретного ключа криптосистемы* $\mathrm{InducedMcE}(C)$.

Доказательство. Утверждение вытекает из того, что для нахождения ключа методом перебора достаточно перебрать множество представителей фактор-классов множества $\mathrm{MP}(n(N)\lambda)/\Theta$. Сложность проверки одного представителя, в соответствии с теоремой 1, составляет $\mathcal{O}(\lambda Q + (k(N)\lambda)^3)$, а всего $\frac{(n(N)\lambda)!}{(n(N)!)^{\lambda}\lambda!}$ смежных классов. Используя формулу Стирлинга ($t! \approx \sqrt{2\pi t}\left(\frac{t}{e}\right)^t$), получим искомую оценку. □

Проиллюстрируем эффект от использования индуцированных кодов на примере. Пусть $\mathbb{F} = \mathbb{F}_2$, $\mathcal{G} = \mathbb{Z}_\lambda \oplus \mathbb{Z}_2^p$, $\mathcal{H} \cong \mathbb{Z}_2^p$, N — двоичный код Бермана, изоморфный двоичному коду Рида-Маллера $\mathcal{RM}(r, p)$ длины $n(N) = 2^p$ и размерности $k(N) = \sum_{i=0}^{r} C_r^i$. Согласно [11], криптосистема $\mathrm{McE}(N)$ при $p \leq 16$ может быть взломана за приемлемое время на ноутбуке с процессором 2.1 ГГц. В частности, при $p \leq 8$ ($n(N) \leq 256$) взлом длится менее одной секунды. Пусть $C = \mathbb{F}_2\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$. Для взлома криптосистемы $\mathrm{InducedMcE}(C)$, согласно теореме 2, необходимо перебрать порядка $K(\lambda, n(N)) = \left(\frac{\lambda^{n(N)-1}}{e}\right)^{\lambda}$ ключей. В Таблице 2 приведены значения $\log_2(K(\lambda, n(N)))$ для $\lambda = 2, ..., 9$ и $n(N) = 2, 4, 8, 16, 32, 64, 128, 256$.

В настоящее время, согласно [18], считается вычислительно неосуществимым перебор по ключевому множеству мощности 2^{128} и более. В таблице 2 жирным шрифтом выделены те соотношения $(n(N), \lambda)$, для которых криптосистема $\mathrm{InducedMcE}(C)$ представляется стойкой. Как видно из таблицы, конструкция индуцированных кодов позволяет строить стойкие криптосистемы даже на кодах малой длины.

Таблица 2. Значения величины $\log_2(K(\lambda, n(N)))$
Table 2. Values of $\log_2(K(\lambda, n(N)))$

	$n(N)$							
λ	2	4	8	16	32	64	128	256
2	0,13	4,13	12,13	28,13	60,13	124,13	**252,13**	**508,13**
3	2,04	11,55	30,57	68,61	**144,69**	**296,84**	**601,16**	**>1024**
4	4,27	20,27	52,27	116,27	**244,27**	**500,27**	**1012,27**	**>1024**
5	6,77	29,99	76,42	**169,30**	**355,06**	**726,56**	**>1024**	**>1024**
6	9,50	40,52	102,56	**226,63**	**474,79**	**971,10**	**>1024**	**>1024**
7	12,43	51,73	**130,34**	**287,55**	**601,97**	**>1024**	**>1024**	**>1024**
8	15,54	63,54	**159,54**	**351,54**	**735,54**	**>1024**	**>1024**	**>1024**
9	18,80	75,86	**189,98**	**418,21**	**874,68**	**>1024**	**>1024**	**>1024**

Примечание: $K(\lambda, n(N))$ — количество перебираемых ключей в атаке AttackInduced на криптосистему InducedMcE($\mathbb{F}_2\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$), $\mathcal{G} = \mathbb{Z}_\lambda \oplus \mathbb{Z}_2^p$, $\mathcal{H} \cong \mathbb{Z}_2^p$, $N \cong \mathcal{RM}(r, p)$, $n(N) = 2^p$, $\lambda = 2, ..., 9$, $p \in \{1; ...; 8\}$.

Заметим, что сохранение в секрете трансверсали Y в системе InducedMcE(C) не усиливает стойкость. Действительно, наблюдатель может выбрать любую трансверсаль Y' и, как следует из (16) и (22), матрица открытого ключа тогда может быть приведена к виду:

$$\widetilde{G}W^{-1}_{(\mathcal{H},Y');\mathcal{G}} = SG_0(C)(M \otimes I_{n(N)})\mathrm{diag}(D_1, ..., D_\lambda)P'. \tag{25}$$

Если $\Pi = TP'$ — произвольная матрица из смежного класса, которому принадлежит неизвестная матрица P', то

$$\widetilde{G}W^{-1}_{(\mathcal{H},Y');\mathcal{G}}\Pi^{-1} = SG_0(C)T',$$

где $T' = (M \otimes I_{n(N)})\mathrm{diag}(D_1, ..., D_\lambda)T^{-1}$ принадлежит Θ в силу леммы 3. В этом случае для нахождения подходящего секретного ключа может быть применен алгоритм AttackInduced.

3. Протокол генерации общего ключа

Пусть N – групповой $[n(N), k(N), d(N)]$-код, C – соответствующий индуцированный $[n(N)\lambda, k(N)\lambda, d(N)]$-код. Как следует из предыдущего раздела, криптосистема InducedMcE(C) на индуцированных групповых кодах обладает высокой стойкостью к атаке на ключ. В то же время представляется, что стойкость этой криптосистемы к атакам на шифрограмму снижается по сравнению со стойкостью криптосистемы McE(N) к аналогичным атакам. Это связано с тем, что длина кодового слова увеличивается в λ раз, а количество ошибок, добавляемых в кодовый вектор при шифровании, остается прежним. В этом случае, во-первых, повышается вероятность успеха атаки путем декодирования по обобщенным информационным совокупностям [19],

а во-вторых, повышается вероятность успеха атаки, в ходе которой анализируется разность двух шифрограмм, соответствующих одному информационному сообщению [20]. Отметим, что одним из способов противодействия последней атаке может быть следующий. Вместо одного из λ информационных подблоков длины $k(N)$ использовать случайный вектор длины $k(N)$, выбираемый равновероятно каждый раз заново при шифровании сообщения. Номер случайного подблока может быть общеизвестен. В этом случае разность двух шифрограмм, соответствующих одному информационному сообщению, уже не будет равна разности векторов ошибок, добавленных при шифровании.

Снижения вероятностей успеха отмеченных атак на шифрограмму $\mathbf{z} = \mathbf{c} + \mathbf{e}$ (см. правило шифрования в Таблице 1) можно добиться, например, путем увеличения веса добавляемой ошибки $\mathbf{e}$ при шифровании. Однако превышение веса $w(\mathbf{e})$ величины $t = \lfloor (d(N) - 1)/2 \rfloor$ может привести к тому, что при расшифровании на стороне получателя не все подблоки длины $n(N)$ вектора $\mathbf{z}P^{-1}W^{-1}_{(\mathcal{H},Y);\mathcal{G}}$ могут быть декодированы корректно с помощью алгоритма декодирования Dec_N. Это связано с тем, что при умножении вектора $\mathbf{z}$ на матрицу $P^{-1}W^{-1}_{(\mathcal{H},Y);\mathcal{G}}$ в одном из таких подблоков может быть сгурппировано более t ошибочных координат. С другой стороны, если в правиле шифрования криптосистемы InducedMcE(C) выполняется неравенство $w(\mathbf{e}) \leq \lambda t$, то, как минимум, один из λ подблоков длины $n(N)$ будет расшифрован корректно. Эта особенность позволяет построить на основе системы InducedMcE(C) следующий протокол выработки общего секретного ключа.

Пусть A и B — пользователи, намеревающиеся выработать общий секретный ключ; $\mathbf{k}^A_{\mathrm{pub}} = (\widetilde{G}^A, \lambda t)$, $\mathbf{k}^A_{\mathrm{sec}} = (S^A, P^A)$, $\mathbf{k}^B_{\mathrm{pub}} = (\widetilde{G}^B, \lambda t)$, $\mathbf{k}^B_{\mathrm{sec}} = (S^B, P^B)$ — соответствующие открытые и секретные ключи пользователей A и B. Пользователи независимо друг от друга генерируют случайные последовательности $\mathbf{s}^A (\in \mathbb{F}^{k(N)\lambda})$ и $\mathbf{s}^B (\in \mathbb{F}^{k(N)\lambda})$, шифруют их на ключах друг друга и обмениваются соответствующими шифрограммами. Каждый из пользователей расшифровывает соответствующий принятый вектор и сообщает другому участнику по открытому каналу номер кодового подблока, декодированного корректно. Даже если декодировано корректно более одного подблока, представляется целесообразным отправлять только номер одного подблока, чтобы защититься от нечестного участника, генерирующего вектора ошибок специального вида с целью получения информации о перестановочной матрице второго легитимного участника. Таким образом, пользователь A отправляет пользователю B номер a, а пользователь B отправляет пользователю A номер b, после чего строится общий ключ. Ниже приведен протокол выработки общего ключа.

Протокол Выработки общего секретного ключа $\mathbf{k}$

1: A, B : Генерируют соответствующие пары ключей $(\mathbf{k}^A_{\text{pub}}, \mathbf{k}^A_{\text{sec}})$ и $(\mathbf{k}^B_{\text{pub}}, \mathbf{k}^B_{\text{sec}})$: $\mathbf{k}^A_{\text{pub}} = (\widetilde{G}^A, \lambda t)$, $\mathbf{k}^A_{\text{sec}} = (S^A, P^A)$, $\mathbf{k}^B_{\text{pub}} = (\widetilde{G}^B, \lambda t)$, $\mathbf{k}^B_{\text{sec}} = (S^B, P^B)$. Публикуют открытые ключи $\mathbf{k}^A_{\text{pub}}$ и $\mathbf{k}^B_{\text{pub}}$;
2: A, B : Случайно и равновероятно генерируют последовательности $\mathbf{s}^A (\in \mathbb{F}^{k(N)\lambda})$ и $\mathbf{s}^B (\in \mathbb{F}^{k(N)\lambda})$;
3: $A \to B : \mathbf{z}^A = (\mathbf{z}^A_1, ..., \mathbf{z}^A_\lambda) = \mathbf{s}^A \widetilde{G}^B + \mathbf{e}^A = \mathbf{c}^A + \mathbf{e}^A$, $w(\mathbf{e}^A) \le \lambda t$;
4: $B \to A : \mathbf{z}^B = (\mathbf{z}^B_1, ..., \mathbf{z}^B_\lambda) = \mathbf{s}^B \widetilde{G}^A + \mathbf{e}^B = \mathbf{c}^B + \mathbf{e}^B$, $w(\mathbf{e}^B) \le \lambda t$;
5: $A : \hat{\mathbf{c}}^B := \text{Dec}_C(\mathbf{z}^B (P^A)^{-1}) \widetilde{G}^A = (\widehat{\mathbf{c}}^B_1, ..., \widehat{\mathbf{c}}^B_\lambda)$;
6: $A \to B : a (\in \{1; ...; \lambda\})$, где $\rho(\widehat{\mathbf{c}}^B_a, \mathbf{z}^B_a) \le t$;
7: $B : \hat{\mathbf{c}}^A := \text{Dec}_C(\mathbf{z}^A (P^B)^{-1}) \widetilde{G}^B = (\widehat{\mathbf{c}}^A_1, ..., \widehat{\mathbf{c}}^A_\lambda)$;
8: $B \to A : b (\in \{1; ...; \lambda\})$, где $\rho(\widehat{\mathbf{c}}^A_b, \mathbf{z}^A_b) \le t$;
9: A, B: $\mathbf{k} := \text{Dec}_N(\hat{\mathbf{c}}^A_b + \hat{\mathbf{c}}^B_a)$.

Список литературы / References

[1] McEliece R.J., “A Public-Key Cryptosystem Based on Algebraic Coding Theory”, *JPL Deep Space Network Progress Report*, 1978, № 42, 114–116.

[2] Niederreiter H., “Knapsack-Type Cryptosystem and Algebraic Coding Theory”, *Probl. Control and Inform. Theory*, **15** (1986), 94–34.

[3] Gabidulin E.M. et al., “Ideals Over a Non-Commutative Ring and Their Application in Cryptology.”, *Advances in Cryptology–EUROCRYPT’91 / Ed. by D.W. Davies. Lect. Notes in Comp. Sci. Springer-Verlag*, **547** (1991), 482–489.

[4] Сидельников В.М., “Открытое шифрование на основе двоичных кодов Рида–Маллера”, *Дискретная математика*, **6**:2 (1994), 3–20; [Sidel’nikov V.M., “Open coding based on Reed–Muller binary codes”, *Diskr. Mat.*, **6**:2 (1994), 3–20, (in Russian)].

[5] Сидельников В.М., Шестаков С.О., “О системе шифрования, основанной на обобщенных кодах Рида–Соломона”, *Дискретная математика*, **3**:3 (1992), 57–63; [Sidel’nikov V.M., Shestakov S.O., “O sisteme shifrovanija, osnovannoj na obobshhennyh kodah Rida–Solomona”, *Diskr. Mat.*, **3**:3 (1992), 57–63, (in Russian).]

[6] Деундяк В.М. и др., “Модификация криптоаналитического алгоритма Сидельникова–Шестакова для обобщенных кодов Рида–Соломона и ее программная реализация”, *Известия высших учебных заведений. Северо-Кавказский регион. Технические науки*, 2006, № 4, 15–20; [Deundyak V.M. et al., “Modifikatsiya kriptoanaliticheskogo algoritma Sidel’nikova–Shestakova dlya obobshchennykh kodov Rida–Solomona i ee programmnaya realizatsiya”, *Izvestiya vysshikh uchebnykh zavedeniy. Severo-Kavkazskiy region. Tekhnicheskie nauki*, 2006, № 4, 15–20, (in Russian).]

[7] Wieschebrin C., “Cryptanalysis of the Niederreiter Public Key Scheme Based on GRS Subcodes”, *Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25–28*, 2010, 61–72.

[8] Gibson J.K., “The Security of the Gabidulin Public Key Cryptosystem”, *Advances in Cryptology EUROCRYPT’ 96. Ed. By U.M. Maurer, LNCS 1070*, **1070** (1996), 212–223.

[9] Overbeck R., “Structural Attacks for Public Key Cryptosystems based on Gabidulin Codes”, *Journal of Cryptology*, **21**:2 (2008), 280–301.

[10] Minder L., Shokrollahi A., “Cryptanalysis of the Sidelnikov cryptosystem”, *Lecture Notes in Computer Science*, **4515** (2007), 347–360.

[11] Чижов И.И., Бородин М.А., "Уязвимость криптосистемы Мак-Элиса, построенной на основе двоичных кодов Рида–Маллера", *Прикладаная дискрет. матем. Приложение*, 2013, № 6, 48–49; [Chizhov I.I., Borodin M.A., "Ujazvimost kriptosistemy Mak-Jelisa, postroennoj na osnove dvoichnyh kodov Rida–Mallera", *Prikladnaya diskr. mat. Prilozhenie*, 2013, № 6, 48–49, (in Russian).]

[12] Чижов И.И., Бородин М.А., "Эффективная атака на криптосистему Мак-Элиса, построенную на основе кодов Рида–Маллера", *Дискрет. матем.*, **26**:1 (2014), 10–20. [Chizhov I.I., Borodin M.A., "Jeffektivnaja ataka na kriptosistemu Mak-Jelisa, postroennuju na osnove kodov Rida–Mallera", *Diskr. Mat.*, **26**:1 (2014), 10–20, (in Russian).]

[13] Сидельников В.М., *Теория кодирования*, ФИЗМАТЛИТ, М., 2011; [Sidel'nikov V.M., *Teoriya kodirovaniya*, FIZMATLIT, M., 2011, (in Russian).]

[14] Циммерман К.-Х., *Методы теории модулярных представлений в алгебраической теории кодирования*, МЦНМО, М., 2011; [Tsimmerman K.-Kh., *Metody teorii modulyarnykh predstavleniy v algebraicheskoy teorii kodirovaniya*, MTsNMO, M., 2011, (in Russian).]

[15] Деундяк В.М., Косолапов Ю.В., "Алгоритмы для мажоритарного декодирования групповых кодов", *Моделирование и анализ информационных систем*, **22**:4 (2015), 464–482; [Deundyak V.M., Kosolapov Yu.V., "Algorithms for Majority Decoding of Group Codes", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 464–482, (in Russian).]

[16] Massey J.L., *Threshold Decoding*, MIT Press, Cambridge, 1963.

[17] Curtis C.W., Reiner I., *Representation Theory of Finite Groups and Associative Algebras*, Intersclence Publishers, New York, 1962.

[18] Lenstra A.K., Verheul E.R., "Selecting Cryptographic Key Sizes", *Journal of Cryptology*, **14** (2001), 255–293.

[19] Федоренко С.В., *Методы быстрого декодирования линейных кодов*, ГУАП, СПб, 2008; [Fedorenko S.V., *Metody bystrogo dekodirovaniya lineynykh kodov*, GUAP, SPb, 2008, (in Russian).]

[20] Berson T., "Failure of the McEliece public-key cryptosystem under message resend and related-message attack", *Proceedings of CRYPTO*, **1294** (1997), 213–220.

Deundyak V. M., Kosolapov Y. V., "Cryptosystem Based on Induced Group Codes", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 137–152.

DOI: 10.18255/1818-1015-2016-2-137-152

Abstract. The code C on a group $\mathcal{G}$, induced by the code N on a subgroup $\mathcal{H}$, has the property that for decoding the code C one can use the decoder for the code N. Therefore, if N has an efficient algorithm for decoding, we can build a class of induced codes with known decoding algorithms. This feature is used in this paper to build the code McEliece-type public key cryptosystems on induced group codes. For this cryptosystem we described operations of encryption and decryption, an analysis of the resistance to the attack on the private key is proposed, and also weak keys are highlighted, which is used while breaking McEliece-type cryptosystem on the induced code C is reduced to breaking this cryptosystem on the code N. It is shown that a practically resistant cryptosystem on the induced code C can be built on the code N with small length. Based on the proposed cryptosystem a common protocol for open channel key generation is developed.

Keywords: group codes, induced group codes, the McEliece cryptosystem

On the authors: Deundyak Vladimir Mikhailovich, orcid.org/0000-0001-8258-2419, PhD,
FGNU NII "Specvuzavtomatika", 51 Gazetniy lane, Rostov-on-Don, 344002, Russia
South Federal University, 105/42 Bolshaya Sadovaya Str., Rostov-on-Don, 344006, Russia, e-mail: vl.deundyak@gmail.com,
Kosolapov Yury Vladimirovich, orcid.org/0000-0002-1491-524X, PhD,
South Federal University, 105/42 Bolshaya Sadovaya Str., Rostov-on-Don, 344006, Russia, e-mail: itaim@mail.ru

Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 153–163
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 153–163

DOI: 10.18255/1818-1015-2016-2-153-163

УДК 517.9

Об эффективности минимизирующего подхода к оптимизации запросов

Мендкович Н. А.

получена 4 апреля 2016

Аннотация. Стандартной проблемой использования СУБД является недостаток эффективности и высокая стоимость доступа к хранимым данным. Допустимый уровень работы системы может достигаться с помощью технологий оптимизации запросов, определяющих наиболее эффективный способ выполнения конкретного запроса с помощью его модификации и определения возможных планов выполнения.

Целью данной работы является доказательство эффективности алгоритмов минимизации запроса, основанных на минимизации ограничения запроса и удаления избыточных условий.

Статья представляет алгоритмы минимизации, основанные на математических преобразованиях, определяющих и удаляющих избыточные условия из ограничения запроса, чтобы упростить его. Она включает алгоритмы, основанные на технологиях «поглощения условий», первичных импликант и минимизации множеств линейных неравенств.

Работа также включает теоретическое доказательство эффективности минимизирующего подхода, основанного на упрощении ограничения. Мы также рассматриваем экспериментальные результаты применения этих технологий оптимизации и их влияния на скорость обработки запроса. В конце мы представляем обзор влияния минимизации запроса на весь процесс оптимизации запроса.

Ключевые слова: оптимизация запросов, лексическая оптимизация запросов

Для цитирования: Мендкович Н. А., "Об эффективности минимизирующего подхода к оптимизации запросов", *Моделирование и анализ информационных систем*, **23**:2 (2016), 153–163.

Об авторах:
Мендкович Никита Андреевич, orcid.org/0000-0003-1694-7947, ООО «Фринет Групп», инженер,
Ленинский проспект, 47, Москва, 119991 Россия, e-mail: mend@f-group.ru

Введение

Решение задачи оптимизации запроса к СУБД путем минимизации его лексического представления относится к числу классических идей в теории баз данных. Применение минимизирующих преобразований запроса направлено на минимизацию числа содержащихся в нем условий путем исключения из его состава плеоназмов. Указанное преобразование оказывает влияние на ход и выполнения, и оптимизации запроса.

Указанный подход был представлен в классической работе П. Холла, в которой рассматривается применение приемов реляционной алгебры по преобразованию логических выражений в целях оптимизации запросов. Основные сформулированные им идеи лексической оптимизации путем упрощения запроса сводятся к следующему: объединение последовательности проекций в одну; исключение избыточных операций; упрощение выражений, использующих пустые отношения и тривиальные условия; вынос общих выражений «за скобки» [1].

Подходы, основанные на минимизации запроса, применяются также в современных СУБД. Например, в системе Oracle, подобная процедура производится над подзапросами на основании «свойства включения» («блок запроса X включает другой блок запроса Y, если результат Y является подмножеством (не обязательно собственным) результата X») [2]. Судя по корпоративным отчетам о модернизации оптимизирующих программ СУБД, минимизирующие процедуры также используются в MySQL [3], PostgreSQL [4] и Oracle [5].

Однако в литературе долгое время не были представлены исследования, теоретически обосновывающие эффективность минимизирующих преобразований, а также ее экспериментальные подтверждения. Кроме того, отсутствует общая характеристика места минимизирующих алгоритмов в процессе оптимизации запросов, особенно в контексте наличия оптимизационных процедур, ведущих к увеличению длины запроса и созданию дополнительных условий. В частности, это характерно для «улучшающих» преобразований запросов на основе так называемых «магических множеств» [6, 7].

Решению указанной проблемы посвящена настоящая статья. В разделе 1 описаны минимизирующие алгоритмы, избранные в качестве объекта изучения. В разделе 2 приводится теоретическое обоснование эффективности описанных минимизирующих преобразований. В разделе 3 описаны результаты экспериментального изучения результатов минимизации запросов. В разделе 4 описаны примеры запросов, содержащих избыточности описанных ниже типов, а также сферы применения СУБД, для которых они наиболее характерны. В разделе 5 анализируется прямое и косвенное влияние предложенных алгоритмов на процесс оптимизации запроса с помощью иных алгоритмов.

1. Описание минимизирующих алгоритмов оптимизации

В настоящей главе дано краткое описание трех алгоритмов оптимизации, описанных и опубликованных автором в более ранних работах [8–10]. Указанные алгоритмы позволяют преобразовать условия SQL-запросов, чьи ограничения представимы в виде ДНФ (дизъюнктивная нормальная форма), к наиболее краткой форме путем устранения избыточных условий, что может способствовать решению задачи определения вхождения для некоторых классов запросов.

Опишем формы плеоназмов, которые могут встречаться в составе ограничений запросов.

1. Плеоназм одного конъюнкта, т. е. включение в один и тот же конъюнкт запрос с одинаковым множеством переменных, характеризующих одно и то же

множество атрибутов, из которых некоторые являются логическим подмножеством других (лексическая тавтология) или образующих вместе тождественно истинное или ложное выражение.

2. Плеоназм множества конъюнктов, где однокоренные условия находятся в различных конъюнктах, однако в результате их присутствия запрос содержит смысловую избыточность.

3. Плеоназм общих атрибутов, при котором условия, входящие в состав запросов, не имеют общего корня, но корни двух ограничений имеют общие атрибуты, что приводит к возникновению плеоназма.

Особенностью предлагаемых алгоритмов являются способы минимизации ограничений, представимых в виде логической функции ДНФ:

- минимизация конъюнктов путем определения плеоназмов;
- минимизация ограничения в ДНФ путем поиска общих импликант;
- минимизация конъюнктов и ограничений, включающих условия, содержащие 2 и более атрибутов, путем определения условий — «следствий» из представленных условий.

Программная реализация представленного комплекса алгоритмов рассматривается в качестве оптимизатора, основанного на правилах (rule based optimization), т. е. преобразует запрос в более оптимальное, чем исходное, представление, что будет доказано в тексте ниже.

Границы применимости указанных алгоритмов определяются грамматикой вводящихся выражений. Алгоритмы предназначены только для обработки выражений, содержащих ограничения, представимые в виде булевых функций в ДНФ.

Формальное описание указанных ограничений запросов представлено с помощью формальной грамматики на основе подмножества правил формы Бакуса–Наура (так называемые BNF-грамматики) для языка SQL-2003 [11] (рис. 1).

```
<boolean value expression> ::= <boolean term>|<boolean value expression>
OR <boolean term>
<boolean term> ::= <boolean factor>|<boolean term> AND <boolean factor>
<boolean factor> ::= [ NOT ] <boolean test>
<boolean test> ::= <boolean primary> [ IS [ NOT ] <truth value> ]
<truth value> ::= TRUE | FALSE | UNKNOWN
<boolean primary> ::= <predicate>
<predicate> ::= <left paren> <boolean value expression> <right paren>
<left paren> ::= (
<right paren> ::= )
<boolean value expression> ::= <variable><operator><number>
<variable> ::= <sumstring>|<sumstring><sign><sumstring>
<sumstring> ::= <string>[<degree>]|
<string> ::= <letter>|<letter><string>
```

```
<letter> ::= A|...|Z|*|/
<degree> ::= ^<number>
<number> ::= <figure> | <figure><number>
<figure> ::= 0|...|9
<operator> ::= >|<|=|=>|<=|
<sign> ::= +|-
```

Рисунок 1
Fig. 1

Представленное множество правил отличается от канонического отсутствием возможности создания вложенных ограничений в составе запросов. Однако данное ограничение само по себе не является препятствием для оптимизации ограничений, содержащих вложения, так как существует ряд алгоритмов, позволяющих преобразовывать ограничения, содержащие вложенные подзапросы, исключая вложения [12, 13].

Кроме этого, необходимо отметить следующие обстоятельства. Для некоторых запросов оптимизация путем исключения плеоназмов может сократить число задействованных в нем отношений, обращение к которым присутствует в избыточном конъюнкте, а также присутствие атрибутов, которые в конечном представлении не предусмотрены запросом.

Примером таких запросов может служить поиск данных сотрудников по нескольким суммам критериев с учетом атрибутов подразделений, в которых они работают. Пример ограничения такого запроса представлен на рис. 2.

```
...WHERE (status.person ='phd' AND salary.person>500) OR
(salary.person>1000 AND person.dep=name.department AND
loc.department='Moscow' AND status.person ='phd')
```

Рисунок 2
Fig. 2

В приведенном примере обращение к отношению department является избыточным, так как задействовано исключительно в рамках логически избыточного конъюнкта, реализация которого не влияет на окончательную выборку.

2. Теоретическая оценка эффективности минимизации запроса

Эффективность указанных алгоритмов определяется влиянием минимизации ограничения на стоимость его обработки. Опубликованные работы предлагают следующую формулу, описывающую затраты на обработку ограничения запроса, представимого в виде ДНФ, с учетом числа и структуры условий (Наиболее ранняя публикация, видимо, [14]):

$$T = \sum_{i=1}^{N}\left(\sum_{j=1}^{M_j} t(C_j^i)\cdot \prod_{l=0}^{l=j-1} s\left(C_l^j \cdot \prod_{l=0}^{l=j-1}(1-s(C_l^j))\right)\right). \quad (1)$$

В формуле (1): t — стоимость обработки условия, N — число конъюнктов в функции, M_j — число условий в конъюнкте j, C_j^i — обозначает условие j конъюнкта i, $s(C_j^i)$ — селективность условия (при $s(C_0^i) = 1$).

В рамках предложенной модели стоимость обработки ограничения (T) может быть уменьшена в результате исключения условия C_j или группы условий $\bigcap_{j=1}^{X} C_j^a$ как плеоназма. Или же в результате его идентификации как условия-дубликата и использования вместо обработки данного условия ранее полученных результатов. Т. е. необходимо идентифицировать пары условий, где $C_j^i \in C_q^i$, или пары конъюнктов a и b, где $\bigcap_{j=1}^{X} C_j^a \in \bigcap_{i=1}^{Y} C_i^b$.

Представленные выше алгоритмы позволяют идентифицировать ряд указанных пар, которые не поддаются идентификации с помощью иных опубликованных на сегодняшний день алгоритмов оптимизации запросов.

Пусть N — множество кортежей в исходной таблице, которые необходимо проверить на соответствие условию C_j^i при обработке запроса, а $s(C_j^i)$ — селективность условия, являющегося частью ограничения, представимого в ДНФ. При этом подразумевается, что в рамках конъюнкта обращение к исходным данным производится один раз при обработке первого конъюнкта, а затем алгоритм обрабатывает промежуточные результаты, полученные при обработке предыдущего условия.

Пусть в конъюнкт i входит M и более условий, часть которых являются избыточными некоему условию C_q^i. Обработка данного конъюнкта в неоптимизированном ограничении потребует проверки следующего числа кортежей в основной таблице и промежуточных результатов (формула (2)):

$$T = s(C_1^i) + \sum_{j=2}^{M} t(C_j^i)\cdot \prod_{l=0}^{l=j-1} s(C_j^i) \quad (2)$$

В реальном выражении избыточные C_q^j условия будут иметь селективность равную 1 при применении после него. (Существующие алгоритмы оптимизации предполагают продвижение условий с наименьшей селективностью и стоимостью к началу конъюнкта, чтобы удешевить обработку, а селективность C_q^i всегда меньше или равна избыточным выражениям, см., например, [15, 16].)

Таким образом, избыточные затраты при обработке плеоназмов в составе конъюнкта могут быть определены формулой (3):

$$T_{\text{избыточное}} = \sum_{p}\left(t(C_p^i)\cdot \prod_{l=0}^{q} s(C_l^i)\right), \quad (3)$$

где p – число, принадлежащее множеству Q, а Q — множество чисел-номеров условий конъюнкта, избыточных относительно C_q^i.

Эффективность оптимизации ограничения, завершающейся исключением избыточного конъюнкта, отличается с учетом вида содержащейся избыточности.

В случае конъюнкта, удаляемого простым поглощением, избыточные затраты на обработку относительно невелики, так как требуется обработка лишь одного

условия, чтобы установить, что исходное множество данных больше не содержит кортежей, отвечающих данному конъюнкту, избыточному относительно обработанного ранее (если СУБД не допускает повторный выбор одних и тех же кортежей в рамках обработки одного ограничения).

Таким образом, избыточной является обработка одного условия избыточного конъюнкта, если изначально обработка происходит оптимальным образом, т.е. начинается с проверки соответствия условий, однокоренных обработанным ранее. В этом случае избыточность определяется формулой (4):

$$T_{\text{избыточное}} = t(C_j^i) \cdot \prod_{l=0}^{l=j-1} \Big(1 - s(C^j)\Big). \tag{4}$$

Избыточность, устраняемая путем «склейки» конъюнктов на основе обнаружения общих импликант, равна общей стоимости обработки избыточного конъюнкта (формула (5)):

$$T_{\text{избыточное}} = \sum_{j-1}^{M_i} t(C_j^i) \cdot \prod_{l=0}^{l=j-1} s(C_l^i). \tag{5}$$

Третий вид оптимизации, разработанный нами на основе алгоритма минимизации систем линейных неравенств, позволяет удалять оба описанных выше вида избыточности, а также путем анализа ограничения идентифицировать случаи, когда они являются тождественно ложными или тождественно истинными. Таким образом, в части возможных запросов оптимизирующее воздействие минимизации ограничения — несомненно. Формулы (3)—(5) позволяют оценить затраты на обработку избыточных условий, устранимых при использовании представленных алгоритмов.

3. Экспериментальная оценка эффективности минимизации

В данном разделе мы описываем проведение тестирования и его результаты. Оно осуществляется с помощью реализации ряда запросов к описанной выше базе данных в среде Oracle Database 10g.

Все запросы были направлены на выборку равного числа атрибутов из таблиц одинаковой размерности и отличались исключительно составом ограничения. Каждый из них содержал те или иные виды плеоназмов. В запросах не используются вложенные подзапросы. Ограничение состояло из множества условий, представленных в дизъюнктивной нормальной форме.

Порядок эксперимента имел следующий вид:

- реализация запроса в первичной форме (время выполнения T_1);
- выполнение программы оптимизации запроса (T_{opt});
- реализация оптимизированного запроса (T_2).

Эффективность алгоритма определялась соотношением

$$I_{eff} = (T_2 + T_{opt})/T_1, \tag{6}$$

где I_{eff} — коэффициент эффективности оптимизации.

Кроме того, целью эксперимента является оценка взаимосвязи числа условий в составе запроса со скоростью его обработки, для чего также проводится учет двух других показателей:

- число условий в составе запроса до оптимизации (A_1);
- число условий в составе запроса после оптимизации (A_2).

Результаты тестов представлены в таблице 1

Таблица 1. Результаты тестов (время указывается в секундах)

	T_1	T_{opt}	T_2	I_{eff}	A_1	A_2
1	0,3	0,00008	0,1	0,3336	3	1
2	0,5	0,00011	0,1	0,2002	6	2
3	0,8	0,00008	0,15	0,1876	5	2
4	0,4	0,00010	0,2	0,5003	7	4
5	0,8	0,00008	0,2	0,2501	7	3
6	0,14	0,00007	0,03	0,2148	5	3
7	0,06	0,00008	0,01	0,1680	4	3
8	0,02	0,00009	0,00(9)*	0,4545	9	6
9	0,23	0,00008	0,02	0,0873	3	1
10	0,15	0,00008	0,08	0,5339	4	2

Примечание: (*)Менее 0,01 секунды, приводится максимально возможное значение из-за погрешности измерения.

Коэффициент эффективности оптимизации демонстрирует возможность сокращения стоимости обработки запроса на 47%—84%. Причем временные затраты на собственно оптимизацию запроса пренебрежимо малы в сравнении с затратами на его обработку. Проведенные тесты показывают, что они в 10^3—10^4 раз меньше, чем временные затраты системы на выполнение запроса.

Таким образом, экспериментально подтвержден оптимизирующий эффект минимизации запроса, что определяется:

- пренебрежимо малыми затратами на выполнение минимизирующих алгоритмов относительно общей стоимости обработки запроса;
- обнаруженным оптимизирующим эффектом минимизации запросов, содержащих избыточные условия.

4. Примеры возникновения избыточности, устраняемой путем минимизации

Вопрос о реальном распространении неоптимальных сложных запросов, включающих множество конъюнктов условий, в реальной работе современных БД и, следовательно, практическом значении представленных алгоритмов является дискуссионным. Зачастую они предполагают логическую ошибку пользователя при составлении сложных запросов, включающих множество конъюнктов условий, и указание избыточных или логически противоречивых критериев.

Для многих реальных СУБД такого рода запросы нетипичны, но существует ряд исключений. Для работы многих систем все же характерны задачи выделения множеств сущностей, отвечающих одной из сумм представленных критериев.

Подобные запросы на практике встречаются в работе систем, предназначенных для хранения и обработки персональных данных большого числа индивидов. В частности, внутренние запросы в системах онлайн рекламы представимы в виде сложных булевых функций, содержащих множество условий (характеристик пользователей). Также наличие подобных запросов характерно для систем «издатель–подписчик» (publish&subscribe systems), где при формировании перечня адресатов выполняются запросы, содержащие многосоставные ограничения, представимые в виде булевых функций в ДНФ [17].

В существующей литературе описаны примеры реальных запросов к СУБД, содержащих большое число плеоназмов, оказывающих значительное влияние на размеры запроса. В частности, в одной из работ в качестве примера упомянут неискусственный запрос, в исходном виде включавший 98 условий, а в результате преобразования выражения в конъюнктивную нормальную форму и тривиальных упрощающих преобразований сократившийся до 12 [18].

Наконец, следует оговориться, что не все ограничения, поддающиеся минимизации с использованием представленных алгоритмов, могут быть определены пользователем как избыточные без проведения математического анализа. Это в особенности касается ограничений, представимых в виде конъюнктов линейных неравенств с использованием 2 и более атрибутов.

5. Влияние минимизации на общий ход оптимизации запроса

Разработка алгоритмов минимизации ограничений запросов, представимых в виде ДНФ, содержащих 2 и более конъюнктов, представляет и общетеоретический интерес. Это связано в первую очередь с задачами оптимизации серий запросов путем идентификации в них общих подвыражений (см., например, [19, 20]).

В контексте данной задачи группа запросов может рассматриваться как подзапросы, объединенные дизъюнкцией. В этом случае для запросов, имеющих общие подмножества элементов в части select и from, актуальна задача поиска общих групп условий, входящих в ограничение. В данном случае ограничения оптимизируемых запросов могут быть рассмотрены как единое ограничение, составные части которого объединены дизъюнкцией.

В множестве независимых запросов, отобранных по признаку наличия общих отношений, вероятно наличие повторяющих и избыточных условий, устранение которых путем сокращения позволяет ускорить обработку. При решении этой задачи возникает необходимость идентификации общих подвыражений в составе ограничения, решению такой задачи может способствовать приведение запроса к единому наиболее лаконичному представлению с использованием всех доступных алгоритмов минимизации.

Без учета фактической оптимальности полученного минимального представления оно способствует решению двух задач:

- упрощение процедуры автоматической идентификации тождества двух выражений;
- сокращение числа возможных планов представления сложных запросов, что упрощает их оптимизацию.

Запрос к СУБД может быть представлен в виде B-дерева, где листья — отношения БД, а узловые элементы содержат операции и условия. При этом выполнение запроса характеризуется движением от листьев вверх по графу с целью последовательной реализации запроса. Задача оптимизации характеризуется выбором из множества существующих представлений запроса лучшего с точки зрения стоимости обработки представления. Учитывая значительное число возможных конечных представлений дерева, актуальной является задача сокращения числа представлений путем исключения однозначно неоптимальных.

В ряде работ в качество параметра, определяющего число вариантов представления графов, рассматривалось число отношений без учета числа операций. Это определялось наличием «тривиального» правила оптимальности представления запроса, сформулированного, в частности, у Ионнидиса, согласно которому только один операнд отношения соединения может являться промежуточным результатом. Выполнение этого требования приводит к тому, что число вариантов представления дерева запроса определяется функцией $O(2^N)$, где N — число отношений [21].

Однако на текущий момент существуют работы, доказывающие, что реальные запросы могут иметь более сложную структуру, включающую множественные соединения одних и тех же отношений, что ведет к влиянию числа операций на число представлений графа. Кроме того, доказывается, что в случае, если запрос представим в виде гиперграфа, возникает задача его упрощения путем сокращения числа соединений, чтобы сократить затраты на поиск оптимального представления. Причем в ряде случаев без предварительного упрощения эта задача рассматривается как не поддающаяся решению [22, 23].

Таким образом, можно выделить следующие способы применения представленных алгоритмов:

- оптимизирующий эффект минимизации ограничения запроса;
- минимизация ограничения как одно из средств приведения ограничений к универсальному виду в целях определения их тождества;
- сокращение числа представлений запроса путем минимизации числа содержащихся в нем условий, что сокращает расходы на оптимизацию запроса.

Заключение

В данной статье проблема влияния минимизации запроса путем исключения из ограничения избыточных условий изучена на примере ранее разработанных алгоритмов лексической оптимизации.

Теоретически доказано, что минимизирующие преобразования ведут к сокращению времени выполнения запроса, и приведены теоретически обоснованные оценки экономии (формулы (3)—(5)).

Также приведены экспериментальные результаты сравнения стоимости обработки исходных и минимизированных запросов, содержащих избыточные условия. Доказано, что реализация алгоритмов поиска избыточности имеет стоимость, в 10^3—10^4 раз уступающую обработке запроса, а ликвидация избыточности может приводить к существенному сокращению времени обработки запроса.

Наконец, в статье рассмотрен контекст минимизации запросов с учетом практики работ в области оптимизации. Показана реальная встречаемость запросов, содержащих избыточности, отвечающие критериям представленных алгоритмов, а также показано косвенное влияние минимизирующих преобразований на процесс обработки запроса.

Список литературы / References

[1] Hall P. A. V., “Optimization of single expressions in a relational data base system”, *IBM Journal of Research and Development*, **20**:3 (1976), 244–257.

[2] Bellamkonda S. at al., “Enhanced Subquery Optimizations in Oracle”, *Proceedings of the 35th international conference on Very large data base*, August, 2009, **2**, 2009, 1368.

[3] “Chapter 7. Optimization”, *MySQL 5.5 Reference Manual.* http://dev.mysql.com/doc/refman/5.5/en/optimization.html.

[4] “PostgreSQL 8.3.3”, postgresql-8.3.3/src/backend/optimizer/util/predtest.c.

[5] “Query Optimization in Oracle Database 10g Release 2. P. 9”.

[6] Seshadri P. at al., “Cost-Based Optimization for Magic: Algebra and Implementation”, *ACM SIGMOD Record*, **25**:2 (1996).

[7] Faber W., Greco G., Leone N., “Sets and their application to data integration”, *Journal of Computer and System Sciences*, **73**:4 (2007).

[8] Mendkovich N., Kuznetcov S., “New Algorithms for Lexical Query Optimization”, *Proceedings of the 31st International Conference on Information Technology Interfaces*, Cavtat/Dubrovnik, June 22–25, 2009 (Zagreb, University of Zagreb), 2009, 187–192.

[9] Кузнецов С.Д., Мендкович Н.А., “Новые алгоритмы лексической оптимизации запросов”, *Моделирование и анализ информационных систем*, **16**:4 (2009), 22–33; [Kuznetsov S.D., Mendkovich N.A., “New algorithms for query modifications”, *Modeling and Analysis of Information Systems*, **16**:4 (2009), 22–33, (in Russian).]

[10] Кузнецов С.Д., Мендкович Н.А., “Оптимизация конъюнктов условий в составе запросов”, *Модел. и анал. информ. систем*, **18**:3 (2011), 144–154; [Kuznetsov S.D., Mendkovich N.A., “Optimization of queries containing conjunctions of conditions”, *Modeling and Analysis of Information Systems*, **18**:3 (2011), 144–154, (in Russian).]

[11] “BNF Grammar for ISO/IEC 9075-2:2003”, http://savage.net.au/SQL/sql-2003-2.bnf.html.

[12] Khaitan P. at al., “Improved query plans for unnesting nested SQL queries”, *Proceedings of 2nd International Conference on Computer Science and its Applications*, December 10–12, South Korea, 2009 (Jeju Island, IEEE), 2009, 147–152.

[13] Muralikrishna M., "Improved unnesting algorithms for join aggregate SQL queries", *Proceedings of the 18th International Conference on Very Large Data Bases*, August 23–27, Vancouver, Canada, 1992 (San Francisco: Morgan Kaufmann Publishers Inc.), 1992, 91–102.

[14] Satoh K. at al., "Local and Global Query Optimization Mechanisms", *Proceedings of 11th International Conference on Very Large Data Bases*, August 21-23, 1985, Stockholm, Sweden (Berlin: Morgan Kaufmann), 1985, 408–409.

[15] Hellerstein J. M., Stonebraker M., "Predicate Migration: Optimizing Queries with Expensive Predicates", *ACM SIGMOD Record*, **22**:2 (1993), 267–276.

[16] Chaudhuri S., "Optimization of queries with user-defined predicates", *Journal ACM Transactions on Database Systems (TODS)*, **24**:2 (1999), 177–228.

[17] Fontoura M. at al., "Efficiently Evaluating Complex Boolean Expressions", *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (New York: ACM), 2010, 3–4.

[18] Vorwerk K., Paulley G. N., "On Implicate Discovery and Query Optimization", *Proceedings of the International Database Engineering and Applications Symposium*, IDEAS 2002. July 17-19, 2002, Edmonton, Canada (Los Alamitos: Computer Society), 2002, 2–12.

[19] Roy P. at al., "Efficient and extensible algorithms for multi-query optimization", *Proceedings of the 2000 ACM SIGMOD International Conference on Management of data* (ACM New York, NY, USA), 2000, 249–260.

[20] Dalvia N. N. at al., "Pipelining in multi-query optimization", *Journal of Computer and System Science*, **66**:4 (2003).

[21] Ioannidis Y. E., "Query Optimization", *ACM Computing Surveys (CSUR)*, **28**:1 (1996), 121–123.

[22] Neumann T., "Query Simplification: Graceful Degradation for Join-Order Optimization", *SIGMOD'09*, June 29–July 2, 2009, Providence, Rhode Island, USA, 2009, 405–406.

[23] Moerkotte G., Neumann T., "Dynamic programming strikes back", *Proceedings of SIGMOD Conference 2008*, June 9–12, 2008, Vancouver, BC, Canada, 2009.

Mendkovich N., "On the Effectiveness of the Minimization Approach to the Query Optimization", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 153–163.

DOI: 10.18255/1818-1015-2016-2-153-163

Abstract. A standard problem of DBMSs usage is a lack of efficiency and high cost of the access to the stored data. The acceptable level of system performance may be achieved by query optimization technics that determine the most efficient way to execute a given query by its modification and considering possible query execution plans. The goal of this paper is to prove the efficiency of the query minimization algorithms based on minimization of the query restriction by elimination of the redundant conditions. The paper represents minimization algorithms based on the mathematical transformations, which detect and remove redundant conditions from query restriction to simplify it. It includes minimization algorithms based on "condition absorption", prime implicants, and a set of linear inequalities minimization technics. The paper also includes theoretical justification of the efficiency of minimization approach to the query optimization based on restriction simplification. We also observe experimental results of the implementation of these optimization techniques and their influence on the query processing speed. In the end, we represent an observation of the query minimization impact on the whole optimization process

Keywords: query optimization, lexical optimization

On the authors: Mendkovich Nikita, orcid.org/0000-0003-1694-7947, FREENet Group Ltd., engineer, Leninsky Ave. 47, Moscow, 119991 Russia, e-mail: mend@f-group.ru

Моделирование и анализ информационных систем. Т. 23, №2 (2016), с. 164–172
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 164–172

DOI: 10.18255/1818-1015-2016-2-164-172

УДК 512.71

О группе Брауэра арифметической модели многообразия над глобальным полем положительной характеристики

Прохорова Т. В.

получена 13 февраля 2016

Аннотация. Пусть V – гладкое проективное многообразие над глобальным полем $k = \kappa(C)$ рациональных функций на гладкой проективной кривой C над конечным полем $\mathbb{F}_q$ характеристики p. Предположим, что существует проективный плоский $\mathbb{F}_q$-морфизм $\pi : X \to C$, где X – гладкое проективное многообразие, общий схемный слой морфизма π изоморфен многообразию V (мы называем морфизм $\pi : X \to C$ *арифметической моделью многообразия* V).

М. Артин высказал гипотезу о конечности группы Брауэра $\mathrm{Br}(X)$, классифицирующей пучки алгебр Адзумаи на X по модулю подобия. Хорошо известно, что группа $\mathrm{Br}(X)$ содержится в когомологической группе Брауэра

$$\mathrm{Br}'(X) = H^2_{et}(X, \mathbb{G}_m).$$

По определению, $\mathrm{non}-p$ – компонента когомологической группы Брауэра $\mathrm{Br}'(X)$ совпадает с прямой суммой l-примарных компонент группы $\mathrm{Br}'(X)$ по всем простым числам l, отличным от характеристики p.

Известно, что структура k-многообразия на V задает канонический морфизм групп $\mathrm{Br}(k) \to \mathrm{Br}'(V)$.

В работе доказана конечность $\mathrm{non}-p$ – компоненты когомологической группы Брауэра $\mathrm{Br}'(X)$ многообразия X при условии, что факторгруппа

$$[\mathrm{Br}'(V)/\operatorname{Im}[\mathrm{Br}(k) \to \mathrm{Br}'(V)]](\mathrm{non}-p)$$

конечная.

В частности, если V – поверхность типа К 3 (другими словами, V – гладкая проективная односвязная поверхность над полем k и канонический класс поверхности V тривиален: $\Omega^2_V = \mathcal{O}_V$), причем характеристика основного поля $p > 2$, то по теореме Скоробогатова — Зархина факторгруппа $[\mathrm{Br}'(V)/\operatorname{Im}[\mathrm{Br}(k) \to \mathrm{Br}'(V)]](\mathrm{non}-p)$ конечна, так что в этом случае группы $\mathrm{Br}'(X)(\mathrm{non}-p)$ и $\mathrm{Br}(X)(\mathrm{non}-p)$ конечные.

Ключевые слова: группа Брауэра, арифметическая модель, К 3–поверхность

Для цитирования: Прохорова Т. В., "О группе Брауэра арифметической модели многообразия над глобальным полем положительной характеристики", *Моделирование и анализ информационных систем*, **23**:2 (2016), 164–172.

Об авторах: Прохорова Татьяна Вячеславовна, orcid.org/0000-0002-6883-2087, канд. физ.-мат. наук, Владимирский государственный университет им. А.Г. и Н.Г. Столетовых, ул. Горького, 87, г. Владимир, 600000, Россия,
e-mail: tvprokhorova@mail.ru

Введение

Пусть V – гладкое проективное многообразие над глобальным полем $k = \kappa(C)$ рациональных функций на гладкой проективной кривой C над конечным полем $\mathbb{F}_q$ характеристики p. Предположим, что существует проективный плоский $\mathbb{F}_q$-морфизм $\pi : X \to C$, где X – гладкое проективное многообразие, общий схемный слой морфизма π изоморфен многообразию V (мы называем морфизм $\pi : X \to C$ *арифметической моделью многообразия* V).

М. Артин высказал гипотезу о конечности группы Брауэра $\mathrm{Br}(X)$, классифицирующей пучки алгебр Адзумаи на X по модулю подобия. Известно, что группа $\mathrm{Br}(X)$ содержится в когомологической группе Брауэра

$$\mathrm{Br}'(X) = H^2_{et}(X, \mathbb{G}_m).$$

По определению, $\mathrm{non}-p$ – компонента когомологической группы Брауэра $\mathrm{Br}'(X)$ совпадает с прямой суммой l-примарных компонент группы $\mathrm{Br}'(X)$ по всем простым числам l, отличным от характеристики p.

Известно, что структура k-многообразия на V задает канонический морфизм групп $\mathrm{Br}(k) \to \mathrm{Br}'(V)$.

В работе доказана конечность $\mathrm{non}-p$ – компоненты когомологической группы Брауэра $\mathrm{Br}'(X)$ многообразия X при условии, что факторгруппа

$$[\mathrm{Br}'(V)/\operatorname{Im}[\mathrm{Br}(k) \to \mathrm{Br}'(V)]](\mathrm{non}-p)$$

конечная.

В частности, если V – поверхность типа K 3 (другими словами, V – гладкая проективная односвязная поверхность над полем k и канонический класс поверхности V тривиален: $\Omega^2_V = \mathcal{O}_V$), причем характеристика основного поля $p > 2$, то по теореме Скоробогатова — Зархина [1, теорема 1.3] факторгруппа $[\mathrm{Br}'(V)/\operatorname{Im}[\mathrm{Br}(k) \to \mathrm{Br}'(V)]](\mathrm{non}-p)$ конечна, так что в этом случае группы $\mathrm{Br}'(X)(\mathrm{non}-p)$ и $\mathrm{Br}(X)(\mathrm{non}-p)$ конечные.

Аналогичный результат доказан С. Г. Танкеевым для арифметической модели K 3–поверхности над числовым полем [2].

Автор благодарит С. Г. Танкеева за ценные советы.

Основные результаты

Теорема 1. *Пусть $\pi : X \to C$ – сюръективный морфизм гладких проективных многообразий над конечным полем $\mathbb{F}_q$ характеристики p, C – кривая, общий схемный слой морфизма π является гладким многообразием V над полем $k = \kappa(C)$. Если группа*

$$[\mathrm{Br}'(V)/\operatorname{Im}[\mathrm{Br}(k) \to \mathrm{Br}'(V)]](\mathrm{non}-p)$$

конечная, то группа $\mathrm{Br}'(X)(\mathrm{non}-p)$ конечная.

Доказательство. Будем обозначать через $\kappa(y)$ поле вычетов точки $y \in X$.

Пусть $i_y : \operatorname{Spec}\kappa(y) \to X$ – каноническое вложение $y \in X$ и

$$\operatorname{Div}_X^{\mathrm{vert}} = \bigoplus_{\substack{y\in X\setminus V\\ \operatorname{codim}_X(y)=1}} i_{y*}\mathbb{Z}$$

– пучок вертикальных дивизоров Картье. Существует точная последовательность пучков (в этальной топологии схемы X)

$$0 \to \mathbb{G}_{m,X} \to h_*\mathbb{G}_{m,V} \to \operatorname{Div}_X^{\mathrm{vert}} \to 0, \tag{1}$$

где $h : V \hookrightarrow X$ – вложение общего схемного слоя морфизма π [3, последняя формула на с. 637].

Мы имеем

$$\operatorname{Div}_X = \bigoplus_{\substack{y\in X\\ \operatorname{codim}_X(y)=1}} i_{y*}\mathbb{Z} = \left(\bigoplus_{\substack{y\in V\\ \operatorname{codim}_X(y)=1}} i_{y*}\mathbb{Z}\right) \bigoplus \operatorname{Div}_X^{\mathrm{vert}}.$$

Хорошо известно, что $H^1(X, \operatorname{Div}_X) = 0$ [4, гл. 3, § 2, пример 2.22]. Следовательно, $H^1(X, \operatorname{Div}_X^{\mathrm{vert}}) = 0$. Значит, (1) даёт точную последовательность

$$\begin{aligned} 0 \to H^2(X, \mathbb{G}_m) \to H^2(X, h_*\mathbb{G}_{m,V}) \to H^2(X, \operatorname{Div}_X^{\mathrm{vert}})\\ \to H^3(X, \mathbb{G}_m) \to H^3(X, h_*\mathbb{G}_{m,V}) \to H^3(X, \operatorname{Div}_X^{\mathrm{vert}}). \end{aligned} \tag{2}$$

Спектральная последовательность Лере

$$E_2^{p,q} = H^p(X, R^q h_*\mathbb{G}_{m,V}) \Rightarrow H^{p+q}(V, \mathbb{G}_{m,V})$$

даёт точную последовательность

$$0 \to E_2^{1,0} \to E^1 \to E_2^{0,1} \xrightarrow{d_2^{0,1}} E_2^{2,0} \to E_1^2 \to E_2^{1,1} \to E_2^{3,0},$$

где $E_1^2 = \operatorname{Ker}[E^2 \to E_2^{0,2}]$ [4, приложение B]. Следовательно, мы имеем точную последовательность

$$\begin{aligned} 0 \to H^1(X, h_*\mathbb{G}_{m,V}) \to H^1(V, \mathbb{G}_{m,V})\\ \to H^0(X, R^1h_*\mathbb{G}_{m,V}) \xrightarrow{d_2^{0,1}} H^2(X, h_*\mathbb{G}_{m,V})\\ \to \operatorname{Ker}[H^2(V, \mathbb{G}_{m,V}) \to H^0(X, R^2h_*\mathbb{G}_{m,V})]\\ \to H^1(X, R^1h_*\mathbb{G}_{m,V}). \end{aligned} \tag{3}$$

С другой стороны, $R^1h_*\mathbb{G}_{m,V} = 0$ в силу аргументов п. (b) доказательства леммы 4.4.1 в [3]. Поэтому (3) даёт изоморфизм

$$H^2(X, h_*\mathbb{G}_{m,V}) \xrightarrow{\sim} \operatorname{Ker}[\operatorname{Br}'(V) \to H^0(X, R^2h_*\mathbb{G}_{m,V})]$$

и (2) даёт точную последовательность

$$0 \to \operatorname{Br}'(X) \to \operatorname{Ker}[\operatorname{Br}'(V) \to H^0(X, R^2h_*\mathbb{G}_{m,V})] \to H^2(X, \operatorname{Div}_X^{\mathrm{vert}}). \tag{4}$$

В дальнейшем мы обозначаем через η общую схемную точку кривой C и через $\kappa(y)^{\mathrm{s}}$ сепарабельное замыкание поля вычетов $\kappa(y)$ в алгебраическом замыкании $\overline{\kappa(y)}$. Существует каноническое отображение $\mathrm{Br}(k) \to \mathrm{Br}'(V)$, индуцированное структурным морфизмом $V \to \operatorname{Spec} k$. С другой стороны, существует каноническая точная последовательность

$$\begin{aligned} 0 \to \mathrm{Br}(C) \to \mathrm{Br}(k) \to \bigoplus_{\substack{v \in C \\ v \neq \eta}} \mathrm{Hom}_{\mathrm{cont}}(\mathrm{Gal}(\kappa(v)^{\mathrm{s}}/\kappa(v)), \mathbb{Q}/\mathbb{Z}) \\ \to H^3(C, \mathbb{G}_m) \to H^3(\mathrm{Spec}(k), \mathbb{G}_m) \end{aligned}$$

[4, гл. III, § 2, пример 2.22(a)], где $\mathrm{Hom}_{\mathrm{cont}}$ обозначает группу непрерывных гомоморфизмов. В нашем случае C – полная гладкая алгебраическая кривая над конечным полем F_q, поэтому $\mathrm{Br}(C) = 0$ и $H^3(C, \mathbb{G}_m) = \mathbb{Q}/\mathbb{Z}$ [4, гл. III, § 2, пример 2.22(g)]. Мы приходим к хорошо известной точной последовательности глобальной теории полей классов

$$0 \to \mathrm{Br}(k) \to \bigoplus_{\substack{v \in C \\ v \neq \eta}} \mathrm{Hom}_{\mathrm{cont}}(\mathrm{Gal}(\kappa(v)^{\mathrm{s}}/\kappa(v)), \mathbb{Q}/\mathbb{Z}) \to \mathbb{Q}/\mathbb{Z}. \tag{5}$$

Эта точная последовательность имеет вид

$$0 \to \mathrm{Br}(k) \to \bigoplus_{\substack{v \in C \\ v \neq \eta}} \mathbb{Q}/\mathbb{Z} \to \mathbb{Q}/\mathbb{Z}$$

[4, гл. III, § 2, пример 2.22(e)].

Для замкнутой точки $v \in C$ рассмотрим каноническое вложение $i_v : \operatorname{Spec} \kappa(v) \hookrightarrow C$. Спектральная последовательность Лере

$$E_2^{p,q} = H^p(C, R^q i_{v*}\mathbb{Z}) \to H^{p+q}(\operatorname{Spec} \kappa(v), \mathbb{Z})$$

дает точную последовательность

$$0 \to E_2^{1,0} \to E^1 \to E_2^{0,1} \xrightarrow{d_2^{0,1}} E_2^{2,0} \to E_1^2 \to E_2^{1,1},$$

где $E_1^2 = \mathrm{Ker}[E^2 \to E_2^{0,2}]$ [4, приложение B]; следовательно, мы имеем точную последовательность

$$\begin{aligned} 0 \to H^1(C, i_{v*}\mathbb{Z}) \to H^1(\operatorname{Spec} \kappa(v), \mathbb{Z}) \\ \to H^0(C, R^1 i_{v*}\mathbb{Z}) \xrightarrow{d_2^{0,1}} H^2(C, i_{v*}\mathbb{Z}) \\ \to \mathrm{Ker}[H^2(\operatorname{Spec} \kappa(v), \mathbb{Z}) \to H^0(C, R^2 i_{v*}\mathbb{Z})] \\ \to H^1(C, R^1 i_{v*}\mathbb{Z}). \end{aligned} \tag{6}$$

Хорошо известно, что $R^q i_{v*}\mathbb{Z} = 0$ для всех $q > 0$ [4, гл. III, § 2, пример 2.22(a)]; поэтому (6) даёт изоморфизм

$$H^2(C, i_{v*}\mathbb{Z}) \xrightarrow{\sim} H^2(\operatorname{Spec} \kappa(v), \mathbb{Z}).$$

С другой стороны,

$$H^2(\operatorname{Spec}\kappa(v),\mathbb{Z}) \xrightarrow{\sim} \operatorname{Hom}_{\mathrm{cont}}(\operatorname{Gal}(\kappa(v)^{\mathrm{s}}/\kappa(v)),\mathbb{Q}/\mathbb{Z})$$

[4, гл. III, § 2, пример 2.22]. Следовательно,

$$\begin{aligned} H^2(C,\operatorname{Div}_C) &\stackrel{\mathrm{def}}{=} H^2\left(C,\bigoplus_{\substack{v\in C\\ v\neq\eta}} i_{v*}\mathbb{Z}\right)\\ &= \bigoplus_{\substack{v\in C\\ v\neq\eta}} H^2(C,i_{v*}\mathbb{Z})\\ &= \bigoplus_{\substack{v\in C\\ v\neq\eta}} \operatorname{Hom}_{\mathrm{cont}}(\operatorname{Gal}(\kappa(v)^{\mathrm{s}}/\kappa(v)),\mathbb{Q}/\mathbb{Z}) \end{aligned}$$

[4, гл. III, § 2, пример 2.22(a)]. Поэтому (5) даёт точную последовательность

$$0 \to \operatorname{Br}(k) \to H^2(C,\operatorname{Div}_C). \tag{7}$$

С другой стороны, можно определить канонический морфизм

$$\pi^* : H^*(C,\operatorname{Div}_C) \to H^*(X,\pi^*(\operatorname{Div}_C)),$$

используя резольвенту Годемана [4, гл. III, § 1, замечание 1.20, (c)]

$$0 \to \operatorname{Div}_C \to \mathcal{C}^0(C,\operatorname{Div}_C) \to \mathcal{C}^1(C,\operatorname{Div}_C) \to \dots,$$

точность функтора π^* на категории пучков для этальной топологии [4, гл. II, § 2, предложение 2.6; начало § 3] и общую конструкцию отображения

$$\pi^* : H^*(C,\operatorname{Div}_C) \to H^*(X,\pi^*(\operatorname{Div}_C))$$

по Годеману [5, гл. II, § 4.16] (для построения этого морфизма не обязательно прибегать к резольвенте Годемана; для любого пучка $\mathcal{F}$ имеется канонический морфизм $\mathcal{F} \to R\pi_*\pi^*(\mathcal{F})$ в производной категории этальных пучков на C, который даёт отображение $H^*(C,\mathcal{F}) \to H^*(C,R\pi_*\pi^*(\mathcal{F})) = H^*(X,\pi^*(\mathcal{F}))$).

Каноническое вложение $\pi^*(\operatorname{Div}_C) \hookrightarrow \operatorname{Div}_X^{\mathrm{vert}}$ даёт каноническое отображение

$$H^2(X,\pi^*(\operatorname{Div}_C)) \to H^2(X,\operatorname{Div}_X^{\mathrm{vert}}).$$

Следовательно, имеются канонические морфизмы

$$\varphi : \operatorname{Br}(k) \to \operatorname{Br}'(V), \tag{8}$$

$$H^2(C,\operatorname{Div}_C) \to H^2(X,\operatorname{Div}_X^{\mathrm{vert}}) \tag{9}$$

Пусть $B = \operatorname{Ker}[\operatorname{Br}'(V) \to H^0(X,R^2h_*\mathbb{G}_{m,V})]$. Очевидно, что (4), (7) – (9) дают коммутативную диаграмму с точными строками

$$\begin{array}{ccccccc} 0 & \longrightarrow & \operatorname{Br}'(X) & \longrightarrow & B & \longrightarrow & H^2(X,\operatorname{Div}_X^{\mathrm{vert}})\\ & & \uparrow & & \varphi\uparrow & & \uparrow\\ & & 0 & \longrightarrow & \operatorname{Br}(k)\cap\varphi^{-1}(B) & \longrightarrow & H^2(C,\operatorname{Div}_C) \end{array} \tag{10}$$

Поскольку группа $[\mathrm{Br}'(V)/\varphi(\mathrm{Br}(k))](\mathrm{non}-p)$ конечная по условию теоремы, то мы видим, что группа

$$B/[\varphi(\mathrm{Br}(k))\cap B](\mathrm{non}-p) = \mathrm{Coker}[\mathrm{Br}(k)\cap\varphi^{-1}(B)\xrightarrow{\varphi} B](\mathrm{non}-p)$$

конечная. Поэтому достаточно показать, что пересечение группы $\mathrm{Br}'(X)(\mathrm{non}-p)$ с образом морфизма групп

$$[\mathrm{Br}(k)\cap\varphi^{-1}(B)](\mathrm{non}-p)\xrightarrow{\varphi} B(\mathrm{non}-p)$$

конечно. В силу коммутативности диаграммы (10) достаточно доказать конечность ядра канонического отображения $H^2(C,\mathrm{Div}_C)\to H^2(X,\mathrm{Div}_X^{\mathrm{vert}})$.

Очевидно, что ядро отображения $H^2(C,\mathrm{Div}_C)\to H^2(X,\mathrm{Div}_X^{\mathrm{vert}})$ является прямой суммой по всем замкнутым точкам v кривой C ядер отображений

$$\pi^*: H^2(\mathrm{Spec}\,\kappa(v),\mathbb{Z})\to\bigoplus_D H^2(\mathrm{Spec}\,\kappa(D),\mathbb{Z}), \tag{11}$$

которые также можно записать в виде

$$\mathrm{Hom}_{\mathrm{cont}}(\mathrm{Gal}(\overline{\kappa(v)}/\kappa(v)),\mathbb{Q}/\mathbb{Z})\to\bigoplus_D \mathrm{Hom}_{\mathrm{cont}}(\mathrm{Gal}(\kappa(D)^{\mathrm{s}}/\kappa(D)),\mathbb{Q}/\mathbb{Z}).$$

Здесь D пробегает неприводимые компоненты слоя морфизма $X\to C$ над точкой v, поле $\kappa(D)^{\mathrm{s}}$ является сепарабельным замыканием поля $\kappa(D)$ рациональных функций на D.

Пусть κ_D – целое замыкание $\kappa(v)$ в поле $\kappa(D)$. Тогда $\kappa(v)\subset\kappa(D)$ – такое расширение полей, что многообразие D является геометрически целым над κ_D. Мы имеем композицию канонических морфизмов полей

$$\kappa(D)=\kappa_D(D)\leftarrow\kappa_D\leftarrow\kappa(v),$$

индицирующую композицию отображений

$$H^2(\mathrm{Spec}\,\kappa(v),\mathbb{Z})\to H^2(\mathrm{Spec}\,\kappa_D,\mathbb{Z})\to H^2(\mathrm{Spec}\,\kappa_D(D),\mathbb{Z})=H^2(\mathrm{Spec}\,\kappa(D),\mathbb{Z}).$$

Очевидно, что композиция

$$H^2(\mathrm{Spec}\,\kappa(v),\mathbb{Z})\xrightarrow{\pi^*}\bigoplus_D H^2(\mathrm{Spec}\,\kappa(D),\mathbb{Z})\to H^2(\mathrm{Spec}\,\kappa(D),\mathbb{Z}) \tag{12}$$

отображения π^* в (11) и канонической проекции

$$\bigoplus_D H^2(\mathrm{Spec}\,\kappa(D),\mathbb{Z})\to H^2(\mathrm{Spec}\,\kappa(D),\mathbb{Z})$$

является композицией канонических отображений

$$H^2(\mathrm{Spec}\,\kappa(v),\mathbb{Z})\to H^2(\mathrm{Spec}\,\kappa_D,\mathbb{Z})\to H^2(\mathrm{Spec}\,\kappa_D(D),\mathbb{Z})=H^2(\mathrm{Spec}\,\kappa(D),\mathbb{Z})$$

$$\xrightarrow{x\mapsto \mathrm{mult}_{\pi^{-1}(v)}(D)\cdot x} H^2(\operatorname{Spec}\kappa(D),\mathbb{Z}), \tag{13}$$

где $\mathrm{mult}_{\pi^{-1}(v)}(D)$ – кратность подмногообразия D в слое $X_v=\pi^{-1}(v)$.

Пусть $\mathbb{F}_q$ – конечное поле порядка q и пусть $W\hookrightarrow\mathbb{P}^n_{\mathbb{F}_q}$ – геометрически неприводимое проективное подмногообразие размерности r и степени d. Хорошо известный результат Ленга и Вейля [6, теорема 1] показывает, что

$$|\operatorname{Card}(W(\mathbb{F}_q))-q^r|\le(d-1)(d-2)\,q^{r-1/2}+c(n,d,r)\,q^{r-1}$$

для константы $c(n,d,r)>0$, зависящей только от n,d и r.

В силу оценки Ленга–Вейля и леммы Гензеля многообразие V имеет точки почти во всех пополнениях глобального поля k. Другими словами, имеется такое конечное множество S замкнутых точек кривой C, что для всех $v\notin S$ морфизм $X\to C$, ограниченный на спектр локального кольца $\mathcal{O}_v$, имеет сечение $\theta:\operatorname{Spec}\mathcal{O}_v\to X\times_C\operatorname{Spec}\mathcal{O}_v$. В силу следствия 2.2 в [7], точка $\theta(v)$ является регулярной точкой схемного слоя $\pi^{-1}(v)$. Поэтому $\mathcal{O}_{\pi^{-1}(v),\theta(v)}$ – регулярное локальное кольцо и слой $\pi^{-1}(v)$ аналитически неприводим в точке $\theta(v)$ [8, гл. 11, замечание 1 к предложению 11.24]; в частности, $\theta(\operatorname{Spec}\mathcal{O}_v)$ пересекает *одну* неприводимую компоненту схемного слоя $\pi^{-1}(v)$ (то, что сечение на регулярной модели пересекает в точности одну неприводимую компоненту кратности 1 слоя, доказано также в [9, лемма 1.1, b]). Поэтому любой вертикальный дивизор Картье D с носителем в слое $\pi^{-1}(v)$ может быть единственным образом записан в виде

$$D=n_0\cdot\pi^{-1}(v)+\sum_i n_i\cdot D_i,$$

где $n_j\in\mathbb{Z}$ и D_i – такие неприводимые компоненты слоя $\pi^{-1}(v)$, что выполнено равенство $D_i\cap\theta(\operatorname{Spec}\mathcal{O}_v)=\varnothing$. Следовательно, получаем разложение пучков

$$\operatorname{Div}^{\mathrm{vert}}_{X\times_C\operatorname{Spec}\mathcal{O}_v}=\pi^*(\operatorname{Div}_{\operatorname{Spec}\mathcal{O}_v})\oplus\left(\bigoplus_{\substack{y\in X\times_C\operatorname{Spec}\mathcal{O}_v\setminus V,\ \operatorname{codim}_{X\times_C\operatorname{Spec}\mathcal{O}_v}(y)=1,\\ \overline{\{y\}}\cap\theta(\operatorname{Spec}\mathcal{O}_v)=\varnothing}} i_{y*}\mathbb{Z}\right) \tag{14}$$

Соотношение $\pi\circ\theta=\mathrm{id}_{\operatorname{Spec}\mathcal{O}_v}$ показывает, что композиция

$$\begin{aligned}H^*(\operatorname{Spec}\mathcal{O}_v,\operatorname{Div}_{\operatorname{Spec}\mathcal{O}_v})&\xrightarrow{\pi^*}H^*(X\times_C\operatorname{Spec}\mathcal{O}_v,\pi^*(\operatorname{Div}_{\operatorname{Spec}\mathcal{O}_v}))\\&\xrightarrow{\theta^*}H^*(\operatorname{Spec}\mathcal{O}_v,\operatorname{Div}_{\operatorname{Spec}\mathcal{O}_v})\end{aligned}$$

является тождественным отображением. Поэтому мы получаем вложение

$$H^2(\operatorname{Spec}\mathcal{O}_v,\operatorname{Div}_{\operatorname{Spec}\mathcal{O}_v})\xhookrightarrow{\pi^*}H^2(X\times_C\operatorname{Spec}\mathcal{O}_v,\pi^*(\operatorname{Div}_{\operatorname{Spec}\mathcal{O}_v})).$$

С другой стороны, разложение (14) даёт вложение

$$H^2(X\times_C\operatorname{Spec}\mathcal{O}_v,\pi^*(\operatorname{Div}_{\operatorname{Spec}\mathcal{O}_v}))\hookrightarrow H^2(X\times_C\operatorname{Spec}\mathcal{O}_v,\operatorname{Div}^{\mathrm{vert}}_{X\times_C\operatorname{Spec}\mathcal{O}_v}).$$

Следовательно, для $v\notin S$ ядро отображения (11) тривиально.

Остаётся доказать, что для $v \in S$ ядро композиции (12) конечно.

Для начала заметим, что отображение $H^2(\operatorname{Spec}\kappa_D, \mathbb{Z}) \to H^2(\operatorname{Spec}\kappa(D), \mathbb{Z})$ в композиции (13) инъективно. Действительно, это отображение совпадает с отображением

$$H^1(\operatorname{Gal}(\overline{\kappa_D}/\kappa_D), \mathbb{Q}/\mathbb{Z}) \to H^1(\operatorname{Gal}(\kappa(D)^{\mathrm{s}}/\kappa(D)), \mathbb{Q}/\mathbb{Z}).$$

Оно инъективно, потому что $\kappa(D)$ и любое алгебраическое замыкание поля κ_D линейно разделены, так что имеется канонический изоморфизм [10, гл. V, § 10, п. 4, теорема 1]

$$\operatorname{Gal}(\overline{\kappa_D}/\kappa_D) \xrightarrow{\sim} \operatorname{Gal}(\overline{\kappa_D}(D)/\kappa(D));$$

остаётся использовать каноническую последовательность инфляции – ограничения [11, гл. IV, § 5, предложение 5.1]

$$\begin{aligned} 0 \to H^1(\operatorname{Gal}(\overline{\kappa_D}(D)/\kappa(D)), \mathbb{Q}/\mathbb{Z}) &\xrightarrow{\inf} H^1(\operatorname{Gal}(\kappa(D)^{\mathrm{s}}/\kappa(D)), \mathbb{Q}/\mathbb{Z}) \\ &\xrightarrow{\mathrm{res}} H^1(\operatorname{Gal}(\kappa(D)^{\mathrm{s}}/\overline{\kappa_D}(D)), \mathbb{Q}/\mathbb{Z}). \end{aligned}$$

Следовательно, достаточно доказать конечность ядра композиции отображений

$$H^2(\operatorname{Spec}\kappa(v), \mathbb{Z}) \to H^2(\operatorname{Spec}\kappa_D, \mathbb{Z}) \xrightarrow{x \mapsto \operatorname{mult}_{\pi^{-1}(v)}(D)\cdot x} H^2(\operatorname{Spec}\kappa_D, \mathbb{Z}). \tag{15}$$

Пусть $d = [\kappa_D : \kappa(v)]$. Поскольку композиция ограничения и кограничения является умножением на d, то мы видим, что ядро композиции (15) содержится в ядре умножения на $\operatorname{mult}_{\pi^{-1}(v)}(D) \cdot d : \mathbb{Q}/\mathbb{Z} \to \mathbb{Q}/\mathbb{Z}$, которое, очевидно, конечно. Теорема 1 доказана.

Теорема 2. *Пусть $\pi : X \to C$ – сюръективный морфизм гладких проективных многообразий над конечным полем $\mathbb{F}_q$ характеристики $p > 2$, C – кривая, общий схемный слой морфизма π является* K3*–поверхностью V над $k = \kappa(C)$. Тогда группа*

$$\operatorname{Br}'(X)(\operatorname{non} -p) \overset{\mathrm{def}}{=} \bigoplus_{l \neq p} \operatorname{Br}'(X)(l)$$

конечная.

Это следует из теоремы 1 и теоремы Скоробогатова – Зархина [1, теорема 1.3], согласно которой группа $[\operatorname{Br}'(V)/\operatorname{Im}[\operatorname{Br}(k) \to \operatorname{Br}'(V)]](\operatorname{non} -p)$ является конечной.

Список литературы / References

[1] Skorobogatov A.N., Zarhin Yu.G., “A finiteness theorem for the Brauer group of K3 surfaces in odd characteristic”, arXiv: arXiv: 1403.0849v1 [math.AG] 4 Mar 2014, 1–10.

[2] Танкеев С.Г., “О конечности группы Брауэра арифметической схемы”, *Матем. заметки*, **95**:1 (2014), 136–149; [Tankeev S.G., “On the finiteness of the Brauer group of an arithmetic scheme”, *Math. Notes*, **95**:1 (2014), 136–149], (in Russian)].

[3] Colliot-Thélène J.-L., Skorobogatov A.N., Swinnerton-Dyer P., “Hasse principle for pencils of curves of genus one whose Jacobians have rational 2-division points”, *Invent. Math.*, **134**:3 (1998), 579–650.

[4] Милн Дж., *Этальные когомологии*, Мир, М., 1983; [Milne J.S., *Etale cohomology*, Princeton Univ. Press, Princeton, 1980].

[5] Годеман Р., *Алгебраическая топология и теория пучков*, ИЛ, М., 1961; [Godement R., *Topologie algébrique et théorie des faisceaux*, Hermann, Paris, 1958].

[6] Lang S., Weil A., "Number of points of varieties in finite fields", *Amer. J. Math.*, **76**:4 (1954), 819–827.

[7] Танкеев С. Г., "О группе Брауэра арифметической схемы. II", *Изв. РАН. Сер. матем.*, **67**:5 (2003), 155–176; [Tankeev S.G., "On the Brauer group of arithmetic scheme. II", *Izv. Math.*, **67**:5 (2003), 1007–1029].

[8] Атья М., Макдональд И., *Введение в коммутативную алгебру*, Мир, М., 1972; [Atiyah M.F., Macdonald I.G., *Introduction to commutative algebra*, Addison–Wesley Publ. Co., Massachusets, 1969].

[9] Skorobogatov A. N., "Descent on fibrations over the projective line", *Amer. J. Math.*, **118**:5 (1996), 905–923.

[10] Бурбаки Н., *Алгебра. Многочлены и поля. Упорядоченные группы, Элементы математики*, Наука, М., 1965; [Bourbaki N., *Éléments de Mathématique. Algébre, livre II*, Hermann, Paris, 1963].

[11] *Алгебраическая теория чисел*, ред. Касселс Дж., Фрёлих А., Мир, М., 1969; [*Algebraic number theory*, Proc. Internat. Conf. Brighton, 1965, eds. Cassels G. W. S., Frölich A., Academic Press, London, and Thompson, Washington, DC, 1967].

Prokhorova T. V., " On the Brauer Group of an Arithmetic Model of a Variety over a Global Field of Positive Characteristic", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 164–172.

DOI: 10.18255/1818-1015-2016-2-164-172

Abstract. Let V be a smooth projective variety over a global field $k = \kappa(C)$ of rational functions on a smooth projective curve C over a finite field $\mathbb{F}_q$ of characteristic p. Assume that there is a projective flat $\mathbb{F}_q$-morphism $\pi : X \to C$, where X is a smooth projective variety and the generic scheme fiber of π is isomorphic to a variety V (we call $\pi : X \to C$ an *arithmetic model of a variety* V).

M. Artin conjectured the finiteness of the Brauer group $\mathrm{Br}(X)$ classifying sheaves of Azumaya algebras on X modulo similitude. It is well known that the group $\mathrm{Br}(X)$ is contained in the cohomological Brauer group

$$\mathrm{Br}'(X) = H^2_{\acute{e}t}(X, \mathbb{G}_m).$$

By definition, the non $-p$ component of the cohomological Brauer group $\mathrm{Br}'(X)$ coincides with the direct sum of the l-primary components of the group $\mathrm{Br}'(X)$ for all prime numbers l different from the characteristic p. It is known that the structure of k-variety on V yields the canonical morphism of the groups $\mathrm{Br}(k) \to \mathrm{Br}'(V)$.

The finiteness of the non $-p$ component of the cohomological Brauer group $\mathrm{Br}'(X)$ of a variety X has been proved if

$$[\mathrm{Br}'(V)/\operatorname{Im}[\mathrm{Br}(k) \to \mathrm{Br}'(V)]](\mathrm{non}-p)$$

is finite.

In particular, if V is a K 3 surface (in other words, V is a smooth projective simply connected surface over a field k and the canonical class of a surface of V is trivial: $\Omega^2_V = \mathcal{O}_V$) and the characteristic of the ground field $p > 2$, then, by the Skorobogatov – Zarhin theorem, $[\mathrm{Br}'(V)/\operatorname{Im}[\mathrm{Br}(k) \to \mathrm{Br}'(V)]](\mathrm{non}-p)$ is finite, so in this case the groups $\mathrm{Br}'(X)(\mathrm{non}-p)$ and $\mathrm{Br}(X)(\mathrm{non}-p)$ are finite.

Keywords: Brauer group, arithmetic model, K 3 surface

On the authors: Prokhorova Tatyana Vyacheslavovna, orcid.org/0000-0002-6883-2087, PhD,
A. G. and N. G. Stoletov Vladimir State University, Gorky str., 87, Vladimir, 600000, Russia, e-mail: tvprokhorova@mail.ru

Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 173–184
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 173–184

DOI: 10.18255/1818-1015-2016-2-173-184

УДК 517.9

Построение CFC-программ ПЛК по LTL-спецификации

Рябухин Д. А., Кузьмин Е. В., Соколов В. А.

получена 1 марта 2016

Аннотация. Статья продолжает серию работ, посвященных подходу к построению и верификации «дискретных» программ логических контроллеров (ПЛК) по LTL-спецификации. Этот подход обеспечивает возможность анализа корректности программ логических контроллеров с помощью метода проверки модели (Model Checking). В рамках подхода в качестве языка спецификации программного поведения используется язык темпоральной логики LTL. Анализ корректности LTL-спецификации относительно программных свойств производится автоматически с помощью программного средства символьной проверки модели Cadence SMV.

Ранее было показано, каким образом по корректной (проверенной на корректность относительно программных свойств) LTL-спецификации происходит построение ST-, LD- и IL-программ. В этой статье описывается технология построения CFC-программ по LTL-спецификации. Язык непрерывных функциональных схем CFC (Continuous Function Chart) представляет собой разновидность языка FBD (Function Block Diagram) — графического языка диаграмм принципиальных схем электронных устройств на микросхемах. В отличие от FBD язык CFC обеспечивает возможность свободного размещения программных компонентов и их соединений на экране. Технология построения CFC-программ демонстрируется на примере.

Представление ПЛК-программы на языке CFC в рамках подхода к программированию по LTL-спецификации в отличие от представлений на других стандартных языках дает возможность наглядного графического изображения потока данных от входных переменных к выходам ПЛК. Явным образом демонстрируется зависимость и влияние значений одних переменных на значения других переменных во время исполнения программы за один проход рабочего цикла ПЛК. Фактически CFC-программа представляет собой схему потоков данных ПЛК-программы.

Ключевые слова: программируемые логические контроллеры (ПЛК), построение и верификация ПЛК-программ, LTL-спецификация, CFC-диаграммы

Для цитирования: Рябухин Д. А., Кузьмин Е. В., Соколов В. А., "Построение CFC-программ ПЛК по LTL-спецификации", *Моделирование и анализ информационных систем*, **23**:2 (2016), 173–184.

Об авторах:
Рябухин Дмитрий Александрович, orcid.org/0000-0002-0799-8868, аспирант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: dmitriy_ryabukhin@mail.ru

Кузьмин Егор Владимирович, orcid.org/0000-0003-0500-306X, доктор физ.-мат. наук, доцент, профессор кафедры теоретической информатики, Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: kuzmin@uniyar.ac.ru

Соколов Валерий Анатольевич, orcid.org/0000-0003-1427-4937, доктор физ.-мат. наук, профессор, зав. кафедрой теоретической информатики, Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: sokolov@uniyar.ac.ru

Благодарности:
Работа проводилась при финансовой поддержке РФФИ, грант №12-01-00281-а.

Введение

Статья продолжает серию работ [1–6, 10–13], посвященных подходу к построению и верификации «дискретных» программ логических контроллеров по LTL-спецификации. Этот подход обеспечивает возможность анализа корректности программ логических контроллеров с помощью метода проверки модели (Model Checking) [7, 8]. В рамках подхода в качестве языка спецификации программного поведения используется язык темпоральной логики LTL. Анализ корректности LTL-спецификации относительно программных свойств производится автоматически с помощью программного средства символьной проверки модели Cadence SMV [14].

Программируемый логический контроллер (ПЛК) — «реагирующая» система, имеющая множество входов, подключаемых посредством датчиков к объекту управления, и множество выходов, подключаемых к исполнительным устройствам [18,19]. Программа ПЛК выполняется в рабочем цикле: считывание входов, выполнение программы и выставление выходов. Применение ПЛК в системах управления сложными производственными процессами предъявляет строгие требования корректности к программам ПЛК.

Языки программирования ПЛК определяются стандартом МЭК 61131-3. Стандарт включает в себя описание пяти языков: SFC, IL, ST, LD и FBD/CFC. Ранее в статьях [3–5] показывалось, каким образом по корректной (проверенной на корректность относительно программных свойств) LTL-спецификации происходит построение ST-, LD- и IL-программ. В этой статье описывается технология построения CFC-программ по LTL-спецификации. Язык непрерывных функциональных схем CFC (Continuous Function Chart) представляет собой разновидность языка FBD (Function Block Diagram) — графического языка диаграмм принципиальных схем электронных устройств на микросхемах. В отличие от FBD язык CFC обеспечивает возможность свободного размещения программных компонентов и их соединений на экране. Технология построения CFC-программ демонстрируется на примере программы ПЛК управления установкой для приготовления смесей.

Информация о других подходах к построению и анализу FBD/CFC-программ ПЛК может быть найдена, например, в работах [15–17].

1. Построение программ по LTL-спецификации

Смысл концепции программирования ПЛК по LTL-спецификации [1–6, 10–13] состоит в обеспечении возможности анализа корректности ПЛК-программ методом проверки модели [7, 8]. В соответствии с этой концепцией значение каждой переменной должно изменяться только в одном месте программы и не более одного раза за одно полное выполнение программы при прохождении рабочего цикла ПЛК. Поэтому изменение значения каждой программной переменной задаётся с помощью пары явных LTL-формул и одной неявной. Первая LTL-формула описывает ситуации, при которых происходит возрастание значения соответствующей переменной, вторая LTL-формула определяет условия, приводящие к уменьшению значения переменной. Третья LTL-формула в явном виде не прописывается, поскольку всегда имеет жёсткую конструкцию, составленную из элементов первых двух формул. Она описывает условия, при которых переменная не меняет своего значения за один проход

рабочего цикла ПЛК. Рассматриваемые для спецификации поведения переменных LTL-формулы являются конструктивными в том смысле, что по ним производится построение ПЛК-программы, которая соответствует темпоральным свойствам, выраженным этими формулами. Таким образом, программирование ПЛК сводится к построению LTL-спецификации поведения каждой программной переменной.

Для описания ситуаций, которые приводят к увеличению и уменьшению значения целочисленной переменной V, используются LTL-формулы вида

$$\mathbf{G}\,\mathbf{X}(\, V{>}\_V \Rightarrow \mathit{OldValCond} \,\wedge \mathit{FiringCond} \,\wedge V = \mathit{NewValExpr}\,); \tag{1}$$

$$\mathbf{G}\,\mathbf{X}(\, V{<}\_V \Rightarrow \mathit{OldValCond}' \wedge \mathit{FiringCond}' \wedge V = \mathit{NewValExpr}'\,). \tag{2}$$

Символ лидирующего подчеркивания «_» в обозначении переменной $\_V$ воспринимается как псевдооператор, позволяющий обратиться к значению переменной V, которое она имела в предыдущем состоянии (после последнего полного прохода рабочего цикла ПЛК). При этом псевдооператор может использоваться только под действием темпорального оператора $\mathbf{X}$.

Условия *FiringCond* и *OldValCond* являются логическими выражениями над программными переменными и константами, которые строятся с применением операторов сравнения, логических и арифметических операторов и псевдооператора «_» (который по определению может быть применим только к переменным). Выражение *FiringCond* описывает ситуации, при которых возникает необходимость изменения значения переменной V, если это, конечно, допускается условием *OldValCond*. Выражение *NewValExpr*, определяющее новое значение переменной V, строится с помощью переменных и констант, операторов сравнения, логических, арифметических операторов и псевдооператора «_».

Для описания всех возможных ситуаций, при которых происходит возрастание значения переменной V, в формуле (1) после символа оператора импликации $\Rightarrow$ может потребоваться несколько наборов рассмотренных конъюнктивных членов $\mathit{OldValCond}_i \wedge \mathit{FiringCond}_i \wedge V = \mathit{NewValExpr}_i$ объединенных в дизъюнкцию.

Аналогичный смысл имеют выражения $\mathit{FiringCond}'$, $\mathit{OldValCond}'$ и $\mathit{NewValExpr}'$.

В случае с переменной логического типа данных для спецификации ее поведения используются более простые LTL-формулы:

$$\mathbf{G}\,\mathbf{X}(\, \neg\_V \wedge \;\; V \Rightarrow \mathit{FiringCond}\,);$$

$$\mathbf{G}\,\mathbf{X}(\;\; \_V \wedge \neg V \Rightarrow \mathit{FiringCond}'\,);$$

которые означают, что всякий раз, когда новое значение переменной V оказывается больше или меньше ее предыдущего значения, записанного в переменной $\_V$, из этого следует, что было выполнено условие соответствующего внешнего воздействия *FiringCond* или $\mathit{FiringCond}'$. В случае наличия равенства $\mathit{FiringCond} = \neg\mathit{FiringCond}'$ эта спецификация может быть заменена на одну LTL-формулу вида

$$\mathbf{G}\,\mathbf{X}(\, V = \mathit{FiringCond}\,).$$

Неявная LTL-формула сохранения прежнего значения переменной V имеет вид

$$\mathbf{G}\,\mathbf{X}(\, V{=}\_V \Rightarrow \neg(\mathit{OldValCond} \wedge \mathit{FiringCond}) \wedge \neg(\mathit{OldValCond}' \wedge \mathit{FiringCond}')\,). \tag{3}$$

Для логической переменной V эта неявная LTL-формула выглядит как

$$\mathbf{G}\,\mathbf{X}(\, V{=}\_V \Rightarrow \neg(\neg\_V \wedge \mathit{FiringCond}) \wedge \neg(\_V \wedge \mathit{FiringCond}')\,).$$

При построении спецификации важно учитывать то, в каком порядке распола-

гаются темпоральные формулы, описывающие поведение переменных. Некоторая переменная без псевдооператора «_» может быть задействована в спецификации поведения другой переменной, только если спецификация ее поведения уже произведена и находится выше по тексту.

В спецификации блок темпоральных формул, описывающий поведение некоторой переменной, по необходимости сопровождается указанием начального значения переменной, которое она получает при инициализации. Для этого задействуется ключевое слово Init. Например, $\mathrm{Init}(V) = 1$ означает, что при инициализации переменная V получает значение 1. Если в спецификации явно не указывается начальное значение переменной, то считается, что это значение равняется нулю.

2. Построение CFC-программ ПЛК по LTL-спецификации

В этом разделе рассматривается способ построения программного CFC-кода по конструктивной LTL-спецификации поведения программных переменных. В общем виде схема трансляции LTL-формул в программный код на языке CFC следующая.

Двум явным темпоральным формулам переменной V (1) и (2) (и неявной темпоральной формуле (3)) ставится в соответствие графический блок на языке CFC. Общий вид CFC-блока для целочисленной и логической переменных представлен на рис. 1 в правом нижнем углу. Возможные реализации этого блока также приведены на рис. 1. Порядок выполнения и расположения блоков устанавливается в соответствии с порядком LTL-формул в спецификации. Блоки соединяются между собой по одноименным маркерам виртуальной или реальной линией. Линии проводятся также в соответствии с порядком следования формул в спецификации.

Представление ПЛК-программы на языке CFC в рамках подхода к программированию по LTL-спецификации в отличие от представлений на других стандартных языках дает возможность наглядного графического изображения потока данных от входных переменных к выходам ПЛК. Явным образом демонстрируется зависимость и влияние значений одних переменных на значения других переменных во время исполнения программы за один проход рабочего цикла ПЛК. Фактически CFC-программа представляет собой схему потоков данных ПЛК-программы.

Отметим, что каждая программная переменная должна быть определена в разделе (локальном или глобальном) описания переменных и проинициализирована в соответствии со спецификацией. В среде разработки CoDeSys [9] по умолчанию все переменные инициализируются нулем.

Кроме того, необходимо реализовать идею псевдооператора лидирующего подчеркивания «_». Для этого последними действиями в программе для каждой переменной V выходное значение соответствующего ей CFC-блока направляется также и в переменную-выход $\_V$, которая на следующем рабочем цикле ПЛК будет выступать в качестве входной переменной. При этом $\_V$ определяют в разделе описания переменных с такой же инициализацией, как и для переменной V.

В следующем разделе идея построения CFC-программ по LTL-спецификации наглядно демонстрируется на примере ПЛК-программы управления установкой для приготовления смесей.

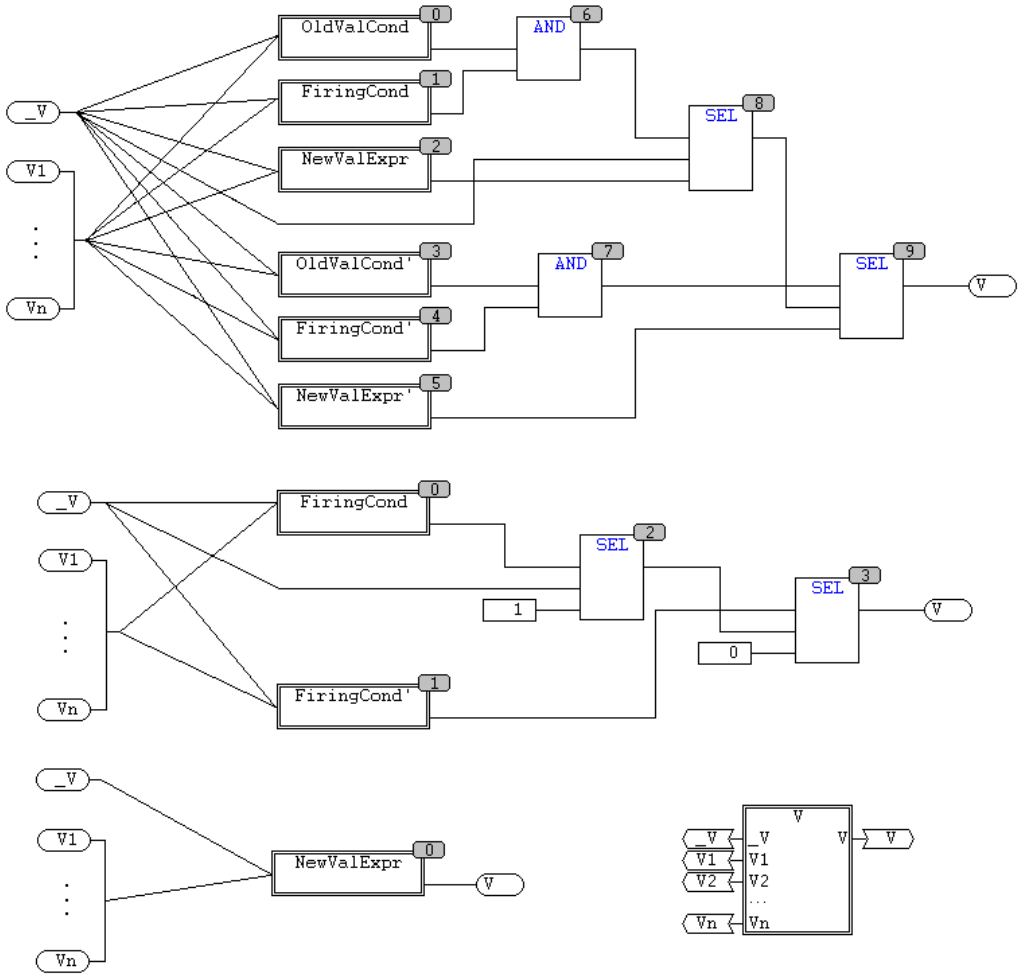

Рис. 1. Общий вид CFC-блока для целочисленной и логической переменных
Fig. 1. General form of CFC-block for integer and boolean variables

3. Установка для приготовления смесей

Установка, схема которой представлена на рис. 2, предназначена для приготовления смесей из двух компонентов. Первый смешиваемый компонент подается из бака 1 посредством клапана «Кл1» в изначально пустой (нет сигнала от датчика «ДУ0») резервуар до тех пор, пока не сработает датчик уровня «ДУ1». По срабатыванию датчика «ДУ1» клапан «Кл1» закрывается. Затем открывается «Кл2» и из бака 2 подается второй смешиваемый компонент до тех пор, пока не сработает датчик уровня «ДУ2», после чего клапан «Кл2» закрывается. Далее в течение T секунд с помощью мешалки в резервуаре происходит перемешивание компонентов при определённой температуре, которая поддерживается нагревателем. По истечении этого времени открывается клапан «РКл» и происходит выгрузка готовой смеси. Выгрузка заканчивается, т. е. клапан «РКл» закрывается, как только пропадает сигнал от датчика уровня «ДУ0». Клапан «АКл» используется для принудительного слива смеси из резервуара. Наличие смешиваемых компонентов в баках определяется по соответствующим датчикам уровня «ДБ1» и «ДБ2». Функционирование мешалки определяется

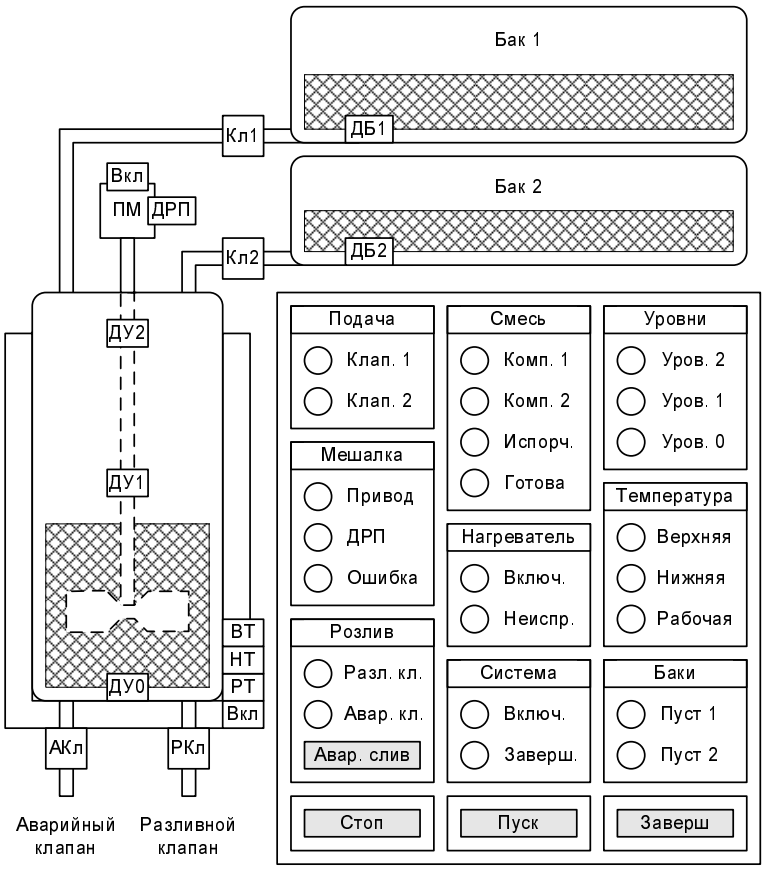

Рис. 2. Схема и панель управления установки для приготовления смесей
Fig. 2. Scheme and control panel of mixing plant

по датчику работы привода мешалки «ДРП». Температура нагревательного элемента отслеживается с помощью датчика верхней температуры «ВТ», датчика нижней температуры «НТ» и датчика рабочей температуры «РТ». Если после запуска установки нагреватель имеет температуру меньше нижней температуры, то он включается и начинает нагреваться. Как только температура нагревателя достигает заданного верхнего значения, то нагреватель выключается и постепенно остывает. Задача нагревателя состоит в поддержании некоторой рабочей температуры, необходимой для смешивания компонентов. Рабочая температура устанавливается на несколько градусов меньше нижней температуры. Запуск, экстренная остановка и плавное завершение работы установки осуществляется соответствующими кнопками панели управления «Пуск», «Стоп» и «Заверш». Под плавным завершением работы понимается выключение установки после того, как будет приготовлена текущая порция смеси и полностью выдана через клапан «РКл».

Установка по приготовлению смесей является автоматической. Управление установкой осуществляется с помощью ПЛК, получающего входные сигналы от датчиков установки и переключателей панели управления оператора и подающего выходные сигналы на приводы установки и лампы панели управления (см. рис. 2).

Задача состоит в написании программы для ПЛК с 13 входами и 14 выходами, предназначенного для управления установкой. Интерфейс ПЛК управления уста-

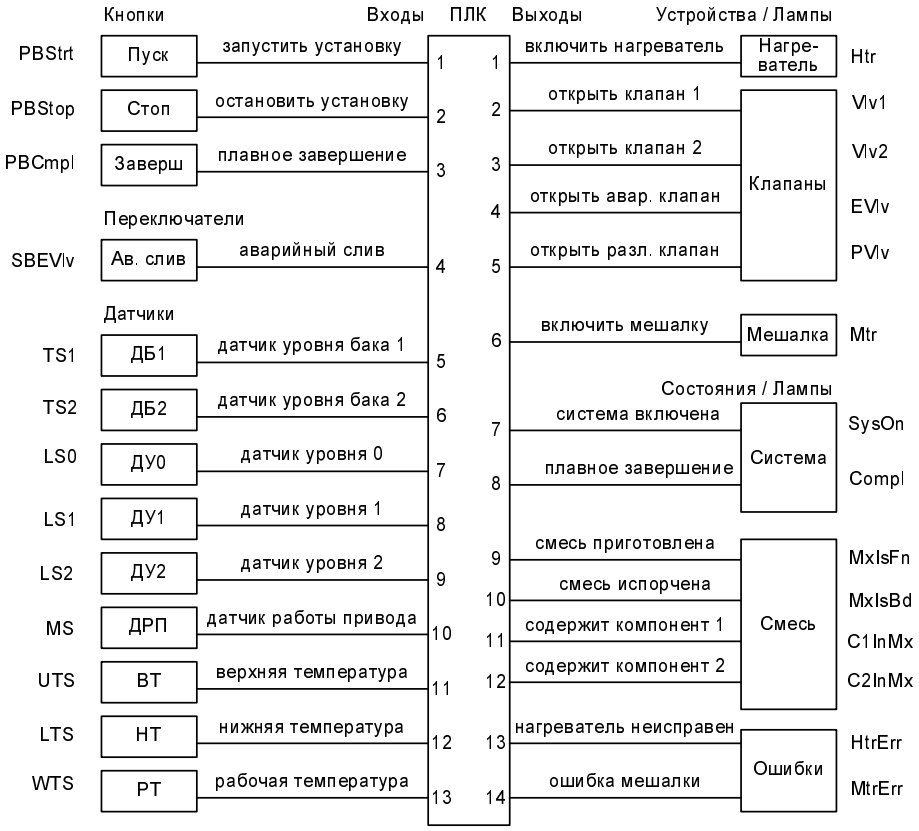

Рис. 3. Интерфейс ПЛК управления установкой для приготовления смесей
Fig. 3. Interface of mixing plant

новкой представлен на рис. 3. Предполагается, что сигналы от датчиков на соответствующие лампы панели управления идут напрямую, минуя ПЛК.

По описанию, схожему с описанием подобной установки из [2,13], была построена следующая LTL-спецификация.

```
/* Таймеры */
GX( MtrTmrIn = Mtr ∧   MS );
GX( ErrTmrIn = Mtr ∧ ¬MS );
GX( HtrTmrIn = Htr ∧ ¬WTS );
/* Состояния/Ошибки/Лампы */
GX( ¬_MtrErr ∧  MtrErr ⇒ (ErrTmrQ ∨ ErrTmrIn) ∧ _MS );
GX(  _MtrErr ∧ ¬MtrErr ⇒ PBStop );
GX( ¬_HtrErr ∧  HtrErr ⇒ (HtrTmrQ ∨ HtrTmrIn) ∧ _WTS );
GX(  _HtrErr ∧ ¬HtrErr ⇒ PBStop );
/* Состояния/Лампы */
GX( ¬_C1InMx ∧  C1InMx ⇒  LS0 ∧ _Vlv1 ∧ _TS1 );
GX(  _C1InMx ∧ ¬C1InMx ⇒ ¬LS0);
GX( ¬_C2InMx ∧  C2InMx ⇒  LS0 ∧ _Vlv2 ∧ _TS2 );
GX(  _C2InMx ∧ ¬C2InMx ⇒ ¬LS0);
GX( ¬_MxIsFn ∧  MxIsFn ⇒  LS0 ∧ MtrTmrQ ∧ ¬MxIsBd );
GX(  _MxIsFn ∧ ¬MxIsFn ⇒ ¬LS0);
```

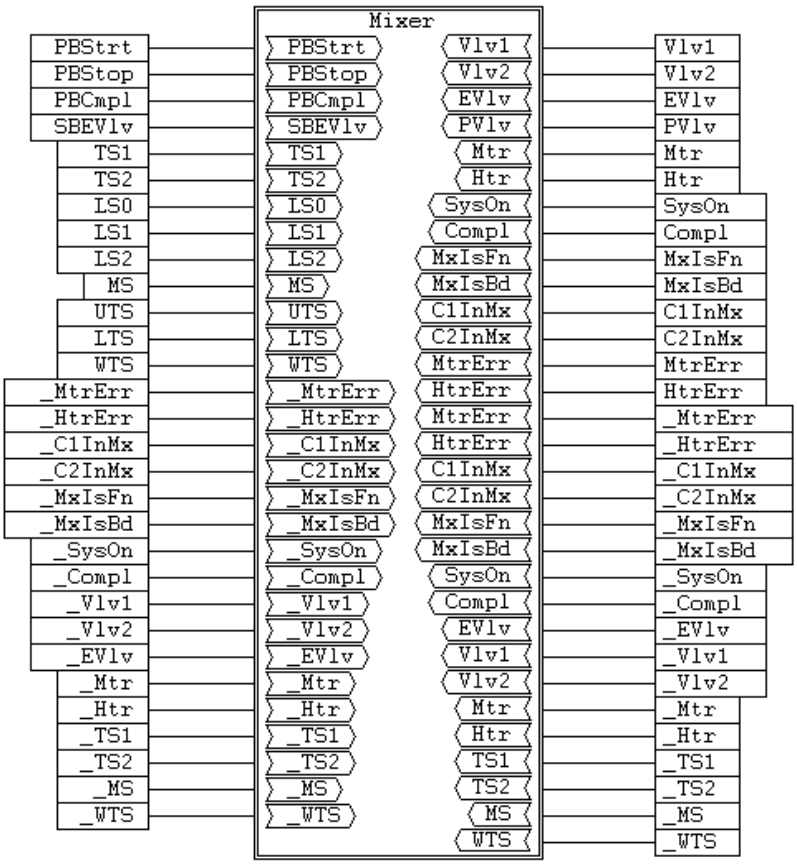

Рис. 4. Общий вид CFC-программы установки для приготовления смесей
Fig. 4. General form of CFC-program of mixing plant

$$\mathbf{GX}(\ \neg\_\texttt{MxIsBd} \wedge \ \texttt{MxIsBd} \Rightarrow \ \texttt{LS0} \wedge \texttt{\_EVlv} \wedge \neg\_\texttt{MxIsFn} \wedge \texttt{\_C2InMx});$$
$$\mathbf{GX}(\ \ \texttt{\_MxIsBd} \wedge \neg\texttt{MxIsBd} \Rightarrow \neg\texttt{LS0});$$
/* Вспомогательные переменные */
$$\mathbf{GX}(\ \texttt{Fin} = (\texttt{PBCmpl} \vee \texttt{\_Compl}) \wedge \neg\_\texttt{Vlv1} \wedge \neg\texttt{LS0} \vee \texttt{HtrErr} \vee \texttt{MtrErr} \vee \texttt{MxIsBd} \vee$$
$$\vee\,\texttt{SBEVlv} \vee \texttt{PBStop} \vee \neg\texttt{TS1} \wedge \texttt{LS1} \wedge \neg\texttt{MxIsFn} \vee \neg\texttt{TS2} \wedge \texttt{LS2} \wedge \neg\texttt{MxIsFn}\ );$$
/* Состояния/Лампы */
$$\mathbf{GX}(\ \neg\_\texttt{SysOn} \wedge \ \texttt{SysOn} \Rightarrow \ \texttt{PBStrt} \wedge \neg\texttt{Fin})\ );$$
$$\mathbf{GX}(\ \ \texttt{\_SysOn} \wedge \neg\texttt{SysOn} \Rightarrow \ \texttt{Fin}\ );$$
$$\mathbf{GX}(\ \neg\_\texttt{Compl} \wedge \ \texttt{Compl} \Rightarrow \ \texttt{SysOn} \wedge \texttt{PBCmpl}\ );$$
$$\mathbf{GX}(\ \ \texttt{\_Compl} \wedge \neg\texttt{Compl} \Rightarrow \neg\texttt{SysOn}\ );$$
/* Приводы/Лампы */
$$\mathbf{GX}(\ \texttt{EVlv} = \texttt{SBEVlv}\ );$$
$$\mathbf{GX}(\ \texttt{Vlv1} = \texttt{SysOn} \wedge \texttt{TS1} \wedge \neg\texttt{LS1} \wedge \neg\texttt{C2InMx} \wedge \neg\texttt{Compl} \wedge \neg\texttt{LS0} \wedge \texttt{WTS}\ );$$
$$\mathbf{GX}(\ \texttt{Vlv2} = \texttt{SysOn} \wedge \texttt{TS2} \wedge \ \texttt{LS1} \wedge \neg\texttt{LS2} \wedge \neg\texttt{MxIsFn} \wedge \texttt{WTS}\ );$$
$$\mathbf{GX}(\ \texttt{Mtr} \ = \texttt{SysOn} \wedge \texttt{LS2} \wedge \neg\texttt{MxIsFn} \wedge \texttt{WTS} \wedge \texttt{C1InMx} \wedge \texttt{C2InMx}\ );$$
$$\mathbf{GX}(\ \texttt{PVlv} = \texttt{SysOn} \wedge \texttt{MxIsFn}\ );$$
/* Нагреватель/Лампа */
$$\mathbf{GX}(\ \neg\_\texttt{Htr} \wedge \ \texttt{Htr} \Rightarrow \ \texttt{SysOn} \wedge \neg\texttt{LTS}\ );$$
$$\mathbf{GX}(\ \ \texttt{\_Htr} \wedge \neg\texttt{Htr} \Rightarrow \neg\texttt{SysOn} \vee \ \texttt{UTS}\ ).$$

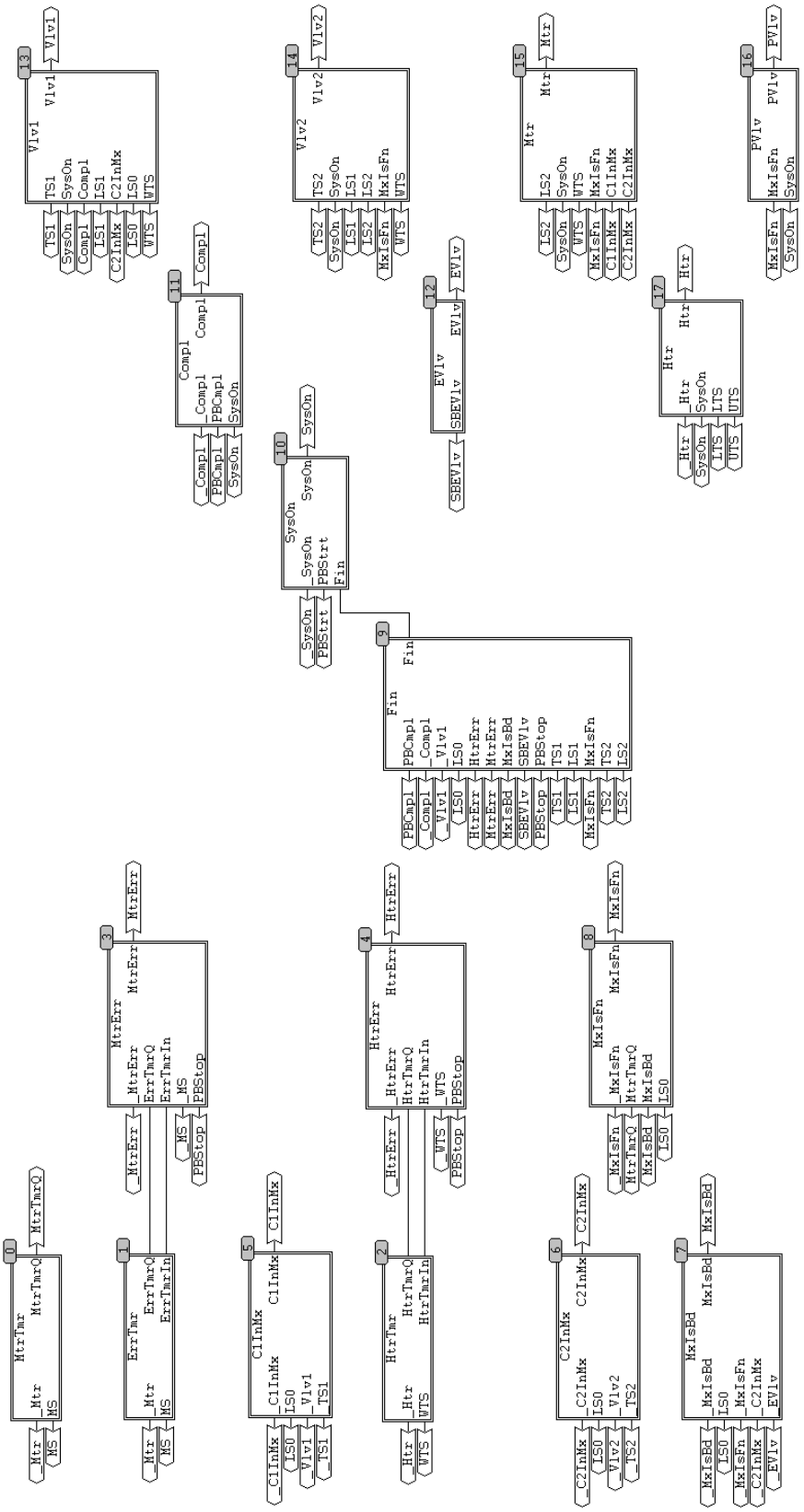

Рис. 5. CFC-программа установки для приготовления смесей
Fig. 5. CFC-program of mixing plant

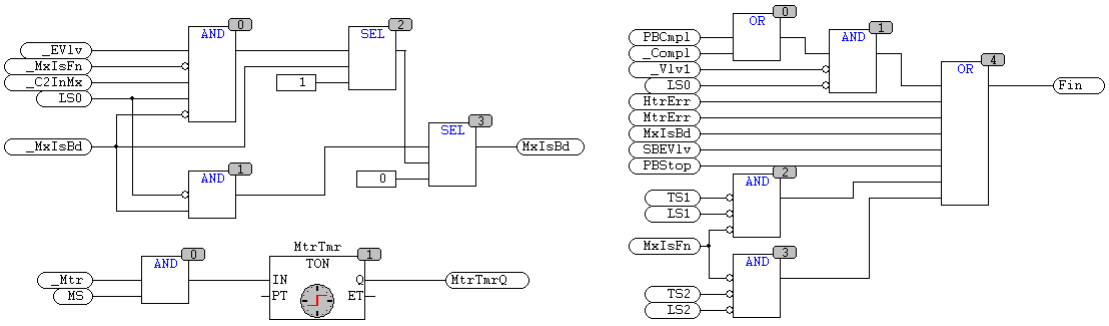

Рис. 6. Примеры CFC-блоков для переменных MxIsBd, MtrTmrQ и Fin
Fig. 6. CFC-blocks for variables MxIsBd, MtrTmrQ and Fin

Примерами программных свойств, которые предполагаются к проверке для данной LTL-спецификации (и соответственно для CFC-программы, построенной по этой спецификации), являются следующие LTL-свойства.

«Выключенная система». Если система выключена, то клапаны подачи и разливной клапан закрыты, нагреватель и привод мешалки выключены, а лампы группы «Система» погашены: $\mathbf{G}(\ \neg\texttt{SysOn} \Rightarrow \neg(\texttt{Vlv1} \vee \texttt{Vlv2} \vee \texttt{PVlv} \vee \texttt{Htr} \vee \texttt{Mtr} \vee \texttt{Compl})\ )$.

«Клапаны». В один момент времени в открытом состоянии может находиться не более одного клапана: $\mathbf{G}(\ \texttt{Vlv1} + \texttt{Vlv2} + \texttt{PVlv} + \texttt{EVlv} \leq 1\ )$.

«Пустые баки». Если хотя бы один из баков и резервуар пустые, система находится в выключенном состоянии: $\mathbf{G}(\ (\neg\texttt{TS1} \vee \neg\texttt{TS2}) \wedge \neg\texttt{LS0} \Rightarrow \neg\texttt{SysOn}\ )$.

«Мешалка». Если привод мешалки включен, все клапаны установки закрыты: $\mathbf{G}(\ \texttt{Mtr} \Rightarrow \neg(\texttt{Vlv1} \vee \texttt{Vlv2} \vee \texttt{PVlv} \vee \texttt{EVlv})\ )$.

«Перерасход компонентов». В испорченную или уже приготовленную смесь подача компонентов не осуществляется: $\mathbf{G}(\ (\texttt{MxIsBd} \vee \texttt{MxIsFn}) \Rightarrow \neg(\texttt{Vlv1} \vee \texttt{Vlv2})\ )$.

«Испорченная смесь». Испорченная смесь не может ни выдаваться через разливной клапан установки, ни перемешиваться мешалкой, ни подогреваться включенным нагревателем: $\mathbf{G}(\ \texttt{MxIsBd} \Rightarrow \neg\texttt{PVlv} \wedge \neg\texttt{Mtr} \wedge \neg\texttt{Htr}\ )$.

«Включенная система». Если система включена и в резервуаре достигнута рабочая температура, то обязательно открыт один из неаварийных клапанов или работает привод мешалки: $\mathbf{G}(\ \texttt{SysOn} \wedge \texttt{WTS} \Rightarrow (\texttt{Vlv1} \vee \texttt{Vlv2} \vee \texttt{PVlv} \vee \texttt{Mtr})\ )$.

«Включенный привод». Если привод мешалки был включен, то он обязательно рано или поздно будет выключен: $\mathbf{G}(\ \texttt{Mtr} \Rightarrow \mathbf{F}(\neg\texttt{Mtr})\ )$.

По описанной в этом разделе LTL-спецификации установки для приготовления смесей строится CFC-программа, которая в общем виде представлена на рис. 4 и 5. Примеры реализаций CFC-блоков для переменных MxIsBd, MtrTmrQ и Fin приведены на рис. 6. CFC-блоки программы на рис. 5 в основном соединены неявно с помощью соответствующих маркеров, но для примера показаны и несколько явных линий связи. Визуально блоки расположены таким образом, что значения переменных-блоков следующего ряда не оказывают влияния на значения переменных-блоков предыдущего ряда в рамках потока данных при исполнении программы за один рабочий цикл ПЛК.

Список литературы / References

[1] Кузьмин Е. В., Рябухин Д. А., Соколов В. А., "О выразительности подхода к построению ПЛК-программ по LTL-спецификации", *Моделирование и анализ информационных систем*, **22**:4 (2015), 507–520; [Kuzmin E. V., Ryabukhin D. A., Sokolov V. A., "On the Expressiveness of the Approach to Constructing PLC-programs by LTL-Specification", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 507–520, (in Russian).]

[2] Кузьмин Е. В., Рябухин Д. А., Соколов В. А., "Моделирование согласованного поведения ПЛК-датчиков", *Моделирование и анализ информационных систем*, **21**:4 (2014), 75–90; [Kuzmin E. V., Ryabukhin D. A., Sokolov V. A., "Modeling a Consistent Behavior of PLC-Sensors", *Modeling and Analysis of Information Systems*, **21**:4 (2014), 75–90, (in Russian).]

[3] Рябухин Д. А., Кузьмин Е. В., Соколов В. А., "Построение IL-программ ПЛК по LTL-спецификации", *Моделиров. и анализ информационных систем*, **21**:2 (2014), 26–38; [Ryabukhin D. A., Kuzmin E. V., Sokolov V. A., "Construction of PLC IL-programs by LTL-specification", *Modeling and Analysis of Information Systems*, **21**:2 (2014), 26–38, (in Russian).]

[4] Кузьмин Е. В., Соколов В. А., Рябухин Д. А., "Построение и верификация LD-программ ПЛК по LTL-спецификации", *Моделирование и анализ информационных систем*, **20**:6 (2013), 78–94; [Kuzmin E. V., Sokolov V. A., Ryabukhin D. A., "Construction and Verification of PLC LD-programs by LTL-specification", *Modeling and Analysis of Information Systems*, **20**:6 (2013), 78–94, (in Russian).]

[5] Кузьмин Е. В., Соколов В. А., Рябухин Д. А., "Построение и верификация ПЛК-программ по LTL-спецификации", *Моделирование и анализ информационных систем*, **20**:4 (2013), 5–22; [Kuzmin E. V., Sokolov V. A., Ryabukhin D. A., "Construction and Verification of PLC-programs by LTL-specification", *Modeling and Analysis of Information Systems*, **20**:4 (2013), 5–22, (in Russian).]

[6] Кузьмин Е. В., Соколов В. А., "Моделирование, спецификация и построение программ логических контроллеров", *Моделирование и анализ информационных систем*, **20**:2 (2013), 104–120; [Kuzmin E. V., Sokolov V. A., "Modeling, Specification and Construction of PLC-programs", *Modeling and Analysis of Information Systems*, **20**:2 (2013), 104–120, (in Russian).]

[7] Baier C., Katoen J.-P., *Principles of Model Checking*, The MIT Press, 2008.

[8] Clark E. M., Grumberg O., Peled D. A., *Model Checking*, The MIT Press, 2001.

[9] CoDeSys, *Controller Development System*, http://www.3s-software.com/.

[10] Kuzmin E. V., Sokolov V. A., Ryabukhin D. A., "Construction and Verification of PLC-Programs by LTL-Specification", *Automatic Control and Computer Sciences*, **49**:7 (2015), 453–465.

[11] Kuzmin E. V., Sokolov V. A., Ryabukhin D. A., "Construction and Verification of PLC LD Programs by the LTL Specification", *Automatic Control and Computer Sciences*, **48**:7 (2014), 424–436.

[12] Kuzmin E. V., Sokolov V. A., "Modeling, Specification and Construction of PLC-programs", *Automatic Control and Computer Sciences*, **48**:7 (2014), 554–563.

[13] Kuzmin E. V., Ryabukhin D. A., Sokolov V. A., "Modeling a Consistent Behavior of PLC-Sensors", *Automatic Control and Computer Sciences*, **48**:7 (2014), 602–614.

[14] SMV, *The Cadence SMV Model Checker*, http://www.kenmcmil.com/smv.html.

[15] Wardana A., *Development of Automatic Program Verification for Continious Function Chart based on Model Checking*, Kassel university press GmbH.

[16] Ovataman T., Aral A., Polat D., Unver A. O., "An overview of model checking practices on verification of PLC software", *Software and Systems Modeling*, 2014.

[17] Pakonen A., Matasniemi T., Lahtinen J., Karhela T., "A toolset for model checking of PLC software", *Proceedings of 18th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA2013, Cagliari, Italy*, 2013.

[18] Parr E. A., *Programmable Controllers. An engineer's guide*, Newnes, 2003.

[19] Петров И. В., *Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования*, М.: СОЛОН-Пресс, 2004; [Petrov I. V., *Programmiruemye kontrollery. Standartnye jazyki i priemy prikladnogo proektirovanija*, M.: SOLON-Press, 2004, (in Russian).]

Ryabukhin D. A., Kuzmin E. V., Sokolov V. A., "Construction of CFC-programs by LTL-specification", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 173–184.

DOI: 10.18255/1818-1015-2016-2-173-184

Abstract. This article continues a cycle of papers, which describe an approach to construction and verification of discrete PLC-programs by an LTL-specification. The approach provides a possibility of PLC-program correctness analysis by the model checking method. For the specification of the program behavior the linear-time temporal logic LTL is used. The correctness analysis of an LTL specification is performed automatically by the symbolic model checking tool Cadence SMV.

Previously it was shown how ST-, LD- and IL-programs are constructed by a correct (with verified program properties) LTL-specification. In this article a technology of CFC-program construction by an LTL-specification is described. The language CFC (Continuous Function Chart) is a variation of FBD (Function Block Diagram). FBD is a graphical language for microcircuits. CFC provides a possibility of free allocation of program components and connections on a screen. The approach to construction of CFC-programs is shown by an example.

PLC-program representation on CFC within the approach to programming by LTL-specification differs from other representations. It gives the visualisation of a data flow from inputs to outputs. Influence and dependence between variables is explicitly shown during program execution within one PLC working cycle. In fact, CFC-program is a scheme of PLC-program data flow.

Keywords: programmable logic controllers (PLC), construction and verification of PLC-programs, LTL-specification, CFC-diagrams

On the authors:
Ryabukhin Dmitriy Aleksandrovich, orcid.org/0000-0002-0799-8868, graduate student,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: dmitriy_ryabukhin@mail.ru

Kuzmin Egor Vladimirovich, orcid.org/0000-0003-0500-306X, doctor of science, associate professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: kuzmin@uniyar.ac.ru

Sokolov Valery Anatolievich, orcid.org/0000-0003-1427-4937, doctor of science, professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: sokolov@uniyar.ac.ru

Acknowledgments:
This work was supported by the Russian Foundation for Basic Research (RFBR), №12-01-00281-a.

Моделирование и анализ информационых систем. Т. 23, №2 (2016), с. 185–194
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 185–194

DOI: 10.18255/1818-1015-2016-2-185-194

UDC 519.987

Asymptotic Formula for the Moments of Bernoulli Convolutions

Timofeev E. A.

Received February 8, 2016

Abstract.
For each λ, $0 < \lambda < 1$, we define a random variable

$$Y_\lambda = (1-\lambda)\sum_{n=0}^{\infty} \xi_n \lambda^n,$$

where ξ_n are independent random variables with

$$\mathsf{P}\{\xi_n = 0\} = \mathsf{P}\{\xi_n = 1\} = \frac{1}{2}.$$

The distribution of Y_λ is called a symmetric Bernoulli convolution. The main result of this paper is

$$M_n = \mathsf{E}Y_\lambda^n = n^{\log_\lambda 2} 2^{\log_\lambda(1-\lambda)+0.5\log_\lambda 2 - 0.5} e^{\tau(-\log_\lambda n)} \left(1 + \mathcal{O}(n^{-0.99})\right),$$

where

$$\tau(x) = \sum_{k\neq 0} \frac{1}{k} \alpha\left(-\frac{k}{\ln\lambda}\right) e^{2\pi i k x}$$

is a 1-periodic function,

$$\alpha(t) = -\frac{1}{2i\,\mathrm{sh}(\pi^2 t)} (1-\lambda)^{2\pi i t} (1 - 2^{2\pi i t}) \pi^{-2\pi i t} 2^{-2\pi i t} \zeta(2\pi i t),$$

and $\zeta(z)$ is the Riemann zeta function.

The article is published in the author's wording.

Keywords: moments, self-similar, Bernoulli convolution, singular, Mellin transform, asymptotic

For citation: Timofeev E. A., "Asymptotic Formula for the Moments of Bernoulli Convolutions", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 185–194.

On the authors:
Timofeev Evgeniy Alexandrovich, orcid.org/0000-0002-0980-2507, ScD, professor
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: timofeevEA@gmail.com

For each λ, $0 < \lambda < 1$ we define the random variable

$$Y_\lambda = (1-\lambda)\sum_{n=0}^{\infty} \xi_n \lambda^n,$$

where ξ_n are independent random variables with

$$\mathsf{P}\{\xi_n = 0\} = \mathsf{P}\{\xi_n = 1\} = \frac{1}{2}.$$

The distribution of Y_λ is called a symmetric Bernoulli convolution.

The cumulative distribution function $F_\lambda(t) = \mathsf{P}\{Y_\lambda < t\}$ can be characterized by the functional equation

$$F_\lambda(x) = \frac{1}{2}F_\lambda(\lambda^{-1}x) + \frac{1}{2}F_\lambda(\lambda^{-1}x - \lambda^{-1} + 1), \quad 0 \le x \le 1. \tag{1}$$

We stress that the Fourier transform is the infinite product

$$\phi(t) = \mathsf{E}e^{itY_\lambda} = \prod_{n=0}^{\infty}\left(\frac{1}{2} + \frac{1}{2}e^{it\lambda^n(1-\lambda)}\right) = e^{it/2}\prod_{n=0}^{\infty}\cos(\lambda^n(1-\lambda)t/2), \tag{2}$$

The early study of $F_\lambda(t)$ was related to some questions of harmonic analysis [1, 14.20].

In Figure 1 we show the histograms for $F'_\lambda(t)$ approximations. This graphics were created as Iterated Functions System of the maps $S_1(x) = \lambda x$, $S_2(x) = \lambda x - \lambda + 1$. We used 2^{20} points and 1000 equally intervals for the histogram.

We stress that these approximations are very crude.

Since the 1930's a lot of work has been done to investigate $F_\lambda(t)$ (see e.g. survey [8]).

One of the fundamental question is to decide for which λ the function $F_\lambda(t)$ is absolutely continuous and for which it is singular.

Results on absolute continuity of $F_\lambda(t)$.

- Jessen and Wintner [7] proved that $F_\lambda(t)$ is either absolutely continuous or purely singular. It is clear that $F_\lambda(t)$ is uniform on $[0,1]$ for $\lambda = \dfrac{1}{2}$ and is purely singular for $\lambda < \dfrac{1}{2}$.

- Wintner [13] proved that $F_\lambda(t)$ is absolutely continuous for $\lambda = 2^{-1/k}$, $k = 1, 2, \ldots$, with a density having $k-1$ derivatives. For example, for $\lambda = 2^{-1/2}$ we have the following density

$$F'_\lambda(x) = \begin{cases} \left(2 + \frac{3}{\sqrt{2}}\right)x, & 0 \le x \le \sqrt{2} - 1; \\ 1 + \frac{1}{\sqrt{2}}, & \sqrt{2} - 1 \le x \le 2 - \sqrt{2}; \\ \left(2 + \frac{3}{\sqrt{2}}\right)(1-x), & 2 - \sqrt{2} \le x \le 1. \end{cases} \tag{3}$$

- Erdös [4] showed that $F_\lambda(t)$ is singular when λ^{-1} is a Pisot (Pisot-Vijayaraghavan) number ($0.5 < \lambda < 1$). Moreover, the Fourier transform $\phi(t)$ does not tend to 0 as $t \to \infty$. Recall that a Pisot number is an algebraic integer $\theta > 1$ all of whose Galois conjugates (other roots of the minimal polynomial) of θ are less than 1 in modulus.

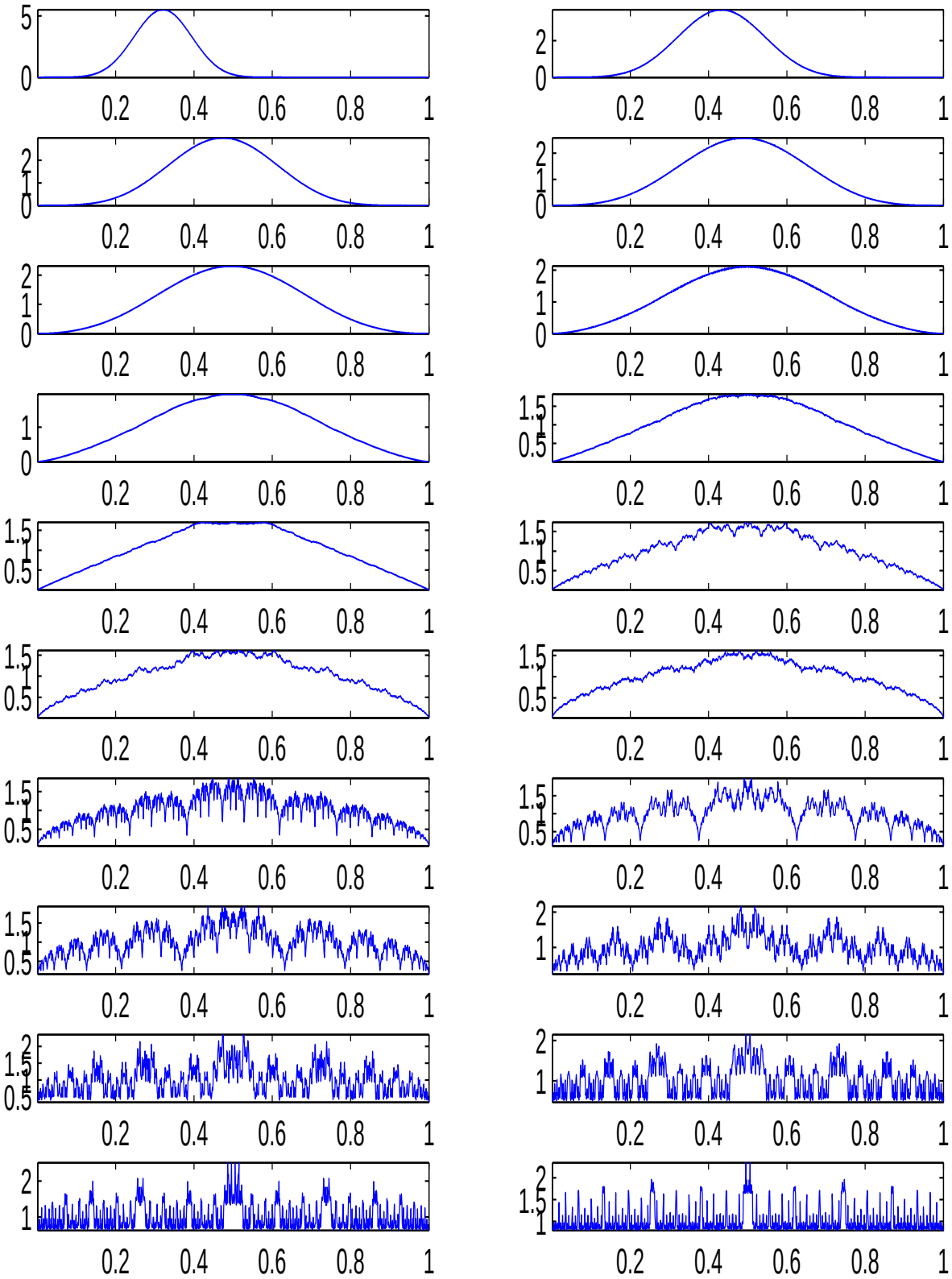

Fig 1. Histograms for $F'_\lambda(t)$ correspond to the following $\lambda^{-1} = 1 + i/21$, $i = 1, 2, \ldots, 20$

- Salem [9–11] proved the converse of the result of Erdos. If $\phi(t)$ does not tend to 0 as $t \to \infty$, then λ^{-1} is necessarily a Pisot number. The reciprocal of Pisot numbers remain today the only known set of λ for which $F_\lambda(t)$ is singular.
- At the next year Erdös [5] proved a result in opposite direction, namely, there is a $a < 1$ such that for almost all λ in the interval $a < \lambda < 1$ we have $F_\lambda(t)$ is absolutely continuous.
- Garsia [6] found new examples of algebraic λ with absolutely continuous $F_\lambda(t)$.
- Solomyak [12] proved that $F_\lambda(t)$ is absolutely continuous with a density in L^2 for a.e. $0.5 < \lambda < 1$.

In this paper we study the moments of Bernoulli convolution. They are defined by

$$M_n = \mathsf{E}Y_\lambda^n = \int_0^1 x^n \, dF_\lambda(x). \tag{4}$$

The main result of this paper

Theorem 1. *Let M_n be defined by (4), then the following holds as $n \to \infty$*

$$M_n = n^{\log_\lambda 2} 2^{\log_\lambda(1-\lambda)+0.5\log_\lambda 2-0.5} e^{\tau(-\log_\lambda n)} \left(1 + \mathcal{O}(n^{-0.99})\right),$$

where

$$\tau(x) = \sum_{k\neq 0} \frac{1}{k} \alpha\left(-\frac{k}{\ln\lambda}\right) e^{2\pi i k x} \tag{5}$$

is 1-periodic function,

$$\alpha(t) = -\frac{1}{2i\,\mathrm{sh}(\pi^2 t)} (1-\lambda)^{2\pi i t} (1-2^{2\pi i t}) \pi^{-2\pi i t} 2^{-2\pi i t} \zeta(2\pi i t), \tag{6}$$

and $\zeta(z)$ is the Riemann zeta function.

Remark 1. *We emphasize that the periodic function $\tau(x)$ is a constant only in the Wintner's cases $\lambda = 2^{-1/k}$, $k = 1, 2, \ldots$.*

Moreover, only in these cases $F_\lambda(x) = Cx^\alpha + \mathcal{O}(x^{\alpha+\varepsilon})$ for some $\alpha > 0$, $\varepsilon > 0$ as $x \to 0$.

We observe that $|\alpha(t)|$ does not depend on λ, $|\alpha(\pm 1/\ln 2)| \approx 10^{-5}$, and due the fast decrease of $|\alpha(t)|$ (see Figure 2) the fluctuating function $\tau(x)$ stays bounded by 10^{-5}.

Remark 2. *We have the following approximations*

$$M_n \approx n^{\log_\lambda 2} 2^{\log_\lambda(1-\lambda)+0.5\log_\lambda 2-0.5}$$

with accuracy 10^{-5} and

$$M_n \approx n^{\log_\lambda 2} 2^{\log_\lambda(1-\lambda)+0.5\log_\lambda 2-0.5} + \alpha\left(-\frac{1}{\ln\lambda}\right) n^{-2\pi i/\ln\lambda} - \alpha\left(\frac{1}{\ln\lambda}\right) e^{+2\pi i/\ln\lambda}$$

with accuracy 10^{-9}.

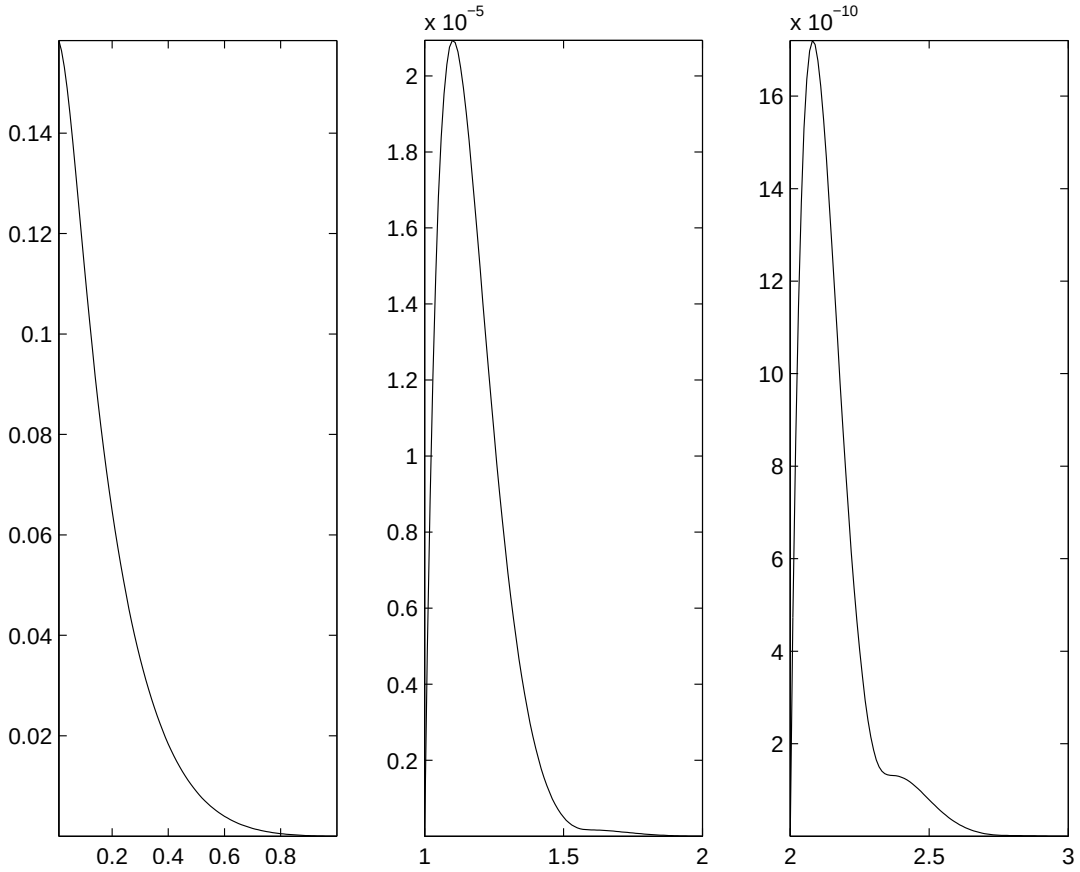

Fig 2. The graphs of $|\alpha(t)|$ for $t \in [0,1]$, $t \in [1,2]$, and $t \in [2,3]$

Proof. Now we give a proof of the theorem using analytic techniques such as poissonization and the Mellin transform

A recursive relation for the moments M_n of any function satisfying (1) is the following

$$M_n = \int_0^1 x^n \, dF_\lambda(x) = \\ = \frac{1}{2}\int_0^1 x^n \, dF_\lambda(\lambda^{-1}x) + \frac{1}{2}\int_0^1 x^n \, dF_\lambda(\lambda^{-1}x - \lambda^{-1} + 1) = \\ = \frac{1}{2}\lambda^n \int_0^1 t^n \, dF_\lambda(t) + \frac{1}{2}\int_0^1 (\lambda t + 1 - \lambda)^n \, dF_\lambda(t).$$

Substituting M_n, we obtain

$$M_n = \frac{1}{2}\lambda^n M_n + \frac{1}{2}\sum_{k=0}^{n} \binom{n}{k} \lambda^k (1-\lambda)^{n-k} M_k, \quad M_0 = 1, \quad n = 0, 1, \ldots. \tag{7}$$

We define the Poisson transform as

$$M(x) = \sum_{n=0}^{\infty} M_n \frac{x^n}{n!} e^{-x}. \tag{8}$$

which exists for all complex x since the series converges due to the estimate $M_m \leq 1$. Substituting (7) в (8), we get

$$M(x) = \\ = \frac{1}{2}\sum_{n=0}^{\infty}\lambda^n M_n \frac{x^n}{n!}e^{-x} + \frac{1}{2}\sum_{n=0}^{\infty}\frac{x^n}{n!}e^{-x}\sum_{k=0}^{n}\binom{n}{k}\lambda^k(1-\lambda)^{n-k}M_k = \\ = \frac{1}{2}M(\lambda x)e^{-(1-\lambda)x} + \frac{1}{2}M(\lambda x). \quad (9)$$

Since $M_0 = 1$, we find that

$$M(x) = \prod_{k=0}^{\infty}\left(\frac{1}{2} + \frac{1}{2}e^{-(1-\lambda)\lambda^k x}\right). \quad (10)$$

Let

$$G(x) = \ln M(x) = \sum_{k=0}^{\infty}\ln\left(\frac{1}{2} + \frac{1}{2}e^{-(1-\lambda)\lambda^k x}\right). \quad (11)$$

The Mellin transform

$$\tilde{G}(z) = \int_0^{\infty} G(x)x^{z-1}dx, \quad (12)$$

is define for

$$-1 < \Re z < 0$$

and satisfies

$$\tilde{G}(z) = \frac{(1-\lambda)^{-z}}{1-\lambda^{-z}}\int_0^{\infty} x^{z-1}\ln\left(\frac{1}{2} + \frac{1}{2}e^{-x}\right)dx,$$

By integration by parts, we obtain

$$\tilde{G}(z) = \frac{(1-\lambda)^{-z}}{1-\lambda^{-z}}\frac{1}{z}\int_0^{\infty}\frac{x^z}{1+e^x}\,dx,$$

Using [15, 3.411.3], we get

$$\tilde{G}(z) = \frac{(1-\lambda)^{-z}}{1-\lambda^{-z}}(1-2^{-z})\Gamma(z)\zeta(z+1), \quad (13)$$

for $-1 < \Re z < 0$.

It is known that the Gamma function and the zeta function are both continuable to the complex plane:

- $\Gamma(z)$ has simple poles at the non-positive integers;
- $\zeta(z)$ has only simple pole at $z = 1$.

Therefore, the function $\tilde{G}(z)$ continuable to the complex plane.

$\tilde{G}(z)$ has a double pole at $z = 0$ and simple poles:

- at the non-positive integers, from $\Gamma(z)$,
- at $z = z_k$, from $(1 - \lambda^{-z})^{-1}$, where

$$z_k = \frac{2\pi i k}{\ln \lambda}, \quad k \neq 0; \tag{14}$$

Via the inverse Mellin transform we find that

$$G(x) = \frac{1}{2\pi i} \int\limits_{-\sigma - i\infty}^{-\sigma + i\infty} \tilde{G}(z) x^{-z}\, dz, \tag{15}$$

where $0 < \sigma < 1$.

In order to find $G(x)$, we apply the residue theorem for the right half-plane $\Re z > -\sigma$.

Now we find residues of the function $\tilde{G}(z)x^{-z}$.

Using the following formula

$$\operatorname{Res}\left(\frac{P(z)}{zQ(z)}, 0\right) = \frac{P'(0)}{Q'(0)} - \frac{P(0)Q''(0)}{2Q'(0)^2},$$

where

$$P(z) = \frac{1 - 2^{-z}}{z}\Gamma(z+1)(1-\lambda)^{-z}x^{-z}(z\zeta(z+1)),$$
$$Q(z) = 1 - \lambda^{-z},$$

we obtain

$$\operatorname{Res}\left(\tilde{G}(z)x^{-z}, 0\right) = \frac{P'(0)}{Q'(0)} - \frac{P(0)Q''(0)}{2Q'(0)^2}.$$

(There $\operatorname{Res}(f(z), z_0)$ denotes the residue of $f(z)$ at the point $z = z_0$.)

Substituting $(z\zeta(z+1))_{z=0} = 1$, $(z\zeta(z+1))'_{z=0} = -\Gamma'(1)$ [15, 9.536], we get

$$P(0) = \ln 2, \quad P'(0) = -\ln x \ln 2 - \ln(1-\lambda)\ln 2 - \frac{1}{2}\ln^2 2,$$
$$Q'(0) = \ln \lambda, \quad Q''(0) = -\ln^2 \lambda.$$

Therefore,

$$\operatorname{Res}\left(\tilde{G}(z)x^{-z}, 0\right) = -\frac{\ln 2}{\ln \lambda}\left(\ln x + \ln(1-\lambda) + \frac{1}{2}\ln 2\right) + \frac{1}{2}\ln 2.$$

The residue at z_k, $k = \pm 1, \pm 2, \ldots$ is

$$\operatorname{Res}\left(\tilde{G}(z)x^{-z}, z_k\right) = \frac{(1-\lambda)^{-z_k}}{\ln \lambda}(1 - 2^{-z_k})\Gamma(z_k)\zeta(z_k+1)x^{-z_k}.$$

$\Gamma(z)$ decreases exponentially fast along vertical lines while $\zeta(z)$ is only polinomial growth as $\Im z \to \pm\infty$. Thus, Collorary 1 to Theorem 4 from [3] applies here and we have

$$G(x) = \frac{\ln 2}{\ln \lambda}\left(\ln x + \ln(1-\lambda) + \frac{1}{2}\ln 2\right) - \frac{1}{2}\ln 2 -$$
$$- \frac{1}{\ln \lambda}\sum_{k \neq 0}(1-\lambda)^{-z_k}(1 - 2^{-z_k})\Gamma(z_k)\zeta(z_k+1)x^{-z_k} + \mathcal{O}(x^{-\gamma}), \tag{16}$$

for every $\gamma > 0$.

Take $\gamma = \log_\lambda 2 + 2$.

Using (16) and (11), we get

$$M(x) = x^{\log_\lambda 2} e^{\tau(\log_\lambda x)} \left(1 + \mathcal{O}(x^{-\gamma})\right),$$

where the function τ is defined in (5).

In order to find M_n, we apply Theorem 10.5 from [14]. To apply Theorem 10.5 from [14], we must check that the conditions required in this theorem are actually satisfied. In particular:

there exist β, $0 < \theta < \pi/2$, and $0 < \eta < 1$ such that the following conditions hold for sufficiently large $|z|$, $z = x + iy$:

- for $z \in S_\theta$
$$\left|\frac{1}{2} + \frac{1}{2}e^{-(1-\lambda)z}\right| \lambda^\beta \le 1 - \eta;$$
- for $z \notin S_\theta$ and some $\alpha < 1$
$$\left|\frac{1}{2} + \frac{1}{2}e^{-(1-\lambda)z}\right| e^{(1-\lambda)x} \le e^{\alpha(1-\lambda)|z|};$$

where $S_\theta = \{z \ : \ |\Im z| \le \theta \Re z\}$.

It can easily be checked that this condition holds for every $\beta > \log_\lambda 2$ and we apply Theorem 10.5 from [14] to yeild

$$M_n = M(n)\left(1 + \mathcal{O}(n^{-0.99})\right).$$

for $\beta = \log_\lambda 2 + 0.01$.

The last step consists of simplifying the function $\alpha(t)$ (6). Using (16), we have

$$\alpha(t) = -\frac{1}{2\pi i}(1-\lambda)^{2\pi i t}(1 - 2^{2\pi i t})\Gamma\left(-2\pi i t + 1\right)\zeta\left(-2\pi i t + 1\right).$$

Applying the functional equation of the Reimann zeta function [15, 9.535.3]

$$\Gamma(z)\zeta(z)\cos\frac{\pi z}{2} = \pi^z 2^{z-1}\zeta(1-z),$$

we obtain (6).

□

References

[1] Bari N.K., *Trigonometric Series*, Holt, Rinehart and Winston, New York, 1967.

[2] Flajolet P., Sedgewick R., *Analytic Combinatorics*, Cambridge University Press, 2008.

[3] Flajolet P., Gourdon X., Dumas P., "Mellin transforms and asymptotics: Harmonic sums", *Theoretical Computer Science*, **144**:1–2 (1995), 3–58.

[4] Erdös P., "On a Family of Symmetric Bernoulli Convolutions", *American Journal of Mathematics*, **61**:4 (1995), 974–976.

[5] Erdös P., "On the Smoothness Properties of a Family of Bernoulli Convolutions", *American Journal of Mathematics*, **62**:1 (1940), 180–186.

[6] Garsia A.M., "Arithmetic Properties of Bernoulli Convolutions", *Transactions of the American Mathematical Society*, **102**:3 (1962), 409–432.

[7] Jessen B., Wintner A., "Distribution Functions and the Riemann Zeta Function", *Transactions of the American Mathematical Society*, **38**:1 (1935), 48–88.

[8] Peres Y., Schlag W., and Solomyak B., "Sixty years of Bernoulli convolutions", *Fractals and Stochastics II (C. Bandt, S. Graf and M. Zaehle, eds.)*, Birkhauser, 2000, 39–65.

[9] Salem R., "Sets of Uniqueness and Sets of Multiplicity", *Transactions of the American Mathematical Society*, **54**:2 (1943), 218–228.

[10] Salem R., "Sets of Uniqueness and Sets of Multiplicity. II", *Transactions of the American Mathematical Society*, **56**:1 (1944), 32–49.

[11] Salem R., "Rectifications to the Papers Sets of Uniqueness and Sets of Multiplicity, I and II", *Transactions of the American Mathematical Society*, **63**:3 (1948), 595–598.

[12] Solomyak B., "On the Random Series $\sum \pm\lambda^n$ (an Erdos Problem)", *The Annals of Mathematics 2nd Ser.*, **142**:3 (1995), 611–625.

[13] Wintner A., "On Convergent Poisson Convolutions", *American Journal of Mathematics*, **57**:4 (1935), 827–838.

[14] Szpankowski W., *Average Case Analysis of Algorithms on Sequences*, John Wiley & Sons, New York, 2001.

[15] Gradstein I. S., Ryzhik I. M., *Table of integrals, Series, and Products*, Academic Press, 1994.

Тимофеев Е.А., "Асимптотика моментов симметричной свертки Бернулли", *Моделирование и анализ информационых систем*, **23**:2 (2016), 185–194.

DOI: 10.18255/1818-1015-2016-2-185-194

Аннотация. Для каждого λ, $0 < \lambda < 1$ определим случайную величину (симметричную свертку Бернулли)

$$Y_\lambda = (1-\lambda)\sum_{n=0}^{\infty} \xi_n \lambda^n,$$

где ξ_n – независимые случайные величины с

$$\mathsf{P}\{\xi_n = 0\} = \mathsf{P}\{\xi_n = 1\} = \frac{1}{2}.$$

Основной результат настоящей работы

$$M_n = \mathsf{E}Y_\lambda^n = n^{\log_\lambda 2} 2^{\log_\lambda(1-\lambda)+0.5\log_\lambda 2 - 0.5} e^{\tau(-\log_\lambda n)} \left(1 + \mathcal{O}(n^{-0.99})\right),$$

где функция

$$\tau(x) = \sum_{k\neq 0} \frac{1}{k} \alpha\left(-\frac{k}{\ln\lambda}\right) e^{2\pi i k x}$$

является периодической с периодом равным 1,

$$\alpha(t) = -\frac{1}{2i\,\mathrm{sh}(\pi^2 t)} (1-\lambda)^{2\pi i t} (1 - 2^{2\pi i t}) \pi^{-2\pi i t} 2^{-2\pi i t} \zeta(2\pi i t),$$

а $\zeta(z)$ – дзета-функция Римана.

Статья публикуется в авторской редакции.

Ключевые слова: моменты, самоподобие, свертка Бернулли, сингулярная функция, преобразование Меллина, асимптотика

Об авторах:
Тимофеев Евгений Александрович, orcid.org/0000-0002-0980-2507,
доктор. физ.-мат. наук, профессор кафедры теоретической информатики,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: timofeevEA@gmail.com

Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 195–210
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 195–210

DOI: 10.18255/1818-1015-2016-2-195-210

УДК 004.048

Коллективные потоковые вычисления: реляционные модели и алгоритмы

Усталов Д. А.

получена 2 апреля 2016

Аннотация. В последнее время краудсорсинг на основе выполения микрозадач получил широкое применение в области анализа неструктурированных данных. Разрабатываются специализированные методики, состоящие из множества этапов обработки исходных данных, требующих согласованности их представления для обеспечения воспроизводимости работы. Данная статья посвящена решению проблемы воспроизводимости и формализации процесса краудсорсинга микрозадачами. Предложена модель коллективных потоковых вычислений на основе расширенной реляционной модели и потоковой модели вычислений. Модель предназначена для обработки исходных данных в виде реляционных отношений путем параллельного выполнения этапов разметки микрозадачами и этапов автоматической синхронизации. Этапы обработки данных и связи между ними записываются с использованием схемы коллективных вычислений, представляющей собой слабо связный ориентированный ациклический граф. Описан синхронный алгоритм выполнения схем коллективных вычислений. Продемонстрированы приложения модели в области компьютерной лингвистики для уточнения лексикализации понятий в электронных тезаурусах и построения родо-видовых отношений между понятиями при помощи краудсорсинга. Процедура «добавить–удалить–подтвердить» позволяет внести в лексикализацию понятий недостающие лексемы и исключить посторонние. Процедура «род–вид–сопоставить» позволяет сформировать гипо-гиперонимические отношения между понятиями на основе соответствующих родо-видовых пар слов. Результаты экспериментов на материалах открытого электронного тезауруса русского языка подтверждают применимость разработанных процедур для развития лексических ресурсов. В экспериментах приняли участие как волонтеры из популярных социальных сетей, так и пользователи бирж краудсорсинга (за вознаграждение в форме микроплатежей).

Ключевые слова: краудсорсинг, потоковые вычисления, реляционная модель, компьютерная лингвистика

Для цитирования: Усталов Д. А., "Коллективные потоковые вычисления: реляционные модели и алгоритмы", *Моделирование и анализ информационных систем*, **23**:2 (2016), 195–210.

Об авторах:
Усталов Дмитрий Алексеевич, orcid.org/0000-0002-9979-2188, аспирант,
Институт математики и механики им. Н.Н. Красовского Уральского отделения Российской академии наук,
ул. Софьи Ковалевской, 16, г. Екатеринбург, 620990 Россия, e-mail: dau@imm.uran.ru

Благодарности:
Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-37-00354 мол_а «Методы автоматизации процесса коллективного построения лингвистических ресурсов». Исследование выполнено при финансовой поддержке РГНФ: проект «Новый открытый электронный тезаурус русского языка» № 13-04-12020 и проект «Интеграция тезаурусов RussNet и YARN» № 16-04-12019.

Введение

Краудсорсинг — способ получения услуг, идей и информации путём соучастия большого количества людей в Интернете [1], заслуживший большую популярность с момента своего появления в 2006 году. Зарубежные исследователи успешно применяют краудсорсинг для построения и разметки языковых ресурсов [2], в том числе лексических онтологий, словарей тональности и др. Отечественные исследователи применяют краудсорсинг преимущественно без денежного вознаграждения участников, вместо этого полагаясь на их *альтруизм* при построении открытых русскоязычных языковых ресурсов, таких как корпус текстов со снятой неоднозначностью [3], корпус текстов для перефразирования [4], электронный тезаурус [5]. В русскоязычной литературе множество участников процесса краудсорсинга получило название «толпа» (англ. *crowd*) [6].

Процесс коллективной работы осуществляется на специализированных платформах — биржах, где участники получают *микроплатежи* за работу — выполнение *микрозадач*. Оплата труда не гарантирует высокого качества результата [7], поэтому каждое задание выполняется несколькими разными участниками одновременно и затем проводится агрегация ответов на основе эвристического или вероятностного алгоритма. Известными биржами краудсорсинга являются MTurk[1], CrowdFlower[2] и «Яндекс.Толока[3]», с недавних пор доступная сторонним заказчикам.

Существует два основных способа повышения эффективности краудсорсинговой разметки: проектирование более совершенных подходов к представлению заданий и разработка математических методов обеспечения качества разметки. Объектом исследования данной работы является проблема представления данных и заданий для разметки перед участниками, не являющимися экспертами в заданной предметной области: биологии, географии, лингвистике и т. д.

В данной статье предложена модель коллективных потоковых вычислений, построенная на основе потоковых вычислений и расширенной реляционной модели. Модель предназначена для обработки исходных данных в виде реляционных отношений путём параллельного выполнения этапов коллективной разметки и этапов автоматической синхронизации. В разделе 1 представлен обзор литературы по тематике работы. В разделе 2 описана модель коллективных потоковых вычислений. В разделе 3 приведён алгоритм выполнения схем таких вычислений. В разделе 4 продемонстрированы приложения разработанных моделей для решения задач компьютерной лингвистики. В заключении подведены итоги работы и предложены пути дальнейших исследований.

1. Обзор литературы

Обзор литературы включает два направления: (1) модели коллективных вычислений и (2) методики решения задач при помощи краудсорсинга.

[1] https://www.mturk.com/mturk/welcome
[2] https://www.crowdflower.com/
[3] https://toloka.yandex.com/

1.1. Модели коллективных вычислений

Попытки адаптации популярных моделей компьютерного программирования к области краудсорсинга производятся с начала 2010-х гг., причём особое внимание уделяется различным вычислительным моделям распределённых и параллельных вычислений. Среди примеров следует отметить CrowdForge — адаптацию парадигмы MapReduce, в которой этапы *отображения* и *редукции* заменены на краудсорсинговые задачи [8], а также CrowdDB — расширение языка SQL, включающее оператор `CROWD` для запуска заданий на MTurk при выполнении запроса на манипулирование данными [9]. Также известны эксперименты по переносу различных нотаций моделирования бизнес-процессов и парадигм декларативного программирования [10].

Активное развитие технологий распределённых вычислений и снижение входных барьеров в данную область привело к росту популярности концепции потоковых вычислений [11], первоначально использованной при создании Манчестерской потоковой ЭВМ, а позднее — мультиклеточных процессоров [12]. Система CrowdWeaver адаптирует данную вычислительную модель для краудсорсинга [13], однако реализация данной системы, как и сведения о принципах её функционирования и используемой системе типов, закрыты.

Интерес к потоковой вычислительной модели проявляется у исследователей «Интернета вещей», использующих данную модель для интеграции датчиков различных производителей в одном помещении [14]. При описании потоковых вычислений применяют различные теоретико-графовые подходы, в том числе сети процессов Кана [15] и сети Петри [16].

1.2. Методики решения задач

Отдельного внимания заслуживают краудсорсинговые методики решения трудноформализуемых задач обработки и анализа неструктурированных данных.

Одной из самых известных методик краудсорсинга является Find-Fix-Verify, предложенная Бернштейном и соавторами для перефразирования текстовых документов [17]: на этапе *поиска* участники выделяют фрагменты исходного текста для последующего исправления, затем на этапе *исправления* участники предлагают варианты исправления для ранее выделенных фрагментов, после чего на этапе *проверки* участники голосуют за лучшие варианты из предложенных.

Норона и соавторы предложили методику PlateMate для определения калорийности пищи на основе фотографий [18]. Методика состоит из трёх последовательных этапов: на этапе *отметить* участники выделяют граничной рамкой блюда на фотографии, на этапе *опознать* участники выбирают из списка соответствующие рамкам блюда, на этапе *измерить* участники указывают размер порции каждого блюда.

Краудсорсинговая методика CrowdER предназначена для обнаружения дубликатов и связывания записей в базах данных различных товаров [19]. Задача решается в два этапа: сначала выполняется *оценка* подобия между парами товаров, затем осуществляется *проверка* пар записей, оставшихся после фильтрации.

При решении задач обработки естественного языка необходимы специализированные словари и иные языковые ресурсы, которые в последнее время всё чаще

создаются при помощи краудсорсинга [2]. Биманн предложил трёхэтапную методику Turk Bootstrap Word Sense Inventory (TWSI) для коллективного построения синсетов на основе явления лексической замещаемости слов [20]: на этапе *подстановки слов* участники предлагают синонимы для слов в заданном контексте, затем на этапе *выравнивания смыслов* участники оценивают семантическую близость пар слов, после чего на этапе *сопоставления значений* участники оценивают качество выведенных значений слов по примерам употребления.

2. Коллективные потоковые вычисления

На практике краудсорсинг на основе выполнения микрозадач предполагает обработку некоторого набора исходных данных в один или несколько этапов, причём множество таких этапов известно заранее. В рамках модели коллективных потоковых вычислений предлагается использовать реляционную модель данных, расширенную операциями вкладывания (англ. *nest*), выкладывания (англ. *unnest*) и расширенной проекции для упрощения работы с массивами и более сложными структурами данных [21,22]. Выбор реляционной модели обусловлен практическими соображениями: строгостью системы типов, популярностью модели, а также возможностью адаптации более новых документоориентированных подходов [23]. Введём понятие схемы коллективных вычислений для описания этапов обработки данных.

Определение 1. *Схема коллективных вычислений — слабо связный ориентированный ациклический граф W со множеством рёбер E, множество вершин которого образуется объединением множества этапов разметки S, множества этапов синхронизации Y и множества источников данных D.*

$$W = (S \cup Y \cup D, E) \tag{1}$$

Важно отметить, что все вершины W являются реляционными отношениями с заданными заголовками. Для удобства записи обозначим множество всех элементов схемы как $V = S \cup Y \cup D$ и множество входящих вершин в вершину $v \in V$ как $In(v)$. Обозначим заголовок отношения $v \in V$ как $H(v)$, тело как $B(v)$, а первичный ключ как $\mathrm{PK}(v) \subseteq H(v)$.

Определение 2. *Этап разметки $s \in S$ — это реляционное отношение, тело которого получено путём преобразования толпой участников кортежей единственного входящего отношения: $In(s) \subset V \wedge |In(s)| = 1$.*

Примером этапа разметки является задание на оценку семантической близости пары слов с одиночным выбором из вариантов «не связаны», «слабо связаны», «связаны» и «сильно связаны» [24]. Входным отношением в данном примере является таблица интересующих пар слов с заданным первичным ключом, а выходным отношением является таблица суждений участников о семантической близости каждой пары слов, образующая внутреннее соединение с исходной таблицей пар слов.

Определение 3. *Этап синхронизации $y \in Y$ — это реляционное отношение, тело которого получено путём автоматической обработки двух и более входящих отношений: $In(y) \subset V \wedge |In(y)| > 1$.*

Этапы синхронизации задают правила агрегации результатов выполнения этапов разметки с другими источниками данных, необходимыми для успешного решения исходной задачи. Примером этапа синхронизации является этап подготовки заданий по выравниванию смыслов методики TWSI, в которой для выбранных участниками лексических замещений подбираются примеры использования по корпусу текстов [20]. Входными отношениями являются таблица выявленных синонимов и табличное представление корпуса текстов. Выходным отношением является внутреннее соединение этих отношений с ограничением по количеству примеров на синоним.

Определение 4. *Источник данных $d \in D$ — это реляционное отношение, тело которого получено заранее и не зависит от других элементов схемы коллективных вычислений: $In(d) = \varnothing$.*

В качестве источников данных выступают словари, справочники, корпусы текстов, коллекции изображений и т. п. Источники данных получены заранее и не изменяются в процессе работы.

3. Выполнение схем коллективных вычислений

Этапы синхронизации и средства предварительной обработки данных реализуются в виде программного обеспечения различной сложности на разных языках программирования, что вносит дополнительные требования к детерминированности описания методики краудсорсинга. Введём понятие согласованности схемы коллективных вычислений.

Определение 5. *Согласованной схемой коллективных вычислений является схема, каждый кортеж каждого элемента которой однозначно идентифицирует породившие его кортежи.*

$$\forall v \in V \Big(\forall v' \in In(v) \big(H(v) \cap H(v') \supseteq \mathrm{PK}(v') \big) \Big). \tag{2}$$

В несогласованных схемах коллективных вычислений сведения между этапами либо передаются не целиком, что исключает возможность определить породившие кортежи, либо возникает неоднозначность из-за использования операций внешнего соединения без ограничений.

Алгоритм 1 осуществляет синхронное выполнение согласованных схем коллективных вычислений, обеспечивая удовлетворение зависимостей каждого элемента схемы. Каждый последующий элемент схемы не может быть выполнен до тех пор, пока каждая из его зависимостей не будет удовлетворена, т. е. завершена. Это обеспечивается применением битовой маски M, содержащей сведения о завершённости этапов: **false** — не завершён, **true** — завершён. При инициализации, битовая маска содержит значения **true** только для источников данных. В результате работы алгоритма такое значение получают все поля битовой маски.

Алгоритм 1 Синхронный алгоритм выполнения схем коллективных вычислений
Algorithm 1 Synchronous algorithm for executing collaborative computation workflows

Require: $V = S \cup Y \cup D, |D| > 0$
  for all $v \in V$ **do**
    $M_v \leftarrow (v \in D)$ ▷ Отметить все источники данных как завершённые.
  end for
  parallel for all $v \in V \setminus D$ **do**
    WAIT$(\forall v' \in In(v)(M_{v'} = \textbf{true}))$ ▷ Подождать завершение всех зависимостей.
    $B(v) \leftarrow$ RUN(v) ▷ Выполнить текущий элемент.
    $M_v \leftarrow \textbf{true}$ ▷ Отметить элемент как завершённый.
  end for
Ensure: $(\forall v \in V)(M_v = \textbf{true})$

Принцип работы алгоритма 1 аналогичен алгоритму обхода графа в ширину с несколькими отличиями: обход может начинаться с нескольких стартовых вершин, каждая вершина обрабатывается параллельно и только один раз, и каждая вершина ожидает пометки всех инцидентных ей вершин как завершённых. Требование $|D| > 0$ обеспечивает возможность обхода всех элементов схемы начиная с источников данных и наряду с условием ацикличности предотвращает взаимные блокировки при выполнении алгоритма.

Параллельная обработка достигается путём запуска каждой итерации рабочего цикла в отдельном потоке выполнения, что позволяет размещать задания соответствующих этапов разметки по мере выполнения предыдущих этапов. В данной работе подход к параллелизму реализован на основе механизма зелёных потоков [25], хотя возможно применение других решений: легковесных процессов, пулов потоков выполнения и т. п.

4. Приложения в компьютерной лингвистике

В данном разделе представлены две краудсорсинговые процедуры очистки и обогащения лексических ресурсов. Краудсорсинговая процедура «добавить–удалить–подтвердить» предназначена для уточнения лексикализации понятий в электронных тезаурусах и состоит из этапов *добавления* недостающих слов, *удаления* посторонних слов и *подтверждения* предложенных изменений [26]. Процедура «род–вид–сопоставить» предназначена для построения семантических отношений между понятиями: участниками соотносятся понятия в парах *род* и *вид*, после чего производится *сопоставление* подтверждённых пар [27]. Эксперименты проводились с использованием сервиса управления процессом краудсорсинга [28], развёрнутом в СКЦ ИММ УрО РАН[4]. Результаты обоих экспериментов доступны в виде открытых данных[5,6].

[4] http://parallel.uran.ru/
[5] http://ustalov.imm.uran.ru/pub/arc-yarn-100.tar.gz
[6] http://ustalov.imm.uran.ru/pub/gsm-ainl.tar.gz

4.1. Уточнение лексикализации понятий

Процедура «добавить–удалить–подтвердить» предназначена для повышения качества синонимических рядов в лексических ресурсах путём добавления в них недостающих слов и удаления из них лишних слов (рис. 1). Входными данными в данной процедуре является множество синсетов $\mathbb{S}$, подлежащих обработке, и множество слов-кандидатов $\mathbb{W}$ к включению в состав соответствующих синсетов. На каждом этапе решаются различные задачи:

- на этапе «добавить» (Add) участнику предоставляется синсет и список слов-кандидатов на включение в него. Участник выбирает недостающие слова для включения в синсет без искажения его общего значения;
- на этапе «удалить» (Remove) участнику предоставляется синсет. Участник может указать слова, искажающие общее значение синсета;
- на этапе «подтвердить» (Confirm) участнику предоставляется исходный синсет и его модифицированный вариант. Участник указывает лучший вариант из предложенных.

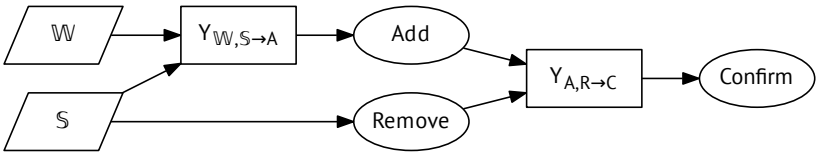

Рис. 1. Процедура «добавить–удалить–подтвердить»

Fig. 1. “Add–Remove–Confirm”

Результатом выполнения процедуры является множество модифицированных синсетов. Важно отметить, что этапы «добавить» и «удалить» выполняются одновременно и независимо друг от друга. Данная процедура записана в виде схемы коллективных вычислений ARC в формуле (3), описание отношений приведено в таблице 1. Подчёркивание при записи заголовка реляционного отношения обозначает элементы его первичного ключа. Поскольку атрибуты *words*, *added* и *removed* представляют собой списки слов, то при определении этапа синхронизации $Y_{A,R\to C}$ выполняются теоретико-множественные операции объединения и вычитания.

$$S_{ARC} = \{A, R, C\} \tag{3.1}$$

$$Y_{ARC} = \{Y_{\mathbb{W},\mathbb{S}\to A}, Y_{A,R\to C}\} \tag{3.2}$$

$$D_{ARC} = \{\mathbb{W}, \mathbb{S}\} \tag{3.3}$$

$$\begin{aligned} E_{ARC} = \big\{&(\mathbb{S}, Y_{\mathbb{W},\mathbb{S}\to A}), (\mathbb{W}, Y_{\mathbb{W},\mathbb{S}\to A}), (\mathbb{S}, R), (Y_{\mathbb{W},\mathbb{S}\to A}, A),\\ &(A, Y_{A,R\to C}), (R, Y_{A,R\to C}), (Y_{A,R\to C}, C)\big\} \end{aligned} \tag{3.4}$$

$$ARC = \big(S_{ARC} \cup Y_{ARC} \cup D_{ARC}, E_{ARC}\big) \tag{3.5}$$

Таблица 1. Элементы процедуры «добавить–удалить–подтвердить»
Table 1. Elements of "Add–Remove–Confirm"

Элемент	**Определение**
$\mathbb{W}$	$H(\mathbb{W}) = \{(\underline{sid}, \texttt{INT}), (candidates, \texttt{TEXT[]})\}$
$\mathbb{S}$	$H(\mathbb{S}) = \{(\underline{sid}, \texttt{INT}), (words, \texttt{TEXT[]})\}$
$Y_{\mathbb{W},\mathbb{S}\to A}$	$\mathbb{W} \bowtie \mathbb{S}$
A	$H(A) = \{(\underline{aid}, \texttt{INT}), (sid, \texttt{INT}), (added, \texttt{TEXT[]})\}$
R	$H(R) = \{(\underline{rid}, \texttt{INT}), (sid, \texttt{INT}), (removed, \texttt{TEXT[]})\}$
$Y_{A,R\to C}$	$\sigma_{words \neq words'}\big(\pi_{sid,aid,rid,words,\ \mathbb{S}.words \cup A.added \setminus R.removed \to words'}(\mathbb{S} \bowtie A \bowtie R)\big)$
C	$H(C) = \{(\underline{cid}, \texttt{INT}), (aid, \texttt{INT}), (rid, \texttt{INT}), (sid, \texttt{INT}), (b, \texttt{BOOL})\}$

Теорема 1. *Схема коллективных вычислений «добавить–удалить–подтвердить» является согласованной.*

Доказательство. Исходные данные представлены в отношениях $\mathbb{W}, \mathbb{S} \in D$. Проверим условие (2) для остальных отношений $V \setminus D$ по табл. 1.

1. $In(Y_{\mathbb{W},\mathbb{S}\to A}) = \{\mathbb{W}, \mathbb{S}\}$, причём $Y_{\mathbb{W},\mathbb{S}\to A}$ является проекцией $\mathbb{W} \bowtie \mathbb{S}$. Значит, $H(Y_{\mathbb{W},\mathbb{S}\to A}) \cap \big(H(\mathbb{W}) \cup H(\mathbb{S})\big) = \mathrm{PK}(\mathbb{W}) \cup \mathrm{PK}(\mathbb{S})$.

2. $In(A) = \{Y_{\mathbb{W},\mathbb{S}\to A}\}$, причём $H(A) \cap H(Y_{\mathbb{W},\mathbb{S}\to A}) = \mathrm{PK}(Y_{\mathbb{W},\mathbb{S}\to A}) = \mathrm{PK}(\mathbb{W}) \cup \mathrm{PK}(\mathbb{S})$.

3. $In(R) = \{\mathbb{S}\}$, причём $H(R) \cap H(\mathbb{S}) = \mathrm{PK}(\mathbb{S})$.

4. $In(Y_{A,R\to C}) = \{A, R\}$, причём $Y_{A,R\to C}$ является фильтрацией, заданной на проекции $\mathbb{S} \bowtie A \bowtie R$. Значит, $H(Y_{A,R\to C}) \cap \big(H(A) \cup H(R)\big) = \mathrm{PK}(A) \cup \mathrm{PK}(R)$.

5. $In(C) = \{Y_{A,R\to C}\}$, причём $H(C) \cap H(Y_{A,R\to C}) = \mathrm{PK}(Y_{A,R\to C}) = \mathrm{PK}(A) \cup \mathrm{PK}(R)$.

Каждый кортеж каждого отношения однозначно идентифицирует породившие его кортежи. Следовательно, схема (3) является согласованной. □

Исследование применимости процедуры «добавить–удалить–подтвердить» проводилось по материалам открытого электронного тезауруса русского языка Yet Another RussNet [5], поскольку он распространяется на условиях свободной лицензии Creative Commons (CC BY-SA), создан при помощи краудсорсинга, и содержит большое количество дублирующих друг друга понятий [5]. В качестве данных для эксперимента использовано подмножество тезауруса, состоящее из ста синсетов, для которых имеется большое количество дубликатов. Обнаружение дубликатов выполнялось эвристическим методом, объединяющим пару синсетов при наличии не менее двух общих слов [29].

Участники были приглашены из социальных сетей VK, Facebook и Twitter путём публикации открытого вызова, в результате чего время выполнения схемы коллективных вычислений (3) алгоритмом 1 составило 231 мин., в течение которых получено 1313 ответов на 300 заданий. В среднем на одно задание этапов «добавить» и «удалить» приходилось пять ответов, на одно задание «подтвердить» — три ответа. Ответы агрегировались простым голосованием большинства.

Для оценки качества результата были привлечены два эксперта. Каждый эксперт видел исходный синсет s и его модифицированную версию s'. От экспертов требовалось выставить оценку «0» в случае, если синсет s' не улучшился по сравнению с исходным синсетом s, или выставить оценку «1», если синсет существенно улучшился в результате применения процедуры. Синсеты, которые, по мнению участников, не изменились в ходе разметки, были исключены из процесса оценки. Таким образом, осталось всего 84 синсета из 100.

Оба эксперта установили, что $\frac{70}{84} = 83\,\%$ оставшихся синсетов существенно улучшились по результатам эксперимента (Таблица 2), хотя согласованность их суждений по значению коэффициента каппа Флейса [30] достаточно низка ($\kappa = 0,143$). Это обусловлено скошенностью распределения ответов: 14 оценок «0» и 70 за оценку «1», что является известной проблемой данного коэффициента [31].

Таблица 2. Результаты эксперимента «добавить–удалить–подтвердить»
Table 2. Results for “Add–Remove–Confirm”

Участники		**Эксперты**	
Изменилось	84	Улучшилось	70
Не изменилось	16	Не улучшилось	14
Всего	100	**Всего**	84

Несмотря на это, ответы экспертов согласуются хорошо: различается лишь 20 суждений из 84, что подтверждается индексом Жаккара, равным $1 - \frac{20}{84} = 74\,\%$. Только $\frac{4}{84} = 5\,\%$ всех синсетов отмечены обоими экспертами как «не улучшившиеся». Каждая из этих ошибок принадлежит разному классу:

- участник перепутал гипероним и синоним;
- участник перепутал гипоним и синоним;
- участник перепутал когипоним и синоним;
- добавлено слово, явно не соответствующее синсету.

Это показывает, что участники, не являющиеся специалистами-лексикографами, не всегда способны корректно разделить отношение синонимии от отношения гипо- и гиперонимии. На практике слова часто замещаются гиперонимами, например: «*Лев* был голоден. *Зверь* рычал.». Оценки экспертов различались в $\frac{20}{84} = 24\,\%$ случаев, разделяемых на три класса:

- синсет, описывающий два или более понятий, остался неоднозначным даже после удаления лишних слов;

- большое количество корректных синонимов добавлено к синсету, однако лишнее слово не было удалено;
- добавленные слова образуют достаточно хороший синсет, однако его значение отличается от значения исходного синсета.

Полученные результаты подтверждают применимость процедуры «добавить–удалить–подтвердить» для уточнения лексикализации понятий в электронных тезаурусах.

4.2. Построение родо-видовых отношений

Процедура «род–вид–сопоставить» предназначена для построения родо-видовых отношений между понятиями в электронных тезаурусах (рис. 2). Входными данными в данной процедуре является множество синсетов $\mathbb{S}$ и множество родо-видовых пар между словами $\mathbb{R}$. Этапы процедуры предполагают решение следующих задач:

- на этапе «род» (Genus) задана родо-видовая пара слов и синсет. Участник должен подтвердить, что синсет корректно представляет *род* для слова из пары;
- на этапе «вид» (Species) задана родо-видовая пара слов и синсет. Участник должен подтвердить, что синсет корректно представляет *вид* для слова из пары;
- на этапе «сопоставить» (Match) задана пара синсетов. Участник должен подтвердить, что данная пара синсетов образует осмысленное родо-видовое отношение.

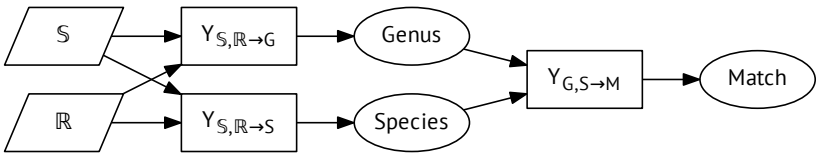

Рис. 2. Процедура «род–вид–сопоставить»

Fig. 2. “Genus–Species–Match”

Результатом выполнения процедуры является множество родо-видовых пар между синсетами. Этапы «род» и «вид» выполняются одновременно и независимо друг от друга. Данная процедура записана в виде схемы коллективных вычислений *GSM* в формуле (4), описания отношений приведены в таблице 3. Запись $\mu(\cdot)$ обозначает операцию выкладывания, упомянутую в разделе 2.

$$S_{GSM} = \{G, S, M\} \tag{4.1}$$

$$Y_{GSM} = \{Y_{\mathbb{S},\mathbb{R}\to G}, Y_{\mathbb{S},\mathbb{R}\to S}, Y_{G,S\to M}\} \tag{4.2}$$

$$D_{GSM} = \{\mathbb{S}, \mathbb{R}\} \tag{4.3}$$

$$E_{GSM} = \big\{(\mathbb{S}, Y_{\mathbb{S},\mathbb{R}\to G}), (\mathbb{S}, Y_{\mathbb{S},\mathbb{R}\to S}), (\mathbb{R}, Y_{\mathbb{S},\mathbb{R}\to G}), (\mathbb{R}, Y_{\mathbb{S},\mathbb{R}\to S}), (Y_{\mathbb{S},\mathbb{R}\to G}, G), (Y_{\mathbb{S},\mathbb{R}\to S}, S), (S, Y_{G,S\to M}), (G, Y_{G,S\to M}), (Y_{G,S\to M}, M)\big\} \tag{4.4}$$

$$GSM = \big(S_{GSM} \cup Y_{GSM} \cup D_{GSM}, E_{GSM}\big) \tag{4.5}$$

Таблица 3. Элементы процедуры «род–вид–сопоставить»
Table 3. Elements of “Genus–Species–Match”

Элемент	**Определение**
$\mathbb{S}$	$H(\mathbb{S}) = \big\{(\underline{sid}, \texttt{INT}), (words, \texttt{TEXT[]})\big\}$
$\mathbb{R}$	$H(\mathbb{R}) = \big\{(\underline{hypernym}, \texttt{TEXT}), (\underline{hyponym}, \texttt{TEXT})\big\}$
$Y_{\mathbb{S},\mathbb{R}\to G}$	$\mathbb{R} \bowtie \pi_{sid,words,\mu(words)\to hypernym}(\mathbb{S})$
$Y_{\mathbb{S},\mathbb{R}\to S}$	$\mathbb{R} \bowtie \pi_{sid,words,\mu(words)\to hyponym}(\mathbb{S})$
G	$H(G) = \big\{(\underline{gid}, \texttt{INT}), (sid, \texttt{INT}), (hypernym, \texttt{TEXT}), (hyponym, \texttt{TEXT}), (b, \texttt{BOOL})\big\}$
S	$H(S) = \big\{(\underline{spid}, \texttt{INT}), (sid, \texttt{INT}), (hypernym, \texttt{TEXT}), (hyponym, \texttt{TEXT}), (b, \texttt{BOOL})\big\}$
$Y_{G,S\to M}$	$\sigma_{\substack{sid_1\neq sid_2 \\ \wedge b_1=b_2=\textbf{true}}} \big(\pi_{\substack{gid,sid\to sid_1,b\to b_1, \\ hypernym,hyponym}}(G) \underset{\substack{G.hypernym=S.hypernym \\ \wedge G.hyponym=S.hyponym}}{\bowtie} \pi_{\substack{spid,sid\to sid_2,b\to b_2, \\ hypernym,hyponym}}(S)\big)$
M	$H(M) = \big\{(\underline{mid}, \texttt{INT}), (gid, \texttt{INT}), (spid, \texttt{INT}), (b, \texttt{BOOL})\big\}$

Теорема 2. *Схема коллективных вычислений «род–вид–сопоставить» является согласованной.*

Доказательство. Исходные данные представлены в отношениях $\mathbb{S}, \mathbb{R} \in D$. Как и при доказательстве теоремы 1, проверим условие (2) для остальных отношений $V \setminus D$ по табл. 3.

1. $In(Y_{\mathbb{S},\mathbb{R}\to G}) = \{\mathbb{S}, \mathbb{R}\}$, причём $Y_{\mathbb{S},\mathbb{R}\to G}$ является естественным соединением $\mathbb{S}$ и $\mathbb{R}$. Значит, $H(Y_{\mathbb{S},\mathbb{R}\to G}) \cap \big(H(\mathbb{S}) \cup H(\mathbb{R})\big) = \mathrm{PK}(\mathbb{S}) \cup \mathrm{PK}(\mathbb{R})$. Аналогичное утверждение справедливо для $Y_{\mathbb{S},\mathbb{R}\to S}$ с точностью до обозначений.

2. $In(G) = \{Y_{\mathbb{S},\mathbb{R}\to G}\}$, причём $H(G) \cap H(Y_{\mathbb{S},\mathbb{R}\to G}) = \mathrm{PK}(Y_{\mathbb{S},\mathbb{R}\to G}) = \mathrm{PK}(\mathbb{S}) \cup \mathrm{PK}(\mathbb{R})$. Аналогичное утверждение справедливо для S с точностью до обозначений.

3. $In(Y_{G,S\to M}) = \{G, S\}$, причём $Y_{G,S\to M}$ является фильтрацией, заданной на проекции тета-соединения G и S по $\mathrm{PK}(\mathbb{R})$. Следовательно, $H(Y_{G,S\to M}) \cap \big(H(G) \cup H(S)\big) = \mathrm{PK}(G) \cup \mathrm{PK}(S)$ с учётом переименований.

4. $In(M) = \{Y_{G,S\to M}\}$, причём $H(M) \cap H(Y_{G,S\to M}) = \mathrm{PK}(Y_{G,S\to M}) = \mathrm{PK}(G) \cup \mathrm{PK}(S)$.

Каждый кортеж каждого отношения однозначно идентифицирует породившие его кортежи. Следовательно, схема (4) является согласованной. □

Исследование применимости процедуры «род–вид–сопоставить» проводилось по материалам открытого электронного тезауруса русского языка Yet Another RussNet [5]. При подготовке данных осуществлялось сопоставление синсетов со словарём предметной области «безопасность жизнедеятельности[7]»: набор данных для построения родо-видовых отношений состоял из 2271 синсета и 383 кандидатов-отношений.

В отличие от предыдущего эксперимента с привлечением волонтёров из социальных сетей, данный эксперимент проводился на краудсорсинговой бирже TurboText[8], поддерживающей размещение и выполнение микрозадач. Минимальное вознаграждение участника на данной бирже составляет 4 руб., среднее вознаграждение на момент первого октября 2015 г. — 11 руб. Для удобства разметки, задания группировались в пакеты из нескольких заданий. Вознаграждение участников на трёх этапах выполнения данной процедуры составляло 5 руб. Каждое задание выполнялось не менее чем пятью различными участниками. При отображении многословных синсетов участникам демонстрировались только первые пять слов, отсортированные по убыванию частоты появления в Национальном корпусе русского языка [32].

Выполнение схемы коллективных вычислений (4) алгоритмом 1 заняло 264 мин. Получено 13 056 ответов на 2558 заданий; на каждое задание собиралось пять ответов от разных участников. На этапах «род» и «вид» ответы агрегировались простым голосованием большинства; на финальном этапе «сопоставить» было проведено сравнение нескольких методов статистического вывода ответов: MV — голос большинства, KOS — итеративный алгоритм Каргера–Оха–Шаха, ZenCrowd — вероятностный метод на основе фактор-графов [27, с. 122].

Отдельный интерес представлял вопрос целесообразности применения краудсорсинга для решения такой задачи, поскольку сведения о синонимических рядах и родо-видовых парах слов могут быть использованы для автоматического построения отношений эвристическим образом. В заданной родо-видовой паре $(g, s) \in \mathbb{R}$, представленной множеством родовых синсетов $\mathbb{S}_g \subseteq \mathbb{S}$ и видовых синсетов $\mathbb{S}_s \subseteq \mathbb{S}$, между парой синсетов $S_g \in \mathbb{S}_g$ и $S_s \in \mathbb{S}_s$ строится отношение, если существует хотя бы ещё одна родо-видовая пара, связывающая эти синсеты:

$$\left|\{s' : \exists(g, s') \in \mathbb{R}\} \cap \{g' : \exists(g', s) \in \mathbb{R}\}\right| > 1. \tag{5}$$

Полученные результаты были обработаны экспертом: определено количество верных положительных TP, верных отрицательных TN, ложных положительных FP и ложных отрицательных FN ответов, по которым вычислена точность P, полнота R и F_1-мера (табл. 4). Дополнительно результаты выполнения эвристического метода (5) приведены в строке «Эвристика».

Исследование результатов выявило три типа ошибок:

- ошибки участников, состоящие в некорректном понимании лексических значений в заданиях;
- ошибки в синсетах, вызванные чрезмерной общностью или узостью выраженных в них понятий;

[7] http://www.mchs.gov.ru/dop/terms
[8] http://www.turbotext.ru/

Таблица 4. Результаты эксперимента «род–вид–сопоставить»
Table 4. Results for “Genus–Species–Match”

Метод	**TP**	**TN**	**FP**	**FN**	**P**	**R**	**F_1**
Эвристика	40	102	17	128	0,70	0,24	0,36
MV	129	57	62	39	0,68	0,77	0,72
KOS	142	63	56	26	0,72	0,84	0,78
ZenCrowd	146	69	50	22	**0,74**	**0,87**	**0,80**

- ошибки в данных, обусловленные некорректными исходными данными из недоверенных источников, например слово «город» в качестве гиперонима в родовидовой паре (место, город), или наличием таких бессмысленных синсетов, как {страна, столица, область, район}.

Несмотря на то, что каждое задание было выполнено не менее чем пятью разными участниками, доля совпавших ответов низка и составляет 55 %. При этом процедура проявила себя устойчивой к ошибкам, хотя их количество можно снизить путём дополнительной очистки исходных данных. Полученные результаты подтверждают применимость процедуры «род–вид–сопоставить» для построения родо-видовых отношений между синсетами.

Заключение

Предложенная модель коллективных потоковых вычислений основана на потоковой вычислительной модели и расширенной реляционной модели для представления и выполнения схем решения различных задач анализа и обработки данных при помощи краудсорсинга. На основе данной модели разработаны краудсорсинговые процедуры «добавить–удалить–подтвердить» — для уточнения лексикализации понятий и «род–вид–сопоставить» — для построения родо-видовых отношений между ними. Результаты проведённых экспериментов подтверждают применимость этих процедур для развития электронных лексических ресурсов. В качестве следующего шага данного исследования будет проведено построение открытого тезауруса предметной области при помощи предложенных процедур.

Большой интерес для дальнейшей работы представляют два направления исследований: интеграция с медицинскими технологиями и адаптация концепций из области распределённых вычислений. Применение электроэнцефалографии и иных электрофизиологических методов исследования функционального состояния головного мозга открывает новые перспективы как для изучения сложности заданий для участников, так и для разработки более совершенных человеко-машинных интерфейсов [33]. Развитие механистических подходов к краудсорсингу позволит создать ещё более эффективные процедуры решения важных задач анализа, обработки и представления данных в самых разных областях науки и промышленности [34].

Благодарности. При выполнении экспериментов использована инфраструктура суперкомпьютера «Уран» ИММ УрО РАН. Автор благодарит А. В. Созыкина и

Д. И. Игнатова за ценные замечания по содержанию работы, а также Ю. А. Киселёва за совместное проведение эксперимента «добавить–удалить–подтвердить» в работе [26].

Список литературы / References

[1] Estellés-Arolas E., González-Ladrón-de Guevara F., "Towards an integrated crowdsourcing definition", *Journal of Information Science*, **38**:2 (2012), 189–200, http://jis.sagepub.com/content/38/2/189.

[2] *The People's Web Meets NLP*, eds. Gurevych I., Kim J., Springer Berlin Heidelberg, 2013, http://dx.doi.org/10.1007/978-3-642-35085-6.

[3] Bocharov V., Alexeeva S., Granovsky D. et al., "Crowdsourcing morphological annotation", Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue", 2013, 109–124, http://www.dialog-21.ru/digests/dialog2013/materials/pdf/BocharovVV.pdf.

[4] Pronoza E., Yagunova E., "Comparison of Sentence Similarity Measures for Russian Paraphrase Identification", Proceedings of the AINL-ISMW FRUCT, 2015, 74–82, http://dx.doi.org/10.1109/AINL-ISMW-FRUCT.2015.7382973.

[5] Braslavski P., Ustalov D., Mukhin M., "A Spinning Wheel for YARN: User Interface for a Crowdsourced Thesaurus", Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics, 2014, 101–104, http://www.aclweb.org/anthology/E/E14/E14-2026.pdf.

[6] Шуровьески Дж., *Мудрость толпы*, Манн, Иванов и Фербер, 2013, http://www.mann-ivanov-ferber.ru/books/paperbook/the-wisdom-of-crowds/; [Surowiecki J., *The Wisdom of Crowds*, Doubleday, 2004, (in Russian).]

[7] Gadiraju U., Kawase R., Dietze S. et al., "Understanding Malicious Behavior in Crowdsourcing Platforms: The Case of Online Surveys", Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2015, 1631–1640, http://dx.doi.org/10.1145/2702123.2702443.

[8] Kittur A., Smus B., Khamkar S., et al., "CrowdForge: Crowdsourcing Complex Work", Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, 2011, 43–52, http://dx.doi.org/10.1145/2047196.2047202.

[9] Franklin M. J., Kossmann D., Kraska T. et al., "CrowdDB: Answering Queries with Crowdsourcing", Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, 2011, 61–72, http://dx.doi.org/10.1145/1989323.1989331.

[10] Kucherbaev P., Daniel F., Tranquillini S. et al., "Crowdsourcing Processes: A Survey of Approaches and Opportunities", *IEEE Internet Computing*, **20**:2 (2016), 50–56, http://dx.doi.org/10.1109/MIC.2015.96.

[11] Johnston W. M., Hanna J. R. P., Millar R. J., "Advances in Dataflow Programming Languages", *ACM Comput. Surv.*, **36**:1 (2004), 1–34, http://dx.doi.org/10.1145/1013208.1013209.

[12] Стрельцов Н. В., "Архитектура и реализация мультиклеточных процессоров", Труды V Международной научной конференции «Параллельные вычисления и задачи управления» (Москва, 26–28 октября 2010 г.), 2010, 1087–1104; [Streltsov N. V., "Arkhitektura i realizatsiya multikletochnykh protsessorov", Trudy V Mezhdunarodnoy nauchnoy konferentsii "Parallelnye vychisleniya i zadachi upravleniya" (Moskva, 26–28 oktyabrya 2010 g.), 2010, 1087–1104, (in Russian).]

[13] Kittur A., Khamkar S., André P., "CrowdWeaver: Visually Managing Complex Crowd Work", Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, 2012, 1033–1036, http://dx.doi.org/10.1145/2145204.2145357.

[14] Giang N. K., Blackstock M., Lea R. et al., “Distributed Data Flow: A Programming Model for the Crowdsourced Internet of Things”, Proceedings of the Doctoral Symposium of the 16th International Middleware Conference, 4:1–4:4, http://dx.doi.org/10.1145/2843966.2843970.

[15] Kahn G., “The semantics of a simple language for parallel programming”, *Information Processing*, **74** (1974), 471–475.

[16] Murata T., “Petri Nets: Properties, Analysis and Applications”, *Proceedings of the IEEE*, **77**:4 (1989), 541–580, http://dx.doi.org/10.1109/5.24143.

[17] Bernstein M. S., Little G., Miller R. C. et al., “Soylent: A Word Processor with a Crowd Inside”, Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology, 2010, 313–322, http://dx.doi.org/10.1145/1866029.1866078.

[18] Noronha J., Hysen E., Zhang H. et al., “PlateMate: Crowdsourcing Nutritional Analysis from Food Photographs”, Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, 2011, 1–12, http://dx.doi.org/10.1145/2047196.2047198.

[19] Wang J., Kraska T., Franklin M. J. et al., “CrowdER: Crowdsourcing Entity Resolution”, Proc. VLDB Endow., **5**:11 (2012), 1483–1494, http://dx.doi.org/10.14778/2350229.2350263.

[20] Biemann C., “Creating a system for lexical substitutions from scratch using crowdsourcing”, *Language Resources and Evaluation*, **47**:1 (2013), 97–122, http://dx.doi.org/10.1007/s10579-012-9180-5.

[21] Schek H.-J., Scholl M. H., “The relational model with relation-valued attributes”, *Information Systems*, **11**:2 (1986), 137–147, http://dx.doi.org/10.1016/0306-4379(86)90003-7.

[22] Garcia-Molina H., Ullman J. D., Widom J., *Database Systems: The Complete Book*, 2nd edition, Prentice Hall Press, 2008.

[23] Zhao G., Huang W., Liang S. et al., “Modeling MongoDB with Relational Model”, Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on, 2013, 115–121, http://dx.doi.org/10.1109/EIDWT.2013.25.

[24] Panchenko A., Loukachevitch N. V., Ustalov D. et al., “RUSSE: The First Workshop on Russian Semantic Similarity”, Computational Linguistics and Intellectual Technologies: papers from the Annual conference “Dialogue”, **2** (2015), 89–105, http://www.dialog-21.ru/digests/dialog2015/materials/pdf/PanchenkoAetal.pdf.

[25] McCartney W. P., Sridhar N., “Abstractions for Safe Concurrent Programming in Networked Embedded Systems”, Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, 2006, 167–180, http://dx.doi.org/10.1145/1182807.1182825.

[26] Ustalov D., Kiselev Y., “Add-Remove-Confirm: Crowdsourcing Synset Cleansing”, Application of Information and Communication Technologies (AICT), 2015 IEEE 9th International Conference on, 2015, 143–147, http://dx.doi.org/10.1109/ICAICT.2015.7338534.

[27] Ustalov D., “Crowdsourcing Synset Relations with Genus-Species-Match”, Proceedings of the AINL-ISMW FRUCT, 2015, 118–124, http://dx.doi.org/10.1109/AINL-ISMW-FRUCT.2015.7382980.

[28] Ustalov D., “A Crowdsourcing Engine for Mechanized Labor”, *Proceedings of the Institute for System Programming*, **27**:3 (2015), 351–364, http://dx.doi.org/10.15514/ISPRAS-2015-27(3)-25.

[29] Kiselev Y., Ustalov D., Porshnev S., “Eliminating Fuzzy Duplicates in Crowdsourced Lexical Resources”, Proceedings of the Eighth Global Wordnet Conference, 2016, 161–167, http://gwc2016.racai.ro/procedings.pdf.

[30] Fleiss J. L., Levin B., Paik M. C., *Statistical Methods for Rates and Proportions*, 3rd edition, John Wiley & Sons, 2003.

[31] Powers D. M. W., "The Problem with Kappa", Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, 2012, 345–355, http://aclweb.org/anthology/E12-1035.

[32] Ляшевская О. Н., Шаров С. А., *Частотный словарь современного русского языка (на материалах Национального корпуса русского языка)*, Азбуковник, 2009; [Lyashevskaya O. N., Sharoff S. A., *Chastotnyy slovar sovremennogo russkogo yazyka (na materialakh Natsionalnogo korpusa russkogo yazyka)*, Azbukovnik, 2009, (in Russian).]

[33] Healy G. F., Gurrin C., Smeaton A. F., "Informed Perspectives on Human Annotation Using Neural Signals", *MultiMedia Modeling: 22nd International Conference, Proceedings, Part II*, Lecture Notes in Computer Science, **9517**, 2016, 315–327, http://dx.doi.org/10.1007/978-3-319-27674-8_28.

[34] Ignatov D. I., Kaminskaya A. Yu., Bezzubtseva A. A. et al., "FCA-Based Models and a Prototype Data Analysis System for Crowdsourcing Platforms", *Conceptual Structures for STEM Research and Education*, Lecture Notes in Computer Science, **7735**, 2013, 173–192, http://dx.doi.org/10.1007/978-3-642-35786-2_13.

Ustalov D. A., "Dataflow-Driven Crowdsourcing: Relational Models and Algorithms", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 195–210.

DOI: 10.18255/1818-1015-2016-2-195-210

Abstract. Recently, microtask crowdsourcing has become a popular approach for addressing various data mining problems. Crowdsourcing workflows for approaching such problems are composed of several data processing stages which require consistent representation for making the work reproducible. This paper is devoted to the problem of reproducibility and formalization of the microtask crowdsourcing process. A computational model for microtask crowdsourcing based on an extended relational model and a dataflow computational model has been proposed. The proposed collaborative dataflow computational model is designed for processing the input data sources by executing annotation stages and automatic synchronization stages simultaneously. Data processing stages and connections between them are expressed by using collaborative computation workflows represented as loosely connected directed acyclic graphs. A synchronous algorithm for executing such workflows has been described. The computational model has been evaluated by applying it to two tasks from the computational linguistics field: concept lexicalization refining in electronic thesauri and establishing hierarchical relations between such concepts. The "Add–Remove–Confirm" procedure is designed for adding the missing lexemes to the concepts while removing the odd ones. The "Genus–Species–Match" procedure is designed for establishing "is-a" relations between the concepts provided with the corresponding word pairs. The experiments involving both volunteers from popular online social networks and paid workers from crowdsourcing marketplaces confirm applicability of these procedures for enhancing lexical resources.

Keywords: crowdsourcing, dataflow model, relational model, computational linguistics

On the authors:
Ustalov Dmitry Alekseevich, orcid.org/0000-0002-9979-2188, graduate student,
N.N. Krasovskii Institute of Mathematics and Mechanics of the Ural Branch of the Russian Academy of Sciences, Sofia Kovalevskaya str., 16, Yekaterinburg, 620990, Russia,
e-mail: dau@imm.uran.ru

Acknowledgments:
The reported study was funded by RFBR according to the research project No. 16-37-00354 мол_а "Adaptive Crowdsourcing Methods for Linguistic Resources". This work was supported by the Russian Foundation for the Humanities project no. 13-04-12020 "New Open Electronic Thesaurus for Russian" and project no. 16-04-12019 "RussNet and YARN thesauri integration".

Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 211–227
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 211–227

DOI: 10.18255/1818-1015-2016-2-211-227

УДК 519.6

Задача маршрутизации, осложненная зависимостью функций стоимости и "текущих" ограничений от списка заданий

Ченцов А. Г., Ченцов А. А.

получена 28 ноября 2015

Аннотация. Рассматривается задача маршрутизации перемещений, осложненная ограничениями различных типов (условия предшествования, ограничения на достижимость состояний при каждом перемещении и др.). Допускается многовариантность на этапе перемещений, что естественным образом приводит к задаче о посещении мегаполисов. Стоимости перемещений и работ, выполняемых при посещении мегаполисов, могут зависеть от списка заданий. Данный список может отвечать уже выполненным, либо, напротив, еще не выполненным заданиям. Допускается также, что "текущие" ограничения (на перемещения) могут зависеть от упомянутого списка заданий. Рассматриваемая постановка ориентирована на приложения к задачам атомной энергетики (проблема снижения облучаемости персонала АЭС при выполнении комплекса работ в условиях повышенной радиации) и машиностроения. В последнем случае, связанном с управлением инструментом при листовой резке деталей на машинах с ЧПУ, "текущие" ограничения на перемещения могут быть обусловлены тепловыми допусками по отношению к уже "пройденным" фрагментам листа. В статье приведена схема построения оптимального решения на основе широко понимаемого динамического программирования. Используемый при этом алгоритм реализован на ПЭВМ; результаты его применения иллюстрируются на модельных примерах.

Ключевые слова: маршрут, трасса, условия предшествования

Для цитирования: Ченцов А. Г., Ченцов А. А., "Задача маршрутизации, осложненная зависимостью функций стоимости и "текущих" ограничений от списка заданий", *Моделирование и анализ информационных систем*, **23**:2 (2016), 211–227.

Об авторах:
Ченцов Александр Георгиевич, orcid.org/0000-0002-8274-1456, член-корр. РАН,
Институт математики и механики им. Н.Н. Красовского,
ул. Софьи Ковалевской, 16, г. Екатеринбург, 620990 Россия,
Уральский Федеральный Университет, профессор,
ул. Мира, 19, г. Екатеринбург, 620002, Россия, e-mail: chentsov@imm.uran.ru

Ченцов Алексей Александрович, orcid.org/0000-0002-0646-9147, канд. физ.-мат. наук,
Институт математики и механики им. Н.Н. Красовского,
ул. Софьи Ковалевской, 16, г. Екатеринбург, 620990 Россия, e-mail: chentsov.a@binsys.ru

Благодарности:
Работа выполнена при финансовой поддержке программы фундаментальных исследований Президиума РАН «Математические задачи современной теории управления».

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проекты 14-08-00419, 15-01-07909).

Работа выполнена при финансовой поддержке Постановления № 211 Правительства Российской Федерации, контракт № 02.A03.21.0006.

Введение

Рассматривается задача маршрутизации перемещений с ограничениями различных типов. Постановка ориентирована на инженерные приложения, среди которых особо отметим некоторые задачи атомной энергетики и задачу управления инструментом при листовой резке деталей на станках с числовым программным управлением (ЧПУ). В упомянутых весьма конкретных задачах возникает необходимость маршрутизации перемещений в условиях большого числа разнообразных ограничений, что крайне затрудняет применение методов дискретной оптимизации, разрабатываемых преимущественно для решения задачи коммивояжера (ЗК) и задач типа ЗК. Серьезные затруднения возникают уже на постановочном уровне. Так, в части, относящейся к задаче о демонтаже энергоблока АЭС, выведенного из эксплуатации, мы сталкиваемся с функциями стоимости, зависящими от списка заданий, не выполненных на текущий момент. В самом деле, исполнитель находится под воздействием радиации от источников, которые еще не демонтированы на данный момент времени. Здесь же возможна ситуация, когда некоторые варианты перемещений, будучи запрещенными в начале процесса демонтажа, с течением времени могут уже стать доступными за счет более раннего демонтирования части радиоактивного оборудования (какие-то источники со временем оказываются "выключенными"). Таким образом, возможно влияние списка заданий и на саму систему ограничений.

Подобная ситуация может складываться и в задаче, связанной с листовой резкой на машинах с ЧПУ (см. [1], [2]). Здесь, однако, могут возникать ограничения на перемещения, зависящие от списка заданий, уже выполненных на момент перемещения. Они при некоторой идеализации могут формулироваться в виде следующего требования: новые точки врезки (в металл) должны быть достаточно удалены от уже вырезанных фрагментов листа, что, в частности, может быть связано со специальными тепловыми допусками. Учет данных ограничений может осуществляться непосредственно (путем введения запретов на определенные типы перемещений), а также косвенно: могут быть введены специальные функции стоимости (по сути, штрафы), при которых нарушение ограничений, формально разрешенное, сопровождается резким ухудшением критерия качества.

Отметим здесь же, что в обоих вышеупомянутых инженерных задачах возникают также ограничения на очередность выполнения заданий, т.е. на маршрут, в виде так называемых условий предшествования (условия типа "одно после другого"). Так, в задаче о демонтаже энергоблока одни излучающие фрагменты оборудования могут располагаться на других, а потому первые (верхние) должны всякий раз демонтироваться раньше нижних. В задаче, связанной с листовой резкой деталей, имеющих один внешний и, возможно, несколько внутренних контуров, резка последних должна осуществляться (всякий раз) раньше.

Полезно отметить также и то, что объектами посещения в упомянутых задачах маршрутизации прикладного характера являются не города (как в ЗК), а мегаполисы, т.е. имеется определенная многовариантность в системе возможных перемещений. Так, например, в задаче об управлении процессом резки пункты прибытия (к каждому контуру) суть точки врезки, а пункты отправления — соответствующие им точки выключения инструмента. Совокупность всех точек упомянутых двух типов образует мегаполис, связанный с вырезаемым контуром. Сама же резка

осуществляется по некоторой эквидистанте данного контура. После прибытия режущего инструмента (резака) в точку врезки начинается этап внутренних работ, который завершается в точке выключения инструмента. Вероятно, было бы еще логичнее создать некоторую вторичную эквидистанту, все точки которой могли бы использоваться в качестве возможных точек врезки, что после должной формализации привело бы к "непрерывной" экстремальной задаче на этапе резки каждого контура. На сегодняшний день, однако, трудности вычислительной реализации вынуждают к дискретизации, т.е. к работе с мегаполисами. При этом точки врезки и точки выключения инструмента группируются в упорядоченные пары, совокупность которых (для каждого контура) также порождает ограничения на возможные перемещения между мегаполисами.

Исследуемая ниже задача имеет своим прототипом известную труднорешаемую (в традиционном понимании [3]) ЗК; см. [4]– [7] и др. Отметим достаточно подробное обсуждение задач маршрутизации, подобных в том или ином отношении ЗК, приведенное в [4] (имеются в виду задачи типа ЗК; см. также [8]). В связи с использованием ДП при решении ЗК отметим [9], [10].

В настоящей работе последовательно развивается подход, восходящий к [11]; см. также [12]– [16]. Непосредственную же основу настоящей работы можно связать с [17], где рассматривалась задача последовательного обхода мегаполисов, в которой имеются условия предшествования, а функции стоимости и "текущие" ограничения зависят от списка заданий. Для данной общей постановки в [17] построен вариант широко понимаемого ДП, который был реализован в виде алгоритма для ПЭВМ. В настоящей статье упомянутая конструкция [17] развивается в направлении организации вычислений в задачах, имеющих достаточно большую для метода ДП размерность.

Итак, новым моментом в настоящем исследовании является прежде всего то, что и функции стоимости, и ограничения допускают зависимость от списка заданий. Сами ограничения формулируются, следовательно, в зависимости от процесса выполнения заданий и являются по сути динамическими.

1. Некоторые общие обозначения и определения

В статье используется стандартная теоретико-множественная символика (кванторы, связки и др.). Символ $\stackrel{\triangle}{=}$ используется для обозначения равенства по определению. Семейством называется множество, все элементы которого — множества. Ниже используются всякий раз семейства подмножеств (п/м) того или иного фиксированного множества. В этой связи через $\mathcal{P}(H)$ (через $\mathcal{P}'(H)$) обозначаем семейство всех (всех непустых) п/м множества H; через $\mathrm{Fin}(H)$ обозначаем семейство всех непустых конечных п/м H. Если p и q — какие-либо объекты, то через $\{p; q\}$ обозначаем (единственное) множество, содержащее p, q и не содержащее никаких других элементов; итак, $\{p; q\}$ — неупорядоченная пара объектов p, q, т.е. двоеточие. Каждому объекту y сопоставляем синглетон $\{y\} \stackrel{\triangle}{=} \{y; y\}$, содержащий y. Поскольку каждое множество — объект, в упомянутых терминах определена (см. [18, с. 67]) упорядоченная пара произвольных объектов: если a и b суть объекты, то $(a, b) \stackrel{\triangle}{=} \big\{\{a\}; \{a; b\}\big\}$. Если z — какая-либо упорядоченная пара, то через $\mathrm{pr}_1(z)$

и $\mathrm{pr}_2(z)$ обозначаем соответственно ее первый и второй элементы, определяемые условием $z = (\mathrm{pr}_1(z), \mathrm{pr}_2(z))$. Если α, β и γ — три произвольных объекта, то определен [19, с. 17] триплет $(\alpha, \beta, \gamma) \triangleq \big((\alpha, \beta), \gamma\big)$, являющийся упорядоченной парой специального вида. Для любых трех множеств A, B и C полагаем, следуя [19, с. 17], что $A \times B \times C \triangleq (A \times B) \times C$; поэтому при $x \in A \times B$ и $y \in C$ имеем $(x, y) \in A \times B \times C$. Данное обстоятельство учитываем при обозначении соответствующих значений функций трех переменных: для всяких непустого множества D, функции φ, действующей из $A \times B \times C$ в D, $x \in A \times B$ и $y \in C$ в виде $\varphi(x, y) \in D$ имеем значение φ в точке $(x, y) \in A \times B \times C$, которое также записываем в виде $\varphi(x_1, x_2, y)$, где $x_1 = \mathrm{pr}_1(x)$ и $x_2 = \mathrm{pr}_2(x)$: $\varphi(x_1, x_2, y) = \varphi(x, y)$.

Через $\mathcal{R}_+[T]$ обозначаем множество всех функций, действующих из непустого множества T в полупрямую $[0, \infty[\triangleq \{\xi \in \mathbb{R} |\ 0 \le \xi\}$ (здесь и ниже $\mathbb{R}$ — вещественная прямая). Полагаем, как обычно, $\mathbb{N} \triangleq \{1; 2; ...\}$, $\mathbb{N}_0 \triangleq \{0\} \cup \mathbb{N} = \{0; 1; 2; ...\}$ и

$$\overline{p, q} \triangleq \big\{j \in \mathbb{N}_0 |\ (p \le j) \& (j \le q)\big\} \quad \forall p \in \mathbb{N}_0\ \forall q \in \mathbb{N}_0$$

(заметим, что $\overline{p, q} = \emptyset$ при $q < p$). Если K — непустое конечное множество, то $|K| \in \mathbb{N}$ есть def мощность (количество элементов) K и определено непустое конечное множество $(\mathrm{bi})[K]$ всех биекций "отрезка" $\overline{1, |K|}$ на K. Полагаем, что $|\emptyset| \triangleq 0$.

2. Постановка задачи

Фиксируем непустое множество X, $x^0 \in X$, число $N \in \mathbb{N}$ со свойством $N \ge 2$, а также множества

$$M_1 \in \mathrm{Fin}(X), ..., M_N \in \mathrm{Fin}(X),$$

именуемые ниже мегаполисами. Полагаем далее, что

$$\big(x^0 \notin M_j \ \ \forall j \in \overline{1, N}\big) \& \big(M_p \cap M_q = \emptyset \ \ \forall p \in \overline{1, N}\ \forall q \in \overline{1, N} \setminus \{p\}\big).$$

Полагаем также заданными (непустые) отношения

$$\mathbb{M}_1 \in \mathcal{P}'(M_1 \times M_1), ..., \mathbb{M}_N \in \mathcal{P}'(M_N \times M_N). \tag{1}$$

Если $j \in \overline{1, N}$, то получаем, что $\mathfrak{M}_j \triangleq \{\mathrm{pr}_1(z) :\ z \in \mathbb{M}_j\} \in \mathcal{P}'(M_j)$ и $\mathbf{M}_j \triangleq \{\mathrm{pr}_2(z) :\ z \in \mathbb{M}_j\} \in \mathcal{P}'(M_j)$ (заметим, что в конкретизации, отвечающей листовой резке на машинах с ЧПУ, элементы $\mathfrak{M}_j$ являются возможными точками врезки, а элементы $\mathbf{M}_j$ — точками выключения инструмента). Получаем, что

$$\mathbb{X} \triangleq \{x^0\} \cup \big(\bigcup_{i=1}^{N} M_i\big) \in \mathrm{Fin}(X), \quad \mathbf{X} \triangleq \{x^0\} \cup \big(\bigcup_{i=1}^{N} \mathbf{M}_i\big) \in \mathrm{Fin}(\mathbb{X}). \tag{2}$$

Пусть $\mathfrak{N} \triangleq \mathcal{P}'(\overline{1, N})$, $A_1 :\ \mathbf{X} \times \mathfrak{N} \to \mathcal{P}'(M_1), ..., A_N :\ \mathbf{X} \times \mathfrak{N} \to \mathcal{P}'(M_N)$. Полагаем, что при $j \in \overline{1, N}$, $x \in \mathbf{X}$ и $K \in \mathfrak{N}$ непустое множество $A_j(x, K)$, $A_j(x, K) \subset M_j$, исчерпывает возможности перемещения в мегаполис M_j из состояния x в условиях,

когда K есть множество всех заданий, не выполненных на момент перемещения. Постулируем, что

$$A_j(x,K)\cap\mathfrak{M}_j\neq\emptyset\ \ \forall x\in\mathbf{X}\ \forall K\in\mathfrak{N}\ \forall j\in K. \tag{3}$$

Заметим, что с точки зрения здравого смысла условие (3), да и предположения относительно непустоты множеств, являющихся образами отображений $A_1,...,A_N$, несколько избыточны. Так, в частности, в (3) допускается, что при $K\in\mathfrak{N}$ и $j\in K$ рассматривается случай $x\in\mathbf{M}_j$ (см. (3)), который не будет возникать при наших построениях. Мы сохраняем, однако, упомянутые избыточные условия, имея в виду, что и при определении $A_1,...,A_N$, и при формулировании (3) рассматриваются на самом деле продолжения "реальных" зависимостей, связанных с фактическими ограничениями, которые имеют силу в пределах некоторых п/м $\mathbf{X}\times\mathfrak{N}$. Упомянутые продолжения, а точнее, доопределения, могут осуществляться (ограничимся сейчас обсуждением (2)), например, следующим образом: при $K\in\mathfrak{N}$, $j\in K$ и $x\in\mathbf{M}_j$ полагаем, что $A_j(x,K)=\mathfrak{M}_j$; тем самым объявляется формальная совместность "петли" , когда из $x\in\mathbf{M}_j$ возможно возвращение в $\mathfrak{M}_j$, чего, конечно, в нашей задаче маршрутизации происходить не будет. Аналогичные соображения можно использовать при доопределении "реальных" зависимостей, связанных с фактическими условиями на возможные перемещения, до мультифункций $A_1,...,A_N$.

Заметим также, что случаи, когда фактические ограничения зависят от списка $\tilde{K}$ выполненных (на текущий момент) заданий, легко сводятся к вышеупомянутому варианту посредством представления $\tilde{K}=\overline{1,N}\setminus K$, где K — множество (список) оставшихся заданий. Использование же именно множества K в последующих построениях представляется целесообразным с точки зрения применения аппарата ДП.

Всюду в дальнейшем полагаем, что $\mathbb{P}\triangleq(\mathrm{bi})[\overline{1,N}]$; элементы $\mathbb{P}$ (а это — перестановки множества $\overline{1,N}$) называем маршрутами. При $\alpha\in\mathbb{P}$ определена перестановка $\alpha^{-1}\in\mathbb{P}$, обратная к α: $\alpha(\alpha^{-1}(k))=\alpha^{-1}(\alpha(k))=k$ при $k\in\overline{1,N}$. Через $\mathbb{Z}$ обозначаем множество всех кортежей $(z_j)_{j\in\overline{0,N}}:\ \overline{0,N}\to\mathbb{X}\times\mathbf{X}$ (итак, $z_0\in\mathbb{X}\times\mathbf{X}$, $z_1\in\mathbb{X}\times\mathbf{X}$, ..., $z_N\in\mathbb{X}\times\mathbf{X}$). При $\alpha\in\mathbb{P}$ полагаем, что

$$\begin{array}{c}\mathcal{Z}_\alpha\triangleq\big\{(z_t)_{t\in\overline{0,N}}\in\mathbb{Z}|\ (z_0=(x^0,x^0))\&(z_t\in\mathbb{M}_{\alpha(t)}\ \ \forall t\in\overline{1,N})\&\\ \&(\mathrm{pr}_1(z_s)\in A_{\alpha(s)}(\mathrm{pr}_2(z_{s-1}),\{\alpha(l):\ l\in\overline{s,N}\})\ \ \forall s\in\overline{1,N})\big\},\end{array} \tag{4}$$

получая в виде элементов (4) траектории, согласованные с маршрутом α (имеется в виду согласование и в смысле отношений (1), и в смысле условий, связанных с (3)). Определение (4) допускает естественное распространение на случай, когда требуется выполнение уже не всех заданий. В этой связи при $K\in\mathfrak{N}$ полагаем, что $\mathbb{Z}_K$ есть def множество всех кортежей $(z_t)_{t\in\overline{0,|K|}}:\ \overline{0,|K|}\to\mathbb{X}\times\mathbf{X}$. Если же $x\in\mathbf{X}$, $K\in\mathfrak{N}$ и $\alpha\in(\mathrm{bi})[K]$, то

$$\begin{array}{c}\mathcal{Z}(x,K,\alpha)\triangleq\big\{(z_t)_{t\in\overline{0,|K|}}\in\mathbb{Z}_K|\ (z_0=(x,x))\&(z_t\in\mathbb{M}_{\alpha(t)}\ \ \forall t\in\overline{1,|K|})\&\\ \&(\mathrm{pr}_1(z_s)\in A_{\alpha(s)}(\mathrm{pr}_2(z_{s-1}),\{\alpha(l):\ l\in\overline{s,|K|}\})\ \ \forall s\in\overline{1,|K|})\big\}.\end{array} \tag{5}$$

Рассуждением по индукции проверяется, что каждое из множеств (5) непусто. Поскольку $\mathcal{Z}_\alpha=\mathcal{Z}(x^0,\overline{1,N},\alpha)$ при $\alpha\in\mathbb{P}$, имеем следующее свойство. Именно, $\mathcal{Z}_\alpha\neq\emptyset$

при $\alpha \in \mathbb{P}$. Итак (при условии (3)), с маршрутами (полными и частичными) связываются непустые множества траекторий. На выбор же самих маршрутов также накладываются ограничения.

Условия предшествования. Фиксируем множество $\mathbf{K} \in \mathcal{P}(\overline{1,N} \times \overline{1,N})$, элементы которого называем адресными парами. Если $z \in \mathbf{K}$, то, в частности, $z = (i,j)$, где $i \in \overline{1,N}$ и $j \in \overline{1,N}$; $i = \mathrm{pr}_1(z)$ и $j = \mathrm{pr}_2(z)$. Всюду в дальнейшем полагаем, что $\forall \mathbf{K}_0 \in \mathcal{P}'(\mathbf{K})\ \exists z_0 \in \mathbf{K}_0 :\ \mathrm{pr}_1(z_0) \neq \mathrm{pr}_2(z)\ \ \forall z \in \mathbf{K}_0$ (конкретные варианты данного условия см. в [11, часть 2]). Тогда

$$\mathbf{A} \triangleq \big\{\alpha \in \mathbb{P}|\ \forall z \in \mathbf{K}\ \forall t_1 \in \overline{1,N}\ \forall t_2 \in \overline{1,N}\ \ ((\mathrm{pr}_1(z) = \alpha(t_1))\& \\ \&(\mathrm{pr}_2(z) = \alpha(t_2))) \Rightarrow (t_1 < t_2)\big\} = \big\{\alpha \in \mathbb{P}|\ \alpha^{-1}(\mathrm{pr}_1(z)) < \alpha^{-1}(\mathrm{pr}_2(z))\ \ \forall z \in \mathbf{K}\big\} \neq \emptyset; \tag{6}$$

итак, у нас существуют $\mathbf{K}$–допустимые (по предшествованию) маршруты. В итоге

$$\mathbf{D} \triangleq \big\{(\alpha, (z_t)_{t\in\overline{0,N}}) \in \mathbf{A} \times \mathbb{Z}|\ (z_t)_{t\in\overline{0,N}} \in \mathcal{Z}_\alpha\big\} \in \mathrm{Fin}(\mathbf{A} \times \mathbb{Z}) \tag{7}$$

и, в частности, $\mathbf{D} \neq \emptyset$; элементы $\mathbf{D}$ рассматриваем в качестве допустимых решений (ДР). Каждый такой элемент есть пара "маршрут-трасса".

Функции стоимости. Фиксируем далее функции

$$\mathbf{c} \in \mathcal{R}_+[\mathbb{X} \times \mathbb{X} \times \mathfrak{N}],\ c_1 \in \mathcal{R}_+[\mathbb{X} \times \mathbb{X} \times \mathfrak{N}], ..., c_N \in \mathcal{R}_+[\mathbb{X} \times \mathbb{X} \times \mathfrak{N}],\ f \in \mathcal{R}_+[\mathbf{X}]. \tag{8}$$

Относительно (8) полагаем, что функция $\mathbf{c}$ оценивает внешние перемещения, функции $c_1, ..., c_N$ — работы, связанные с посещением мегаполисов (далее именуем эти работы внутренними), а f — терминальное состояние процесса. С учетом этого полагаем, что при $\alpha \in \mathbb{P}$ и $(z_i)_{i\in\overline{0,N}} \in \mathbb{Z}$

$$\mathfrak{C}_\alpha[(z_i)_{i\in\overline{0,N}}] \triangleq \sum_{s=1}^{N} \big[\mathbf{c}(\mathrm{pr}_2(z_{s-1}), \mathrm{pr}_1(z_s), \{\alpha(t) :\ t \in \overline{s,N}\}) + \\ + c_{\alpha(s)}(z_s, \{\alpha(t) :\ t \in \overline{s,N}\})\big] + f(\mathrm{pr}_2(z_N)); \tag{9}$$

для наших целей важен вариант (9), в котором $\alpha \in \mathbf{A}$ и $(z_i)_{i\in\overline{0,N}} \in \mathcal{Z}_\alpha$, т.к. тогда (9) оценивает качество ДР из множества (7). Итак, в качестве основной рассматриваем задачу

$$\mathfrak{C}_\alpha[(z_t)_{t\in\overline{0,N}}] \to \min,\ \ \alpha \in \mathbf{A},\ (z_t)_{t\in\overline{0,N}} \in \mathcal{Z}_\alpha. \tag{10}$$

С учетом (7) получаем, что данная задача совместна по ограничениям; ей сопоставляется экстремум

$$V \triangleq \min_{\alpha\in\mathbf{A}} \min_{(z_t)_{t\in\overline{0,N}} \in \mathcal{Z}_\alpha} \mathfrak{C}_\alpha[(z_t)_{t\in\overline{0,N}}] \in [0, \infty[\tag{11}$$

и непустое множество оптимальных ДР, т.е. таких ДР $(\alpha^0, (z_t^0)_{t\in\overline{0,N}}) \in \mathbf{D}$, для которых $\mathfrak{C}_{\alpha^0}[(z_t^0)_{t\in\overline{0,N}}] = V$. Мы ставим своей целью нахождение V (11) и какого-либо оптимального ДР. Для достижения этой цели предлагается вариант широко понимаемого ДП.

3. Динамическое программирование

В настоящем разделе излагается схема работы [17] с некоторыми добавлениями, связанными с описанием вспомогательных конструкций. Прежде всего отметим, что в самой задаче (10) процедуру ДП применять затруднительно в связи с большим числом разнообразных условий, ограничивающих конкретный выбор как собственно маршрута, так и траектории. Поэтому, следуя [11, часть 2], мы используем редукцию основной задачи, подменяя условия предшествования соответствующими условиями вычеркивания. В этой связи полагаем при $K \in \mathfrak{N}$, что $\Xi[K] \stackrel{\triangle}{=} \{z \in \mathbf{K}|\ (\mathrm{pr}_1(z) \in K)\&(\mathrm{pr}_2(z) \in K)\}$. Оператор $\mathbf{I}$, действующий в $\mathfrak{N}$, определяем правилом: при $K \in \mathfrak{N}$

$$\mathbf{I}(K) \stackrel{\triangle}{=} K \setminus \{\mathrm{pr}_2(z) :\ z \in \Xi[K]\}. \tag{12}$$

Собственно в (12) как раз и определено правило вычеркивания (заданий из списка). Теперь мы вводим в рассмотрение (частичные) маршруты, соблюдающие данное правило, полагая, что

$$(\mathbf{I}{-}\mathrm{bi})[K] = \{\alpha \in (\mathrm{bi})[K]|\ \alpha(s) \in \mathbf{I}(\{\alpha(t) :\ t \in \overline{s, |K|}\})\ \ \forall s \in \overline{1, |K|}\}\ \ \forall K \in \mathfrak{N}. \tag{13}$$

При этом [11, часть 2] $(\mathbf{I}{-}\mathrm{bi})[K] \neq \emptyset\ \ \forall K \in \mathfrak{N}$. Посредством (13) определено, в частности, множество $(\mathbf{I}{-}\mathrm{bi})[\overline{1, N}]$, для которого [11, часть 2]

$$\mathbf{A} = (\mathbf{I}{-}\mathrm{bi})[\overline{1, N}] = \{\alpha \in \mathbb{P}|\ (\alpha(k) \in \mathbf{I}(\overline{1, N} \setminus \{\alpha(l) :\ l \in \overline{1, k-1}\})\ \ \forall k \in \overline{2, N})\& \\ \&(\alpha(1) \in \mathbf{I}(\overline{1, N}))\}. \tag{14}$$

Введем теперь частичные критерии качества, полагая сначала при $K \in \mathfrak{N}$, $\alpha \in (\mathrm{bi})[K]$ и $(z_t)_{t \in \overline{0, |K|}} \in \mathbb{Z}_K$, что

$$\hat{\mathfrak{C}}_\alpha[(z_t)_{t \in \overline{0, |K|}}|K] \stackrel{\triangle}{=} \sum_{s=1}^{|K|} \big[\mathbf{c}(\mathrm{pr}_2(z_{s-1}), \mathrm{pr}_1(z_s), \{\alpha(t) :\ t \in \overline{s, |K|}\}) + \\ + c_{\alpha(s)}(z_s, \{\alpha(t) :\ t \in \overline{s, |K|}\})\big] + f(\mathrm{pr}_2(z_{|K|})). \tag{15}$$

Разумеется, (15) будет использоваться при $\alpha \in (\mathbf{I}{-}\mathrm{bi})[K]$ и $(z_t)_{t \in \overline{0, |K|}} \in \mathcal{Z}(x, K, \alpha)$. С учетом этого при $x \in \mathbf{X}$ и $K \in \mathfrak{N}$ рассматриваем экстремальную задачу

$$\hat{\mathfrak{C}}_\alpha[(z_t)_{t \in \overline{0, |K|}}|K] \to \min,\ \ \alpha \in (\mathbf{I}{-}\mathrm{bi})[K],\ (z_t)_{t \in \overline{0, |K|}} \in \mathcal{Z}(x, K, \alpha), \tag{16}$$

ограничения которой совместны, а потому определено значение

$$v(x, K) \stackrel{\triangle}{=} \min_{\alpha \in (\mathbf{I}{-}\mathrm{bi})[K]} \min_{(z_t)_{t \in \overline{0, |K|}} \in \mathcal{Z}(x, K, \alpha)} \hat{\mathfrak{C}}_\alpha[(z_t)_{t \in \overline{0, |K|}}|K] \in [0, \infty[. \tag{17}$$

Пусть, кроме того, $v(x, \emptyset) \stackrel{\triangle}{=} f(x)\ \ \forall x \in \mathbf{X}$. Теперь (см. (17)) определена функция Беллмана $v \in \mathcal{R}_+[\mathbf{X} \times \mathcal{P}(\overline{1, N})]$. Нам потребуется одна несущественная модификация мультифункций $A_1, ..., A_N$. Итак, при $j \in \overline{1, N}$, $x \in \mathbf{X}$ и $K \in \mathbf{N}$ полагаем

$$\mathbb{A}_j(x, K) \stackrel{\triangle}{=} \{z \in \mathbb{M}_j|\ \mathrm{pr}_1(z) \in A_j(x, K)\};$$

при этом (см. (3)) $\mathbb{A}_j(x, K) \in \mathcal{P}'(\mathbb{M}_j)$ в случае $j \in K$, где в качестве j можно, в частности, использовать элемент $\mathbf{I}(K)$. По аналогии с [14] устанавливается

Теорема 1. *Если* $x \in \mathbf{X}$ *и* $K \in \mathfrak{N}$, *то*

$$v(x,K) = \min_{j \in \mathbf{I}(K)} \min_{z \in \mathbb{A}_j(x,K)} \big[\mathbf{c}(x, \mathrm{pr}_1(z), K) + c_j(z,K) + v(\mathrm{pr}_2(z), K \setminus \{j\})\big].$$

Поскольку, как легко видеть, $V = v(x^0, \overline{1,N})$, из теоремы 1 следует, что

$$V = \min_{j \in \mathbf{I}(\overline{1,N})} \min_{z \in \mathbb{A}_j(x^0, \overline{1,N})} \big[\mathbf{c}(x^0, \mathrm{pr}_1(z), \overline{1,N}) + c_j(z, \overline{1,N}) + v(\mathrm{pr}_2(z), \overline{1,N} \setminus \{j\})\big]. \quad (18)$$

4. Рекуррентная процедура построения слоев функции Беллмана

Ниже приведена схема, являющаяся развитием [11, §4.9] и не предусматривающая построения всего массива значений функции Беллмана; последнее связано с использованием условий предшествования для целей снижения вычислительной сложности. Пусть

$$\mathcal{G} \triangleq \big\{K \in \mathfrak{N} |\ \forall z \in \mathbf{K}\ \ (\mathrm{pr}_1(z) \in K) \Rightarrow (\mathrm{pr}_2(z) \in K)\big\}. \quad (19)$$

Множества — элементы (19) — именуем существенными списками заданий. Пусть, кроме того,

$$\mathcal{G}_s \triangleq \big\{K \in \mathcal{G} |\ s = |K|\big\}\ \ \forall s \in \overline{1,N}.$$

В виде $\{\mathcal{G}_1, ..., \mathcal{G}_N\}$ имеем разбиение $\mathcal{G}$. При этом $\mathcal{G}_N = \{\overline{1,N}\}$ (синглетон, содержащий $\overline{1,N}$) и $\mathcal{G}_1 \triangleq \big\{\{t\} :\ t \in \overline{1,N} \setminus \mathbf{K}_1\big\}$, где $\mathbf{K}_1 \triangleq \{\mathrm{pr}_1(z) :\ z \in \mathbf{K}\}$ (итак, синглетоны из $\mathcal{G}_1$ соответствуют индексам "неотправителей" в смысле $\mathbf{K}$). Известно [14], [15], что

$$\mathcal{G}_{s-1} = \big\{K \setminus \{t\} :\ K \in \mathcal{G}_s,\ t \in \mathbf{I}(K)\big\}\ \ \forall s \in \overline{2,N}.$$

Получили рекуррентную процедуру $\mathcal{G}_N \to \mathcal{G}_{N-1} \to ... \to \mathcal{G}_1$ (имеется в виду случай $N > 2$). Следующий шаг состоит в построении слоев пространства позиций, обозначаемых через $D_0, D_1, ..., D_N$. При этом полагаем, что $D_N \triangleq \big\{(x^0, \overline{1,N})\big\}$ (синглетон, содержащий позицию $(x^0, \overline{1,N})$) и $D_0 \triangleq \big\{(x, \emptyset) :\ x \in \tilde{\mathbf{M}}\big\}$, где

$$\tilde{\mathbf{M}} \triangleq \bigcup_{j \in \overline{1,N} \setminus \mathbf{K}_1} \mathbf{M}_j.$$

Итак, определены крайние слои. Если $s \in \overline{1,N-1}$, то при $K \in \mathcal{G}_s$ определяем последовательно

$$\mathcal{J}_s(K) \triangleq \big\{j \in \overline{1,N} \setminus K |\ \{j\} \cup K \in \mathcal{G}_{s+1}\big\} \in \mathcal{P}'(\overline{1,N}),$$
$$\mathcal{M}_s[K] \triangleq \bigcup_{j \in \mathcal{J}_s(K)} \mathbf{M}_j,\ \ \mathbb{D}_s[K] \triangleq \big\{(x,K) :\ x \in \mathcal{M}_s[K]\big\} \in \mathcal{P}'(\mathbf{X} \times \mathcal{G}_s),$$

после чего полагаем

$$D_s \triangleq \bigcup_{\mathbb{K} \in \mathcal{G}_s} \mathbb{D}_s[\mathbb{K}]. \quad (20)$$

Таким образом (см. (20)), определение слоев $D_0, D_1, ..., D_N$ завершено. При этом, конечно, $D_j \neq \emptyset \ \ \forall j \in \overline{1,N}$. Отметим следующее важное свойство (см. [17, (8)]): если $s \in \overline{1,N}$, $(x,K) \in D_s$, $j \in \mathbf{I}(K)$ и $z \in \mathbb{A}_j(x,K)$, то непременно

$$(\mathrm{pr}_2(z), K \setminus \{j\}) \in D_{s-1}. \tag{21}$$

Учитывая непустоту слоев D_l, $l \in \overline{0,N}$, введем в рассмотрение сужения функции Беллмана v на каждый из этих слоев: если $t \in \overline{0,N}$, то $v_t \in \mathcal{R}_+[D_t]$ определяем условиями

$$v_t(x,K) \stackrel{\triangle}{=} v(x,K) \ \ \forall (x,K) \in D_t. \tag{22}$$

Из (21) и (22) имеем, в частности, что при условиях, обеспечивающих (21), определено значение $v_{s-1}(\mathrm{pr}_2(z), K \setminus \{j\}) \in [0,\infty[$. Поэтому при $s \in \overline{1,N}$ и $(x,K) \in D_s$ имеем следующее значение:

$$\min_{j \in \mathbf{I}(K)} \min_{z \in \mathbb{A}_j(x,K)} \left[\mathbf{c}(x, \mathrm{pr}_1(z), K) + c_j(z,K) + v_{s-1}(\mathrm{pr}_2(z), K \setminus \{j\}) \right] \in [0,\infty[.$$

Из теоремы 1 имеем с учетом (21), (22), что (см. [17]) справедливо

Предложение 1. *Если $s \in \overline{1,N}$, то преобразование v_{s-1} в v_s имеет следующий вид:*

$$v_s(x,K) = \min_{j \in \mathbf{I}(K)} \min_{z \in \mathbb{A}_j(x,K)} \left[\mathbf{c}(x, \mathrm{pr}_1(z), K) + c_j(z,K) + \right.$$
$$\left. + v_{s-1}(\mathrm{pr}_2(z), K \setminus \{j\}) \right] \in [0,\infty[\ \ \forall (x,K) \in D_s.$$

Получили рекуррентную процедуру $v_0 \to v_1 \to ... \to v_N$, где $v_0 \in \mathcal{R}_+[D_0]$ определяется явным образом: $v_0(x,\emptyset) = f(x) \ \ \forall x \in \tilde{\mathbf{M}}$. Отметим, что из предложения 1, в частности, следует, что (см. (22))

$$V = v_N(x^0, \overline{1,N}) = \min_{j \in \mathbf{I}(\overline{1,N})} \min_{z \in \mathbb{A}_j(x^0, \overline{1,N})} \left[\mathbf{c}(x^0, \mathrm{pr}_1(z), \overline{1,N}) + \right.$$
$$\left. + c_j(z, \overline{1,N}) + v_{N-1}(\mathrm{pr}_2(z), \overline{1,N} \setminus \{j\}) \right]. \tag{23}$$

В заключении раздела рассмотрим (совсем кратко) схему построения оптимального ДР. Полагаем $\mathbf{z}^{(0)} \stackrel{\triangle}{=} (x^0, x^0)$. Далее с учетом (23) выбираем $\eta_1 \in \mathbf{I}(\overline{1,N})$ и $\mathbf{z}^{(1)} \in \mathbb{A}_{\eta_1}(x^0, \overline{1,N})$ из условия

$$V = \mathbf{c}(x^0, \mathrm{pr}_1(\mathbf{z}^{(1)}), \overline{1,N}) + c_{\eta_1}(\mathbf{z}^{(1)}, \overline{1,N}) + v_{N-1}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \overline{1,N} \setminus \{\eta_1\}). \tag{24}$$

Из (21) получаем следующее свойство: $(\mathrm{pr}_2(\mathbf{z}^{(1)}), \overline{1,N} \setminus \{\eta_1\}) \in D_{N-1}$. Поэтому согласно предложению 1

$$v_{N-1}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \overline{1,N} \setminus \{\eta_1\}) =$$
$$= \min_{j \in \mathbf{I}(\overline{1,N} \setminus \{\eta_1\})} \min_{z \in \mathbb{A}_j(\mathrm{pr}_2(\mathbf{z}^{(1)}), \overline{1,N} \setminus \{\eta_1\})} \left[\mathbf{c}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \mathrm{pr}_1(z), \overline{1,N} \setminus \{\eta_1\}) + \right.$$
$$\left. + c_j(z, \overline{1,N} \setminus \{\eta_1\}) + v_{N-2}(\mathrm{pr}_2(z), \overline{1,N} \setminus \{\eta_1; j\}) \right]. \tag{25}$$

С учетом (25) выбираем $\eta_2 \in \mathbf{I}(\overline{1,N} \setminus \{\eta_1\})$ и $\mathbf{z}^{(2)} \in \mathbb{A}_{\eta_2}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \overline{1,N} \setminus \{\eta_1\})$ из условия

$$v_{N-1}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \overline{1,N} \setminus \{\eta_1\}) = \mathbf{c}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \mathrm{pr}_1(\mathbf{z}^{(2)}), \overline{1,N} \setminus \{\eta_1\}) +$$
$$+ c_{\eta_2}(\mathbf{z}^{(2)}, \overline{1,N} \setminus \{\eta_1\}) + v_{N-2}(\mathrm{pr}_2(\mathbf{z}^{(2)}), \overline{1,N} \setminus \{\eta_1; \eta_2\}). \tag{26}$$

Из (24) и (26) вытекает, что справедливо равенство

$$V = \mathbf{c}(\mathrm{pr}_2(\mathbf{z}^{(0)}), \mathrm{pr}_1(\mathbf{z}^{(1)}), \overline{1,N}) + \mathbf{c}(\mathrm{pr}_2(\mathbf{z}^{(1)}), \mathrm{pr}_1(\mathbf{z}^{(2)}), \overline{1,N} \setminus \{\eta_1\}) + \\ + c_{\eta_1}(\mathbf{z}^{(1)}, \overline{1,N}) + c_{\eta_2}(\mathbf{z}^{(2)}, \overline{1,N} \setminus \{\eta_1\}) + v_{N-2}(\mathrm{pr}_2(\mathbf{z}^{(2)}), \overline{1,N} \setminus \{\eta_1;\eta_2\}). \tag{27}$$

Дальнейшие построения аналогичны (24), (26); их следует продолжать вплоть до исчерпывания полного списка заданий $\overline{1,N}$. Точнее, после N шагов (этапов), подобных (24) и (26), будут построены

$$\eta \triangleq (\eta_j)_{j\in\overline{1,N}} \in \mathbf{A}, \quad (\mathbf{z}^{(j)})_{j\in\overline{0,N}} \in \mathcal{Z}_\eta,$$

для которых $\mathfrak{C}_\eta[(\mathbf{z}^{(j)})_{j\in\overline{0,N}}] = V$ (при $N = 2$ данный вывод следует из (27) непосредственно по определению функции v_0). Иными словами, будет построено оптимальное ДР $(\eta, (\mathbf{z}^{(j)})_{j\in\overline{0,N}}) \in \mathbf{D}$. Данные очевидные построения сейчас опустим, отсылая за подробностями к подобным в идейном отношении конструкциям [14, раздел 7].

5. Вычислительный эксперимент

Рассматриваемые в данной работе теоретические конструкции, являющиеся общим описанием концепции алгоритмов решения маршрутной задачи обхода мегаполисов, будем использовать для решения конкретной задачи маршрутизации движения резака в задаче оптимизации листовой резки. Будем использовать следующую упрощенную постановку данной задачи. Зафиксируем вещественные числа $a_1 \in \mathbb{R}$, $a_2 \in \mathbb{R}$, $b_1 \in \mathbb{R}$ и $b_2 \in \mathbb{R}$, $a_1 < a_2$, $b_1 < b_2$, которые задают на плоскости размеры прямоугольной области (листа), являющейся в нашей постановке множеством X, а именно: $X \triangleq [a_1, a_2] \times [b_1, b_2]$. Данный лист размечен для резки некоторого количества деталей, задаваемых посредством N замкнутых контуров, которые в нашей модели соответствуют мегаполисам $M_1, ..., M_N$; пересечение контуров не допустимо, но одни контуры могут находиться внутри других (например, деталь в виде шайбы). Более подробно рассмотрим, что представляют собой в данной постановке контуры деталей. Каждый контур представлен двумя эквидистантами:

1) Основная эквидистанта — это замкнутая кривая, по которой движется резак при вырезании контура.

2) Вспомогательная эквидистанта — множество пар точек, у которых первый элемент является точкой врезки (точкой включения резака; из данной точки совершается перевод инструмента с включенным резаком на основную эквидистанту для выполнения реза по контуру), а второй — точка выключения инструмента (резак при завершении движения по основной эквидистанте, т.е. по завершении цикла реза по контуру, совершает перемещение в данную точку для выключения перед перемещением к другому контуру; выключение необходимо для исключения повреждения еще не вырезанных деталей).

В рассматриваемой модели основные эквидистанты дискретизированы для удобства вычислительной реализации, а вспомогательные эквидистанты отождествляются с множествами $\mathbb{M}_1, ..., \mathbb{M}_N$ (1), в которых первые элементы каждой пары — точки включения, а вторые элементы — точки выключения резака.

Будем придерживаться модели, в которой весь процесс резки начинается из точки x^0, являющейся положением парковки резака, и завершается отводом резака

после реза последнего контура в состояние x^0 (терминальный этап цикла резки листа — отвод в позицию парковки).

Ограничения на порядок резки контуров в данной концепции выглядят вполне естественно: внутренние (по порядку вложенности) контуры должны вырезаться раньше внешних, что, собственно, определяется технологическими соображениями.

Функции стоимости (8) в данной модели будем определять следующим образом.

1) Функция **c** оценивает затраты на перемещение с выключенным резаком из точки парковки резака x^0 в первую (по порядку реализации раскройного плана) точку врезки или из очередной точки выключения резака в следующую точку врезки. Данная функция в нашей модели — евклидово расстояние.

2) Функции $c_1, ..., c_N$ оценивают внутренние работы, а именно, работы, связанные с выходом из точки врезки на основную эквидистанту и отводом резака от нее по завершении реза в точку выключения резака, расположенную на вспомогательной эквидистанте. Следует отметить, что в оценку затрат на внутренние работы также должны входить затраты на перемещение резака по основной эквидистанте при выполнении реза данного контура, но их величина является постоянной и не зависит ни от точки врезки, ни от точки выключения резака, поэтому (в нашей модели) в оптимизационной задаче данные слагаемые мы не учитываем. Итак, всякий раз функция c_i, где $i \in \overline{1, N}$, определяется суммой $3 * \rho(x, y) + \rho(y, z)$, где ρ — евклидово расстояние, x — точка врезки, y — точка начала цикла резки на вырезаемом контуре (основной эквидистанте), а z — точка выключения резака; коэффициент 3 в формуле характеризует дополнительные затраты, связанные с "пробивкой" материала при врезке. Здесь x и z используются в качестве аргументов, $y = y_{x,z}$ однозначно сопоставляется паре (x, z); зависимость от списка заданий, как и в случае с **c**, здесь отсутствует.

3) Функция f — евклидово расстояние от последней (по ходу реализации раскройного плана) точки выключения резака до точки парковки инструмента.

Технология резки деталей, как уже упоминалось выше, диктует свои ограничения на выбор допустимых точек врезки, которые формализуются в виде отображений $A_1, ..., A_N$. В нашей модели рассматриваем следующие критерии допустимости точек врезки: предотвращение деформации деталей (соблюдение тепловых допусков) и отсеивание наиболее удаленных точек врезки (минимизация холостого хода инструмента). Рассмотрим подробнее каждое из этих условий, при этом будем учитывать, что на момент принятия решения индексы еще не вырезанных контуров образуют множество K, $K \subset \overline{1, N}$.

1) Предотвращение тепловой деформации деталей: для каждого i-го контура задано вещественное число $\delta_i \in \mathbb{R}$, $\delta_i > 0$, определяющее минимально допустимый отступ от данного контура по отношению к уже вырезанным участкам листа. При рассмотрении возможностей для выбора точки врезки для контура с индексом i допустимыми признаются те точки врезки y, которые удовлетворяют системе неравенств:

$$\big(\rho(y, z) > \delta_i \ \ \forall k \in \overline{1, N} \setminus K \ \forall z \in \mathfrak{M}_k\big) \& \big(\rho(y, \tilde{z}) > \delta_i \ \ \forall k \in \overline{1, N} \setminus K \ \forall \tilde{z} \in \tilde{M}_k\big),$$

где $\tilde{M}_k$ — "сетка" , являющаяся дискретизацией основной эквидистанты контура с индексом k. Данное условие призвано минимизировать деформацию материала при врезке за счет рассеивания тепла в еще не вырезанных частях листа.

2) Отсеивание наиболее удаленных точек врезки: обозначим через $\mathfrak{M}_i^{(1)}$ множество точек врезки i-контура, удовлетворяющих 1). Пусть порядок вырезания контуров определяется маршрутом α и $\alpha(j) = i$. Среди точек $\mathfrak{M}_i^{(1)}$ выберем ближайшую к предыдущей точке выключения резака или (для перехода из позиции парковки резака) к x^0 точку врезки y', а именно такую точку, для которой $\mathbf{c}(x, y') = \min\limits_{y \in \mathfrak{M}_i^{(1)}} \mathbf{c}(x, y)$, где $\left(x = x^0,\ y \in \mathfrak{M}_{\alpha(1)}^{(1)}\right) \vee \left(x \in \mathbf{M}_{\alpha(j-1)},\ j \in \overline{2, N}, \alpha(j-1) \in \overline{1, N} \setminus K\right)$. Точка y' признается годной согласно 2). Помимо точки y' допустимыми по 2) полагаем также элементы y множества $\mathfrak{M}_i^{(1)}$, удовлетворяющие неравенству $\rho(x, y) - \rho(x, y') \le \varepsilon_i$, где x — точка начала перемещения на контур с индексом i (см. выше), а вещественное число $\varepsilon_i,\ \varepsilon_i \ge 0$, — допуск на отклонение от ближайшей точки планируемого для осуществления резки контура.

Допустимыми являются точки врезки контура, удовлетворяющие 1) и 2). Отметим, что возможна ситуация, когда из-за вырезанных вокруг контура деталей ни для одной точки врезки контура не выполняется 1), иными словами, это означает, что все точки врезки расположены таким образом, что при включении резака не будет обеспечиваться рассеивание тепловой энергии в еще не вырезанных участках листа и возможна деформация заготовки. Для обеспечения работы алгоритма в таком случае производится отступление от правил выбора точек врезки, основанных на последовательном применении критерия 1) и затем критерия 2). Итак, пусть для контура с индексом i нет ни одной допустимой точки врезки, при этом индексы еще не вырезанных контуров образуют множество $K,\ K \subset \overline{1, N}$. Применяем ко всему контуру правило 2), т.е. находим множество $\mathfrak{M}_i^{(2)}$ ε_i–ближайших (к текущей точке нахождения инструмента) точек врезки. Среди элементов $\mathfrak{M}_i^{(2)}$ находим такой элемент y'', который характеризуется наибольшим минимальным расстоянием до уже вырезанных контуров, т.е. состояние, для которого справедливо

$$c(x, y'') = \max_{k \in \overline{1,N} \setminus K} \min_{y \in \tilde{M}_k} \rho(x, y),$$

где, как было отмечено выше, $x = x^0$ для случая перемещения инструмента из позиции парковки или x есть точка выключения резака на предыдущем (по порядку реализации раскройного плана) контуре. Точка y'' признается допустимой точкой врезки контура i. Такой прием позволяет исключить коллизии, связанные с невозможностью выбора новой точки врезки.

Описанный в настоящей статье алгоритм был реализован в виде программы для ПЭВМ, написанной на языке программирования C++, работающей под управлением 64-разрядной операционной системы семейства Windows, начиная с версии 7, применительно к приведенной в данном разделе модели для задачи оптимизации листовой резки. Программа работает в многопотоковом режиме: вычислительная часть выполняется в отдельном от интерфейса пользователя потоке. Для случая решения задачи на плоскости имеется возможность графического представления траектории движения по мегаполисам с возможностью увеличения отдельных участков графика и сохранения изображения в файл графического формата bmp. Исходные данные и результаты работы программы хранятся в текстовом файле специальной структуры.

Вычислительный эксперимент проводился на ПЭВМ с процессором Intel Core i7, объемом ОЗУ 64 гБ с установленной операционной системой Windows 7 Максимальная SP1 x64.

Пусть, для определенности, $a_1 = -115$, $a_2 = 110$, $b_1 = -115$, $b_2 = 110$; $x^0 = (0, 0)$, т.е. x^0 совпадает с началом координат, а $\delta_i = 6 \quad \forall i \in \overline{1, 34}$; также пусть задано $N = 34$ контуров и 32 адресных пары, образующие множество $\mathbf{K}$, т.е. ограничения на порядок резки контуров. По соображениям объема мы опустим описание контуров, условий предшествования, а также маршрутов и трасс, ограничившись лишь приведением величины затрат и графиков обхода контуров. На графиках значками $\triangleleft$ отмечены точки врезки, а символами $\triangleright$ — соответствующие им точки выключения резака.

Рассмотрим сначала решение задачи для случая $\varepsilon_i = 10 \quad \forall i \in \overline{1, 34}$. Получены следующие результаты:

величина совокупных затрат: $V(x^0, \overline{1, N}) = 1031.4$;

время счета составило: 14 час. 0 мин. 53 сек.

График маршрута и трассы приведен на рис. 1.

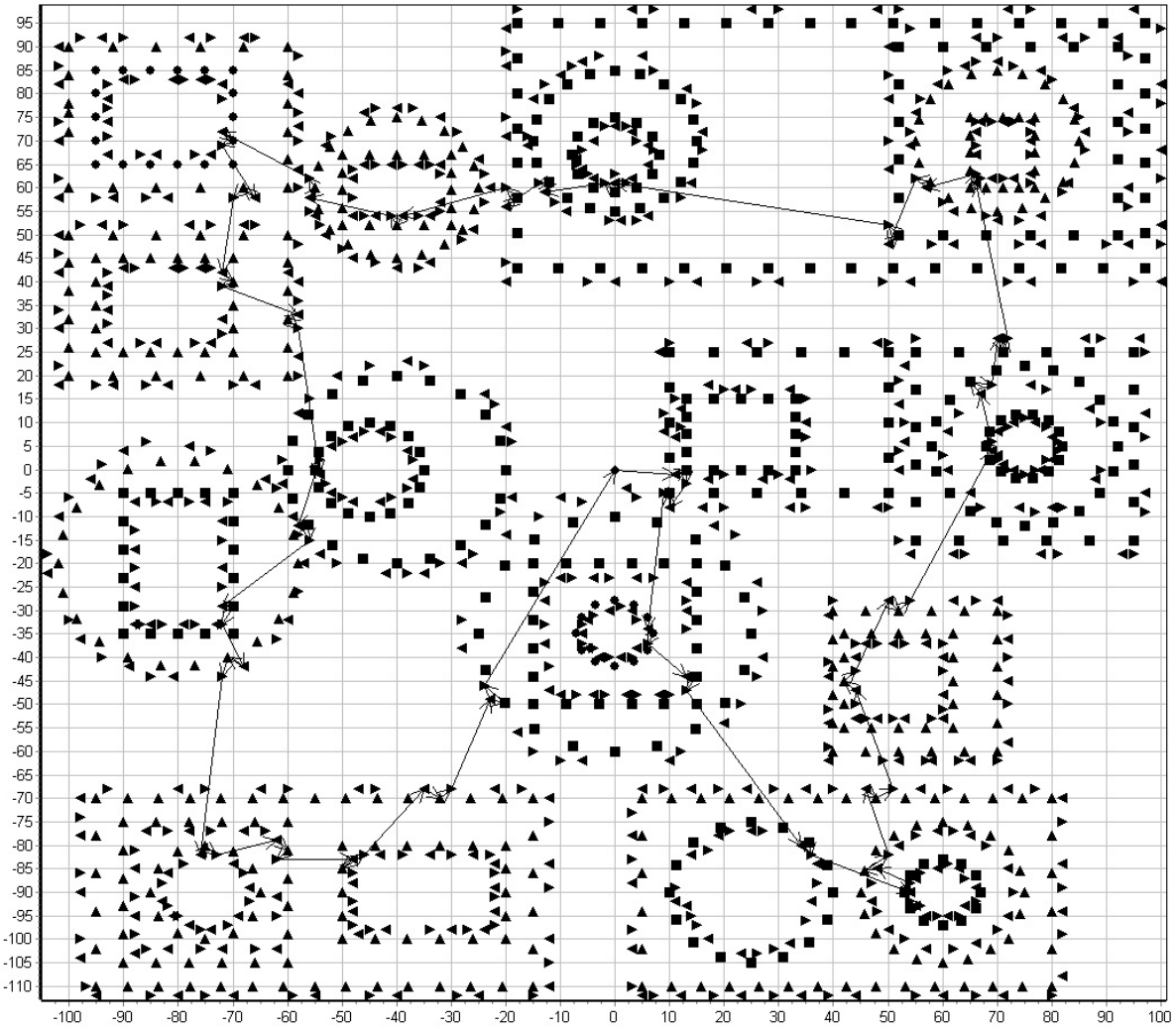

Рис. 1. Маршрут и трасса при значении ε_i равном 10

В ситуации, когда $\varepsilon_i = 50 \ \ \forall i \in \overline{1,34}$, получены следующие результаты:
величина совокупных затрат: $V(x^0, \overline{1,N}) = 1026.4$;
время счета составило: 15 час. 13 мин. 39 сек.
График движения по контурам приведен на рис. 2.

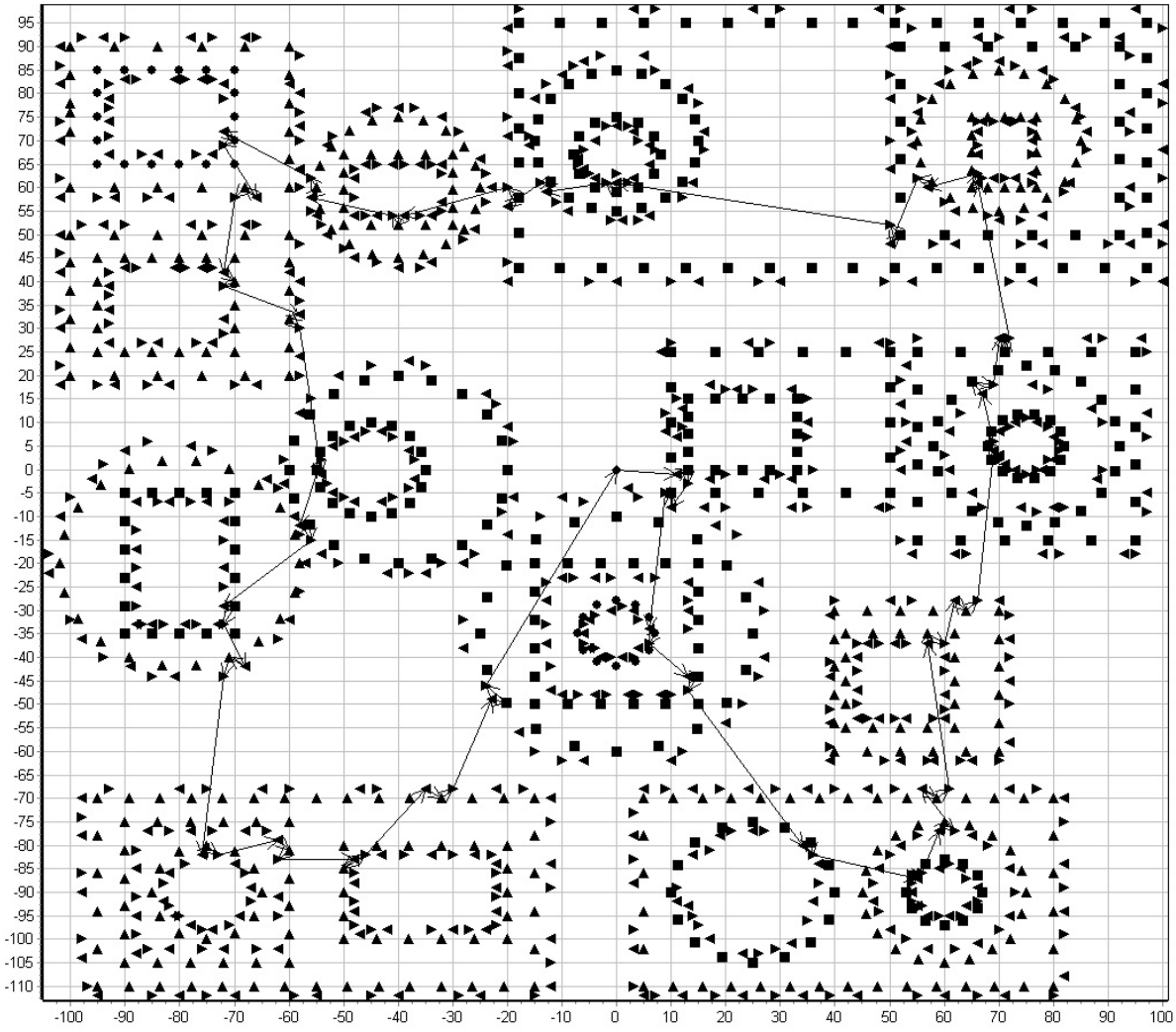

Рис. 2. Маршрут и трасса при значении ε_i равном 50

Как видно из приведенных результатов, с увеличением величин допусков ε_i растет время счета, но уменьшается величина совокупных затрат, что вполне логично: чем больше допуск, тем больше точек врезки доступно для выбора на каждом контуре и тем больше вариантов для оптимизации, но при этом увеличивается объем перебора, что приводит к увеличению времени счета.

Заключение

В статье приведен новый вариант ДП, применимый для построения оптимального решения маршрутной задачи с ограничениями различных типов. На основе данного варианта ДП построен оптимальный алгоритм, реализованный на ПЭВМ; при этом удалось увеличить размерность задач, допускающих точное решение с соблюдением

всех ограничений, с $N \approx 31$ до $N \approx 34$. Последнее существенно в задачах (восходящих идейно к ЗК), для которых своеобразной доминантой размерности является $N!$.

Список литературы / References

[1] Петунин А.А., "О некоторых стратегиях формирования маршрута инструмента при разработке управляющих программ для машин термической резки материала", *Вестник УГАТУ. Серия: Управление, вычислительная техника и информатика*, **13**:2 (35) (2009), 280—286; [Petunin A.A., "O nekotoryh strategijah formirovanija marshruta instrumenta pri razrabotke upravljajushhih programm dlja mashin termicheskoj rezki materiala", *Vestnik UGATU. Serija: Upravlenie, vychislitelnaja tehnika i informatika*, **13**:2 (35) (2009), 280–286, (in Russian).]

[2] Фроловский В.Д., "Автоматизация проектирования управляющих программ тепловой резки металла на оборудовании с ЧПУ", *Информационные технологии в проектировании и производстве*, 2005, №4, 63—66; [Frolovskij V.D., "Avtomatizacija proektirovanija upravljajushhih programm teplovoj rezki metalla na oborudovanii s ChPU", *Informacionnye tehnologii v proektirovanii i proizvodstve*, 2005, №4, 63–66, (in Russian).]

[3] Гэри М., Джонсон Д., *Вычислительные машины и труднорешаемые задачи*, Мир, Москва, 1982; [Gjeri M., Dzhonson D., *Vychislitelnye mashiny i trudnoreshaemye zadachi*, Mir, Moskva, 1982, (in Russian).]

[4] Меламед И.И., Сергеев С.И., Сигал И.Х., "Задача коммивояжера. Вопросы теории", *Автоматика и телемеханика*, 1989, №9, 3–34; [Melamed I.I., Sergeev S.I., Sigal I.H., "Zadacha kommivojazhera. Voprosy teorii", *Avtomatika i telemehanika*, 1989, №9, 3–34, (in Russian).]

[5] Меламед И.И., Сергеев С.И., Сигал И.Х., "Задача коммивояжера. Точные алгоритмы", *Автоматика и телемеханика*, 1989, №10, 3–29; [Melamed I.I., Sergeev S.I., Sigal I.H., "Zadacha kommivojazhera. Tochnye algoritmy", *Avtomatika i telemehanika*, 1989, №10, 3–29, (in Russian).]

[6] Меламед И.И., Сергеев С.И., Сигал И.Х., "Задача коммивояжера. Приближенные алгоритмы", *Автоматика и телемеханика*, 1989, №11, 3–26; [Melamed I.I., Sergeev S.I., Sigal I.H., "Zadacha kommivojazhera. Priblizhennye algoritmy", *Avtomatika i telemehanika*, 1989, №11, 3–26, (in Russian).]

[7] Gutin G., Punnen A.P., *The Traveling Salesman Problem and Its Variations*, Kluwer, 2002.

[8] Сигал И.Х., "Алгоритмы для решения бикритериальной задачи коммивояжера большой размерности", *Техническая кибернетика*, 1990, №6, 143–155; [Sigal I.H., "Algoritmy dlja reshenija bikriterialnoj zadachi kommivojazhera bolshoj razmernosti", *Tehnicheskaja kibernetika*, 1990, №6, 143–155, (in Russian).]

[9] Беллман Р., "Применение динамического программирования к задаче о коммивояжере", *Кибернетический сборник*, **9** (1964), 219–228; [Bellman R., "Primenenie dinamicheskogo programmirovanija k zadache o kommivojazhere", *Kiberneticheskij sbornik*, **9** (1964), 219–228, (in Russian)].

[10] Хелд М., Карп Р.М., "Применение динамического программирования к задачам упорядочения", *Кибернетический сборник*, **9** (1964), 202–218; [Held M., Karp R.M., "Primenenie dinamicheskogo programmirovanija k zadacham uporjadochenija", *Kiberneticheskij sbornik*, **9** (1964), 202–218, (in Russian)].

[11] Ченцов А.Г., *Экстремальные задачи маршрутизации и распределения заданий: вопросы теории*, РХД, Москва–Ижевск, 2008; [Chentsov A.G., *Jekstremalnye zadachi marshrutizacii i raspredelenija zadanij: voprosy teorii*, RHD, Moskva–Izhevsk, 2008, (in Russian).]

[12] Ченцов А.А., Ченцов А.Г., Ченцов П.А., "Экстремальная задача маршрутизации перемещений с ограничениями и внутренними потерями", *Изв. ВУЗов. Математика*, 2010, № 6, 64–81; [Chentsov A.A., Chentsov A.G., Chentsov P.A., "Jekstremalnaja zadacha marshrutizacii peremeshhenij s ogranichenijami i vnutrennimi poterjami", *Izv. VUZov. Matematika*, 2010, № 6, 64–81, (in Russian).]

[13] Ченцов А.А., Ченцов А.Г., Ченцов П.А., "Элементы динамического программирования в экстремальных задачах маршрутизации", *Проблемы управления*, 2013, № 5, 12–21; [Chentsov A.A., Chentsov A.G., Chentsov P.A., "Jelementy dinamicheskogo programmirovanija v jekstremalnyh zadachah marshrutizacii", *Problemy upravlenija*, 2013, № 5, 12–21, (in Russian).]

[14] Ченцов А.Г., "К вопросу о маршрутизации комплекса работ", *Вестник Удм. ун-та. Математика. Механика. Комп. науки.*, 2013, № 1, 59–82; [Chentsov A.G., "K voprosu o marshrutizacii kompleksa rabot", *Vestnik Udm. un-ta. Matematika. Mehanika. Komp. nauki.*, 2013, № 1, 59–82, (in Russian).]

[15] Ченцов А.Г., "Задача последовательного обхода мегаполисов с условиями предшествования", *Автоматика и телемеханика*, 2014, № 4, 170–190; [Chentsov A.G., "Zadacha posledovatelnogo obhoda megapolisov s uslovijami predshestvovanija", *Avtomatika i telemehanika*, 2014, № 4, 170–190, (in Russian).]

[16] Ченцов А.А., Ченцов А.Г., "Динамическое программирование в задаче маршрутизации с ограничениями и стоимостями, зависящими от списка заданий", *Доклады Академии Наук*, **453**:1 (2013), 20–23; [Chentsov A.A., Chentsov A.G., "Dinamicheskoe programmirovanie v zadache marshrutizacii s ogranichenijami i stoimostjami, zavisjashhimi ot spiska zadanij", *Doklady Akademii Nauk*, **453**:1 (2013), 20–23, (in Russian).]

[17] Ченцов А.А., Ченцов А.Г., "Задача маршрутизации с ограничениями, зависящими от списка заданий", *Доклады Академии Наук*, **465**:2 (2015), 154–158; [Chentsov A.A., Chentsov A.G., "Zadacha marshrutizacii s ogranichenijami, zavisjashhimi ot spiska zadanij", *Doklady Akademii Nauk*, **465**:2 (2015), 154–158, (in Russian).]

[18] Куратовский К., Мостовский А., *Теория множеств*, Мир, Москва, 1970; [Kuratovskij K., Mostovskij A., *Teorija mnozhestv*, Mir, Moskva, 1970, (in Russian).]

[19] Дьедонне Ж., *Основы современного анализа*, Мир, Москва, 1964; [D'edonne Zh., *Osnovy sovremennogo analiza*, Mir, Moskva, 1964, (in Russian).]

Chentsov A. G., Chentsov A. A., "Routization Problem Complicated by the Dependence of Costs Functions and «Current» Restrictions From the Tasks List", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 211–227.

DOI: 10.18255/1818-1015-2016-2-211-227

Abstract. The problem of a routization of the movements complicated by the restrictions of different type (preceding conditions, the restrictions on the attainability of states by each movement and others) is considered. A multivariance at the movement step is permitted, it is naturally resulted in the problem about the visiting of megalopolises. The costs of movements and jobs executed when visiting megalopolises may depend on the list of tacks. This list may correspond to performed or unperformed tasks. "Current" restrictions (on movements) may depend on the aforementioned list of tasks. The considered setting is oriented to the application with regard to a nuclear power engineering problems (the problem of decreasing irradiation of the nuclear power station staff when executing a complex of tasks under high radiation intensity) and the machine building. In the second case, which consists in controling a machine for the sheet cutting of details by the numerical program control machines, "current" restrictions on movements may be conditioned by temperature tolerance relative to the fragments of sheet which have already been "visited" by the cutting machine. The scheme of constructing the optimal solution based on the widely understood dynamic programming is considered in this article. The used algorithm is realized on a personal computer; the results of its application are illustrated by the modelling examples.

Keywords: route, trace, preceding conditions

On the authors:
Chentsov Alexander Georgievich, orcid.org/0002-0007-1896-0951, correspondent-member of RAS
N.N. Krasovskii Institute of Mathematics and Mechanics,
S. Kovalevskaya str., 16, Ekaterinburg, 620990, Russia,
Ural Federal University, professor,
Mira str., 19, Ekaterinburg, 620002, Russia, e-mail: chentsov@imm.uran.ru

Chentsov Alexey Alexandrovich, orcid.org/0009-0203-2514-7755, candidate of science,
N.N. Krasovskii Institute of Mathematics and Mechanics,
S.Kovalevskaya str., 16, Ekaterinburg, 620990, Russia, e-mail: chentsov.a@binsys.ru

Acknowledgments:
This work was supported by the Federal targeted Program "Mathematical problems of modern control theory".
This work was supported by the Russian foundation for basic reseach (projects 14-08-00419, 15-01-07909).
This work was supported by Act 211 Government of the Russian Federation, contract № 02.A03.21.0006