ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 34 Выпуск 1

ISSN Online 2220-6426
Volume 34 Issue 1

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2022

ИСП РАН

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

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

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.
Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

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

Журнал издается по одному тому в год, шесть выпусков в каждом томе.
Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.
Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.
The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

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

The journal is published one volume per year, six issues in each volume.
Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

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

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

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2022-34(1)-1

Elicitation of functional requirements from the application programming interface documentation for functional testing

[1] *E.A. Gerlits, ORCID: 0000-0002-1747-075X <gerlits@ispras.ru>*
[1] *D.S. Kildishev, ORCID: 0000-0002-1000-0165 <kildishev@ispras.ru>*
[1,2,3,4] *A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia,*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*
[3] *National Research University, Higher School of Economics 20, Myasnitskaya Ulitsa, Moscow, 101978, Russia*
[4] *Moscow Institute of Physics and Technology (MIPT) 141700, Russia, Moscow region, Dolgoprudny, Campus per., 9*

Abstract. We address a common problem in this paper. The only available documentation for a computer program consists of a user API documentation while we need to identify functional requirements and build test suite to test them. We describe a technique for functional requirements elicitation from the user API documentation. Requirements management tool Requality is exploited in this technique. The tool has been used in several industrial software verification projects.

Keywords: requirements elicitation; requirements extraction; API documentation; Requality; requirements management; functional requirements; requirements markup; requirements catalog.

For citation: Gerlits E.A., Kildishev D.S., Khoroshilov A.V. Elicitation of functional requirements from the application programming interface documentation for functional testing. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 7-22. DOI: 10.15514/ISPRAS–2022–34(1)–1

Выявление функциональных требований в документации программного интерфейса приложения для функционального тестирования

[1] *Е.А. Герлиц, ORCID: 0000-0002-1747-075X <gerlits@ispras.ru>*
[1] *Д.С. Кильдишев, ORCID: 0000-0002-1000-0165 <kildishev@ispras.ru>*
[1,2,3,4] *А.В. Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*
[3] *НИУ Высшая школа экономики, 101978, Россия, г. Москва, ул. Мясницкая, д. 20*
[4] *Московский физико-технический институт, 141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9*

Аннотация. Настоящая работа посвящена решению следующей достаточно распространённой проблемы. Единственной существующей документацией программы является пользовательская документация, описывающая программный интерфейс. Требуется выявить функциональные требования к функциям из программного интерфейса и разработать набор тестов. В работе мы описали метод, руководствуясь которым, можно выявить функциональные требования в пользовательской документации программного интерфейса приложения. Для автоматизации этого метода мы используем инструмент для управления требованиями Requality. Инструмент был использован в нескольких индустриальных проектах по верификации программного обеспечения.

Ключевые слова: выявление требований; извлечение требований; выделение требований; программный интерфейс приложения; Requality; управление требованиями; функциональные требования; разметка требований; каталог требований.

Для цитирования: Герлиц Е.А., Кильдишев Д.С., Хорошилов А.В. Выявление функциональных требований в документации программного интерфейса приложения для функционального тестирования. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 7-22. DOI: 10.15514/ISPRAS–2022–34(1)–1

1. Introduction

Many computer programs provide an application programming interface (API). These are operating systems, software libraries and even social networks, messengers and online services.

API is usually specified in a user API documentation. This kind of documentation commonly includes overall description of the computer program, its subsystems and classes, describes class attributes and methods (functions) including attribute types, possible attribute values, method signatures, types of return values and behavior of methods and functions.

The behavior of functions is usually written in a natural language. Unlike the requirements specification the function description in the API documentation is often incomplete and even may include conflicting and ambiguous statements. The reason is that the API documentation is intended for the needs of users while the requirements specification is intended for the needs of software developers who need carefully written complete set of requirements. In addition, the requirements specification document is often reviewed, verified and corrected during the software development life cycle.

In this paper, we suppose the only written source of the requirements is the user API documentation. The task is to create functional tests for a subset of functions from this API. This situation is quite common in practice. In our experience, a high quality functional requirements specification is usually available if it is required by a standard like DO-178C [1] or an error in the software under development can lead to significant losses (consequences).

Software test engineers often design tests by reading and analyzing the API documentation text and do not explicitly build a requirements catalog over the API documentation. This approach can sometimes be justified, for instance, when smoke tests are to be developed. However, a high quality functional testing implies assessment of test completeness. A common test adequacy criterion for functional testing is the percentage of the requirements verified by the tests. Every test should somehow be traced to the requirements it verifies to be able to calculate the test coverage. As the behavior of functions is described in the user API documentation in plain text, an additional layer of requirements is needed in which every requirement is isolated explicitly and has a unique identifier [2]. This layer of requirements over the user API documentation is usually called the requirements catalog.

In addition, standard ISO/IEC/IEEE 29148-2018 [3] defines key requirements characteristics. Some of these characteristics are difficult to check in plain texts. These are completeness, verifiability and traceability.

In this paper, we publish a technique to build a catalog of functional requirements over the API documentation. Functional requirements are not explicitly listed in the API documentation as it is in the functional requirements specification. Therefore, our technique aims at elicitation of functional requirements from the API documentation. As we already explained the API documentation may contain ambiguous and conflicting statements and other problems typical for plain texts. We address these challenges in our technique.

This paper is structured as follows. We explain our reasons to perform this study in section 2. The goal of this study and the reachability criterion for the goal are expressed in section 3. In section 4, we represent our technique for functional requirements elicitation. This technique has been elaborated during a series of industrial projects on software requirements elicitation. We briefly mention these projects in section 5. We overview investigations related to our study and show the novelty of our study in section 6. We explain why we reach the goal of this study with our technique and come to some conclusions in section 7. Section 8 contains a reference list.

2. Motivation

To our mind, challenges associated with elicitation of functional requirements from the API documentation may be overcome by applying:

- a proper requirements elicitation process;
- effective solutions for technical and scientific problems;
- automation of labor intensive tasks.

We address all these aspects in this paper. We present a requirements elicitation process elaborated during a number of industrial projects. We also offer solutions to some markup issues of API documentation text. We automate routine and labor intensive tasks with our requirements management solution Requality [4].

Industrial requirements management tools [5] like Requality usually come along with a requirements management process [6]. These processes do not usually address the problem of requirements elicitation from written sources. We cover this gap with this paper presenting a requirements elicitation process for API documentation and similar documents like API-related standards.

3. Problem statement

Suppose we have an API documentation. One should extract functional requirements from this documentation, build a catalog of functional requirements and supplement the catalog with new requirements obtained from other sources.

A proper requirements elicitation technique should meet the following requirements:

1) The technique should construct a functional requirements catalog in which every requirement has the unique identifier.

2) The technique should support traceability of requirements to text fragments that represent those requirements in the API documentation.

   Such a traceability relation can be used to estimate coverage of the documentation text by the requirements markup.

3) The technique should tolerate possible changes in the API documentation text.

   Problems are often discovered in the documentation during requirements analysis. The author of the documentation fixes them and issues a corrected version of the documentation text. Some text fragments may be changed due to fixes. These changed text fragments might already be traced to existing requirements in the previous version of the documentation. In this case, we should transfer the existing mapping of requirements to text fragments onto the next (fixed) version of the documentation.

4) The technique should assist in requirements refinement which is the primary way to improve understanding of the system under test.

5) The technique should assist in building a complete set of requirements for functions, subsystems and the whole system.

6) The technique should not rely on a particular natural language.

4. Requirements elicitation technique

In this section, we describe different aspects of our technique for functional requirements elicitation.

4.1 Demo example

All examples in this paper refer to function *select* from POSIX [7] standard. This function waits until an event is registered for one of the file descriptors provided. There are three types of events:

- a file descriptor is ready for reading;
- a file descriptor is ready for writing;
- an error is registered for a file descriptor.

Function *select* is a complex one. The main reason is that the function handles different file types differently. There are three main file types:

- regular files;
- sockets;
- terminals and pseudo terminals.

In this paper, we assume for simplicity that *select* function is applied to regular files only.

4.2. Requirements catalog structure

Let us build our requirements catalog in the form of a *tree* [8] in which:

- vertices are requirements;
- an arc between two incident requirements represents the refinement relation, i.e. the requirement having the higher depth refines an aspect of the requirement having the lower depth.

Before inserting a new requirement into the requirements catalog the following characteristics should be checked [3]:

- the new requirement is interpreted unambiguously;
- the new requirement does not contradict the existing requirements;
- the new requirement cannot be logically deduced from the existing requirements;
- the new requirement has already been implemented or is going to be implemented and meets the actual system or user need;
- the new requirement defines a single aspect of a function, subsystem or system.

Proving each characteristic is a challenge but it is often not necessary. For instance, a requirement can be considered unambiguous if all team members interpret the requirement in the same way. A meeting [9] of all team members can be appointed to address this issue.

If one of the above conditions is not met, we should correct the new requirement or the requirements catalog or both. Thus, by adding a new requirement to the requirements catalog we either:

- refine another requirement;
- or improve completeness of requirements.

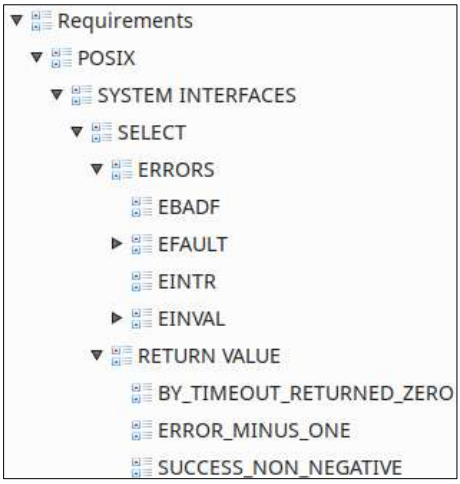

Fig. 1. A fragment of a requirements catalog

The requirement analyst assigns mnemonic names to the requirements while Requality requirements management tool automatically generates unique identifiers for them. The path from the root of the requirements catalog to a requirement may be used as a *unique link* to the requirement. Such a link is often used to trace a test to the requirement verified by the test. For instance, the unique link to requirement *EBADF* on fig. 1 is the following: */Requirements/SELECT/ERRORS/EBADF*.

The structure of the requirements catalog usually reproduces the structure of the API documentation up to a certain level. For instance:

1) The root requirement formulates the main task, goal or mission of the computer program.
2) The root requirements for the subsystems or main components of the computer program reside on the first level of the requirements tree (catalog) and formulate the main task or goal of the corresponding subsystem or component.
3) The root requirements for the classes and global functions (C programming language) reside on the second level and formulate the main task or goal of the corresponding class or function.
4) The root requirements for the methods of classes reside on the third level. Functional requirements for the global functions start to appear on third level too.
5) Functional requirements for the class methods begin on the fourth level.

Description of individual functions in the API documentation is often generated from structured source code comments, e.g. written with Javadoc, Doxygen and etc. Structured nature of source code comments leads to some structure in the description of functions in the API documentation. The structural description of functions in documentation, in turn, is projected onto the requirements catalog. For instance, description of *select* function in POSIX [7] contains the following structure of headings: *NAME*, *SYNOPSIS*, *DESCRIPTION*, *RETURN VALUE*, *ERRORS*. Sections *NAME* and *SYNOPSIS* do not contain any functional requirements. Sections *RETURN VALUE* and *ERRORS* describe the return value of the function and possible error codes respectively. The corresponding headings become child requirements of requirement *SELECT*.

4.3. Requirements text structure

Each requirement should have a textual description:

- a non-empty set of text fragments from the documentation;
- or a manually written text by a template.

Text fragment is a continuous word sequence in a document. Using Requality one can mark text fragments in a document and then link them to a requirement. It is possible to switch between text fragments and corresponding requirements with a single mouse click. There is, for example, the following text fragment in the description of function *select*: *if function select ends by time limit then return 0*. We assign this text fragment to requirement BY_TIMEOUT_RETURNED_ZERO. An additional manual description of the requirement is not mandatory because the text fragment describes the requirement properly.

Requirements have built-in attributes in Requality, e.g. a mnemonic name, a manual description, as well as user defined attributes. If needed, we write manual descriptions for functional requirements according to the following template [10]:

The function/subsystem/system MUST [actions set], [if/while/until CONDITION]. Actions can be:

- a modification of the internal state of the computer program;
- or return of a specific value from the function;
- or an exception.

Conditions are usually made up of logical predicates and logical operations like conjunction, disjunction and negation.

Here is an example. A manual description for requirement ERRORS can be as follows: *Function select MUST write the unique error code into variable [errno] if an error happened.*

A condition is skipped if actions are unconditional. If the subject of the actions is obvious then the phrase *Function/subsystem/system* MUST may also be omitted.

All requirements without text fragments from the API documentation (but with manual descriptions) are considered to come from other sources like an interview with developers or completeness analysis of requirements.

It is sometimes better to represent requirements in the form of tables, images and models. Requality supports tables and arbitrary images in manual descriptions of requirements.

4.4. Heuristic technique

We formulate a set of heuristics in this section that make a requirements elicitation technique effective. We tried to implement them in our technique.

1) Attempts to achieve the best possible requirements quality characteristics are often irrational. The quality of requirements should be enough to achieve the planned testing quality defined in the completeness criterion.
2) The API documentation is not supposed to contain the complete set of functional requirements. Therefore, the authors of the documentation, designers, developers and test engineers become an important source of functional requirements. They help to elicit and improve the requirements catalog [11].
3) The use of special requirements management tools like Requality greatly facilitates and simplifies the requirements elicitation process and the maintenance of large requirements catalogs.
4) Improvement of the API documentation is required to improve the quality of the requirements extracted from that documentation.
5) Representation of some requirements in a more suitable form rather than textual one and visual modeling of unclear requirements help to improve the requirements quality [7]. Effective non-textual requirements representations include tables and formulas. Data flow diagrams [12],

UML action diagrams, UML state diagrams, decision tables [13] and other models [14] can be used to model different aspects of requirements.

6) A group work on a complex problem like requirements elicitation is efficient since it implies mutual assistance and participation of engineers having complementary qualifications. However, an excessive team may have a negative impact on the efficiency of requirements elicitation.
7) The use of a task management system, bug tracking system and a version control system helps to support controllable, goal-oriented, responsible interaction of team members and to meet project deadlines.

4.5. Roles of participants

The process of requirements elicitation is a process of organized and controlled interaction of participants:

1) Requirement analyst (analyst):
   - helps technical project manager to assign priorities to functions;
   - extracts functional requirements from the API documentation;
   - supplements the requirements catalog with requirements from other sources;
   - reveals issues in the requirements catalog and fixes them;
   - fixes issues in the requirements catalog reported by other participants;
   - reveals issues in the API documentation;
   - participates in requirements verification procedures.
2) Author of the API documentation:
   - creates the API documentation;
   - reveals issues in the API documentation;
   - fixes issues in the API documentation;
   - as an important source of functional requirements provides them to the analyst;
   - participates in requirements verification procedures.
3) Test engineer:
   - formulates a test completeness criterion;
   - traces tests to requirements;
   - as an important source of functional requirements provides them to the analyst;
   - reports issues found in the requirements catalog during testing;
   - helps technical project manager to assign priorities to functions;
   - participates in requirements verification procedures.
4) Developer of the computer program (developer):
   - as an important source of functional requirements provides them to the analyst;
   - answers to the questions about the implementation of the computer program;
   - helps technical project manager to assign priorities to functions;
   - participates in requirements verification procedures.
5) Technical project manager (manager):
   - ensures the requirements elicitation process to meet its deadline;
   - assigns priorities to functions;
   - organizes productive interaction of all team members;

- helps other participants to solve various issues they find it difficult to solve by oneself.

All participants use a version control system to track changes in the API documentation and requirements catalog as well as to assign release versions to them. All issues discovered in the API documentation and requirements catalog are trucked in a bug tracking system. We track all tasks (as opposed to issues) in a task management system. Typical tasks are the following: markup requirements for a function, verify completeness of the set of requirements for a function, assign priorities to functions.

4.6. Requirements elicitation process

4.6.1 Preparatory stage

At first, the analyst structures functions provided for testing by dividing them into subsets. A subset may contain interface functions to a single subsystem or functions performing related operations like send and receive, write and read.

Then the analyst looks up the pages in the API documentation describing each subset of functions generally and each function individually in details, i.e. the analyst constructs the appropriate relations between subsets of functions and documentation pages and between individual functions and documentation pages. The analyst supplements this relation during the whole requirements elicitation process.

Then the manager, the analyst and the test engineer assign priorities to the subsets of functions and to the individual functions as well. Priorities depend on different factors: relative complexity of functions, the size of the describing documentation text, restrictions imposed by the terms of reference document, available human and time resources, etc. Priorities are not static and are subject to change during the requirements elicitation process.

The participants of the requirements elicitation process track the progress of all activities using a task management system or a bug tracking system. They assign priorities to the tasks taking into account the priorities of the interface functions that are being worked on.

4.6.2 Requirements markup

The analyst marks up requirements in the documentation respecting the priorities of the interface functions. This process is accompanied by a number of problems.

Does the fragment of text describe a requirement? One can focus on the keywords that may indicate the presence of a requirement in a sentence [15]. For example, POSIX [7] standard requires to use the verb **must** in requirement statements: *Upon successful completion of the function, pselect () and select () must return the total number of bits specified in bitmasks.*

In general, a requirement cannot be recognized in a given set of text fragments on the basis of syntax rules. A functional requirement is a statement about what the computer program, a subsystem or a function should do under a condition or unconditionally. The object of actions in a software system is data, e.g. a return value of a function, the internal state of the computer program, an exception, a message. Let us assume that a set of text fragments formulates one or more requirements if:

- an action is performed or a number of actions;
- the subject of the action is the computer program, a subsystem or a function;
- the object of the action is some data;
- the action should be performed under a condition or unconditionally.

What should be done with text fragments that do not contain any requirements? Requality marks text fragments containing requirements (assigned to a requirement node) with a color. Unmarked text fragments have not been analyzed yet. To be able to control the completeness of documentation markup, text fragments that have been analyzed and are known to not contain any requirements should be separated from unmarked text fragments. Requality tool supports nodes of

type *text node* (as opposed to *requirement* nodes). All text fragments that do not contain any requirements are assigned to nodes of this kind and are marked with a color.

How should the analyst resolve a text problem found in the API documentation? Some examples of text problems are the following: unclear meaning of a certain text fragment, a contradictory statement, an ambiguous statement.

The analyst should create a task in the bug tracking system and make the author of the API documentation responsible for it to be resolved.

Should the analyst refine context dependent text fragments? The context of text fragments can affect their meaning. We recommend to clarify context-dependent text fragments in the requirements catalog. Requality provides *description* attribute in requirement nodes for this purpose.

4.6.3 Suspension criterion for the requirements markup of a function

The requirements markup process for a function is gradually approaching a state when further progress is either limited or difficult. The main natural reason for this is the limited function description in the API documentation. In addition, a large number of issues may slow down the requirements markup process for a function. An iterative markup of functions has proven to be productive.

Criterion 4.1 (Suspension criterion for the requirements markup of a function). *For every text fragment in the function description the following should hold:*

- either the text fragment is marked up, i.e. referred to a requirement or a text node;
- or a task has been created in a bug tracking system concerning the text fragment.

The criterion can examine extra values if needed:

- the number of issues revealed in the function description;
- the amount of marked-up text;
- the amount of time spent on elicitation of requirements for the function;
- the complexity of the function expressed numerically.

4.6.4 Transfer of requirements catalog onto a new API documentation release

While the requirements catalog is being built over a certain documentation release, the API documentation is being naturally improved. When a new documentation release comes out, the requirements analyst should reflect the new improvements in the requirements.

Requality helps to transfer the existing requirements catalog onto a new documentation release. Text fragments that have not changed are mapped to the requirements catalog automatically. Other text fragments are mapped in a semiautomatic manner.

4.6.5 Verification of requirements catalog for a function

When the function description in the documentation has been completely marked up, it is necessary to decide whether the requirements are ready for testing or should be preliminary verified.

High complexity of a function is the key factor in favor of verification. However, complexity is difficult to estimate. The following values can be taken into account:

- the time spent to build the requirements catalog for the function;
- the number of issues in a bug tracking system related to the function;
- the size of the function description in the API documentation;
- the number of completed iterations of the requirements elicitation and etc.

Many existing verification techniques demonstrate effectiveness when applied to requirements: formal inspection [16], equivalence partitioning [17], boundary value analysis [18], decision table analysis [13], meeting [9], consultation, interview [19] and questionnaires [19] [20].

All issues found during verification should be tracked through the bug tracking system. We recommend to verify requirements according to a preliminary elaborated plan which may include the following information:

- the object of verification, i.e. the requirements for a function or a set of functions;
- the place, the date, the start and the finish times;
- a list of participants and their tasks;
- the subject of verification, i.e. the requirements properties to be verified such as uniqueness, completeness, consistency;
- verification method.

4.6.6 Necessary conditions to stop the requirements elicitation process for a function

The requirements analyst, software tester and developer should come to the same understanding of the functions' behavior. This can be achieved when they successfully complete all tasks dedicated to the function.

Proposition 4.1 (Necessary conditions to stop the requirements elicitation process for a function):

- *the markup of the function description in the API documentation should have been completed;*
- *all tasks in the bug tracking system directed to correction of the function description in the API documentation should have been completed;*
- *all tasks in the bug tracking system directed to correction of the requirements catalog for the function should have been completed;*
- *all tasks in the task management system directed to supplementation of the requirements catalog for the function from other sources should have been completed;*
- *all requirements verification tasks for the function in the task management system should have been completed and the subject of the verification included:*
  - *unambiguity of the requirements;*
  - *the accuracy of the leaf requirements in the requirements catalog;*
  - *consistency of the set of requirements;*
  - *completeness of the set of requirements.*
- *the function has been tested and all found errors have been fixed.*

4.6.7 Feedback from functional testing

API testing is always automated. Tests should be written in accordance with the functional requirements. Tracing establishes links between tests and requirements being verified. Thus, testing allows us to naturally confirm verify-ability of requirements.

The test engineer usually discovers many issues in the requirements during the requirements-based test design. In addition, some inconsistencies between the requirements and the observed program's behavior may be due to issues in the requirements. The test engineer should create a task in the bug tracking system for all discovered issues and make the requirements analyst responsible to resolve requirements' related issues.

Test development and analysis of the discovered issues highly improve the test engineer's understanding of the function's behavior. Therefore, the analyst should engage the test engineer in requirements verification and supplementation procedures. This will help to ensure the completeness of the requirements catalog and improve the overall quality of the requirements.

4.6.8 Necessary conditions to stop the requirements elicitation process

The whole requirements elicitation process finishes when all participants have successfully completed their tasks.

Proposition 4.2 (Necessary conditions to stop the requirements elicitation process)*:*

- *the markup of the API documentation or its target part should have been completed;*
- *all tasks in the bug tracking system directed to correction of the API documentation should have been completed;*
- *all tasks in the bug tracking system directed to correction of the requirements catalog should have been completed;*
- *all tasks in the task management system directed to supplementation of the requirements catalog from other sources should have been completed;*
- *all requirements verification tasks in the task management system should have been completed;*
- *all target API functions should have been tested and all found errors have been fixed.*

5. Approbation

Our technique for functional requirements elicitation has evolved during a number of industrial projects. Those projects have been performed by the authors of this study and other researchers from the software engineering department of ISPRAS [21].

The technique has recently been used for requirements elicitation on input-output multiplexing functions from POSIX [7] standard:

- poll;
- select;
- pselect.

The above functions were implemented in a real time operating system. The API of the operating system was described in an API documentation. A requirements catalog consisting of 317 functional requirements has been built as a result of a multi-iterative requirements elicitation process. Dozens of errors were found in the API documentation including:

- incompleteness of information;
- inaccuracy of statements;
- conflicts with POSIX [7] standard.

We have also used Requality to build requirement catalogs for some parts of the following standards and specifications:

- ARINC 653 [22];
- TTCN-3 interface specifications;
- several RFCs including RFC 826, RFC 760 and RFC 768.

6. Related work

One way or another, individual ideas or solutions expressed in this paper might already be published in books or applied in practice. However, we couldn’t find any requirements elicitation techniques characterized as ours:

- the method is aimed at a specific type of documentation, i.e. the user API documentation;
- the method uses feedback from functional testing to enhance the quality of requirements;
- the method is effective in practice due to the use of a specialized software tools like Requality requirements management tool.

Our research group developed requirements management tool Requality. We know for sure there are no publications concerning techniques for requirements elicitation on the basis of this software tool. We fill this gap by publishing this study.

There are several alternative industrial tools [5]. Most of them are used in requirements development from scratch. They assist in building requirements catalogs and support relations between requirements originated at different levels of the software life cycle:

- business requirements;
- system requirements;
- functional requirements.

But most of these requirements management tools cannot maintain links between documentation fragments and related requirements as Requality does.

NLP (natural language processing) methods [23] are widely used to extract various information like requirements from texts written in a natural language. NLP methods are usually well automated therefore they effectively analyze big text data.

Information about hierarchy of classes and methods of these classes is extracted with NLP methods from the API documentation in study [24]. Methods are divided into categories:

- create a resource;
- lock access to a resource;
- modify a resource;
- unlock access to a resource;
- delete a resource.

Then an automaton is created for every resource on the basis of the extracted information. The automaton is then used to reveal defects in the computer program. For instance, the use of a resource before creating it.

Our study and work [24] are similar in that we analyze the same type of documentation, i.e. the API documentation. The both studies have the same goal to improve the quality of computer programs. However, the methods to reach this goal are different. We look for functional requirements. The authors of study [24] look for defects.

NLP methods can be used to verify some characteristics of requirements. For instance, software tool QuARS [25] can reveal ambiguity of text and subjectivity of text (not a requirement but a personal opinion). LOLITA [26] analyzes text and incorporates it in a semantic net [27]. Then possible text interpretations are looked up.

Complex lexis, syntax and morphology of some natural languages and many exceptions from the language rules make it more difficult to apply NLP methods to analyze texts written in those languages. From one hand, these complexities restrict application of NLP methods. From the other hand, these complexities simply become responsibilities of the analyst in our technique.

Application of several requirements elicitation techniques leads to synergistic effect. The choice of a complementary technique depends on human resources available, i.e. the number of engineers, their experience, qualification and etc. Techniques effectively complementing our requirements elicitation technique do not strictly relate to mark up of texts. Among them are the following methods:

- formal inspection [16];
- equivalence partitioning [17];
- boundary value analysis [18];
- decision table analysis [13];
- meeting [9];
- consultation;
- interview [19];
- questionnaires [19] [20].

Correction of problems in the API documentation is an important part of our requirements elicitation technique. There are methods specially designed to reveal errors in the API documentation. For instance, texts written in a natural language are analyzed with NLP methods and source code snippets are analyzed by a code analyzer in study [28]. Combination of two different types of analyzes helps to find inconsistencies between a text fragment and a source code snippet.

A method for requirements elicitation from the user documentation for a legacy system is proposed in study [29]. The extracted requirements are then used to create a functional specification document for a new system similar to the legacy system. Text structure, key words, lexical, syntactical and other text characteristics are used to extract key features of the system, functional requirements, use cases and nonfunctional requirements.

7. Conclusion

In this paper, we propose a technique for functional requirements elicitation from the user API documentation. By means of this technique the requirements analyst can create a catalog of functional requirements suitable for functional testing.

The requirements catalog is a tree in which every requirement has a unique identifier. This tree structure assists in functional requirements refinement and usually reproduces the structure of the API documentation up to a certain level.

Markup of all documentation text with requirements is a necessary condition for a requirements set to be complete. Our requirements management tool Requality maintains links between documentation fragments and related requirements helping us to transfer the requirements catalog onto upcoming documentation versions.

We support requirements obtained from non-written sources, e.g. provided by team members or as a result of a requirements analysis. We write down them in a natural language by a template and replenish the requirements catalog with them.

An acceptable quality of requirements is obtained due to systematic verification procedures, feedback from functional testing and team collaboration through a version control system, a bug tracking system and a task management system.

References / Список литературы

[1] DO-178C. Software Considerations in Airborne Systems and Equipment Certification. RTCA SC-205 and EUROCAE WG-12 Std., 01 2012.

[2] V.V. Kulyamin, N.V. Pakulin et al. Formalization of requirements in practice. Preprints of the Institute for System Programming of the Russian Academy of Sciences, Preprint 13, 2006, 70 p. (in Russian) / В.В. Кулямин, Н.В. Пакулин и др. Формализация требований на практике. Препринты Института системного программирования РАН, Препринт 13, 2006 г., 70 стр.

[3] ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering, ISO and IEC and IEEE Std., 11 2018.

[4] D. Kildishev and A. Khoroshilov. Developing requirements management tool for safety-critical systems. In Proceedings of the International Conference on Actual Problems of Systems and Software Engineering, 2019, pp. 50-57.

[5] N. Gorelits, D. Kildishev, and A. Khoroshilov. Requirement management for safety-critical systems. Overview of solutions. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 25-48 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-2 / Н.К. Горелиц, Д.С Кильдишев, А.В. Хорошилов. Управление требованиями к ответственным системам. Обзор решений. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 25-48.

[6] P. Zielczynski. Requirements Management Using IBM Rational RequisitePro. IBM Press, 2007, 360 p.

[7] Portable Operating System Interface, ISO and IEC and JTC 1/SC 22 Std., Rev. 9945:2009, 09 2009.

[8] A. Khoroshilov and D. Kildishev. Formalizing metamodel of requirements management system. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 5, 2018, pp. 163-176. DOI: 10.15514/ISPRAS-2018-30(5)-10.

[9] R. Ocker, J. Fjermestad et al. Effects of four modes of group communication on the outcomes of software requirements determination. Journal of Management Information Systems, vol. 15, issue 6, 1998, pp. 99-118.
[10] K. Wiegers and J. Beatty. Software Requirements. Microsoft Press, 2013, 672 p.
[11] Z. Zhang. Effective requirements development – A comparison of requirements elicitation techniques. In Proc. of the International Conference on Software Quality Management, 2007, pp. 225–240.
[12] T. DeMarco. Structure Analysis and System Specification. In Pioneers and Their Contributions to Software Engineering, Springer, 1979, pp. 255-288.
[13] R. Shiffman and R. Greenes. Improving clinical guidelines with logic and decision-table techniques: Application to hepatitis immunization recommendations. Medical Decision Making, vol. 14, no. 3, 1994, pp. 245-254.
[14] J. Beatty and A. Chen. Visual models for software requirements. Microsoft Press, 2012, 480 p.
[15] N. Niu and S. Easterbrook. Extracting and modeling product line functional requirements. In Proc. of the 16th IEEE International Requirements Engineering Conference, 2008, pp. 155-164.
[16] M. Fagan. Design and code inspections to reduce errors in program development. IBM Systems Journal, vol. 15, no. 3, 1976, pp. 182-211
[17] D. Richardson and L. Clarke. A partition analysis method to increase program reliability. In Proc. of the 5th International Conference on Software Engineering, 1981, pp. 244-253.
[18] S. Reid. An empirical analysis of equivalence partitioning, boundary value analysis and random testing. in Proc. of the Fourth International Software Metrics Symposium, 1997, pp. 64-73.
[19] S. Sharma and S. Pandey. Article: Revisiting requirements elicitation techniques. International Journal of Computer Applications, vol. 75, no. 12, 2013, pp. 35-39.
[20] S. Kimani, E. Panizzi et al. Digital Library Requirements: A Questionnaire-Based Study. In Handbook of Research on Digital Libraries: Design, Development, and Impact. Information Science Reference, 2009, pp. 287-297.
[21] Open-Source Projects. Available: https://forge.ispras.ru/projects
[22] S. Santos, J. Rufino et al. A portable ARINC 653 standard interface. In Proc. of the 2008 IEEE/AIAA 27th Digital Avionics Systems Conference, 2008, pp. 1.E.2-1-1.E.2-7.
[23] J. Eisenstein. Introduction to Natural Language Processing. MIT Press, 2019, 536 p.
[24] H. Zhong, L. Zhang et al. Inferring specifications for resources from natural language api documentation. In Proc. of the 2009 IEEE/ACM International Conference on Automated Software Engineering, 2009, pp. 307-318.
[25] G. Lami. Quars: A tool for analyzing requirement. Technical report CMU/SEI-2005-TR-014 ESC-TR-2005-014, Carnegie Mellon University, Software Engineering Institute, 2005.
[26] L. Mich. Nl-oops: From natural language to object oriented requirements using the natural language processing system Lolita. Natural Language Engineering, vol. 2, no. 2, 1996, pp. 161-187.
[27] L. Schubert, R. Goebel, and N. Cercone. The structure and organization of a semantic net for comprehension and inference. In Associative Networks. Academic Press, 1979, pp. 121-175.
[28] H. Zhong and Z. Su. Detecting api documentation errors. In Proc. of the 2013 ACM SIGPLAN International Conference on Object-Oriented Programming Systems, Languages and Applications, 2013, pp. 803-816.
[29] I. John and J. Dörr. Elicitation of requirements from user documentation. In Proc. of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality, 2003.

Information about authors / Информация об авторах

Evgeny Anatolyevich GERLITS – researcher at the Software Engineering Department of the Institute for System Programming of the RAS. Main research interests: software dynamic verification, software testing, software quality assurance, static and dynamic program analysis.

Евгений Анатольевич ГЕРЛИЦ – научный сотрудник Института системного программирования. Сфера научных интересов: методы динамической верификации программ, методы тестирования и обеспечения качества программного обеспечения, статический и динамический анализ программ.

Denis Stepanovich KILDISHEV is a junior researcher of the Software Engineering Department. His research interests include requirements management tool development.

Денис Степанович КИЛЬДИШЕВ является младшим научным сотрудником отдела технологий программирования. Его научные интересы включают разработку инструмента управления требованиями.

Alexey Vladimirovich KHOROSHILOV – Ph.D. in Physics and Mathematics, Leading Researcher, Director of the Linux OS Verification Center at ISP RAS, Associate Professor of System Programming Departments at Moscow State University, the Higher School of Economics, and Moscow Institute of Physics and Technology. Main research interests: design and development methods for critical systems, formal methods of software engineering, verification and validation methods, model-based testing, requirements analysis methods, Linux operating system.

Алексей Владимирович ХОРОШИЛОВ – кандидат физико-математических наук, ведущий научный сотрудник, директор Центра верификации ОС Linux в ИСП РАН, доцент кафедр системного программирования МГУ, ВШЭ и МФТИ. Основные научные интересы: методы проектирования и разработки ответственных систем, формальные методы программной инженерии, методы верификации и валидации, тестирование на основе моделей, методы анализа требований, операционная система Linux.

DOI: 10.15514/ISPRAS-2022-34(1)-2

Requirements traceability as the basis for designing a functional and logical architecture of a software system

[1,2] *B.A. Pozin, ORCID: 0000-0002-0012-2230 < bpozin@ec-leasing.ru >*
[3] *G.N. Tsiperman, ORCID: 0000-0001-7740-5629 <gntsip@gmail.com>*

[1] *National Research University Higher School of Economics,*
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia
[2] *EC-leasing,*
125 Varshavskoye shosse, Moscow, 117405, Russia
[3] *National University of Science and Technology "MISIS"*
Leninskiy Prospekt 4, Moscow, 119049, Russia

Abstract. The paper deals with the formation and transformation of stakeholder requirements for the information system throughout the entire life cycle. It is shown how the seamless architecture provides traceability of requirements from the level of the business process, to the functional and logical architectures of systems, to the selection of criteria and identification of microservices. It is shown how maintaining the traceability of requirements in the presence of business, functional and logical architecture models can reduce the cost of planning complex functional and load testing of systems, as well as ensure the interaction of operation, maintenance services and contractors that form the entire system, maintain its integrity during the life cycle.

Keywords: information system; requirements engineering; requirements traceability; adaptive clustering method; seamless design; microservice criteria; business architecture; functional architecture

For citation: Pozin B.A., Tsiperman G.N. Requirements traceability as the basis for designing a functional and logical architecture of a software system. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 23-34. DOI: 10.15514/ISPRAS–2022–34(1)–2

Acknowledgements. The article is based on the materials of the report at the Seventh International Conference «Actual Problems of System and Software Engineering» (APSSE 2021).

Прослеживаемость требований как основа проектирования функционально-логической архитектуры программной системы

[1,2] *Б.А. Позин, ORCID: 0000-0002-0012-2230< bpozin@ec-leasing.ru >*
[3] *Г.Н. Циперман, ORCID: 0000-0001-7740-5629 <gntsip@gmail.com>*

[1] *НИУ Высшая школа экономики,*
101978, Россия, г. Москва, ул. Мясницкая, д. 20
[2] *ЕС-лизинг*
117587, Россия, г. Москва, Варшавское шоссе, д. 125, стр.1
[3] *Национальный исследовательский технологический университет «МИСИС»*
119049, Москва, Ленинский пр-кт, д. 4, стр. 1

Аннотация. В статье рассматриваются вопросы формирования и трансформации требований заинтересованных сторон к информационной системе на протяжении всего жизненного цикла.

Показано, как бесшовная архитектура обеспечивает прослеживаемость требований от уровня бизнес-процесса к функциональной и логической архитектурам систем, к выбору критериев и идентификации микросервисов. Показано, как поддержка прослеживаемости требований при наличии моделей бизнес-архитектуры, функциональной и логической архитектур позволяет снизить затраты на планирование комплексного функционального и нагрузочного тестирования систем, а также обеспечить взаимодействие служб эксплуатации, сопровождения и подрядчиков, формирующих систему в целом, сохранять ее целостность в течение всего жизненного цикла.

Ключевые слова: информационная система; разработка требований; прослеживаемость требований; метод адаптивной кластеризации; бесшовный дизайн; критерии микросервиса; бизнес-архитектура; функциональная архитектура

Для цитирования: Позин Б.А., Циперман Г.Н. Прослеживаемость требований как основа проектирования функционально-логической архитектуры программной системы. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 23-34. DOI: 10.15514/ISPRAS–2022–34(1)–2

Благодарности. Статья подготовлена по материалам доклада на Седьмой Международной конференции «Актуальные проблемы системной и программной инженерии» (АПСПИ 2021).

1. Introduction

Due to increasing complexity of software systems, an increasing number of stakeholders (customers, users, developers, administrators, maintenance staff) are involved in ensuring their life cycle. This leads to the need to ensure the transfer of knowledge about requirements to the software system and the system itself between stakeholders. The key task of this process is to provide an understanding of how the original business requirements are translated into functional and non-functional requirements, requirements for the deployment and maintenance of the system.

For understanding, it is necessary to provide a model representation of the system architecture at various levels of abstraction, where the initial requirements are determined based on the stakeholders needs at the business level, and then consistently (seamlessly), without loss or distortion, are converted into requirements for the software system.

Thus, three main tasks are formulating that arise at different stages of the system life cycle (creation, commissioning and maintenance), the solution of which will be considered in this work:

- Adequate implementation and completeness of the functionality of the system, which is validated by the end user, easily understood by him.
- Deployment, i.e., adaptability to easy deployment by the operating unit of the owner of the system, while maintaining an understanding of which business automation functions or entities are located on each element of the deployment.
- Maintenance, i.e., ease of understanding of the logical architecture of the system by the end user and the ability to set tasks for changes to counterparties.

The solution of these problems will be considered as the evolution of the requirements for the system: their definition, transformation and modification. In [1], the need to obtain a coherence set of requirements that provides two-way traceability between the requirements for system functions and the sources of these requirements based on the attribution of requirements.

In this paper, the method of adaptive clustering (ACM) of information systems (IS) is used as a tool for managing a set of requirements for the system that is consistent at all levels of abstraction [2]. CM is a top-down design method, where detailed architectural models are built on the basis of decomposition of the higher abstraction level models elements, ensuring their seamless connection.

Seamless connection is a connection in which the process of decomposition does not violate the traceability of transitions between elements of architectural models.

2. Formulation of the problem

To describe the functional requirements for a software system, it is customary to use particularly Use Case diagrams, which provide a good definition of the requirements for the system functions, linking them to specific business needs Use Case diagrams use knowledge about the business process implicitly, reflecting on a variety of formalized use cases, existing separately from them, without giving a complete picture of the possibilities of automating the business process.

To ensure the completeness of the definition of business requirements, it is necessary to obtain such requirements directly from the business process model, at the level of the business architecture. This is how the initial business requirements for the system are determined in the ACM method [2], in which UML activity diagrams and business process decomposition techniques are used to describe them.

At the next level of architecture (functional architecture), the requirements for the functions of the system are determined based on the business requirements accepted by the stakeholders. The functional architecture models the interaction of the system with users, as well as with other external agents. The functional architecture forms the appearance of the system based on the representation of the compositions of the system's dialogues, defines the requirements for the dialogues and specifies the interfaces with external systems.

The modern practice of designing a functional architecture of a system involves a heuristic transition between architectural models of various levels of abstraction: a more detailed functional architecture is built in such a way that technical solutions meet business requirements as much as possible. In doing so, the architect uses his experience and knowledge to build a detailed model, and then proves (or considers it obvious) that the resulting model satisfies the requirements arising from a more abstract architecture.

As a result, there is no obvious transition from the requirements for automated functions defined during the business architecture stage to the requirements for system functions defined by the functional architecture. This means that there is a technological gap («seam») between architectures of different levels of abstraction. The presence of a seam does not allow tracing the relationship between the requirements for the system functions and the requirements for the functions of the business process that need to be automated.

At the logical architecture level, the technological gap does not allow to transparently identify which microservices implement specific business needs [3] declared in the system, do not provide an opportunity at the business architecture or functional architecture level to define the bounded context and microservice aggregates [4], to give “physical meaning" to the components of the microservice architecture, thereby concretizing the system deployment plan that is understandable to the system owner. In this case, the traceability of the initial requirements in the logical architecture of the system is violated.

Finally, the technological gap does not allow to correctly build a comprehensive testing plan for the system, and also complicates the process of localizing errors and making changes to the current implementation of the system.

Summarizing, we can formulate the following problems that this work is devoted to:

1) The emergence of technological gaps (seams) between the levels of decomposition and the need to form and use rules for eliminating seams to ensure a traceable functional decomposition of business requirements for the system.
2) The technological gap between the business architecture and the functional architecture of the system and how to bridge it.
3) Lack of traceability of requirements in the design of a microservice architecture, considered as a set of units of development, deployment and maintenance.
4) The impossibility of using tracked functional models for planning complex testing of the system, its regression testing and assessing its performance using load testing methods.

5) Impossibility to use the traceability mechanism when planning work on making changes to the maintenance process.

3. Decomposition as a method of representing a business process model

In ACM, it is the business architecture that is the starting point for the design of IS. Business architecture in the context of IS design is understood as a set of models of automated business processes of an enterprise [5]. In the context of this work, a target business architecture is considered, that is, a model of business processes taking into account the use of an information system. The task is to determine, based on the decomposition of the business process, a complete set of business process functions that can be automated.

Decomposition, as a method of representing a business process model, creates a classification scheme that groups operations in accordance with a system of intermediate results of a business process. Decomposition is not the only way to represent a business process. In the limiting case, it can be described as a sequence of operations leading to the desired result. However, such a description for real processes, where the number of operations is in the hundreds, is difficult to understand and analyze.

It is operations (hereinafter we will call them "business operations") that are the ultimate goal of decomposition and are objective in nature. Business operations are located on the leaves of the decomposition tree, and the rest of the structure of the decomposition tree is intended to define a complete list of business process operations.

3.1 Definition of business process decomposition

A business process is understood as a set of actions in which, on the basis of one or more types of initial data [resources], a result is created that is valuable to the client [6]. To describe the decomposition of a business process, the concepts of business function and business operation are sufficient, which are based on the concept of action (*A*), understood as the transformation (*T*) of an initial resource (*Ir*) into a target product (*Tp*).

Decomposition defines the following elements of a business architecture:

- Business function - an action to obtain an intermediate target product required to solve the task of a business process. The business function is always decomposing.
- Business operation is an action that represents the final result of the decomposition within the description of a business process. The result of the business operation itself is not a decomposable target product, but an executor – a specific role function of personnel.

The transformation function T can include components that are predefined information objects (*Po*) that are part of the transformation mechanism (for example, directories or values of the necessary available constants and variables). Predefined objects are created and modified in other contexts. Thus, the action can be defined by a four:

$$A := (Ir, Po, T, Tp).$$

Initial resource (*Ir*) is an information object of the system that initiates an action:

- defines that part of the context of the current state of the system, which is necessary to obtain the target product of this action;
- can represent one or more input information objects, which are the target products of previous actions that initiate this action.
- Target product (*Tp*) is an information object of the system that represents the result of an action:
- defines a part of the context of the new state of the system, which is necessary for the further advancement of the business process or its completion;

- can be one or more information objects that represent an intermediate result of a business process.

Functional decomposition of an action is an operator ($\widehat{D}$), defined on a set of actions ($\boldsymbol{\mathcal{A}}$), representing an action as a set of action elements that implement the original action:

$$\widehat{D}A \to (A_1, A_2, \dots, A_n), A_i \in \mathcal{A}.$$

The action elements obtained as a result of the decomposition are interconnected by relations representing the control flows that initiate the decomposition elements (Fig. 1). By the scenario of a business function, we mean a logical composition that includes:

- business function decomposition elements $(A_1, A_2, \dots, A_n)$;
- initial and final script element from a set of decomposition elements $A_{in}, A_{fin} \subset (A_1, A_2, \dots, A_n)$;
- set of binary relationships between elements $R(A_i, A_j) \in (\mathcal{R}, \emptyset)$, where $\mathcal{R}$ - is the set of admissible connections between actions.

3.2 Coherence requirements of the decomposition model

The coherence of the decomposition model of a business process is understood as the absence of contradictions between the elements of the model, compliance with the reality in terms of describing business operations.

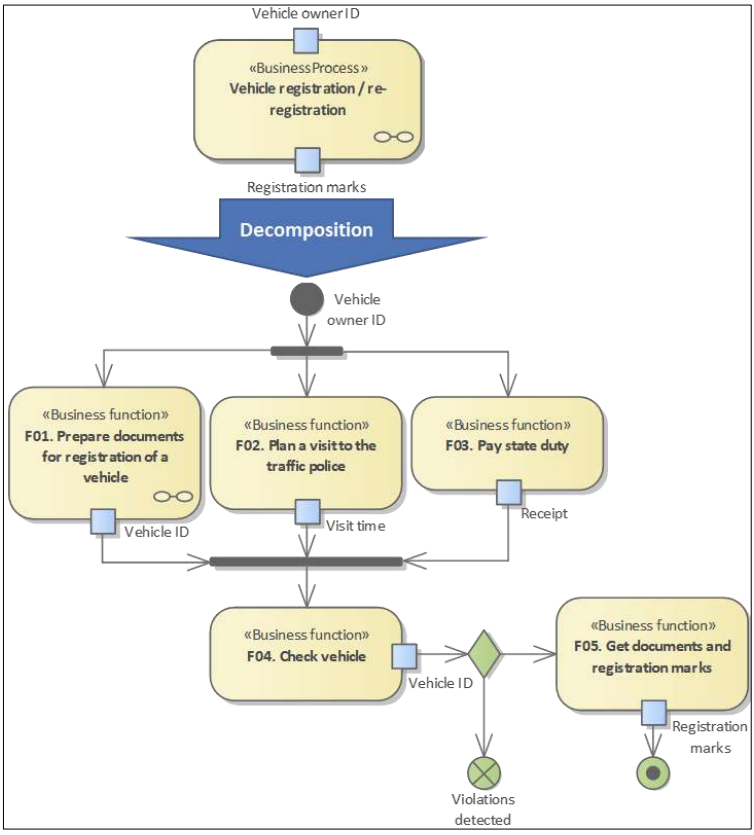

Fig. 1. The first level of a coherent business process decomposition

Two types of coherence are defined:

- *Functional coherence*. Assumes that the action goals of the decomposition elements correspond to the action goals of the decomposed element. To achieve functional consistency, it is necessary to ensure [7]:
  - a) continuity of decomposition - compliance with the sequence of transitions between the levels of the decomposition tree;
  - b) absence of defects - each lower level must reveal the previous one completely, without missing a single element;
  - c) lack of redundancy - it is unacceptable to add during division something that is not in the original.
- *Information (product) coherence*. As an action, the business function converts the *Ir* to *Tp* using the required predefined information objects. The business function decomposition elements, organized into a business function script, should collectively perform the same transformation: the *Ir* at the entrance to the business function decomposition script must match the *Ir* of the decomposed business function, and the final of the script must have a *Tp* that matches the *Tp* of decomposed business function (see fig. 1). Within the script, there must be decomposition elements using the predefined information objects defined for the business function being decomposed. In the scenario itself, new information objects can arise corresponding to the *Ir* and *Tp* of the decomposition elements, and additional predefined information objects can also be used.

Decomposition of a business process allows you to cluster a subject area, dividing it into subdomains corresponding to business functions, with well-defined functional and informational boundaries. Each subdomain represents a bounded context [4], which can be used as the basis for defining a development task for a development team when implementing a microservice architecture of a software system.

4. Formation of microservices

4.1 The task of identifying microservices

In [3], the principle of organizing microservices is declared: the division into services in accordance with the needs of the business. This means that the identification of microservices must be done at the level of the business architecture, i.e., the microservice must have a "physical" meaning. At the level of the architectural (logical) model, there must be an element that defines the boundaries of what we mean by a microservice.

Approaches to the identification of microservices ([8], [9]) mainly dealt with the issues of clustering the IS code according to the criteria of strong and weak connectivity, while simultaneously clarifying the principles that the identified microservices should comply with. The resulting technologies give unstable good results.

The main task for the solution of which the microservice architecture was created is the struggle with the complexity of the development, deployment and maintenance of IS. However, no less important, in our opinion, is the link between the requirements for the system and the system architecture. The key element in this is the concept of a business operation: each business operation must correspond to a microservice that implements its functionality. The input and output of the microservice is determined by the respective Ir and Tp of the business operation. Such a microservice as an architectural element, in our opinion, should correspond to a deployment unit, which will allow tracing in which microservices of the developed and deployed system a particular business operation is implemented, as well as checking the composition of the implemented information context.

This approach requires clustering the functionality and, accordingly, the subject area of the system into maximally independent elements, each of which defines its own limited development context.

This bounded context is determined by the business function. At the same time, there is sufficient freedom in choosing a development unit: the closer a business function is to the root of the decomposition tree, the greater the amount of development it defines.

The independence of development and maintenance objects implies a minimal connection between such objects both in terms of interaction and in terms of using common code: a quick response to changing requirements for one microservice does not imply an order to change the code in another microservice.

With this approach, the criteria for identifying microservices is determined only by business needs, and not by the needs of optimization of development. This is quite consistent with the concept of development and maintenance outlined in [3].

4.2 Criteria for identifying microservices

To determine the criteria for identifying microservices, it is appropriate to give the following analogy. Imagine a car manufacturing industry. Each final product (car) is the result of industry cooperation of various kinds of enterprises that produce both components for assembling a car and the necessary resources. Each enterprise forms its own production infrastructure for the corresponding type of product and, to a reasonable extent, minimizes dependence on related enterprises that are part of the cooperation.

The consistency of the components of the final product is ensured by the necessary standards and contracts between enterprises for the supply of the corresponding component or resource. The contract with the enterprise fixes the terms of delivery of the manufactured product, determined by its name and code within the framework of cooperation.

By this analogy, a microservice is understood as an enterprise included in a cooperation and responsible for the production of one product, and the basis for defining a microservice is a business operation since it meets two main criteria:

- Single Responsibility Principle (SRP) – a business operation is performed under the responsibility of a specific role function of personnel to change the state of one Tp.
- Common Closure Principle (CCP) – The non-decomposable result of a business operation is represented by a single information aggregate (*Tp*).
- The proposed criteria for identifying microservices [10] as deployment units can be formulated as follows:
- When creating an IS, it is difficult to predict which parts of the system will change more often and which less often as a result of operation. To ensure the independence of changes to microservices during operation, the SRP principle can be applied, according to which a microservice should have one responsibility and be associated with one role of an IS user, since the main reason for changes is users.
- Microservice is a software component responsible for changing the state of one specific type of object during its life cycle. In this case, the type of object is an aggregate [4]: a cluster of related objects, which are considered as a whole for the purpose of changing data. External references are limited to one member of the aggregate, designated as the root. This is in line with the CCP principle.

In [8], 16 criteria for the identification of microservices were proposed, concerning, in addition to CCP and SRP, also the issues of determining the development boundaries. Based on a catalog of such criteria, the required system specification artifacts are identified, which can be processed in a structured, semi-automated manner to propose service decomposition that promotes weak communication between services and a high degree of consistency within them. At the same time, the criteria included in the catalog are determined only because, according to the authors, the methods of forming aggregates and defining bounded contexts in the requirements are not known.

5. Seamless transition between business and functional architectures

5.1 Operational Service

The business architecture defines the functional requirements of stakeholders to IS. Business operations are the places where user interactions with the system occur, and therefore the description of business operations allows you to determine the functions that should be automated (automated functions). The set of these functions, defined for a business operation, constitutes the content of the business operation service (hereinafter referred to as the operational service). Operational services are generated for each business operation that includes automated functions. They are architectural elements that are abstract in nature, do not imply any implementation and are intended only to explicitly define the functions to be automated (fig. 2).

An operational service defines the boundary between the business architecture and the functional architecture of the IS. It formalizes the requirements for the automation of business operations and serves as the basis for the formation of technical specifications for the creation or development of IS.

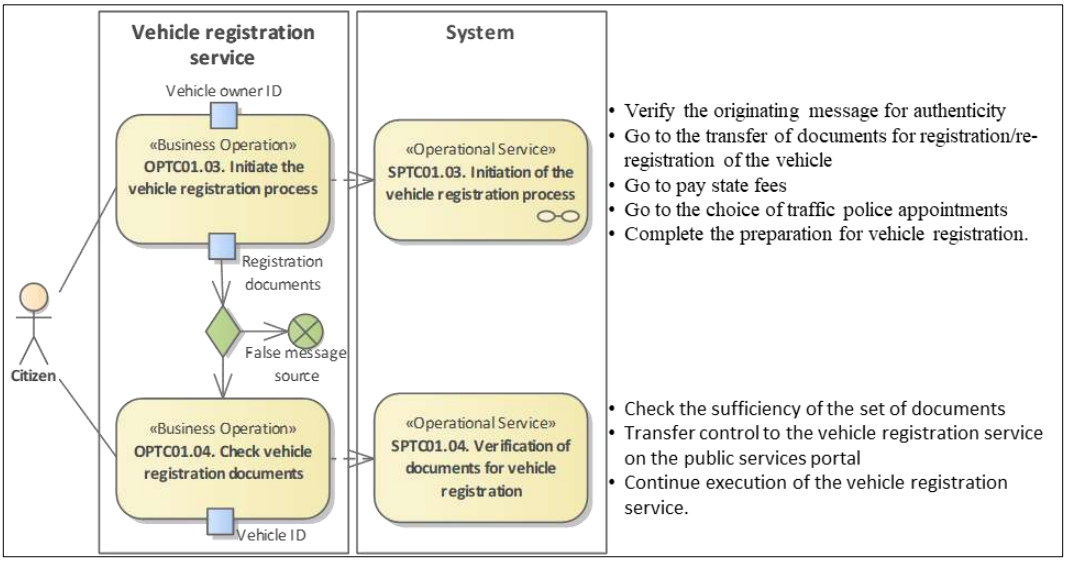

Fig. 2. Business operations, operational services and automated functions (fragment of the diagram)

5.2 Transition to functional architecture

Functional architecture is understood as an architectural representation that includes architectural models of the structure and composition of functional components of the IS that provide access to the "internal" functions of the IS that implement automated functions. In other words, the functional architecture models the interaction of the IS with users, as well as with other external agents. The functional architecture forms the appearance of the IS based on the presentation of the compositions of the system's dialogues, defines the requirements for the dialogues and specifies the interfaces with external systems.

The operational service is the connecting element that defines the seamless connection between the business architecture and the functional architecture: the operational service is decomposed into IS dialogs, the composition of which forms the operational service scenario. Automated functions of a business operation are implemented in dialogs [2].

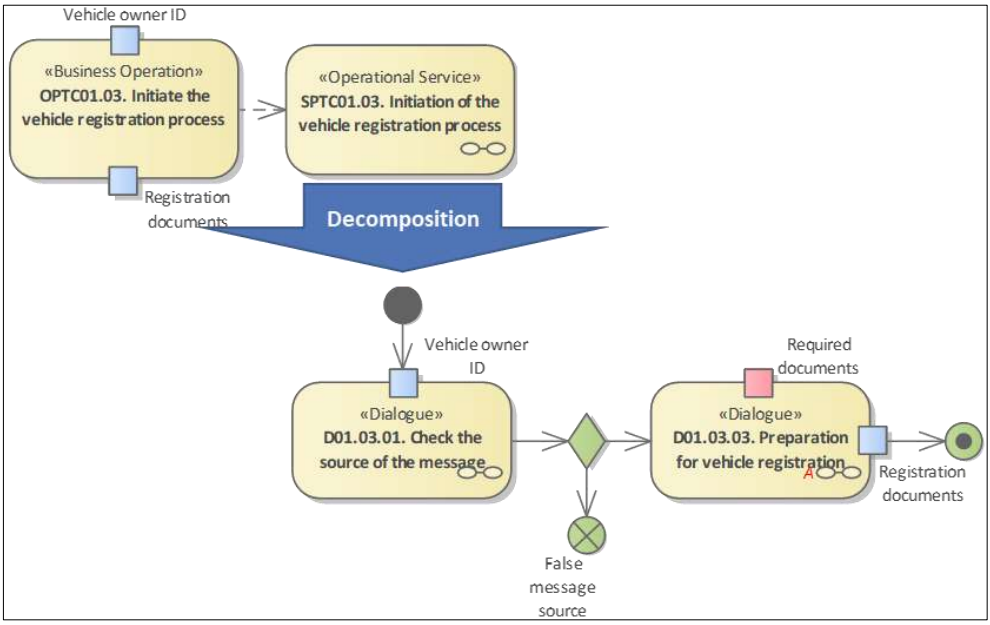

Fig. 3. Seamless transition to functional architecture

In the example (fig. 3), the operational service SPTC01.03 is decomposed into two dialogs D01.03.01 and D01.03.03, which form the service script. The informational coherence of the operational service script with the decomposed service is determined by the fact that the information object "Vehicle owner ID" is passed to the script input, and the "Registration documents" is sent to the output. For dialog D01.03.03, the predefined information object "Required documents" is specified, which is a directory of documents required for registration of vehicles.

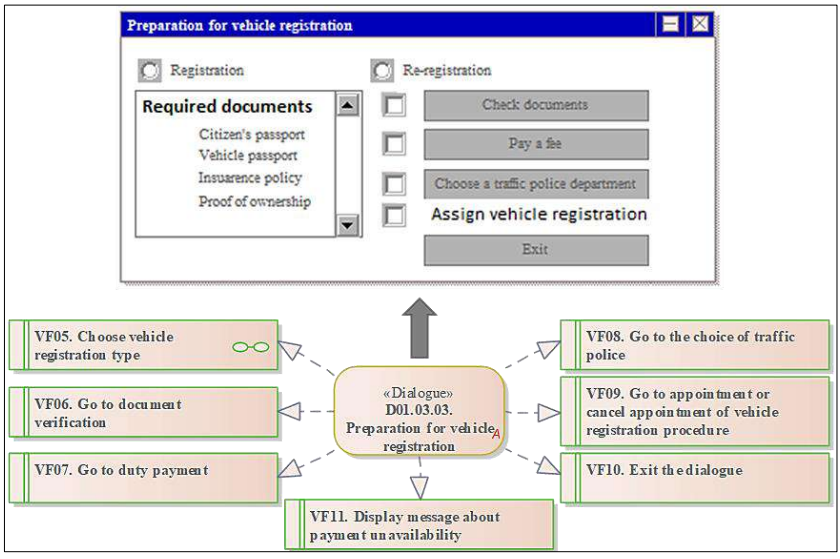

Fig. 4. Description of the dialog: form and view functions

A system dialogue is understood as any act of interaction between agents that causes a change in the state of the IS by launching the corresponding software components [11]. Thus, the dialogue is understood in a broad sense - it is not only the interaction of the user with the system, but also the exchange of messages between any specified IS objects. Dialogues are described [2]:

- form (if it is a dialogue between the user and the system);
- functions of the IS presentation level (view functions), i.e., functions that provide access to the implementation of automated functions;
- various constraints (preconditions, postconditions, error handling, etc.), which are also considered as view functions (fig. 4).

Since a business operation corresponds to a microservice, then, ultimately, it is the implementation of the operational service that determines its functional and informational boundaries, as well as the interfaces between the IS microservices.

6. Complex system testing

The described approach, based on creating a seamless decomposition from a business process model to a microservice that implements a business operation, while maintaining the traceability of system requirements, makes it possible to build plans for complex testing of systems, including integrated complex testing, on a single basis.

Static complex testing to verify the implementation of system functionality can be planned based on business function models as a basis for designing complex tests, and detailed development of test cases and database tables - based on the analysis of business operation models, taking into account business function scenarios.

A static complex plan [12] is easy to document: based on a business function scenario, it is easy to develop and document test quality criteria. This allows not only to generate tests for a single run, but also, taking into account the traceability of requirements, to accumulate tests for regression complex testing, taking into account the achievable degree of coverage of business function scenarios with complex tests.

Dynamic (load) complex testing also requires planning and building (as rule with the participation of stakeholders) dynamic load testing scenarios [12]. For each scenario of carrying out load testing of the system, the formation of a load model is carried out on the basis of scenarios of business functions, taking into account the assessment of the frequency (or probability) of using each business function in the process of functioning of the system being validated on a model or perspective load. In this case, load flows are determined from the standpoint of business rules - scenarios for the implementation of business functions, and the load model is formed taking into account the frequency of calls of end users and other actors to business operations of the system.

7. Planning for change

The planning of functional changes that are supposed to be introduced into the system, when using the current models of all the described levels, is greatly simplified. When shaping changes in the business process model, even a not very deep analysis allows you to understand which business functions from the existing ones will be affected by the changes or what new business functions will need to be implemented. In both cases, changing the boundaries of the system is explicitly reflected in the business function models and their scenarios.

Likewise, the analysis shows which business operations are subject to change or creation. In this case, the control of the integrity of changes is carried out according to the available documents describing the corresponding models, taking into account the traceability of requirements. Based on the lists of business operations subject to change, an appropriate list of microservices to be changed is determined. Thus, the volume of changes can be quickly assessed down to specific software and information components. The same mechanism allows you to estimate the labor intensity of preparing tests for static complex testing.

8. Conclusion

Integration processes are extremely important for complex custom software systems with a lifetime of 10 years or more. The correct formulation of such processes in the life cycle of the system can significantly reduce the costs of creating, maintaining and developing such systems and significantly affect their integrity and quality. The key issue in solving the issues of ensuring the integrity of systems is to ensure the traceability of requirements at the levels of business processes, the formation of functional and logical architectures of systems up to the selection of criteria and identification of microservices.

The paper shows how to provide a functional decomposition of business requirements, describing it in the UML language, correctly identifying the structured elements of the decomposition. Such a structured approach provides up to the logical architecture model to ensure the preservation of the traceability property of requirements, makes it possible to completely get rid of" seams " not only during the design and development of the system, but also during its operation, maintenance and development. The result of such work makes it possible to support the processes of system operation, its maintenance and development, to provide a fairly simple way to manage the configuration of the system being operated by the personnel of the organization that owns the system.

It is shown how maintaining the traceability of requirements in the presence of business, functional and logical architecture models can reduce the cost of planning complex functional and load testing of systems, as well as ensure the interaction of operation, maintenance services and contractors that form the entire system, maintain its integrity during the life cycle. This interaction using the models described in the work allows you to quickly localize the errors that have appeared and identify microservices that are subject to changes, as well as identify requirements that are violated and require more accurate implementation – starting with business requirements.

References / Список литературы

[1] V.K. Batovrin, B.A. Pozin. Requirements Engineering at a Modern Industrial Enterprise. Software Engineering, vol. 10, no. 3, 2019, pp. 114-124 (in Russian) / В.К. Батоврин, Б.А. Позин. Инженерия требований на современном промышленном предприятии. Программная инженерия, том 10, no. 3, 2019 г., стр. 114–124.

[2] G. Tsiperman. Seamless design of information system architecture based on adaptive clustering method. Proceedings of the 6th International Conference Actual Problems of System and Software Engineering (APSSE 2019), CEUR Workshops, vol. 2514, 2019, pp. 38-44.

[3] James Lewis, Martin Fowler. Microservices. A definition of this new architectural term. 25 March 2014. Available at: https://martinfowler.com/articles/microservices.html.

[4] Eric Evans. Domain-Driven Design Reference. Definitions and Pattern Summaries. Dog Ear Publishing, LLC, 2014, 88 p.

[5] ISO/IEC/IEEE FDIS 29148:2017 Systems and software engineering. Life cycle processes. Requirements engineering. Available at: https://www.iso.org/standard/72089.html.

[6] Michael Hammer, James Champy. Reengineering the Corporation. A manifesto for business revolution, Harper Collins Inc., 1993, 272 p.

[7] I.G. Fedorov. A principles of a process model decomposition. Applied informatics, vol. 11, no. 5(65), pp. 19-30 (in Russian), 2016 / И.Г. Фёдоров. Принципы декомпозиции модели процесса. Прикладная информатика, том 11, no. 5(65), 2016, стр. 19-30.

[8] Michael Gysel, Lukas Kölbener et al. Service Cutter: A Systematic Approach to Service Decomposition. Lecture Notes in Computer Science, vol. 9846, 2016, pp. 185-200.

[9] Shanshan Li, He Zhang et al. A dataflow-driven approach to identifying microservices from monolithic applications. The Journal of Systems and Software, vol. 157, 2019, article no. 110380.

[10] C. Richardson. Microservices Patterns: Microservices Patterns: With examples in Java. Manning, 2019, 520 p.

[11] I. Graham. Object-Oriented Methods. Principles and Practice, Third Edition, Addison-Wesley, 2001. 864 p.

[12] B. Pozin, I. Galakhov. Experience in automated functional and load testing in the life cycle of the mission-critical system Baltic Journal of Modern Computing, vol. 8, no. 2, 2020, pp. 241-258.

Information about authors / Информация об авторах

Boris Aronovich POZIN – Doctor of Technical Sciences, Professor of HSE University, CTO of EC-leasing Co. Expert in Software Engineering field, ESSENCE, Software Systems Performance and Integration Testing, Life Cycle Supporting Systems development, implementation and automation. Author of over 200 publications.

Борис Аронович ПОЗИН – доктор технических наук, профессор НИУ ВШЭ, технический директор ЗАО «ЭК-лизинг». Эксперт в области программной инженерии, ESSENCE, тестирования производительности и интеграции программных систем, разработки, внедрения и автоматизации систем поддержки жизненного цикла. Автор более 200 публикаций.

Grigory Naumovich TSIPERMAN – Information system analysis and design expert. Worked in a number of Russian IT companies. Currently, an independent expert. Being Lead Architect and Technical Supervisor, implemented a number of projects of corporate systems, public administration projects, including the Russian system of execution, production, and issuance of biometric citizen passports. The scope of scientific interest includes the development of information systems and knowledge bases, and project management. The author of over 30 scientific publications.

Григорий Наумович ЦИПЕРМАН – специалист по анализу и проектированию информационных систем. Работал в ряде российских ИТ-компаний. В настоящее время независимый эксперт. В качестве ведущего архитектора и технического руководителя реализовал ряд проектов корпоративных систем, проектов государственного управления, в том числе российской системы оформления, производства и выдачи биометрических паспортов граждан. В сферу научных интересов входит разработка информационных систем и баз знаний, управление проектами. Автор более 30 научных публикаций.

DOI: 10.15514/ISPRAS-2022-34(1)-3

Security threat level estimation for untrusted software based on TrustZone technology

D.O. Markin, ORCID: 0000-0001-5823-0632 <mdo@academ.msk.rsnet.ru>
S.M. Makeev, ORCID: 0000-0002-7451-8115 <maksm57@yandex.ru>
T.T. Ho, ORCID: 0000-0003-1149-8791 <mdo@academ.msk.rsnet.ru>
Russian Federation Security Guard Service Federal Academy,
35, Priborostroitel'naya st., Oryol, 302015, Russia

Abstract. The paper proposes a model for assessing the security of information processed by untrusted software from the components of the TrustZone technology. The results of vulnerability analysis of TrustZone technology implementations are presented. The structure of the trustlets security analysis tool has been developed. The paper deals with the problem of assessing the credibility of foreign-made software and hardware based on processors with the ARM architecture. The main results of the work are the classification of trustlets using their threat level assessment and the model of security threat level estimation of information processed by trustlets. Trustlets are software that operates in a trusted execution environment based on TrustZone technology in computers with ARM processors. An assessment of the security of information processed by trustlets for some implementations of trusted execution environments was carried out. The structural scheme of the analysis tool that allows identifying potentially dangerous code constructs in binary files of trustlets is presented. Also analysis tool's algorithm performing syntactic analysis of trustlet data is described. The calculation of the security assessment is carried out on the basis of a set of features proposed by authors. Calculated security assessment levels can be used to classify trustlets that are part of «trusted» operating systems based on TrustZone technology. The levels of potential threat to the security of the information they process are used to differ trustlets during certification tests and vulnerability search. It is advisable to use the results of the work in the interests of conducting certification tests of computer software based on processors with ARM architecture.

Keywords: ARM; TrustZone; trustlet; security estimation; software vulnerabilities

For citation: Markin D.O., Makeev S.M., Ho T.T. Security threat level estimation for untrusted software based on TrustZone technology. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 35-48. DOI: 10.15514/ISPRAS–2022–34(1)–3

Acknowledgements. The article is based on the materials of the report at the Seventh International Conference «Actual Problems of System and Software Engineering» (APSSE 2021).

Оценка уровня защищенности недоверенного программного обеспечения на основе технологии TrustZone

Д.О. Маркин, ORCID: 0000-0001-5823-0632 <mdo@academ.msk.rsnet.ru>
С.М. Макеев, ORCID: 0000-0002-7451-8115 <maksm57@yandex.ru>
Ч. Т. Хо, ORCID: 0000-0003-1149-8791 <mdo@academ.msk.rsnet.ru>
Академия Федеральной службы охраны Российской Федерации,
302015, Россия, г. Орёл, ул. Приборостроительная, д. 35

Аннотация. В работе предлагается модель оценки защищенности информации, обрабатываемой недоверенным программным обеспечением, состоящим из компонентов технологии TrustZone. Представлены результаты анализа уязвимостей реализаций технологии TrustZone. Разработана

структура инструмента анализа защищенности трастлетов. В статье рассматривается проблема оценки надежности программно-аппаратных средств иностранного производства на базе процессоров с архитектурой ARM. Основными результатами работы являются классификация трастлетов с использованием оценки уровня их угроз и модель оценки уровня угроз безопасности информации, обрабатываемой трастлетами. Трастлеты – это программное обеспечение, работающее в доверенной среде выполнения на основе технологии TrustZone на компьютерах с процессорами ARM. Проведена оценка защищенности информации, обрабатываемой трастлетами, для некоторых реализаций доверенных сред исполнения. Представлена структурная схема инструмента анализа, позволяющего выявлять потенциально опасные конструкции кода в бинарных файлах трастлетов. Также описан алгоритм инструмента анализа, выполняющего синтаксический анализ данных трастлета. Расчет оценки безопасности осуществляется на основе комплекса признаков, предложенных автором. Вычисленные уровни оценки безопасности можно использовать для классификации трастлетов, которые являются частью «доверенных» операционных систем на основе технологии TrustZone. Уровни потенциальной угрозы безопасности обрабатываемой ими информации используются для разграничения трастлетов при сертификационных испытаниях и поиске уязвимостей. Результаты работы целесообразно использовать в интересах проведения сертификационных испытаний программного обеспечения для ЭВМ на базе процессоров с архитектурой ARM.

Ключевые слова: ARM; TrustZone; трастлет; оценка безопасности; уязвимости программного обеспечения

Для цитирования: Маркин Д.О., Макеев С.М., Хо Т.Т. Оценка уровня угроз безопасности для ненадежного ПО на основе технологии TrustZone. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 35-48. DOI: 10.15514/ISPRAS–2022–34(1)–3

Благодарности. Статья подготовлена по материалам доклада на Седьмой Международной конференции «Актуальные проблемы системной и программной инженерии» (АПСПИ 2021).

1. Introduction

In the modern market of microprocessor technology, an impressive share is occupied by processors with the ARM architecture, which are a licensed development of the British company of the same name. The vast majority of mobile devices, numerous sensors, sensors, "Internet of Things" devices, on-board vehicle systems and even multiprocessor high-performance data processing systems are built on the basis of processor data. The developer of the circuitry of these devices has laid down the functionality of the processor in two modes: the so called "Secure World" – "trusted" mode, and "Normal World" – untrusted.

At the same time, in order for the software to work in the "trusted" mode, its developer must use an electronic signature, which can be obtained only if the corresponding license of the copyright holder is available. The described technology is called TrustZone, which determines the features of the functioning of computer hardware based on ARM processors, as well as ways to implement system and application software, their interaction when working in different operating modes of the ARM processor, the composition of technical and software tools that protect system components from unauthorized access, modification or blocking.

It is important to note that when using the vast majority of devices based on ARM processors, the user does not have the ability to manage the so-called "trusted" software loaded by the device provider into permanent memory. The described "trusted" software in terms of the developer the ARM company is called trustlet (TA – Trusted Application). These factors and numerous studies suggest that the use of such devices for processing protected information is strictly prohibited in a number of structures due to the lack of trust in both the technical means of the devices and its software. In this regard, there is an objective need to use a set of measures to assess the level of security of information processed by these devices and their software (trustlets and other software components), including within the framework of certification test procedures (certificate of state registration of a computer program No. 2021610311 Russian Federation).

In the absence of documentation for trustlets and system software, it is necessary to use methods for obtaining source code from trustlets, as well as syntactic analysis of binary data of trustlets for the presence of functional or information software objects that pose a threat, potential or immediate, to the protected information being processed. These circumstances determine the relevance of research devoted to improving the methods and tools of software research for identifying vulnerabilities and undeclared opportunities.

This paper describes a model for assessing the security of information processed by trustlets, which takes into account some binary code constructs that can be identified as potentially dangerous functional and information objects and thereby assess the threat level of information processed by trustlets.

Numerous works of both domestic and foreign studies are devoted to the problem of software security analysis. The most famous of them are Avetisyan A. I., Belivantsev A. A., Kurmangaleev Sh. F., Padaryan V. A., Gamayunov D. Yu. [1], Skovorody A. A. [1] and Gaivoronskaya S. A. [2], Zegzda P. D., Boyko V. P., Zaborovsky V. S., Podlovchenko R. I., Ivannikov V. P., Bagaev A. N. [3], Markov A. S. [4], Zakalkin P. V., Matskevich A. G. and Goryunov M. N. [5] etc. Among foreign researchers, the most famous works are Ruan X., Costan V. [6], Pinto S. [7], Cerdeira D. etc.

Attacks on Trustzone components based on device lacking memory protection are described here [8]. Example of Breaking TrustZone memory isolation and secure boot through malicious hardware on a modern FPGA-SoC described in work [9]. Samsung's TrustZone Keymaster Design is described in the article [10]. Developing memory-safe ARM TrustZone applications is described here [11]. A Secure Cache for Arm TrustZone describes in the article [12].

In these works, various aspects of the problems of software research were touched upon, but not enough attention was paid to the issues of obtaining numerical estimates of the degree of security of information processed by trustlets in ARM systems.

2. Task description

2.1 The object of the study

The object of this work is a software solution for a trusted execution environment (TEE) – trusted applications created on the basis of TrustZone technology in computer systems based on processors with ARM architecture.

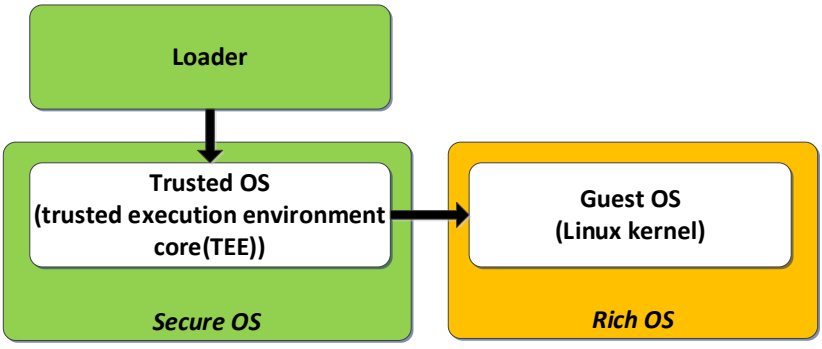

Fig. 1. Loading order in platforms based on ARM processors

TrustZone technology is a hardware-based secure boot environment that allows you to create a TEE. TEE based on TrustZone is called in the terminology of ARM-Secure World or Secure OS, "untrusted" - Rich Execution Environment or Rich OS (iOS, Android, Sailfish, Tizen, Linux, Windows, etc.). The order of loading a computer with an ARM processor is shown in fig. 1, and the interaction of the TEE (Secure OS) and guest operating system (OS) (Rich OS) modes is shown in fig. 2.

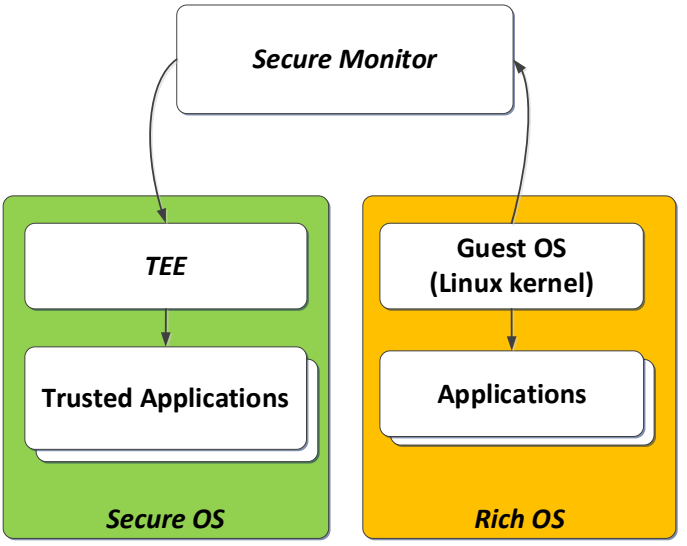

Fig. 2. Interaction of trusted and guest operating systems in platforms based on ARM processors

The essence of the TrustZone technology is to manipulate the operating modes of the processor using a signal set by the NS (Non-Secure) bit. If NS=1, then the processor is in Non-Secure mode (guest OS, or Rich OS, or Normal World OS), if NS=0, then in Trusted, that is, Secure mode (Trusted OS or Trusted OS, or Trusted Execution Environment (TEE)).

The NS bit does not just tell the processor core in which mode it should work. It is also an external signal connected from the processor to almost all the peripherals. In general, the peripheral is connected to the CPU by address, data and control buses. NS is part of the control signals for those processors where TrustZone is implemented. Thus, not just Read, Write commands go from the CPU to the device, but Secure Read, NonSecure Read, Secure Write, NonSecure Write.

ARM TrustZone, built into mobile devices, has been available to users for many years. Over the past few years, the attention to TrustZone has increased significantly. Projects were developed to apply it in various fields of activity: in mobile, industrial, automotive and aerospace. Attention to the technology is supported by the publications of large manufacturers of technical parts and recommendations for development.

The open nature of the TrustZone technology makes it possible to conduct research in the field of improving the security of data processing, in contrast to the proprietary solutions discussed above. Availability of the certificate of the "Aladdin TSM" module [13,14], based on the use of TrustZone technology, according to the requirements of the FSTEC "Requirements for trusted download tools" and the protection profile "IT.SD3.UB2.PZ", confirms this.

2.2 Comparative analysis of known implementations of "trusted" OS

Currently, there are about ten [15] so-called "trusted" operating systems that use the TrustZone functionality. A number of implementations of the TrustZone technology are closed from researchers (Trustonic TCP, Qualcomm SEE). A comparative analysis of the known implementations of "trusted" OS [18] is shown in Table 1.

Table 1. Comparative analysis of "trusted" operating systems

Name	**Developer**	**Features**
GlobalPlatform	Non-profit association of companies G&D Mobile Security,	Industrial standard TEE

	ARM, FIME, Trustonic, Gemalto, Oracle	
General Dynamics OKL4	Open Kernel Labs	Based on the L4 (L3) OS, a microkernel for i386 systems. Developed for iOS devices, since 2012 the source texts are closed
Google Trusty TEE	Google (only fo Android)	It is compatible with ARM and Intel processors. It consists of three main elements: 1) the OS kernel (based on Little Kernel); 2) the kernel driver for managing the interaction of TEE (Trusty) and REE (Android); 3) the user space library for managing the interaction of REE (Android) and TEE (Trusty)
Linaro OP-TEE	Research group Linaro, STMicroelectronics	Based on GlobalPlatform 1.1. An open source project. It consists of three main elements: 1) the OS kernel (memory management modules, interrupts, etc.); 2) the client of the untrusted user space-the monitor-intermediary between the user and kernel spaces, the libraries of the GlobalPlatform TEE Client API implementation; 3) the kernel driver for performing transactions between the trusted and untrusted OS
Jailhouse	Siemens	In a general sense, it is not an OS, but a monitor of resource accesses. It can be run as part of the OS FreeRTOS, Erika3, Linux, Zephyr. Supports processors with the following architectures: ARMv8, ARMv7, x86_64. It requires 2 processors and 50 MB of RAM
QSEE	Qualcomm Secure Execution Environment	Based on General Dynamics OKL4. Closed source text
seL4	Open Kernels labs	It is based on the L4 OS. It has passed a formal verification of correctness by determining the functionality specification and proving its correctness by means of strict logical inference. Open source text. Real-time OS for the firmware of Qualcomm wireless modem processors. The code volume is about 9600 lines. Interrupts during code execution are disabled. Supports ARM processors up to ARMv8
TrustTonic Kinibi	TrustTonic	Closed source text. For the Android OS. For Samsung devices (smartphones and tablet computers). Provides data encryption and device authentication. The trustletshave access to the network. There is a developer kit (SDK) that is compatible with the GlobalPlatform API standards
Xen	University of Cambridge	The microkernel hypervisor. Supports ARM and Intel processors
Xvisor	Developer Community Xvisor	Type 1 hypervisor. Open source text. It has a memory management module, a scheduler, a load balancer and a thread balancer
Aladdin TSM [13, 14]	Aladdin R. D.	Supports i.MX6 processors. It is certified according to the requirements of the FSTEC for the means of trusted loading of the level of the basic I / O system of the second class of protection "IT. SDZ.UB2. PZ" (certificate No. 4155)

The analysis of the features of the TEE functioning in modern ARM processors allows us to conclude that the adaptation of TrustZone technology to the needs of the domestic economy through the use of approaches to increase trust in it, including certification and development of domestic software based on TrustZone, is an important direction for improving domestic information security technologies in the absence of production of a sufficient number of elements. Especially considering the number of modern devices based on processors with ARM architecture.

2.3 Comparative analysis of file formats for trusted application execution environments based on TrustZone technology

The analysis of publications on the TEE research based on the TrustZone technology allowed us to identify the peculiarities of binary files (trustlets) of software executed in the TEE data. According to the list of well-known "trusted" OS for computer systems based on ARM TrustZone, the following binary file formats are used:

for TrustTonic Kinibi OS, the MobiCore Load Format (MCLF) is used, for trustlet files-the extension *.tlbin*;

for Linaro OP-TEE and Aladdin TSM OS-HSTO form-mate, trastlet files have an extension *.ta*.

Trustlets are located in the normal file system of the device and are files containing executable code. They have a similar format to ELF (Executable Linux Format), but with a smaller amount of header and additional structures.

Unlike. ta trustlets,. tlbin does not have the usual ELF file header. To determine the address of the beginning of the executable code, it is necessary to add its length (0x80 + TSHL) to the offset of the text segment header. The formats of the HSTO and MCLF trustlets are shown in fig. 3, 4.

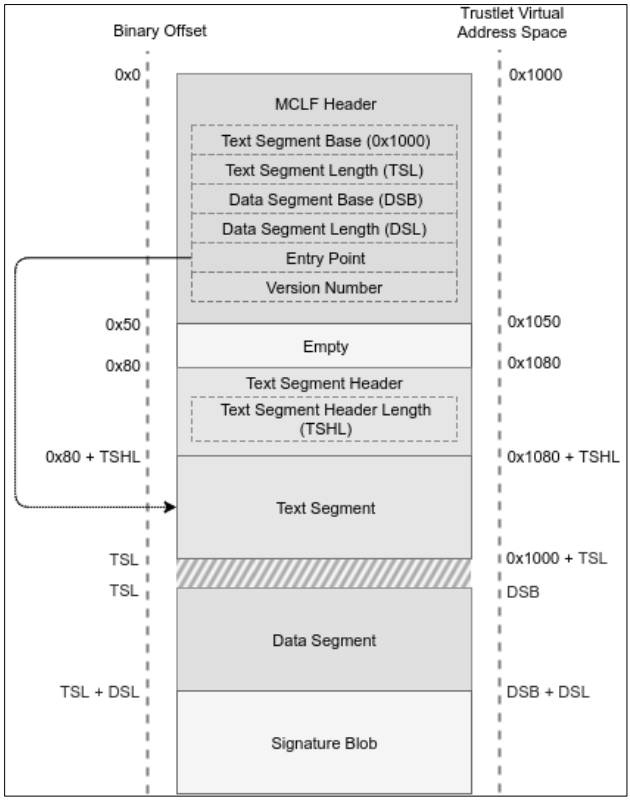

Fig. 3. File structure .tlbin

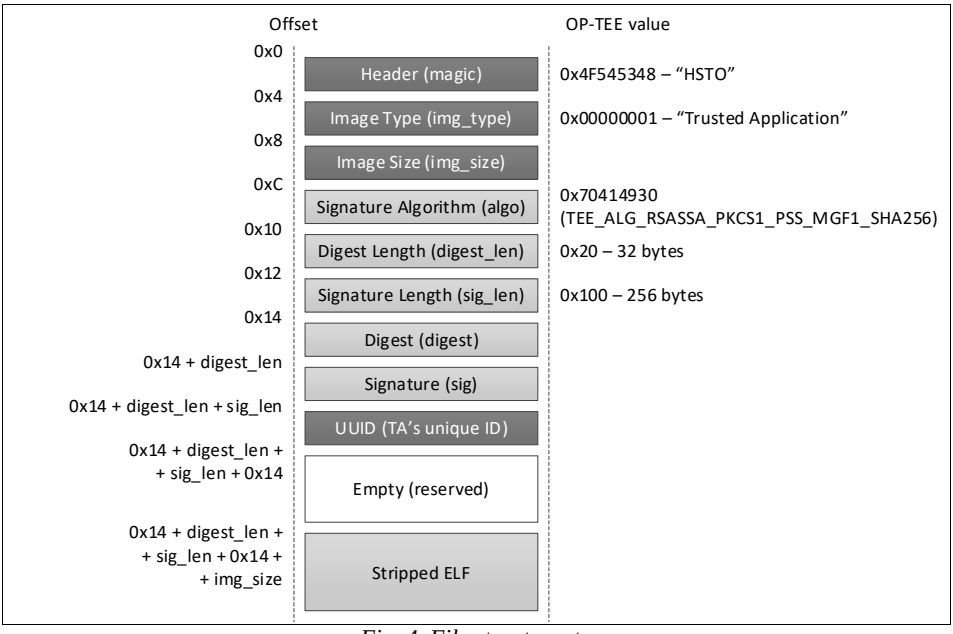

Fig. 4. File structure .ta

The digital signature is located after the "Data Segment" field, the address of its beginning can be determined by adding the lengths of "Text Segment"and" Data Segment". The SHA-256 algorithm is used as a digital signature. The first 4 bytes determine the length of the module, followed by the module itself. After the module, 4 bytes are allocated to describe the length of the public key (0x1), followed by the public key itself, the final 256 bytes are occupied by the signature.

The main differences in the file structure .ta and .tlbin:

- format .ta allows you to specify which crypto algorithm will be used to sign the trustlet,
- .tlbin always uses the SHA-256 algorithm as a signature .ta uses an ELF file without service information (stripped elf) as an executable code,
- .tlbin does not use the ELF file format.

At the same time, when extracting stripped elf from .ta its use is impossible without modification.

From the analysis of the formats, it can be seen that they differ from the ELF format, so it is impossible to run the trustlet in the Unix system emulator without additional preparation of the trustlet or the environment [16]. The task of preparing a trustlet to run in an emulator (for example, QEMU, Virtual Box, etc.) can be solved using static instrumentation methods [3]. To do this, you need to convert the trustlet file to the ELF format.

An example of the names of traslites in the Kinibi Trustonic OS is presented in Table 2. Examples of the names of trustletsin the OS based on Aladdin TSM and OP-TEE are presented in Table 2.

Table 2. Examples of MCLF format TAs` names in Kinibi TrustZone

vendor.app.mcRegistry
07010000000000000000000000000000.tlbin
07060000000000000000000000000000.tlbin
ffffffff0000000000000000000000f.tlbin
system.app.mcRegistry
000603080605010200000000000000000.tlbin
ffffffff000000000000000000000000c.tlbin

ffffffff000000000000000000000000d.tlbin
ffffffffd0000000000000000000000004.tlbin

Table 3. Examples of HSTO format TAs` names in Alladdin TSM TrustZone (JaCarta Box)

Aladdin TSM (JacartaBox)
5b9e0e40-2636-11e1-ad9e0002a5d5c51b.ta
5ce0c432-0ab0-40e5-a056782ca0e6aba2.ta
48002366-708f-4b23-ab0921ce46766a11.ta
e13010e0-2ae1-11e5-896a0002a5d5c51b.ta
OP-TEE
5b9e0e40-2636-11e1-ad9e-0002a5d5c51b.ta
5ce0c432-0ab0-40e5-a056-782ca0e6aba2.ta
5dbac793-f574-4871-8ad3-04331ec17f24.ta
ffd2bded-ab7d-4988-95ee-e4962fff7154.ta

In the file systems of computer systems based on ARM processors, the trustlets files are usually available to unprivileged users.

2.4 Task description

The task description is as follows.

Source data:

1) the set of trusted applications (truslets) extracted from the TEEs based on the TrustZone technology:"Aladdin TSM", "OP-TEE", "TrustTonic Kinibi";

Necessary:

1) to develop a model for assessing the level of security of information processed by the TEE software based on TrustZone technology, which differs from the known ones by taking into account the logical features of information input-output flows and API calls.

Limitations and assumptions:

1) the source code of the trustlets is not available;
2) the studied trustletshave one of the following formats: "HSTO" (Linaro OPTI OS) and "MCLF" (Trustonic Kinibi OS);
3) the format of trustlets (ELF-like) does not allow them to be executed in a traditional environment (untrusted UNIX-like operating system – Android, Debian, Ubuntu, etc.).

3. Investigation of vulnerabilities of the trusted application execution environment based on TrustZone technology

Trusted Execution Environments is one of the modern security mechanisms for protecting the integrity and confidentiality of applications. This technology is de facto a hardware technology for implementing TEE in mobile environments, as well as industrial control systems, servers, and budget household devices. The most common implementations of TEE are the developments of Qualcomm, Trustonic, Huawei, Nvidia and Linaro [17].

The features of vulnerabilities detected in the TrustZone-based TEE [18,19] are usually associated with:

- classic input data validation errors, such as "buffer overflow";
- numerous architectural shortcomings of TEE systems, such as the lack of ASLR technology and other system protection tools for applications;
- the lack of consideration of hardware properties at the architectural and microarchitectural level when implementing system TEE, for example, associated with the appearance of side channels

for transmitting information in cache memory or interaction with the memory of reconfigurable equipment capable of accessing confidential data.

The E1 exploit exploits an error in the LCB kernel and is able to execute arbitrary code with the EL1 privilege level, and, as a result, is able to develop the attack further, up to extracting secret keys and decrypting the disk and unlocking the device bootloader [20].

The E6 exploit allows an attacker to take control of the Linux kernel by sending a special set of data from a user-level application to trusted Widevine applications [21].

The TEE system requires drivers in the software to access protected software, technical components, I/O devices that process critical information. The complexity of implementing drivers that are traditional sources of errors and their functioning with an extended privilege level lead to the appearance of critical vulnerabilities.

Interfaces between the TEE components allow you to exchange sufficiently large amounts of data with privileged access rights to a sufficiently large number of trusted applications – trustlets. For example, in some implementations of the "trusted" OS Trustonic TEE, their number reaches 32, and in Widevine-70.

Another architectural problem is the excessively large amounts of data that are transferred between trusted and untrusted environments. For example, in Qualcomm's TEE, the buffer size for such data reaches 1.6 MB, and this volume can indirectly increase.

In some implementations of TEE, trustlets can be displayed in the physical memory of the unprotected mode. This applies primarily to Qualcomm's TEE. After scanning the physical address space of the Linux kernel and fixing it, it is possible to introduce a backdoor into the system (exploit E6 [21]). Unlike Qualcomm's TEE, in the implementation of Trustonic TEE, the display of the trustlet in the physical memory of the unprotected mode is excluded.

Some security problems are associated with the appearance of side channels through application debugging tools. Such channels are found in Huawei's TEE implementations [20]. In particular, using the TEE system call, the trustlet (trusted application) uploads its own stack trace to public memory, which allows an attacker to study the address space of the trustlet and use it to create an exploit in the future. The same vulnerability was found in the Trustonic TEE.

It should be noted that in all known implementations of TrustZone there is no ASLR technology (with the exception of its limited implementation in Qualcomm TEE), i.e. all trusted applications are always loaded at a fixed address of the virtual address space (0x1000). In addition, a shared library (mcLib) is provided, which is also downloaded at a permanent address for each trustlet (0x7D01000). Thus, any vulnerability found in the trustlet can be used without much effort when determining the address of loading the trustlet into memory. In addition, the shared library used by trustlets contains a significant amount of code that serves as a source of gadgets (data) for identifying system calls and calls to trusted applications.

Another system vulnerability is the lack of a separate stack for cookie data, as well as the lack of protective (barrier) pages between the data of protected processes, which leads to the appearance of a technical possibility for implementing a "buffer overflow"attack.

The TEE has mechanisms for protecting against code execution in memory areas by means of the WXN bit in the SCTLR register, as well as the XN memory page attribute. However, well-known TEE implement these protection mechanisms partially. For example, there is no stack for cookie data in the Trustonic TEE. Cookie data is stored as global variables from the data segment of the trustlet without buffer protection pages. In addition, the layout of data in memory is implemented in such a way that the stack is located at the end of the data segment, and global variables are in front of it, which creates ideal conditions for exploiting a buffer overflow vulnerability.

Unlike Trustonic, Qualcomm's TEE creates a stack with a random location in the address space, but there is also no mechanism for monitoring the integrity of protective buffer pages.

Huawei's TEE lacks both data execution protection and protective buffer pages.

A common disadvantage of TEE is also the lack of hardware control of the integrity of the TEE code, which weakens the overall level of trust in "trusted" software.

In addition to the considered architectural vulnerabilities, there are numerous typical software errors in the TEE implementations associated with:

- incorrect processing of input or output data;
- name verification errors;
- errors that lead to the possibility of exploiting a vulnerability such as "buffer overflow";
- incorrect processing of parameters.

Such errors can be used as starting points for extending privileges and are found in almost all existing components of the TEE of various manufacturers.

Thus, the analysis of a number of detected vulnerabilities in the implementation of TEE based on ARM TrustZone shows that even the implementation of the technology that increases the overall level of security of the system has numerous vulnerabilities and errors that can be used by attackers for unauthorized access and performing illegitimate operations with data.

4. Security threat level estimation for untrusted software based on TrustZone technology

The analysis of open sources of information [2,3,5] allowed us to identify signs that the presence of which in the software implementation may indicate the potential danger of this software from the point of view of the possible implementation of threats to the security of the processed information.

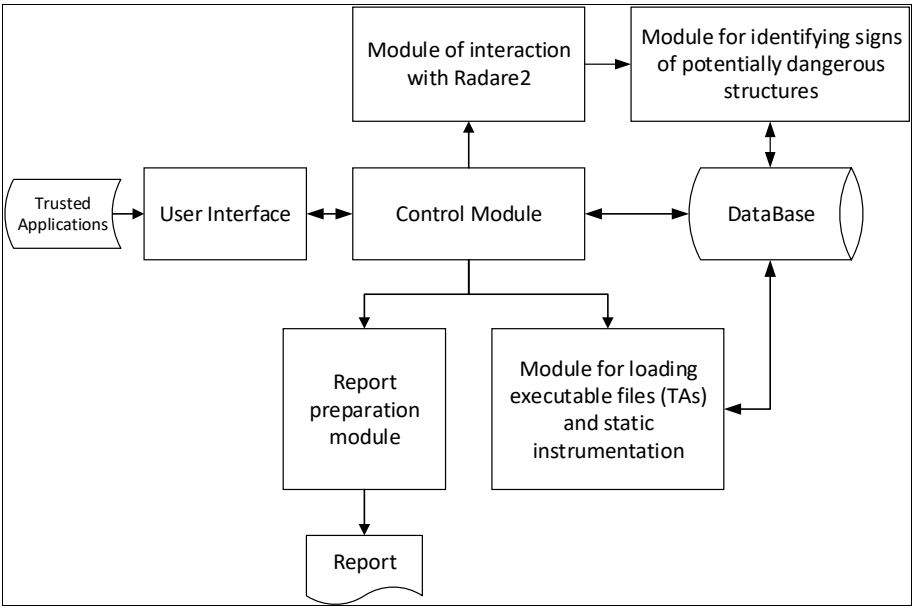

Fig. 5. Block diagram of the program for identifying signs of potentially dangerous structures

These include [18]:

- the presence and number of data entry and exit points (N_{in} and N_{out}, respectively), the amount of input/output data (V_{in} and V_{out}, respectively);
- the variety of APIs $F_{API\_TEE}^{unsafe}$, including specific API functions of TEE that handle critical data and information (account and authentication information, key information for the cryptographic subsystem that identifies(address) of the network, and information about the technical characteristics of the computer, etc.);

- the presence of calls to function objects (functions) of the standard system libraries F_{system}^{unsafe}, including the processing critical information and data;
- the presence of specified sequences of calls to functional objects, including calls to certain API functions and calls to functional objects of standard system libraries IF^{unsafe};
- the volume V_{WSM} and frequency of processing data N_{WSM} coming from the "untrusted" OS via the WSM (World Shared Memory) buffer $\langle V_{in}^{WSM}, N_{in}^{WSM} \rangle$ and transmitted from the TEE to the "untrusted" OS – $\langle V_{out}^{WSM}, N_{out}^{WSM} \rangle$.

Thus, the model for assessing the level of security of information processed by the TEEe based on TrustZone technology can be represented as a tuple:

$$Z = \langle N_{in}, N_{out}, V_{in}, V_{out}, F_{API\_TEE}^{unsafe}, F_{system}^{unsafe}, IF^{unsafe}, \langle V_{in}^{WSM}, N_{in}^{WSM} \rangle, \langle V_{out}^{WSM}, N_{out}^{WSM} \rangle \rangle.$$

In order to identify the listed features, a program was developed (certificates of state registration of a computer program No. 2020666135, 2021610311 Russian Federation), the block diagram of which is shown in the fig. 5.

In order to study the machine code instructions in the binary files of trustlets, a tool for analyzing binary files of trustlets was developed and implemented in Python.

Table 4 shows a list of some functions found using a Python software tool.

Table 4. Names of system functions in the ffd2bded-ab7d-4988-95ee-e4962fff7154.ta file

utee_storage_free_enum, utee_storage_reset_enum, utee_storage_start_enum, utee_storage_next_enum, utee_storage_obj_read, utee_storage_obj_write, utee_storage_obj_trunc, utee_storage_obj_seek	Working with the file system
utee_hash_final, utee_cipher_init, utee_cipher_update, utee_cipher_final, utee_cryp_obj_get_info, utee_cryp_obj_restrict_usage, utee_cryp_obj_get_attr	Cryptographic functions
utee_open_ta_session, utee_invoke_ta_command, utee_close_ta_session	Working with sessions (data exchange between the trustler and the application via the OS)

After studying the last group of system calls, we can conclude that all calls are preceded by a call to the_open_ta_session function. This function returns the session ID, which is used in subsequent calls. And after working with the system functions, the session is closed by ut ee_close_t a_session. That is, the exchange of translit information with other applications will occur between the call of ut ee_open_t a_session and ut ee_close_ta_session.

5. Conclusions

As a result, based on this information, it is possible to identify the places of system calls in the machine code and determine the type of functions for further classification according to the degree of vulnerability or the information being processed.

As part of the work, the analysis of executable file formats for TEE based on TrustZone technology – trustlets for Kinibi Trustonic OS (Samsung smartphones), OP-TEE (Linaro) and JacartaBox (Aladdin TSM) was carried out. Their distinctive features and approaches, such as static instrumentation, are identified, which allow using dynamic analysis methods in relation to trustlets in the future.

References / Список литературы

[1] Zakharkin P.V., Melnikov P.V. The system of software analysis for the absence of undeclared capabilities. Software Engineering, vol. 9, no. 2, 2018, pp. 69-75 (in Russian) / Закалкин П.В., Мельников П.В. Система анализа программного обеспечения на предмет отсутствия недекларированных возможностей. Программная инженерия, том 9, no. 2, 2018 г., стр. 69-75.

[2] Skovoroda A.A., Gamayunov D.Yu. Dynamic analysis of mobile applications. Software Engineering. 2019, No. 7-8, pp. 324-333 (in Russian) / Сковорода А.А., Гамаюнов Д.Ю. Динамический анализ мобильных приложений. Программная инженерия, том 10, no. 7-8, 2019 г., стр. 324-333.

[3] Gaivoronskaya S.A. Hybrid method for detecting shellcodes. Systems of high availability. 2012, vol. 2, No. 8, pp. 33-44 (in Russian) / Гайворонская С.А. Гибридный метод обнаружения шеллкодов. Системы высокой доступности, vol. 8, no. 2, 2012 г., pp. 33-44

[4] Begaev A.N., Kashin S.V. et al. Identification of vulnerabilities and undeclared opportunities in software. St. Petersburg, ITMO University, 2020, 38 p. (in Russian) / Бегаев А.Н., Кашин С.В. и др. Выявление уязвимостей и недекларированных возможностей в программном обеспечении. Учебно-методическое пособие. Санкт-Петербург, Университет ИТМО, 2020 г., 38 стр.

[5] Markov A. S., Cirlov V.L., Barabanov A.V. Methods for assessing the inconsistency of information security tools. Moscow, Radio and Communications, 2012, 192 p. (in Russian) / Марков А.С., Цирлов В.Л., Барабанов А.В. Методы оценки несоответствия средств защиты информации. Москва, «Радио и связь», Москва, «Радио и связь», 2012 г., 192 стр.

[6] Goryunov M.N., Eremenko V.T. et al. Recognition of functional objects of software in the absence of source texts. Information systems and technologies, no. 5, 2013, pp. 112-120 (in Russian) / Горюнов М.Н., Ерёменко В.Т. и др. Распознавание функциональных объектов программного обеспечения в условиях отсутствия исходных текстов. Информационные системы и технологии, no. 5, 2013 г., стр. 112-120.

[7] Costan V., Devadas S. Intel SGX Explained. URL: https://eprint.iacr.org/2016/086.pdf, 2016.

[8] Pinto S., Santos N. Demystifying Arm TrustZone: A Comprehensive Survey. ACM Computing Surveys, vol. 51, issue 6, 2019, article no. 130, 36 p.

[9] Stajnrod R., Yehuda R.B, Zaidenberg N.J. Attacking TrustZone on devices lacking memory protection. Journal of Computer Virology and Hacking Techniques, 2021, 11 p.

[10] Gross M., Jacob N. et al. Breaking TrustZone memory isolation and secure boot through malicious hardware on a modern FPGA-SoC. Journal of Cryptographic Engineering, 2021, 16 p.

[11] Shakevsky A., Ronen E. Avishai Wool Trust Dies in Darkness: Shedding Light on Samsung's TrustZone Keymaster Design. URL: https://eprint.iacr.org/2022/208.pdf, 2022.

[12] Wan S., Sun M. et al. RusTEE: developing memory-safe ARM TrustZone applications. In Proc. of the Annual Computer Security Applications Conference (ACSAC 2020), 2020, pp. 442–453.

[13] Benedito O., Delgado-Gonzalo R., Schiavoni V. KeVlar-Tz: A Secure Cache for Arm TrustZone. Lecture Notes in Computer Science, vol 12718, 2021, pp. 109-124.

[14] Trusted platform for ARM processors. URL: https://www.aladdin-rd.ru/catalog/tsm, accessed: 10.12.2021 (in Russian) / Доверенная платформа для процессоров ARM.

[15] Certificate of Conformity FSTEC of Russia No. 4155. URL: https://www.aladdin-rd.ru/upload/certified/sertifikat_fstek_4155_tsm.pdf, accessed: 10.12.2021 (in Russian) / Сертификат соответствия ФСТЭК России No 4155.

[16] Yehuda R.B., Leon R., Zaidenberg N.J. ARM Security Alternatives. In Proc. of the 18th European Conference on Cyber Warfare and Security, 2019, pp. 604–612.

[17] Markin D.O., Makeev S.M. et al. Methodology of research of system software of network equipment of the Cisco family for the presence of undeclared capabilities. Scientific Notes of the Orel State University, no. 3(88), 2020, pp. 215-221 (in Russian) / Маркин Д.О., Макеев С.М. и др. Методика исследования системного программного обеспечения сетевого оборудования семейства cisco на предмет наличия недекларируемых возможностей. Ученые записки Орловского государственного университета, no. 3(88), 2020 г., стр. 215-221.

[18] Cerdeira, D., Santos N. et al. SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems. In Proc. of the IEEE Symposium on Security and Privacy, 2020, pp. 1416-1432.

[19] Markin D.O., Ho T.T., Meshkov N.P. Features of the search for software vulnerabilities based on TrustZone technology. Problems of information security. Computer systems, no. 4, 2020, pp. 79-87 (in Russian) / Маркин Д.О., Хо Т.Ч., Мешков Н.П. Особенности поиска уязвимостей программного обеспечения на основе технологии TrustZone. Проблемы информационной безопасности. Компьютерные системы, no. 4, 2020 г., стр. 79-87.

[20] Markin D. O., Ho T. T. Research of vulnerabilities of the trusted application execution environment based on TrustZone technology. Izvestiya Tula State University. Technical sciences, issue 9, 2020, pp. 316-328

(in Russian) / Маркин Д.О., Хо Т.Ч. Исследование уязвимостей доверенной среды исполнения приложении на основе технологии TrustZone. Известия тульского государственного университета технические науки, вып. 9, 2020 г., pp. 316-328.

[21] Extracting Qualcomm's KeyMaster Keys – Breaking Android Full Disk Encryption (Jun 2016). URL: https://bits-please.blogspot.com/2016/06/extracting-qualcomms-keymaster-keys.html, accessed: 10.12.2021.

[22] Di Shen. Attacking your "Trusted Core". Exploiting TrustZone on Android. URL: https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android.pdf, accessed: 10.12.2021.

Information about authors / Информация об авторах

Dmitry Olegovich MARKIN – candidate of engineering sciences, employer of the Academy of Federal Guard Service. His research interests include: information security, machine learning methods, pattern recognition, cryptographic methods for protecting information.

Дмитрий Олегович МАРКИН – кандидат технических наук, сотрудник Академии ФСО России. В круг его научных интересов входят: информационная безопасность, методы машинного обучения, распознавание образов, криптографические методы защиты информации.

Sergey Mikhailovich MAKEEV – candidate of engineering sciences, employer of the Academy of Federal Guard Service. His research interests include: decision support systems, machine learning methods, pattern recognition, information security.

Сергей Михайлович МАКЕЕВ – кандидат технических наук, сотрудник Академии ФСО России. Его исследовательские интересы включают: системы поддержки принятия решений, методы машинного обучения, распознавание образов, информационную безопасность.

Trung Thaj HO is an employer of the Academy of Federal Guard Service. His research interests include: information security, machine learning methods, cryptographic methods for protecting information.

Чунг Тхай ХО — сотрудник Академии ФСО России. В круг его научных интересов входят: информационная безопасность, методы машинного обучения, криптографические методы защиты информации.

DOI: 10.15514/ISPRAS-2022-34(1)-4

Исследование применимости аппаратной компрессии данных в межпроцессорных каналах связи процессоров с архитектурой Эльбрус

А.В. Сурченко, ORCID: 0000-0001-5200-7982 <Alexander.V.Surchenko@mcst.ru>
АО «МЦСТ»,
117105, Россия, г. Москва, ул. Нагатинская, д.1, стр.23
Московский физико-технический институт,
141701, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. В современных процессорных системах наблюдается увеличение нагрузки на подсистему памяти, вызванное преимущественно тенденцией к увеличению числа процессорных ядер. В частности, одним из наиболее критических мест с точки зрения пропускной способности становятся межпроцессорные каналы связи, темп передачи информации в которых заметно ниже, чем на шинах внутри процессора. В качестве одного из способов повышения пропускной способности межпроцессорных каналов связи можно рассматривать аппаратную компрессию данных, призванную уменьшить объем информации, передаваемой по межпроцессорным каналам. В данной работе производится оценка актуальности применения аппаратной компрессии данных в межпроцессорных каналах связи процессоров с архитектурой Эльбрус. В качестве рассматриваемого алгоритма компрессии выбирается алгоритм BΔI*-HL, демонстрирующий достаточно высокую эффективность при малых задержках и затратах на реализацию. Исследования производятся на FPGA-прототипе процессора «Эльбрус-16С» для задач пакета SPEC CPU2000. Результаты исследования показывают, что за счет аппаратной компрессии данных удается сжать 38,0% с данными, а в целом объем передаваемой по межпроцессорным каналам связи информации за счет компрессии данных снижается на 13,4%. Полученные результаты позволяют сделать вывод об актуальности применения аппаратной компрессии данных в межпроцессорных каналах памяти процессоров с архитектурой Эльбрус с целью увеличения производительности подсистемы памяти.

Ключевые слова: архитектура Эльбрус; аппаратная компрессия данных; межпроцессорные каналы связи; производительность подсистемы памяти

Для цитирования: Сурченко А.В. Исследование применимости аппаратной компрессии данных в межпроцессорных каналах связи процессоров с архитектурой Эльбрус. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 49-58. DOI: 10.15514/ISPRAS–2022–34(1)–4

Evaluation of Hardware Data Compression in Interprocessor Links of Elbrus Processors

A.V. Surchenko, ORCID:0000-0001-5200-7982 <Alexander.V.Surchenko@mcst.ru>
MCST, 1, Nagatinskaya st., Moscow, 117105, Russia
Moscow Institute of Physics and Technology (National Research University),
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia

Abstract. The tendency to increase core count in modern processor systems leads to a higher strain on memory subsystem. In particular, one of the most critical points in terms of throughput is interprocessor links, where

bandwidth is significantly less than in processor data buses. Hardware data compression can be considered as one of the ways to increase throughput in interprocessor links, as it allows to decrease the amount of information transmitted over the links. This paper presents the evaluation of hardware data compression in interprocessor links of Elbrus processors. BΔI*-HL compression algorithm is chosen for the evaluation. The results are obtained of FPGA prototype of "Elbrus-16C" processor for the tasks of SPEC CPU2000 benchmark suite. They show that by using hardware data compression 38,0% of all data packets were compressed and that the amount of information transmitted overall has decreased by 13,4%. These results demonstrate that the use of hardware data compression in interprocessor links of Elbrus processors is justified and has potential to significantly increase memory subsystem performance.

Keywords: Elbrus architecture; hardware data compression; interprocessor links; memory subsystem performance

For citation: Surchenko A.V. Evaluation of Hardware Data Compression in Interprocessor Links of Elbrus Processors. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 49-58 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–4

1. Введение

Увеличение производительности в современных процессорных системах достигается преимущественно за счет увеличения числа процессорных ядер. В связи с этим производится большее число обращений в память, т.к. увеличивается число устройств, осуществляющих подобные запросы. Как следствие, увеличивается нагрузка на подсистему памяти, и ее пропускной способности может оказаться недостаточно, чтобы своевременно и без задержек обрабатывать запросы в память, поступающие от ядер.

В частности, одним из наиболее критических мест с точки зрения пропускной способности являются каналы связи, находящиеся вне чипа (off-chip) [1][2]. Это объясняется малой шириной подобных каналов, зависящей от числа контактов (пинов) процессорного чипа.

При рассмотрении многопроцессорных систем аналогичное ограничение по пропускной способности накладывается и на межпроцессорные каналы связи. Темп передачи пакетов с данными в этих каналах, как правило, в несколько раз ниже, чем внутри процессора. Несмотря на то, что передача по межпроцессорным каналам связи осуществляется на большей частоте, чем частота процессора, разницы в частотах все еще недостаточно, чтобы темп передачи данных по межпроцессорным каналам связи соответствовал темпу передачи внутри процессора. Кроме того, помимо пакетов с данными по межпроцессорным каналам связи передаются также и другие пакеты системного протокола (пакеты с первоначальными запросами, пакеты-подтверждения и др.), а также служебная информация, необходимая для корректной работы каналов связи на физическом уровне передачи. Все эти факторы дополнительно снижают темп передачи данных по межпроцессорным каналам связи.

В качестве одного из способов повышения пропускной способности каналов связи, находящихся вне чипа, можно рассматривать аппаратную компрессию данных [2][4][5][6]. За счет ее использования можно добиться уменьшения объема информации, передаваемой через межпроцессорные каналы связи, тем самым увеличивая пропускную способность подсистемы памяти.

В рамках данной работы производилось исследование применимости аппаратной компрессии данных в межпроцессорных каналах связи процессоров с архитектурой Эльбрус. В последующих разделах статьи приводятся детали исследования. В разд. 2 приводится информация о выборе алгоритма компрессии, наиболее подходящего для использования в межпроцессорных каналах связи процессоров с архитектурой Эльбрус, а также производится выбор передаваемой через каналы информации, которая будет подвергаться сжатию. В разд. 3 описывается, какие именно статистические данные были собраны для оценки применимости аппаратной компрессии данных в межпроцессорных каналах связи, а также приведен тестовый стенд, разработанный для сбора этих статистических данных. В разд.4

представлены результаты анализа применимости компрессии данных в межпроцессорных каналах связи. Разд. 5 завершает работу.

2. Выбор алгоритма компрессии и объекта компрессии

2.1 Критерии, предъявляемые к алгоритму

В ходе исследования был выдвинут ряд критериев, определяющих выбор алгоритма компрессии, подходящего для использования в подсистеме памяти. Ниже приведены сами критерии и пояснения к каждому из них:

1) процедуры компрессии и декомпрессии при использовании выбранного алгоритма должны производиться быстро;
2) Реализация алгоритма в аппаратуре не должна приводить к существенным структурным изменениям;
3) алгоритм должен быть эффективным (т.е. должен сжимать достаточно заметный процент данных).

Высокая скорость компрессии и декомпрессии необходима, поскольку в противном случае, несмотря на, возможно, бóльшую эффективность, алгоритм будет вызывать большие задержки при передаче, что окажет негативное воздействие на пропускную способность.

Требование к отсутствию существенных структурных изменений вводится с целью предотвратить потенциальное уменьшение производительности устройств, которое может произойти из-за изменений в их структуре.

Наконец, требование к эффективности алгоритма позволит гарантировать, что прирост производительности от реализации алгоритма будет превышать затраты на его реализацию.

Согласно проведенным ранее исследованиям, одним из наиболее оптимально подходящих под эти критерии алгоритмов является алгоритм компрессии BΔI*-HL [6]. Следующий подраздел описывает принципы работы этого алгоритма, а также приводит сведения о его производительности.

2.2 Алгоритм BΔI*-HL

Алгоритм BΔI*-HL является модификацией предложенного в [7] алгоритма BΔI (Base-Delta-Immediate). Основной принцип работы алгоритма BΔI основывается на наблюдении о том, что данные, передаваемые в подсистеме памяти (являющиеся, как правило, строками данных кэш-памяти последнего уровня), обладают определенными зависимостями.

Если разбить отдельную кэш-строку на несколько сегментов одинакового размера, то данные в пределах этих сегментов зачастую являются близкими по значению либо друг к другу, либо к нулю. Именно эта близость в значениях позволяет уменьшить объем хранимой информации, поскольку в таком случае можно выбрать какое-то базовое значение и хранить его, а для остальных сегментов хранить только смещения («дельты») относительно этого значения.

Рис. 1 и рис. 2 иллюстрируют процесс компрессии и декомпрессии данных с использованием алгоритма BΔI*-HL. Для упрощения вычислений в качестве базового значения («базы») выбираются старшие разряды сегментов. Смещениями («дельтами») являются младшие разряды сегментов, которые могут различаться между собой. Дополнительно сжатая кэш-строка содержит т.н. «маску», которая позволяет определить, являются ли старшие разряды конкретного сегмента нулевыми или равными базе. Компрессия производится параллельно с разными размерами сегментов и смещений, каждая пара из размера сегмента и размера смещения при этом называется схемой компрессии. Компрессия считается успешной, если хотя бы для какой-то схемы компрессии старшие разряды всех сегментов либо совпадают, либо равны нулю.

Декомпрессия по алгоритму BΔI*-HL в то же время сводится к простой конкатенации смещения с базой или нулями в зависимости от соответствующего бита маски.

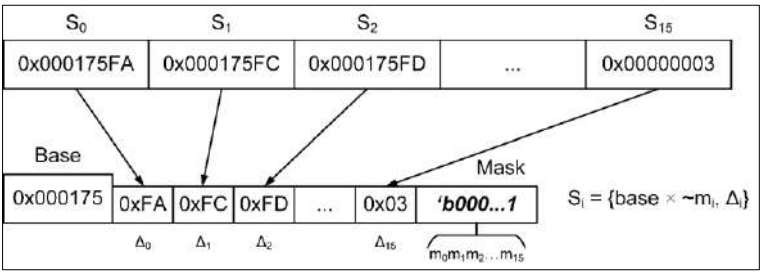

Рис. 1. Компрессия по алгоритму BΔI-HL*
Fig. 1. Compression using BΔI-HL algorithm*

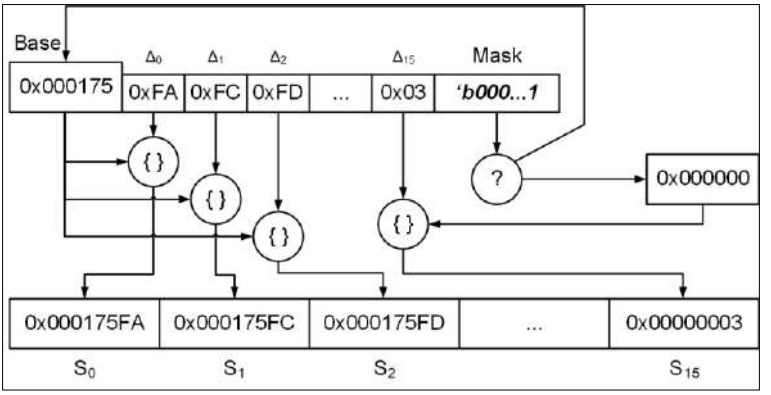

Рис. 2. Декомпрессия по алгоритму BΔI-HL*
Fig. 2. Decompression using BΔI-HL algorithm*

Важно отметить, что схемы компрессии для алгоритма BΔI*-HL были подобраны таким образом, чтобы размер сжатой кэш-строки был равен половине от размера полной. Это связано с тем, что в пакетах с данными могут передаваться как полные кэш-строки, так и половины кэш-строк. Таким образом удается практически без каких-либо накладных расходов переиспользовать пакеты, предназначенные для отправки половины кэш-строк, для упаковки туда кэш-строк, сжатых по алгоритму. Единственным дополнением к такому пакету будет являться информация о сжатии, необходимая для декомпрессии данных на принимающей стороне.

В табл. 1 демонстрируются характеристики по эффективности и накладным расходам на реализацию алгоритма BΔI*-HL, полученные в ходе предыдущих исследований [8]. Результаты по задержкам критического пути, и числу логических элементов, затрачиваемых на реализацию алгоритма, были получены в ходе пробного синтеза на технологии 16 нм с целевой частотой 2 ГГц. Результаты по доле сжимаемых строк и теоретической степени сжатия по алгоритму собирались на данных, поступающих по записи в кэш-память последнего уровня процессора Эльбрус.

Табл. 1. Характеристики алгоритма BΔI-HL*
Table 1. BΔI-HL algorithm characteristics*

	Compression	**Decompression**
Critical path delay (2 GHz)	300 ps (1 cycle)	70 ps (1 cycle)
Number of logical elements	6700	2400
Lines compressed (from lines written to LLC), %	24.2%	
Theoretical compression ratio	1.246	

Как можно видеть, на строках, поступающих по записи в кэш-память последнего уровня алгоритм показывает достаточно высокую долю сжатых строк, и при этом компрессия и декомпрессия при его использовании занимают всего 1 такт при частоте 2 ГГц.

Таким образом, алгоритм удовлетворяет всем критериям, поставленным в предыдущем разделе.

2.3 Выбор объекта компрессии

Как уже было отмечено ранее, алгоритм BΔI*-HL ориентирован на сжатие кэш-строк. Таким образом, проецируя этот факт на типы пакетов, передаваемых через межпроцессорные каналы связи, можно прийти к заключению о том, что рассмотренный алгоритм наиболее подходит для сжатия пакетов с данными.

Прочие пакеты содержат только информацию, необходимую для корректной работы системного протокола. В этой информации уже не будут проявляться зависимости между сегментами данных, а потому применение там рассмотренного алгоритма компрессии не является целесообразным.

Это приводит к решению о том, что среди всей информации, передаваемой по межпроцессорным линкам, компрессия будет рассматриваться только для пакетов с данными. Для остальных типов пакетов попытка сжатия производиться не будет.

3. Тестовый стенд

Для произведения окончательной оценки того, насколько применимой является аппаратная компрессия данных в межпроцессорных каналах связи, необходимо определить, существенно ли уменьшается объем передаваемой по каналам связи информации при использовании компрессии.

При ответе на этот вопрос недостаточно полагаться только на ранее полученные характеристики по доле сжатых строк для алгоритма BΔI*-HL, поскольку, во-первых, они были получены на более высоком уровне иерархии в подсистеме памяти (не все кэш-строки, поступающие в кэш-память последнего уровня, дойдут так же и до межпроцессорных каналов связи), а во-вторых, необходимо также учесть и информацию, передаваемую с другими типами пакетов.

В связи с этим было решено собрать следующую статистическую информацию. Во-первых, можно оценить соотношение объема информации, относящейся к передаваемым пакетам с данными, ко всей передаваемой информации. Таким образом можно определить объем информации, который в принципе можно попытаться подвергнуть сжатию. Во-вторых, можно также оценить, какая доля информации в пакетах с данными является сжимаемой. Наконец, из совместного анализа обоих результатов можно сделать вывод о том, как сильно уменьшится объем передаваемой информации за счет компрессии.

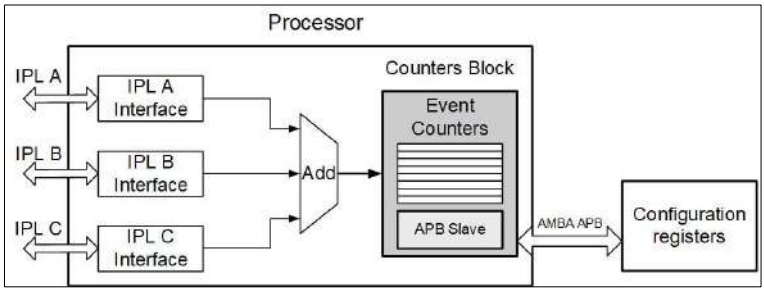

Рис. 3. Тестовый стенд
Fig. 3. Testing bench

Для сбора статистической информации был разработан тестовый стенд, представленный на рис. 3. В ходе работы процессора информация о типах пакетов, отправляемых по межпроцессорным каналам связи, а также информация о компрессии пакетов с данными, записывается на добавленные в систему счетчики событий, суммируясь по всем каналам (межпроцессорные каналы связи на рисунке обозначены как IPL, Interprocessor Links).

Значения счетчиков каждого из процессоров при этом являются программно доступными на конфигурационных регистрах процессора через шину AMBA APB, позволяющую как считывать, так и записывать значения этих счетчиков.

Сбор статистики производился на FPGA-прототипе процессора «Эльбрус-16С», собранном в двухпроцессорной конфигурации. Для тестирования запускались задачи пакета SPEC CPU2000 в однопоточном режиме, причем для создания достаточно высокой нагрузки на межпроцессорные каналы связи задачи запускались с помощью утилиты numactl таким образом, что сама задача запускалась на одном процессоре, а память для нее выделялась на другом.

Полученные статистические результаты, усредненные по обоим процессорам, представлены в следующем разделе.

4. Результаты

На рис. 4 показан график, отражающий долю информации, относящейся к пакетам с данными, среди всей передаваемой по межпроцессорным каналам связи информации. Целочисленные задачи пакета SPEC CPU2000 показаны красным цветом, а задачи с плавающей точкой – синим.

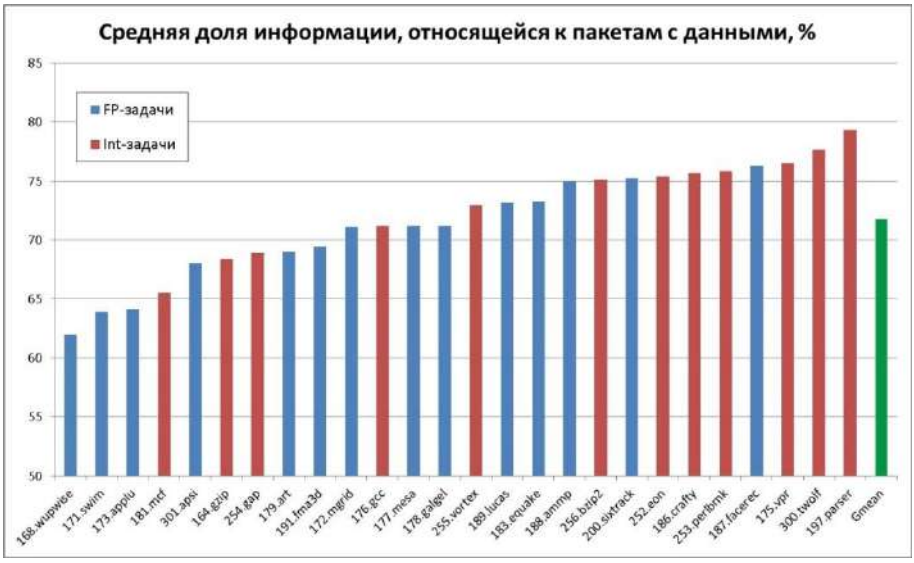

Рис. 4. Средняя доля информации, передаваемой по межпроцессорным линкам, относящаяся к пакетам с данными

Fig. 4. Average percentage of information sent through interprocessor links that relates to data packets

Как можно видеть, для большинства задач пакета SPEC CPU2000 доля информации, относящейся к пакетам с данными, является достаточно высокой, что частично объясняется заметно большим размером пакетов с данными по отношению к другим пакетам. В крайнем правом столбце представлено усредненное значение по всем задачам пакета SPEC CPU2000, составляющее примерно 71,7%.

На рис. 5 показан график, демонстрирующий долю сжатых по алгоритму BΔI*-HL строк, передаваемых в пакетах с данными по межпроцессорным каналам связи. Аналогично

предыдущему графику, целочисленные задачи пакета SPEC CPU2000 показаны красным цветом, а задачи с плавающей точкой – синим.

Рис. 5. Средняя доля сжатых по алгоритму BΔI-HL пакетов с данными среди передаваемых по межпроцессорным каналам связи*

Fig. 5. Average percentage of data packets sent through interprocessor links that can be compressed

Можно видеть, что компрессия в целом работает лучше на целочисленных задачах, что объясняется отличиями в двоичном представлении между целыми числами и числами с плавающей точкой.

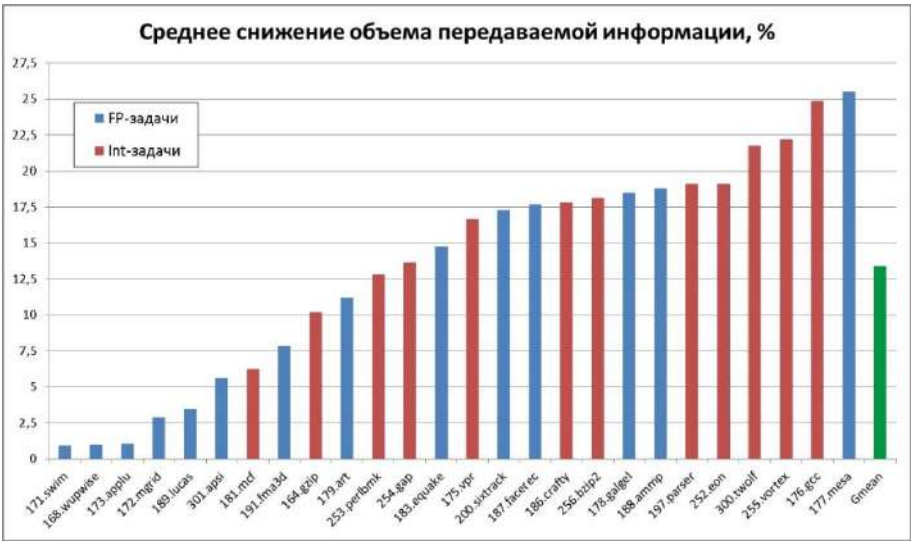

Рис. 6. Средний процент, на который удается сократить объем передаваемой по межпроцессорным каналам связи информации за счет использования аппаратной компрессии данных

Fig. 6. Average percentage by which the amount of information sent over interprocessor links can be reduced by using hardware data compression

Также стоит отметить, что средняя доля сжимаемых данных получилась больше, чем полученная ранее для кэш-памяти последнего уровня, составляя 38,1%. Это можно объяснить, во-первых, различиями в наборах кэш-строк, поступающих в кэш-память последнего уровня и на межпроцессорные каналы связи, а во-вторых, тем, что для

межпроцессорных каналов связи использовалась усовершенствованная версия алгоритма BΔI*-HL, где почти без накладных расходов было добавлено две новые схемы компрессии [8].

Рис. 6 представляет собой график, показывающий уменьшение передаваемой по межпроцессорным каналам связи информации при использовании компрессии. Аналогично предыдущим графикам, целочисленные задачи пакета SPEC CPU2000 показаны красным цветом, а задачи с плавающей точкой – синим.

Так же, как и для графика с долей сжатых пакетов с данными, объем передаваемой по каналам информации наиболее заметно уменьшается для целочисленных задач, поскольку для таких задач больше и доля сжимаемых пакетов с данными. В среднем же по всем задачам пакета SPEC CPU2000 удается уменьшить объем передаваемой информации на 13,4%, что является весьма существенным результатом.

5. Заключение

В ходе данной работы был рассмотрен вопрос актуальности использования аппаратной компрессии данных в межпроцессорных каналах связи процессоров с архитектурой Эльбрус.

Были представлены критерии для алгоритма компрессии, используемого в подсистеме памяти. Согласно этим критериям, для проведения исследования применимости аппаратной компрессии данных был выбран алгоритм компрессии BΔI*-HL.

Был разработан тестовый стенд, где для произведения измерений были добавлены счетчики событий с возможностью программного доступа по чтению и записи.

Согласно результатам измерений, произведенных на FPGA-прототипе процессора «Эльбрус-16С» для задач пакета SPEC CPU2000, аппаратная компрессия данных по алгоритму BΔI*-HL позволяет сжать до половины размера 38,0% пакетов с данными, снижая тем самым объем всей передаваемой по межпроцессорным каналам связи неслужебной информации на 13,4%.

Полученные результаты позволяют считать аппаратную компрессию данных применимой по отношению к межпроцессорным каналам связи. Дальнейшие исследования будут посвящены полноценной интеграции механизма аппаратной компрессии данных в межпроцессорные каналы связи процессоров с архитектурой Эльбрус.

Список литературы / References

[1]. Rogers B.M., Krishna A. et al. Scaling the bandwidth wall: challenges in and avenues for CMP scaling. In Proc. of the 36th Annual International Symposium on Computer Architecture, 2009, pp. 371-382.

[2]. Thuresson M., Spracklen L., & Stenstrom P. Memory-link compression schemes: A value locality perspective. IEEE Transactions on Computers, vol. 57, issue 7, 2008, pp. 916-927.

[3]. Sardashti S., Arelakis A. et al. A primer on compression in the memory hierarchy. Synthesis Lectures on Computer Architecture, vol. 10, issue 5, 2015, pp. 1-86.

[4]. Alameldeen A.R. & Wood D.A. Interactions between compression and prefetching in chip multiprocessors. In Proc. of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture, 2007, pp. 228-239.

[5]. Sathish V., Schulte M.J., & Kim N.S. Lossless and lossy memory I/O link compression for improving performance of GPGPU workloads. In Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), 2012, pp. 325-334.

[6]. Кожин А.С., Сурченко А.В. Исследование применимости компрессии данных в кэш-памяти микропроцессоров с архитектурой «Эльбрус». Вопросы радиоэлектроники, no 2, 2018 г., стр. 32–39 / Kozhin A.S., Surchenko A.V. Evaluation of cache compression for Elbrus processors. Voprosy radioelektroniki, no. 2, 2018, pp. 32–39 (in Russian).

[7]. Pekhimenko G., Seshadri V. et al. (2012, September). Base-delta-immediate compression: Practical data compression for on-chip caches. In Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), 2012, pp. 377-388.

[8]. Kozhin A.S., Surchenko A.V. Design of Data Compression Mechanism in Cache Memory of Elbrus Processors. In 2020 International Conference Engineering and Telecommunication (En&T), 2020, pp. 1-5.

Информация об авторе / Information about the author

Александр Викторович СУРЧЕНКО проходит обучение в аспирантуре в МФТИ, а также работает в качестве старшего инженера в АО «МЦСТ». Область научных интересов: производительность и когерентность подсистемы памяти процессоров и процессорных систем, аппаратная компрессия данных, кэш-память.

Alexander Viktorovich SURCHENKO is a PhD student in MIPT. He is currently also working as a senior engineer at JSC «MCST». Research interests: memory subsystem performance and coherence in processors and processor systems, hardware data compression, cache memory.

DOI: 10.15514/ISPRAS-2022-34(1)-5

Parallelism reduction method in the high-level VLSI synthesis implementation

[1,2] *D.S. Romanova, ORCID: 0000-0002-9020-4802<daryaooo@mail.ru>*
[1] *O.V. Nepomnyashchiy, ORCID: 0000-0002-2459-6414 <2955005@gmail.com>*
[1] *I.N. Ryzhenko, ORCID: 0000-0002-3069-9102 <rodgi.krs@gmail.com>*
[3] *A.I. Legalov, ORCID: 0000-0002-5487-0699 <legalov@mail.ru>*
[1] *N.Y. Sirotinina, ORCID: 0000-0001-8816-4226 <nsirotinina@sfu-kras.ru>*

[1] *Siberian Federal University,*
79, Svobodny pr.79, 660041, Krasnoyarsk, Russia
[2] *Krasnoyarsk State Agrarian University*
Mira pr. 90, 660049, Krasnoyarsk, Russia
[3] *National Research University - Higher School of Economics*
Myasnitskaya str.20, 101000, Moscow, Russia

Abstract. In the article the problems and solutions in the field of ensuring architectural independence and implementation of digital integrated circuits end-to-end design processes are considered. The method and language of parallel programming for functional flow synthesis of design solutions is presented. During the method implementation, the tasks of reducing parallelism and estimating the occupied resources were highlighted. The main feature of the developed method is the introduction of the additional meta-layer into the synthesis process. Algorithms for the parallelism reduction have been developed. The results of software tools development for design support and practical VLSI projects are presented.

Keywords: integrated circuit; algorithm; program; parallel computing model; high-level synthesis; functional-stream language

For citation: Romanova D.S., Nepomnyashchiy O.V., Ryzhenko I.N., Legalov A.I., Sirotinina N.Y. Parallelism reduction method in the high-level VLSI synthesis implementation. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 59-72. DOI: 10.15514/ISPRAS–2022–34(1)–5

Acknowledgements. The article is based on the materials of the report at the Seventh International Conference «Actual Problems of System and Software Engineering» (APSSE 2021).

Метод редукции параллелизма в процессе высокоуровнего синтеза цифровых интегральных схем

[1,2] *Д.С. Романова, ORCID: 0000-0002-9020-4802<daryaooo@mail.ru>*
[1] *О.В. Непомнящий, ORCID: 0000-0002-2459-6414 <2955005@gmail.com>*
[1] *И.Н. Рыженко, ORCID: 0000-0002-3069-9102 <rodgi.krs@gmail.com>*
[3] *А.И. Легалов, ORCID: 0000-0002-5487-0699 <legalov@mail.ru>*
[1] *Н.Ю. Сиротинина, ORCID: 0000-0001-8816-4226 <nsirotinina@sfu-kras.ru>*

[1] *Сибирский федеральный университет,*
660041, Россия, г. Красноярск, пр. Свободный, 79
[2] *Красноярский государственный аграрный университет*
660049, Россия, г. Красноярск, пр. Мира, 90
[3] *Национальный исследовательский университет «Высшая школа экономики»*
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Рассмотрены проблемы и решения в области обеспечения архитектурной независимости и организации процесса сквозного проектирования цифровых интегральных схем. Представлен метод и язык параллельного программирования для функционально потокового синтеза проектных решений. При реализации метода выделены задачи редукции параллелизма и оценки занимаемых ресурсов. Предложен способ свертки, базирующийся на введении дополнительного, мета-слоя в процесс синтеза. Разработан принцип и алгоритмы редукции параллелизма. Представлены результаты разработки программного инструментария поддержки проектирования и реализованные на практике проекты СБИС.

Ключевые слова: интегральная схема; алгоритм; программа; модель параллельных вычислений; высокоуровневый синтез; функционально-потоковый язык

Для цитирования: Романова Д.С., Непомнящий О.В., Рыженко И.Н., Легалов А.И., Сиротинина Н.Ю. Метод редукции параллелизма в процессе высокоуровнего синтеза цифровых интегральных схем. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 59-72. DOI: 10.15514/ISPRAS–2022–34(1)–5

Благодарности. Статья подготовлена по материалам доклада на Седьмой Международной конференции «Актуальные проблемы системной и программной инженерии» (АПСПИ 2021).

1. Introduction

The current level of digital integrated circuits (IC) development is characterized by constantly increasing requirements for the systemic organization of the entire design flow. One of the most important tasks is to reduce the time to get the final result. The main reasons for slowing down the design flow are iterative operations that lead to returning to previous stages. The elimination of iterative operations provides an “end-to-end design flow” resulting in lower financial costs and increased product competitiveness. Another urgent task is to ensure the architectural independence of the product being developed, in other words, design solutions portability between the target platforms of the IC. Portability allows the developer to optout of being tied to the target implementation platform that provides more efficient solutions for key product specifications such as speed, chip area, etc.

An integrated circuit is, in fact, a system for parallel processing of information flows. At the final stages of the synthesis, the architecture and operation algorithm of the IC are presented in hardware description languages. Therefore, efficient solutions for ensuring architectural independence can be found in the area of portable parallel programs. In turn, end-to-end design can be ensured through the use of a parallel computing model of the functional flow and the presentation of the original algorithms in the form of acyclic structures [5, 10, 12, 13].

2. Related work

A key feature of well-known research in this area is the use of a functional-flow parallel computing model and language for the initial description of algorithms for the operation of IC.

Currently, there is a steady tendency to increase the abstraction of the initial algorithms from the final implementation of the project. At the same time, methods for describing the IC architecture develop in several ways. The most common is the introduction of constructs for high-level description into existing hardware description languages (HDL) [1]. Such solutions led to the emergence of the SystemVerilog language based on the classic Verilog [1]. But even in this case, the level of abstraction from the specific architecture of the target chip is not fully provided.

There are a number of solutions based on the use of adapted imperative high-level programming languages (primarily C and C ++) as hardware description languages, for instance, SystemC [2], Handel-C [3], and Impulse-C [4]. However, these languages were created to solve narrow problems, for example, to implement streaming applications or to support alternative programming models, so they do not provide architectural independence of the designed solutions. They are predominantly sequential languages; therefore, they do not support parallelism, which is necessary for describing parallel processes occurring in the IC.

Of particular interest is the COLAMO programming language [5], which is a high-level language with an implicit description of parallelism. Parallelization is achieved by declaring variable access types and indexing array elements, which is typical for data flow languages. Currently, this language is used for programming reconfigurable computing systems. It allows developing parallel application programs with high specific performance [5]. However, this language is focused on solving applied problems in the field of high performance computing for multichip systems.

The most effective solutions are obtained using functional languages that have a more powerful abstraction mechanism and a developed type system. The initial IC operating algorithms described in similar languages are easier to transform and verify [6]. For example, the languages Hydra [7] and uFP [8] use flows to describe signals and recursive expressions to provide schema transformations. Lava, like Hydra, is a built-in subset of Haskell. It possesses powerful circuit design tools derived from its predecessor. At the same time, Lava is simpler and more convenient to use due to the extended type system for describing hardware. However, the mechanism of "lazy" calculating inherent in Haskell and the sequential structure of lists impose restrictions on parallelism transformations and do not allow efficient automatic parallelization of programs [12].

The works of Donnagara [10, 11] show that the effective portability of the IC initial description can be implemented by using functional programming languages and presenting algorithms as data flow graphs. The high efficiency of this approach is proved in practice using the example of parallel software (PaRSEC) for high performance computing [11].

In this case, the solution lies in the application of a programming paradigm, which must meet the following conditions [10]:

- Lack of explicit control over computations (control by data availability / readiness);
- Data flow model;
- Parallelism at the level of operations.

The model that meets the listed requirements is the basis of the functional-flow parallel (FFP) programming languages [12]. This gives grounds to consider the FFP programming methods and the corresponding computation model as the most suitable for solving the problem of high-level architecture-independent IC synthesis.

Among the programming languages that support the FFP model, the Pifagor language is chosen [12]. This language allows a developer to describe an initial algorithm without resource constrains, supports a data flow model and parallelism at the operation level, which is the most important for the architecture-independent end-to-end design of IC. In Pifagor language, architectural independence is achieved by describing only informational connections existed in the program.

Unlike Haskell, Pifagor only supports explicitly describing delayed computations. This allows executing alternative program fragments only when they are needed. Also, there are no loop operators in Pifagor. This prevents conflicts when using different data with the same fragments of a parallel program. In theory, this allows a program to start executing any function as soon as its input is ready. In practice, only resource limits are imposed on the maximum concurrency specified in the language.

The authors have proposed a method for an architecture-independent high-level VLSI synthesis on the basis of the FFP computation model and the Pifagor language that supports it [13]. When implementing the method, the dynamic type system was replaced by a static one, which is supported in hardware description languages, delayed computations were excluded, and it was proposed a method to transform them in compliance with the target IC platform. Also, in the modified Pifagor language, a mechanism for converting recursion into an iterative scheme with a subsequent transition to a pipeline scheme was introduced [13].

The proposed method for ensuring architectural independence in the course of high-level synthesis assumes that a program in Pifagor is transformed into an intermediate representation in the form of a pair of graphs: a data–flow graph (DFG) and a control-flow graph (CFG). DFG specifies information connections, and presenting CFG in an explicit form allows a more detailed description of the computational control process.

The DFG is shaped during the program translation, and the CFG can be elaborated both dynamically and statically. The last approach is used to switch from a program in Pifagor to a program for describing IC in an HDL language.

The intermediate presentation of the FFP of the program during translation is developed in two stages. At the first stage, the initial code is translated into DFG. Then, a control-flow graph is formed from the obtained data-flow graph. CFG can be formed from DFG in various ways. For example, in accordance with the model of computation control by data flow, either it can be reduced to a sequential traversal of the DFG, or provide another strategy for managing computations.

To switch to the target IC platform, the DFG is converted into a pipeline scheme. The pipeline computing scheme is a tiered-parallel form of DFG.

To convert parallelism of the initial algorithm to the target IC platform taking in account the specific resource constraints, the following stages are performed: determining the boundaries of changing parallelism, an algorithm for changing parallelism, and evaluating the result based on resource constraints.

The main task of the transition from the unrestricted parallelism of the FFP model to the target platform is reduction of parallelism. This is the key moment of all ongoing transformations in the architecture-independent method of VLSI synthesis. To implement the reduction mechanism, a new meta-layer called "HDL graph" was introduced. It is an intermediate layer for making changes to the FFP model. According to the authors, the term "HDL graph" most fully corresponds to the method of VLSI synthesis. HDL graph allows you to specify connections between elements of lists of connected vertices, which in turn allow performing optimization transformations by calculating the operation of selecting data from the list. With the help of the introduced meta-layer, when processing types in a statically typed model, a restriction is introduced on the dynamic resizing of lists during computations.

Recursive computations often become an obstacle when porting such programs to some real computing platforms, since at a significant depth of recursion memory overflow can occur. The introduction of the HDL graph made it possible to solve this problem by transforming such computations into iterative ones using tail recursion and specifying the recursion depth at the translation stage.

*3. Parallel conversion*3.1 Parallelism reduction algorithm

The main feature of the introduced synthesis method is the transition from parallelism induction into the algorithm description to reducing the initial maximum-parallel algorithm description according to specific resource constraints of the target platform. As shown in [10, 11], this ensures the portability of parallel algorithms to various platforms. The main advantage of parallelism reduction compared to induction is significant decreasing in the number of steps required to obtain the final result. When reducing a maximally parallel data flow graph, the number of maximally admissible transformations is set at the synthesis stage.

In the process of synthesis, the following tasks are solved:

- Assessment of the resources of the resulting architectural solution;
- Evaluation of the performance of the resulting solution;
- Calculation of the reduction factor for each class of resources;
- Parallelism reduction of the circuit to achieve the required coefficients.

To estimate resources, an intermediate representation of the program HDL graph is used, in which architecture-dependent data are already specified. An HDL graph is an acyclic graph in a tiered-parallel form, at each node of which types and widths of data are specified [14]. The N_k resource classes whereby the scheme should be evaluated are determined depending on the target platform.

For example, the main resource classes of the FPGA platform include:

1) Number of registers N_r;
2) Number of logical cells N_{lc};
3) The amount of block memory N_m Nm;
4) Number of arithmetic and other specialized computing units N_{dsp}.

Resources can be divided into two subsets:

- Memory resources $N_{mem} = \{N_r, N_m\}$;
- Computing resources $N_{comp} = \{N_{lc}, N_{dsp}\}$;

Within the subset, there are restrictions for fungible resources. For memory resources, any data storage can be implemented on block memory, while its implementation on triggers has volume restrictions. For computational resources, it is possible to implement any computation on logical cells, while the type and set of operations for specialized blocks is limited. Resource estimation results are used to further parallelism reduction.

3.2 Memory resources estimation

To estimate the required memory amount, the total amount of resources can be referred to the total amount measured in bits (kbit).

As an example for estimating the circuit resource, consider its HDL graph. Each k-th layer of the tiered-parallel form consists of a set of information inputs B_k and a set of operations O_k. After data typing, each information input of HDL graph vertices has a width of W_k. Based on the input number and the bit width of each input of a specific graph layer, it is possible to determine the amount of memory required to store the result in the corresponding graph state:

To calculate the resource, it is necessary to traverse the graph and sum up the bit widths of the inputs and outputs of all vertices:

$$NR = \sum NR_k.$$

After an initial evaluation of the required memory resource for the initial maximum parallel implementation of the circuit, two options are possible:

The required resource is less than the available one $NR < N_{m/lc}$, where $N_{m/lc}$ is an available resource;

The required resource is greater than the available one $NR \geq N_{m/lc}$

The first option does not require the calculation of the reduction factor for memory resources G_m. But when the scheme is changed during its reduction, for other resources, the memory resource must be evaluated and checked again.

In the second case, the reduction factor G_m is calculated using the following formula:

$$G_m = \frac{NR}{N_{m/lc}}$$

This factor is used in the parallelism reduction algorithm of the scheme.

In addition to the memory limitation, the memory performance limitation must also be considered. If the memory is built on registers/ flip-flops, his limitation does not exist, since the width of the data bus is equal to the amount of data. For block memory, the amount of data read per clock cycle is less than the amount of stored data. If resource NR exceeds the available volume of N_r registers, it is necessary to calculate the total data interface to the memory and the reduction factor by the memory interface G_{md}.

3.3 Algorithm for determining the reduction coefficient

To determine the minimum required reduction factor over the memory interface, a set of stages with the maximum total interface implemented in registers is selected from all stages of the pipeline. To do this, a subset of the stages is selected from the set of pipeline stages, such that:

$$(N_{m/lc} - NR_r) > max \sum NR \;||\; \sum NR_k.$$

The || sign denotes logical addition.

The G_{md} factor is defined as the ratio of the total memory interface of the remaining stages to the total block memory interface IM.

The following algorithm is used to determine the reduction factor over the memory interface:

To select a subset of the pipeline stages such that the sum of resources NR_k of the selected stages will be less than NR_r and selected values sum NR_k will be maximum;

To calculate the memory resource implemented on block/sequential memory:

$$NR_m = NR - \sum NR_k,$$

where k belongs to a subset of the pipeline stages selected at step 1.

Calculate the coefficient G_{md}:

$$G_{md} = 1 + Int(NR_m/IM),$$

where Int is rounding to an integer value.

As an example, the calculation of a 4-point FFT (Fast Fourier Transform) will be considered, where the discrete Fourier transform is calculated by the formula:

$$c_n = \frac{1}{n}\sum_j^{n-1} S_j * W_n^{-kj} + \frac{1}{n}\sum_j^{n-1} S_j * W_n^{kj}.$$

The input data type is signed 16-bit.

The Data-Flow graph after transformation into HDL graph and reduction to a tier-parallel form (TPF) is shown in fig. 1.

The value of recourses NR_k for each stage of the pipeline is calculated as follows:

$$NR_1 = 10 * 2 * 16 = 320;$$
$$NR_2 = 16 * 4 + 8 * 32 = 320;$$
$$NR_3 = 16 * 4 + 4 * 33 = 196;$$
$$NR_4 = 33 * 4 + 4 * 34 = 268.$$

Total resource value is:

$$NR = 1104\ bits.$$

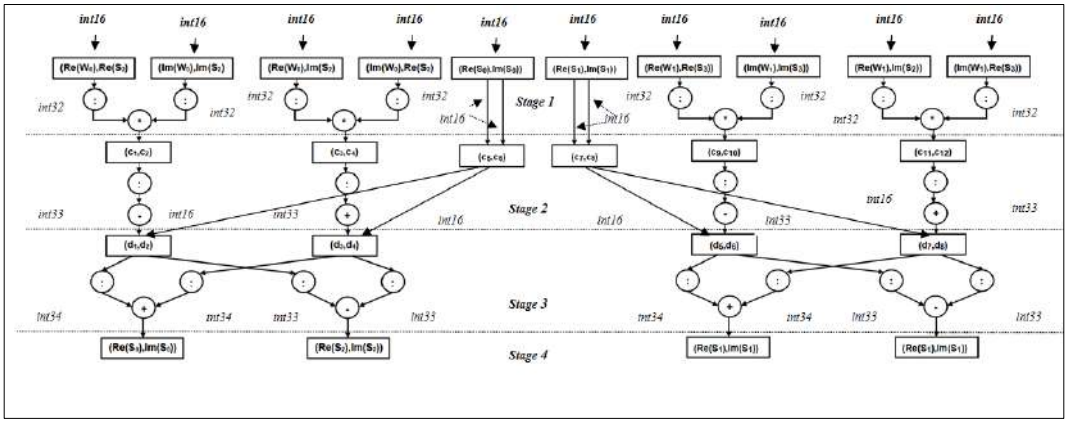

Fig. 1. HDL graph of the maximum parallel form of 4-point FFT

Let's consider two architectures, A1 and A2. Value $N_{m/lc}$ for both architectures is 1536 bits. In the A1 architecture, the entire memory resource is in registers. For such architecture, the 4-point FFT scheme in maximum parallel form is implemented unchanged, as in fig. 2.

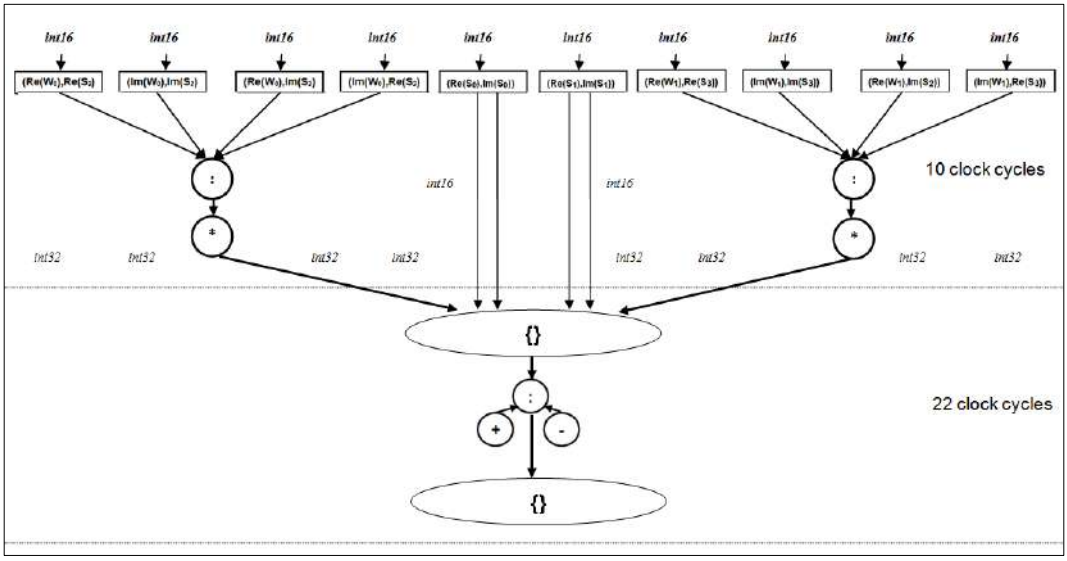

Fig. 2. HDL graph after reduction

In the A2 architecture, the register resource is 512 bits, and 1024 bits are presented in the form of block memory with a data interface of 36 bits. The total block memory interface value *IM* is 36 bits. In accordance with the algorithm for calculating G_{md}, a subset of stages that are implemented in registers and have a maximum interface is selected. In this example, it can be either stage 1 or stage 2.

In this case, value NR_m will be calculated as follows:

$$NR_m = NR - NR_1 = 1104 - 320 = 784 \text{ bits.}$$

The value of the reduction factor by the memory interface will be:

$$G_{md} = \frac{784}{36} = 22_{..}$$

In this case, the data feed period becomes equal to G_{md} and pipeline stages 2, 3, 4 or 1,3,4 are implemented sequentially, since the result is written to one memory block. The scheme obtained as a result of the reduction with the corresponding ratio G_{md} is shown in fig. 2.

The values c_1-c_{12}, d_1-d_8 and S_0-S_3 are calculated in this scheme sequentially. Since stages 2, 3 of the original scheme after reduction require 22 cycles to execute, stage 1 can also be increased to 22 cycles without affecting the overall performance of the system, which will lead to a proportional decrease in the computational resource of stage 1.

3.4 Estimation of the required computing resources

The total number of layers (stages of the pipeline) is M. At each j-th layer of the graph, a certain set of operations is implemented: O_j , where $j = 1, \dots, k$.

For the entire HDL graph, the number of each operation:

$$F_k = \sum_{j=0}^{M} O_k^j.$$

Let L_k be the total number of different types of operations, where $k = 0, \dots, L$. The type of operation here means the type of arithmetic / logical, etc. operations along with the indication of the data width. For example, adding 16-bit data, comparing 20-bit data, etc.

After calculating the total number of operations of each type based on the available a specific architecture resource, it is necessary to assess the degree of parallelism with which it is possible to implement the scheme.

It is supposed that the amount of resource for each specific type of operation is known. Let Y be the type of resource (logical cells, specialized arithmetic blocks DSP, etc.), $V(Y)_k$ will be the amount of resource of type Y required to implement an operation of type k. Then the total resource of type Y for all operations in the HDL graph will be:

$$V(Y) = \sum_{k=0}^{L} VY_k * F_k.$$

For each class of computing resources, the reduction factor G_y can be calculated as the ratio of the total required resource to the resource of the target architecture, rounded to the nearest larger integer:

$$G_y = Int\big(V(Y)/N_y\big).$$

So, the final reduction factor for computing resources is determined as the maximum among all reduction factors:

$$G_{calc} = max\{G_y\}$$

This algorithm is considered using the example of the graph diagram shown in fig. 2. The total number of pipeline stages for a given graph is 4. The number of operation types L will be 5: 16-bit multiplication ($k = 0$), addition and subtraction of 33 and 16 bits ($k = 1.2$), addition of 34-bit data, and subtraction of 33-bit data ($k = 3.4$). The number of operations on the layers of thegraph will be:

$$O_1 = \{8, 0, 0, 0, 0\};$$
$$O_2 = \{0, 2, 2, 0, 0\};$$
$$O_3 = \{0, 0, 0, 4, 4\};$$
$$O_4 = \{0, 0, 0, 0, 0\};$$

The number of each operation of the k-th type:

$F_0 = 8,\ F_1 = 2,\ F_2 = 2,\ F_3 = 4,\ F_4 = 4.$

Let there be architecture with two types of resources for implementation of computations: logical cells ($Y = 0$) and DSP sections ($Y = 1$). The following resource values for each type of operation are taken:

For logic cells:

$$V(0)_0 = 0,\ V(0)_1 = 5,$$
$$V(0)_2 = 5,\ V(0)_3 = 10,\ V(0)_4 = 10.$$

For DSP sections:

$$V(1)_0 = 1, V(1)_1 = 0,$$
$$V(1)_2 = 0, V(1)_3 = 0,\ V(1)_4 = 0.$$

The total resource for each type for all operations in the graph will be:

$V(0) = \sum_{k=0}^{L} VY_k * F_k = 0 * 8 + 5 * 2 + 5 * 2 + 4 * 10 + 4 * 10 = 100,$

$V(1) = \sum_{k=0}^{L} VY_k * F_k = 1 * 8 + 0 * 2 + 0 * 2 + 0 * 10 + 0 * 10 = 8$

The architecture where the number of DSPs is $N_1 = 2$, the number of logical cells is $N_0 = 200$ is considered. According to it, the reduction factor for each type of resource is:

$$G_0 = Int\left(\frac{V(0)}{G_0}\right) = \frac{100}{200} = = 0.5;$$

$$G_1 = Int\left(\frac{V(1)}{G_1}\right) = \frac{8}{2} = 4.$$

The value of the reduction factor for computing resources G_{calc} is:

$$G_{calc} = \max\{0.5, 4\} = 4.$$

After reducing each stage with a factor of 4, the delay of each stage will increase to 4 clock cycles, while the resource will also decrease by 4 times. The resulting circuit is shown in fig. 3.

Note that changing the computational resource may change the memory resource.

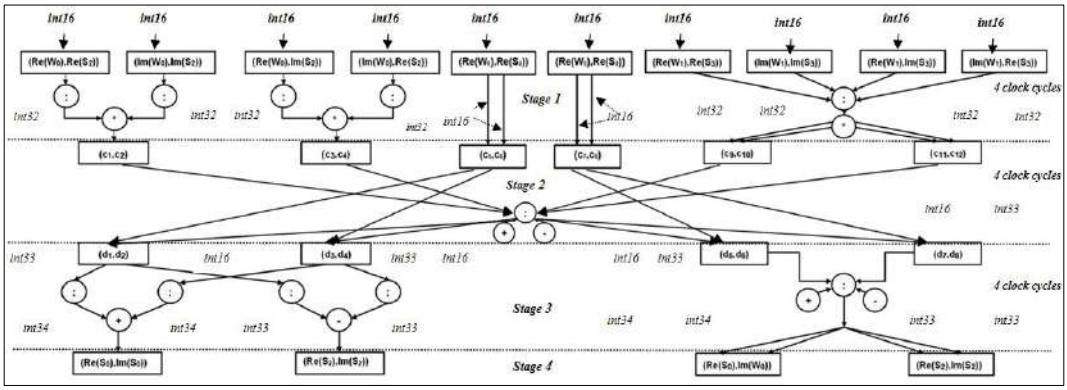

Fig. 3. Scheme of HDL graph after reduction by Gcalc

3.5 Generalized Parallelism Conversion Algorithm

Let the ratio of the resource required to implement the circuit in the original maximally parallel form to the available resource be denoted as R. It is assumed that the available resource is the smallest of the resources. There are three options for transforming the original maximum parallel scheme can be distinguished depending on the available resource of the target platform.

The first option: if $R > 1$, a reduction in parallelism is required.

In this case it is possible to increase performance by placing several circuits in parallel:

$$S = Int(1/R).$$

Consider a reduction algorithm using the reduction coefficients described in subsections 3.2-3.4.

1) Calculating the reduction factors G_m and G_{calc};
2) Choosing the maximum coefficient $G_{max} = \max(G_m, G_{calc})$;
3) Reducing the parallelism of the circuit to G_{max};
4) Recalculating the coefficients G_m and G_{calc} for the modified circuit;
5) If they are less than 1, the algorithm is finalized;

6) If some operations can be implemented using a different type of resources, change these operations to another type of resources without changing the R coefficient and recalculate the G_m and G_{calc} coefficients;
7) If any of the coefficients are greater than 1: $G_{max} = 1 + G_{max}$ and return to step 3.

A sequential increase in the reduction factor allows selecting the minimum possible ratio to meet the resource requirements and at the same time achieve maximum performance (the minimum possible reduction in performance relative to the initial maximum parallel version).

Second option: when $R < 1/2$, an increase in the number of circuits is possible.

The third option: when $1/2 < R < 1$ the resource is enough to accommodate 1 maximum parallel version of the circuit.

In the second and third options, no conversions of the maximally parallel circuit are required.

4. Results

In the framework of the research, the authors have implemented a set of software tools that perform the following functions:

- Transformation of the source code in the FFP language into an intermediate representation in the form of an DFG and CFG;
- Optimization transformations that increase the efficiency of FFP programs;
- Debugging and analysis of FFP code at runtime, including finding errors and tracing;
- Compilation of the intermediate representation of FFP programs into the description of VLSI in HDL languages [15].

Fig. 4 shows the architecture of the developed design support tools based on the proposed high-level synthesis method

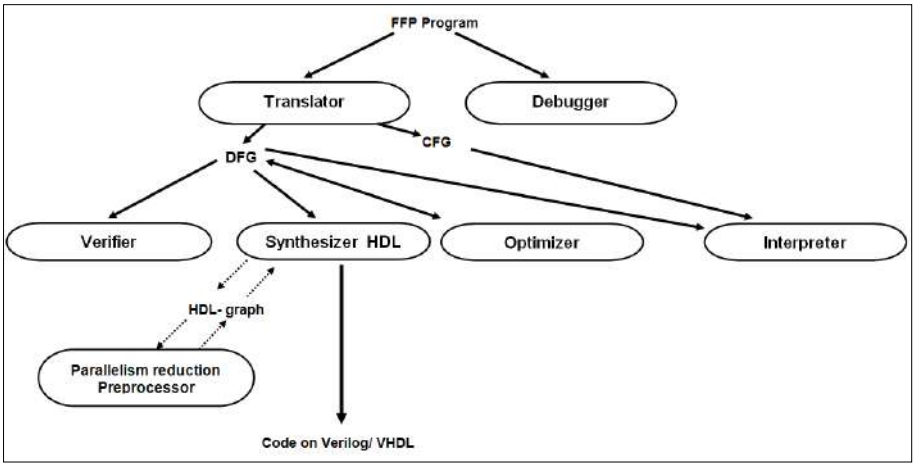

Fig. 4. Architecture of design support tools

The interpreter provides executing the program developed in the FFP language. The input data for the interpreter is the data flow and control graph, as well as the argument of the top-level function. The argument is presented in the format of the DFG description, for this it is processed by the translator.

The optimizer also uses the input intermediate representation and performs optimization of the DFG, the result is saved as an intermediate representation of the DFG.

Optimizing transformations carried out at this stage include:

A. Removing unused code;
B. Optimizing repetitive calculations;
C. Direct function substitution;
D. Removing duplicate code;
E. Optimizing based on equivalent transformations of the FFP algebra of the model.

The translator checks the syntax of programs developed in the Pifagor functional data flow parallel programming language and converts the program into its intermediate representation in the form of DFG and CFG [16]. It transforms a functional-flow description into a description at the level of combinational logical circuits. The translator includes a debugger, a DFG generator, and a CFG generator. The result of the translator's functioning is a set of debugged functions implemented in the Verilog / VHDL languages.

The developed software package also includes a parallelism reduction preprocessor and resource estimation preprocessor. The parallelism reduction preprocessor automatically converts the parallelism of programs intended for translation into an HDL language, taking into account resource restrictions. The preprocessor gets an intermediate representation of the IC operating algorithm in FFP language in the form of a typed data flow graph (HDL graph) and resource constraints of the target platform obtained using the resource constraints preprocessor. The result of the operation of the parallelism reduction preprocessor is the data flow graph, transformed taking into account resource constraints. The resulting representation is used by the circuit synthesizer to obtain a description of the IC in HDL languages.

Translators have been developed for the Verilog and VHDL languages [17]. The program implements checking the initial description for suitability for synthesis, assembling the initial description from a set of functions, assigning data types in the original description and synthesizing the output circuit description in Verilog / VHDL languages.

A set of software tools functions as part of an integrated development environment and allows a developer to form a set of debugged functions for their implementation in the form of a IC. The shell provides information resources for organizing the entire process of high-level VLSI synthesis based on the FFP approach.

By means of the developed tools, a number of scientific and technical solutions have been obtained: a set of complex functional blocks of a single-chip driver of the on-board network of a spacecraft [18], VLSI of the DSP unit based on the BMK K5540TN014A from the MRK06 navigation device [19] and others.

5. Conclusion

The review of recent languages and methods for designing logical circuits made it possible to substantiate the choice of the functional-flow parallel computing model and the Pifagor parallel programming language for the development of an architecture-independent method for synthesizing integrated circuits.

In the process of developing the proposed method for synthesizing IC based on a modified FFP model, a parallelism transformation method was proposed, which consists in reducing the maximum parallelism of the IC operating algorithm when switching to specific target architecture. This approach provides portability of parallel architectures to different platforms.

The parallelism transformation includes resource estimation, calculating the parallelism reduction factor for each resource class, and reducing circuit parallelism to achieve the required characteristics of a specific target platform.

The presented reduction methods make it possible to change the parallelism of the initial algorithm description and provide the implementation of the transfer mechanism to different architectures with different resource constraints.

The proposed method, in contrast to the parallelism induction methods, reduces the complexity of the transfer process by reducing the enumeration of the number of implementation options obtained in the synthesis process, taking into account resource constraints. At the same time, resource estimation methods at the high-level stage require accounting for overheads when changing parallelism to more sequential schemes. For this case, it is necessary to increase the accuracy of the estimate. For this, neural networks and machine learning methods can be used, which, based on the parameters of the circuit evaluation at a high-level stage, can predict the values of the circuit parameters with the required accuracy after implementation at a low level.

Direct calculation of resource required to implement the developed circuit is complicated due to the many transformations that occur during the synthesis and implementation at the low-level stage. The implementation of accurate methods for assessing the resource will further reduce the number of circuit options considered in the synthesis process under resource restrictions.

On the basis of the proposed methods of parallelism reduction and resource estimation, a number of tools have been implemented, such as a translator of an architecture-independent description of automaton and combinational schemes, resource estimation and parallelism reduction preprocessors. Preprocessors calculate and quantify the required circuit resource for selected types of unit blocks (cells, memory, triggers, LUTs, etc.) on supported target platforms.

References / Список литературы

[1] IEEE Std 1800-2012: IEEE Standard for SystemVerilog-Unified Hardware Design, Specification, and Verification Language, 2013.

[2] V.A. Alekhin. SystemC. Simulation of electronic systems. Goryachaya liniya – Telecom, 2018, 320 p. (in Russian) / Алехин В.А. SystemC. Моделирование электронных систем. Горячая линия – Телеком, 2018, 320 стр.

[3] Vivado Design Suite User Guide. High-Level Synthesis. UG902–Xilinx–2015. URL: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_4/ug902-vivado-high-level-synthesis.pdf

[4] Handel-C Language Reference Manual. Celoxica Limited, 2005, 26 p.

[5] I.I. Levin, V.A. Gudkov. Extension of high level language COLAMO for reconfigurable computer systems programming on the level of FPGA logic cells. Herald of computer and information technologies, no. 12, 2010, pp.10-17 (in Russian) / И.И. Левин, В.А. Гудков. Расширение языка высокого уровня COLAMO для программирования реконфигурируемых вычислительных систем на уровне логических ячеек ПЛИС. Вестник компьютерных и информационных технологий. no. 12, 2010 г., стр. 10-17.

[6] J. Sérot, G. Michaelson. Compiling Hume down to gates. In Draft Proc. of the 11th International Symposium on Trends in Functional Programming, 2011, pp, 191-226.

[7] J. O'Donnell, M.R. Barbacci, and C.J. Koomen. Hardware description with recursion equations. In Proc. of the 8th International Symposium on Computer Hardware Description Languages and Their Applications (CHDL '87), 1987, pp. 363-382.

[8] M. Sheeran. μFP, a language for VLSI design. In Proc. of the Conference Record of the ACM Symposium on LISP and Functional Programming (LISP '84), 1984, pp. 104-112.

[9] P. James-Roxby, S. Singh. Lava and JBits: From HDL to bitstream in seconds. In Proc. of the 9th IEEE Symposium on Field-Programmable Custom Computing Machines, 2001, pp. 91-100.

[10] J. Dongarra, G. Bosilca et al. PaRSEC: A programming paradigm exploiting heterogeneity for enhancing scalability. IEEE Computing in Science and Engineering, vol. 6, no. 15, 2013, pp. 36-45.

[11] J. Dongarra, A. Danalis et al. PTG: An Abstraction for Unhindered Parallelism. In Proc. of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing, 2014, pp. 21-30.

[12] A.I. Legalov. A functional language for creating architecture-independent parallel programs. Computational technologies, vol. 10, no. 1, 2005, pp. 71-89 (in Russian) / А.И. Легалов. Функциональный язык для создания архитектурно-независимых параллельных программ. Вычислительные технологии, том 10, no. 1, 2005 г., стр. 71-89.

[13] O.V. Nepomnyashchy, A.I. Legalov et al. Methods and algorithms for a high-level synthesis of the very-large-scale integration. WSEAS Transactions on Computers, vol. 15, 2016, pp. 239-247.

[14] H.J. Nussbaumer. Fast Fourier Transform and Convolution Algorithms. Springer, 1982, 288 p. / Г. Нуссбаумер. Быстрое преобразование Фурье и алгоритмы вычисления сверток. Радио и связь, 1985 г., 248 стр.

[15] O. V. Nepomnyashchiy, I. N. Ryzhenko et al. The VLSI High-Level Synthesis for Building Onboard Spacecraft Control Systems. In Proc. of the Scientific-Practical Conference "Research and Development – 2016", 2017, pp. 229–238.

[16] O.V. Nepomnyashchy, I.N. Ryzhenko et al. Translator of architecture-independent description of automata and combinational circuits. Certificate of state registration of software for computers No. 2021610682, 02/01/2021 (in Russian) / О.В. Непомнящий, И.Н. Рыженко и др. Транслятор архитектурно-независимого описания автоматных и комбинационных схем. Свидетельство о государственной регистрации ПО для ЭВМ № 2021610682, 01.02.2021.

[17] A.A. Komarov, I.N. Ryzhenko, O.V. Nepomnyashchy Program for synthesizing circuit descriptions in HDL hardware description languages from the Pifagor functional-parallel programming language. Certificate of state registration of software for computers No. 2015619175, 08/26/2015 (in Russian) / А.А. Комаров, И.Н. Рыженко, О.В. Непомнящий. Программа синтеза описания схем на языках описания аппаратуры HDL с языка функционально-параллельного программирования «Пифагор». Свидетельство о государственной регистрации ПО для ЭВМ № 2015619175, 26.08.2015.

[18] O.V. Nepomnyashchy, A.A. Komarov et al. Program for the driver of the onboard network of the spacecraft. Certificate of state registration of software for computers No. 2015616896, 06/26/2015 (in Russian) / О.В. Непомнящий, А.А. Комаров и др. Программа драйвера бортовой сети космического аппарата. Свидетельство о государственной регистрации ПО для ЭВМ №2015616896, 25.06.2015.

[19] O.V. Nepomnyashchy, A.A. Komarov, I.N. Ryzhenko Complex-functional block of the lowering adder-limiter. Certificate of state registration of software for computers No. 2016619714, 08/26/2016 (in Russian) / О.В. Непомнящий, А.А. Комаров, И.Н. Рыженко. Сложно-функциональный блок понижающего сумматора-ограничителя. Свидетельство о государственной регистрации ПО для ЭВМ № 2016619714, 26.08.2016.

Information about authors / Информация об авторах

Darya Sergeevna ROMANOVA – post-graduate student of the Department of Computer Engineering of the Siberian Federal University, assistant of the Department of IT&MOIS of the Krasnoyarsk State Agrarian University. Research interests: parallel algorithms, high-performance systems.

Дарья Сергеевна РОМАНОВА – аспирант кафедры вычислительной техники Сибирского федерального университета, ассистент кафедры ИТиМОИС Красноярского государственного аграрного университета. Сферы научных интересов: параллельные алгоритмы, высокопроизводительные системы.

Oleg Vladimirovich NEPOMNYASHCHIY – Professor, Ph.D. in technical sciences, Head of the Computer Science Department at the Siberian Federal University. Research interests: microprocessor systems, high-level analysis and synthesis of complex single-chip systems.

Олег Владимирович НЕПОМНЯЩИЙ – профессор, кандидат технических наук, заведующий кафедрой вычислительной техники. Область научных интересов: микропроцессорные системы, высокоуровневый анализ и синтез сложных однокристальных систем.

Igor Nikolayevich RYZHENKO is an Assistant Professor at the Computer Science Department. Research interests: VLSI high-level synthesis technologies, digital signal processing.

Игорь Николаевич РЫЖЕНКО является ассистентом кафедры вычислительной техники. Сферы научных интересов: технологии высокоуровневого синтеза СБИС, цифровая обработка сигналов.

Alexander Ivanovich LEGALOV – Doctor of Technical Sciences, Professor of the Faculty of Computer Science. His research interests include programming technologies, evolutionary software development, architecture-independent parallel programming.

Александр Иванович ЛЕГАЛОВ – доктор технических наук, профессор факультета компьютерных наук. Его научные интересы включают технологии программирования,

эволюционная разработка программного обеспечения, архитектурно-независимое параллельное программирование.

Natalya Yurievna SIROTININA – Ph.D. in technical sciences, Associate Professor, Department of Computer Science. Her research interests include neural networks and parallel programming.

Наталья Юрьевна СИРОТИНИНА – кандидат технических наук, доцент кафедры «Вычислительная техника». Ее научные интересы включают нейронные сети и параллельное программирование.

DOI: 10.15514/ISPRAS-2022-34(1)-6

Обобщенная контекстно-зависимая теоретико-графовая модель фольклорных и литературных текстов

Н.Д. Москин, ORCID: 0000-0001-5556-5349 <moskin@petrsu.ru>
А.А. Рогов, ORCID: 0000-0002-8815-7920 <rogov@petrsu.ru>
Р.В. Воронов, ORCID: 0000-0003-0104-6409 <rvoronov@petrsu.ru>
Петрозаводский государственный университет,
185910, Россия, г. Петрозаводск, пр. Ленина, д. 33

Аннотация. Одной из проблем при автоматической обработке текстов является их атрибуция. Под этим термином понимают установление атрибутов текстового произведения (определение авторства, времени создания, места записи и др.). В статье представлена обобщенная контекстно-зависимая теоретико-графовая модель, предназначенная для анализа фольклорных и литературных текстов. Минимальной структурной единицей модели (примитивом) является слово. Множества слов объединяются в вершины, причем одно и то же слово может иметь отношение к разным вершинам. Ребра и графовые подструктуры отражают лексические, синтаксические и семантические связи текста. Характеристиками модели являются ее нечеткость, иерархичность и темпоральность. В качестве примеров приводятся иерархическая теоретико-графовая модель составляющих (на примере литературных произведений А. С. Пушкина), темпоральная теоретико-графовая модель сказочного сюжета (на примере русских волшебных сказок А. М. Афанасьева) и нечеткая теоретико-графовая модель «сильных» связей грамматических классов (на примере анонимных статей из дореволюционных журналов «Время», «Эпоха» и еженедельника «Гражданин», которые редактировал Ф. М. Достоевский). Модель строится таким образом, чтобы в дальнейшем ее можно было исследовать с помощью методов искусственного интеллекта (например, деревьев решений или нейронных сетей). Для этой цели в информационной системе «Фольклор» был разработан формат для хранения подобных данных, а также реализованы процедуры для ввода, редактирования и анализа текстов и их теоретико-графовых моделей.

Ключевые слова: теоретико-графовая модель; атрибуция текстов; лексика; синтаксис; семантика; нечеткий граф; иерархический граф; темпоральный граф; информационная система «Фольклор»

Для цитирования: Москин Н.Д., Рогов А.А., Воронов Р.В. Обобщенная контекстно-зависимая теоретико-графовая модель фольклорных и литературных текстов. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 73-86. DOI: 10.15514/ISPRAS–2022–34(1)–6

Generalized context-dependent graph-theoretic model of folklore and literary texts

N.D. Moskin, ORCID: 0000-0001-5556-5349 <moskin@petrsu.ru>
A.A. Rogov, ORCID: 0000-0002-8815-7920 <rogov@petrsu.ru>
R.V. Voronov, ORCID: 0000-0003-0104-6409 <rvoronov@petrsu.ru>
Petrozavodsk State University,
33, Lenin st., Petrozavodsk, 185910, Russia

Abstract. One of the problems of automatic text processing is their attribution. This term is understood as the establishment of the attributes of a text work (determination of authorship, time of creation, place of recording,

etc.). The article presents a generalized context-dependent graph-theoretic model designed for the analysis of folklore and literary texts. The minimal structural unit of the model (primitive) is a word. Sets of words are combined into vertices, and the same word can be related to different vertices. Edges and graph substructures reflect the lexical, syntactic and semantic links of the text. The characteristics of the model are its fuzziness, hierarchy and temporality. As examples, a hierarchical graph-theoretical model of components (on the example of literary works by A. S. Pushkin), a temporal graph-theoretic model of a fairy tale plot (on the example of Russian fairy tales by A. M. Afanasyev) and a fuzzy graph-theoretic model of «strong» connections of grammatical classes (on the example of anonymous articles from the pre-revolutionary magazines «Time», «Epoch» and the weekly «Citizen», edited by F. M. Dostoevsky). The model is built in such a way that it can be further explored using artificial intelligence methods (for example, decision trees or neural networks). For this purpose, a format for storing such data was implemented in the information system «Folklore», as well as procedures for entering, editing and analyzing texts and their graph-theoretic models.

Keywords: graph-theoretic model; text attribution; lexis; syntax; semantics; fuzzy graph; hierarchical graph; temporal graph; information system «Folklore»

For citation: Moskin N.D., Rogov A.A., Voronov R.V. Generalized context-dependent graph-theoretic model of folklore and literary texts. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 73-86 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–6

1. Введение

Задачи анализа текстов, как одно из направлений искусственного интеллекта [12], все чаще решаются с помощью современных математических методов и компьютерных технологий. В научной литературе обозначены различные подходы и методы решения задач классификации и поиска текстов, атрибуции текстов, машинного перевода, автоматического реферирования, выявления плагиата, анализа тональности текстов, генерации текстов, реконструкции текстов и др. [7] Эти задачи объединяет необходимость поиска нестандартных, скрытых закономерностей, присущих текстам, которые можно обнаружить, например, с помощью методов искусственного интеллекта и машинного обучения. Также отметим, что результаты, полученные при решении одного класса задач обработки текстов можно применить и для другого класса.

В настоящее время активно развивается направление обработки естественного языка (машинный перевод, создание чат-ботов и т.д.), связанное с использованием нейросетевых технологий (Transformer, RNN, CNN) [11, 17]. Эти технологии позволяют выявлять скрытые закономерности языка, однако для их настройки и адекватной работы требуется большой объем данных. Другим недостатком этих технологий является закрытость получаемых языковых закономерностей (моделей) и отсутствие обоснования принимаемых решений. Это допустимо при их технологическом использовании, но часто не подходит для научных исследований по атрибуции текстов.

Основная идея математических подходов к решению задачи атрибуции текстов заключается в подсчете статистических параметров, которые, с одной стороны, идентифицируют стиль автора, а, с другой стороны, им слабо контролируются [15, 16]. Это направление известно под названием стилеметрия (stylometry). Ученые изучают такие характеристики как длина предложения, длина слова, богатство словарного запаса и т.п. Иногда исследования базируются на подсчете *n*-грамм - последовательностей текстовых элементов (букв, слов, идентификаторов частей речи и т. д.), взятых в порядке их появления в тексте [18]. Однако часто подобные эксперименты не давали убедительных результатов и являлись труднообъяснимыми для филологов.

Отметим, что по причине своей многоплановой и многоуровневой структуры текст является сложным объектом для изучения. С одной стороны, в нем можно выделить разные структурные единицы (например, на лексическом, синтаксическом и семантическом уровнях), а с другой стороны, установить разные виды связей, т.е. в результате одному и тому же тексту могут соответствовать несколько различных моделей. Такие модели можно

представить в виде графов, которые состоят из множества объектов (вершин) и связей между этими объектами (ребер). На наш взгляд, теоретико-графовые модели являются перспективным направлением в области атрибуции текстов. Например, использование технологии GNN (Graph Neural Network или в переводе с английского *графовые нейронные сети*) позволяет не преобразовывать структуру в числовой вектор, теряя при этом часть важной информации, а сохранить топологические отношения для последующего анализа [19].

Разработка обобщенной модели обусловлена несколькими факторами. Разные теоретико-графовые модели позволяют получить новую информацию об исследуемых текстах. Особенно это важно в случае фольклорных и литературных текстов, когда их число в коллекциях в силу исторических причин невозможно увеличить. При этом получаемые выборки могут быть несбалансированными (например, произведения одного автора сильно преобладают над произведениями другого автора). Интерес представляют и новые характеристики, получаемые в результате синтеза разных моделей, и возможности для дальнейшей разработки более совершенных гибридных структур. Реализованные в программе математические методы и алгоритмы без труда можно перенести с одного класса моделей на другой. В случае же, когда они имеют разное описание, подобный подход выглядит труднореализуемым и сложно интерпретируемым для филологов. Отметим также, что становится проще на одних и тех же данных провести сравнение результатов классификации, полученных с помощью разных методик, и выявить наиболее эффективные. Поэтому значимым являются не только единообразное структурное описание теоретико-графовых моделей для атрибуции текстов, но и разработка общего формата для их хранения и дальнейшего анализа. Данное исследование в целом представляется полезным для систематизации теоретико-графовых моделей текстов и методологии их построения.

В данной работе предлагается обобщенная контекстно-зависимая теоретико-графовая модель. Она была апробирована на различных коллекциях фольклорных и литературных текстов [10, 11, 14]:

- теоретико-графовая модель синтаксической структуры, рассмотренная на материале фольклорных песен (Лужские песни, бесёдные песни) и стилизованных под фольклор текстов (Н. А. Клюев, А. К. Толстой, С. А. Есенин и др.);
- иерархическая модель синтаксической структуры предложения (на материале текстов П. А. Вяземского, Э. По, И. А. Бродского, а также переводов С. Андреевского, Д. Мережковского, К. Брюсова, Г. Голохвастова, Н. Голя, В. Топорова и др.);
- нечеткая теоретико-графовая модель на основе деревьев зависимостей (на материале духовных стихов о Голубиной книге из сборника Кирши Данилова и «Собрания народных песен П. В. Киреевского», былин в записи П. Н. Рыбникова);
- теоретико-графовая модель семантической структуры фольклорных песен (на материале песен Заонежья XIX – начала XX века в записи Ф. Студитского, В. Дашкова, В. Лысанова и пр.);
- деревья решений, полученные на основе анализа анонимных статей из дореволюционных журналов «Время» (1861-1863), «Эпоха» (1864-1865) и еженедельника «Гражданин» (1873-1874), которые редактировал Ф. М. Достоевский.

2. Обобщенная контекстно-зависимая теоретико-графовая модель фольклорных и литературных текстов

2.1 Нечеткие, темпоральные и иерархические графы

Как отмечается в [10], важными характеристиками теоретико-графовых моделей текстов при решении задачи атрибуции являются нечеткость, иерархичность и темпоральность.

Одним из проявлений *нечеткости* на разных уровнях языковой структуры текста являются случаи омонимии. Омонимами (греч. *homos* – одинаковый, *onyma* – имя) называются слова, разные по значению, но одинаковые по звучанию и написанию. Различают лексическую омонимию, морфологическую омонимию, лексико-морфологическую омонимию (наиболее частый вид) и синтаксическую омонимию [5].

Понятие нечеткого графа (fuzzy graph) основано на определении функции принадлежности, которая ставит в соответствие вершине или ребру графа значение от 0 до 1. Более строго нечеткий граф второго рода $\tilde{G} = (\tilde{V}, \tilde{E})$ определяется следующим образом [3]. Пусть имеется некоторое универсальное множество X и задано нечеткое множество $\tilde{V}$ в X имеющее вид

$$\tilde{V} = \{\langle \mu_V(v) \backslash v \rangle\}, v \in X,$$

где $0 \leq \mu_V(v) \leq 1$ – значение функции принадлежности для вершины v (здесь V – носитель множества $\tilde{V}$ Зададим также нечеткое множество ребер

$$\tilde{E} = \{\langle \mu_E(v_i, v_j) \rangle\}, v_i, v_j \in V,$$

где $0 \leq \mu_E(v_i, v_j) \leq 1$ – значение функции принадлежности для ребра (v_i, v_j). Если множество вершин является четким в отличие от множества ребер, то такой граф называется графом первого рода.

Также в ряде работ Л.С. Берштейна (например, в [2]) вводится понятие *темпорального графа*, т.е. модели, где связи между элементами (вершинами графа) изменяются во времени (в случае моделирования текста под этим термином будем понимать упорядоченность слов в тексте и соответствующих им вершин). Автор отмечает, что понятие темпорального графа (temporal graph) в литературе трактуется в достаточно широком диапазоне – от временных графиков до ориентированных ациклических графов и сетей Петри.

В математических терминах назовем темпоральным графом тройку $G = (X, \{\Gamma_t\}, T)$, где X – множество вершин графа с числом вершин $n = |X|, T = \{1,2, \dots, N\}$ – множество натуральных чисел, определяющих (дискретное) время; $\{\Gamma_t\}$ – семейство соответствий, или отображений множества вершин X в себя в момент времени $t \in T$, т.е. $(\forall t \in T)\Gamma_t : X \to X$. Причем, для различных моментов времени эти отображения, в общем случае, различные:

$$(\forall x \in X)(\forall t_1, t_2 \in T | t_1 \neq t_2)[\Gamma_{t_1}(x) \neq \Gamma_{t_2}(x)].$$

Во многих случаях требуются более сложные графовые формализмы, обладающие иерархической структурой (hierarchical graph). Известны *иерархические графовые модели*, описание которых приводится, например, в [8]. Граф C называется фрагментом графа G (обозначим $C \subseteq G$), если C – это подмножество элементов графа G. Обозначим F – иерархию фрагментов G, если $G \in F$ и для любых двух фрагментов C_1 и C_2 из F либо фрагменты C_1 и C_2 не пересекаются, либо один из них является частью (подфрагментом) другого. Фрагмент G – основной (главный) фрагмент иерархии F. Фрагмент $C \in F$ – элементарный, если в F нет фрагментов G, являющихся подфрагментами фрагмента C.

Пусть задана некоторая иерархия фрагментов F графа G. Для любых $C_1, C_2 \in F$ фрагмент C_1 – прямой подфрагмент C_2, если C_1 – подфрагмент C_2 и не существует такого $C_3 \in F$ отличного от C_1 и C_2, что $C_1 \subseteq C_3 \subseteq C_2$. Иерархический граф $H = (G, T)$ состоит из графа G и корневого дерева T, вершины которого соответствуют элементам некоторой иерархии в G, а дуги отражают отношение их непосредственной вложенности. T называется деревом вложенности, а G – основным графом иерархического графа H.

Однако отметим, что подобные иерархические графы не подходят для описания структуры текста. Как показано в п. 2.3, связи могут существовать не только между вершинами, но и между какой-либо вершиной и фрагментом графа.

2.2 Рекурсивное определение обобщенной контекстно-зависимой теоретико-графовой модели

Дадим рекурсивное определение обобщенной контекстно-зависимой теоретико-графовой модели для решения задачи анализа текстов. Это набор $G = (V, H, E, \alpha, \beta, \mu, \gamma)$ для текста T, который определим в три этапа:

1) Сегментация текста T:

- Пусть текст T состоит из упорядоченной последовательности слов $W = \{w_k\}_{k=1}^{K}$, где $K > 0$ – общее количество слов (индекс k соответствует порядку появления слова w_k в тексте);
- $W_l \subset W$ – упорядоченные подмножества слов в тексте ($l = 1, 2, \dots, L$). Подмножество W_l может состоять как из одного слова, так и из совокупности слов (необязательно следующих подряд). Допускается, что подмножества могут пересекаться;

2) Определение элементов теоретико-графовой модели G:

- $V = \{v_i\}_{i=1}^{n}$ – непустое конечное множество вершин;
- $V_j \subset V, j = 0,1, \dots, m$ – подмножества вершин теоретико-графовой модели, таких что их объединение совпадает с $V = \bigcup_{j=1}^{m} V_j$. Допускается, что подмножества могут пересекаться;
- $H = \left\{\{v_i\}_{i=1}^{n} \cup \{G_j\}_{j=1}^{m}\right\}$ – множество, объединяющее вершины из V и совокупность вложенных теоретико-графовых структур $G_j = (V_j, H_j, E_j)$ уровня j, где множество H_j уровня $j = 2,3, \dots, m$ определяется либо как пустое множество, либо как подмножество вложенных теоретико-графовых структур уровней меньших j, т.е. $H_j \subset \{G_l\}_{l=1}^{j-1}$, а E_j представляет собой подмножество упорядоченных пар из $V_j \cup H_j$, т.е. ребер вложенной модели (иерархичность);
- $E \subset H \times H$ (подмножество упорядоченных пар элементов из H) – множество ребер G, которое состоит из s элементов. При этом подмножества ребер $E_j \subseteq E, j = 1,2, \dots, m$ и попарно не пересекаются;

3) Определение атрибутов элементов теоретико-графовой модели G:

- γ – отображение, задающее соответствие между объектами теоретико-графовой модели $x_i \in H \cup E$ и подмножествами слов в тексте $W_l \subset W$. Допускается, что некоторые вершины или подструктуры могут быть «фиктивные» (не связанные со словами в тексте), т. е. $\gamma(x_i) = \emptyset$. Таким образом, γ определяет упорядоченность (темпоральность) объектов теоретико-графовой модели.
- A – множество атрибутов (меток) вершин, которые определяются характеристиками текста. Элемент множества A может быть вектором, который определяет несколько атрибутов;
- $\alpha: V \to A$ – функция, задающая атрибуты (метки) вершинам;
- B – множество атрибутов (меток) ребер, которые определяются характеристиками текста. Элемент множества B может быть вектором, который определяет несколько атрибутов (в том числе, например, отсутствие направленности у ребра);
- $\beta: E \to B$ – функция, задающая атрибуты (метки) ребрам;
- $\mu: H \cup E \to [0,1]$ – функция, задающая нечеткость объектов теоретико-графовой модели.

Рассмотрим, как можно представить в терминах обобщенной модели три теоретико-графовые структуры, обладающие соответственно свойствами иерархичности (п. 2.3), темпоральности (п. 2.4) и нечеткости (п. 2.5).

2.3 Иерархическая теоретико-графовая модель составляющих

В литературе известны два вида деревьев, которые описывают синтаксическую структуру текста. *Деревья зависимостей* обычно используются в описаниях языков со свободным порядком слов (например, русского). Для описания языков с фиксированным порядком слов преимущественно используется второй тип графов – *деревья составляющих* [5]. При этом в предложении выделяются группы слов, функционирующие как отдельные синтаксические единицы – составляющие. Система составляющих – это множество отрезков предложения, которое обладает тем свойством, что каждые два входящих в него отрезка либо не пересекаются, либо один из них содержится в другом. Речь идет о так называемых *синтагмах*. Это совокупность нескольких слов, объединённых по принципу семантико-грамматической сочетаемости, единица синтагматики.

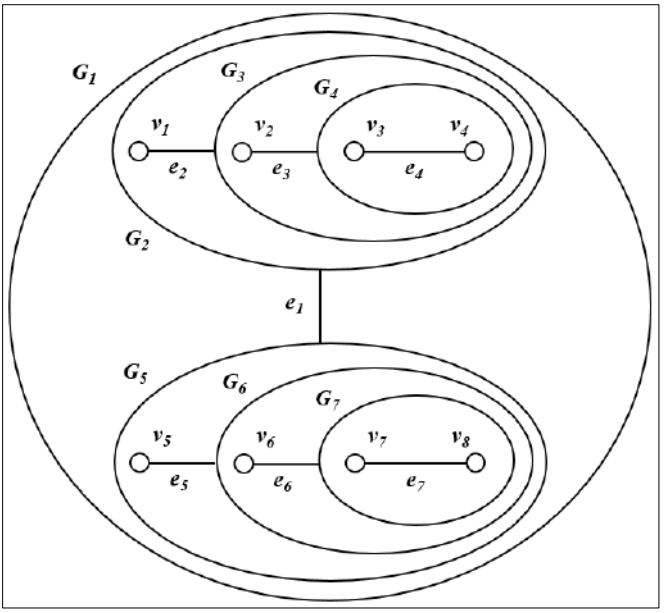

Рис. 1. Модель составляющих фрагмента «Онегин, добрый мой приятель, родился на брегах Невы»
Fig. 1. Model of the components of the fragment «Onegin, my good friend, was born on the banks of the Neva»

Табл. 1. Соответствие вершин и слов текста
Table 1. Correspondence of vertices and words of the text

№	Подмножества слов	Вершина или ребро	Группы	Функция принадлежности
1	$W_1 = \{w_1\}$ = {“Онегин”}	v_1	$\alpha(v_1) = N$	$\mu(v_1) = 1$
2	$W_2 = \{w_2\}$ = {“добрый”}	v_2	$\alpha(v_2) = A$	$\mu(v_2) = 1$
3	$W_3 = \{w_3\}$ = {“мой”}	v_3	$\alpha(v_3) = C$	$\mu(v_3) = 1$
4	$W_4 = \{w_4\}$ = {“приятель”}	v_4	$\alpha(v_4) = N$	$\mu(v_4) = 1$
5	$W_5 = \{w_5\}$ = {“родился”}	v_5	$\alpha(v_5) = V$	$\mu(v_5) = 1$
6	$W_6 = \{w_6\}$ = {“на”}	v_6	$\alpha(v_6) = P$	$\mu(v_6) = 1$
7	$W_7 = \{w_7\}$ = {“брегах”}	v_7	$\alpha(v_7) = N$	$\mu(v_7) = 1$
8	$W_8 = \{w_8\}$ = {“Невы”}	v_8	$\alpha(v_8) = N$	$\mu(v_8) = 1$

Рассмотрим модель составляющих (рис. 1) на примере фрагмента из романа в стихах А. С. Пушкина: «Онегин, добрый мой приятель, родился на брегах Невы» [5]. Отметим, что

существует множество текстов, которые приписываются Александру Сергеевичу и до сих пор в этом вопросе нет окончательного ответа [13]. Общее количество слов фрагмента $K = 8$. В данном случае подмножества W_l $(l = 1, \dots, L = 8)$ будут соответствовать не только словам w_i, но и вершинам графа v_i (табл. 1). Функция α может задавать, например, часть речи слова (т.е. быть атрибутом вершины). Описание множества A представлено в п. 3.2. Сразу отметим, функция μ для всех вершин и ребер принимает значение 1 (нечеткие связи отсутствуют), т.е. $\mu(v_i) = \mu(e_j) = 1$.

Данная теоретико-графовая модель содержит семь подструктур $G_j = (V_j, H_j, E_j), j = 1, 2, \dots, m = 7$. Первая подструктура G_1 содержит G_2 и G_5, которые соединяются ребром e_1, т.е. $V_1 = \emptyset$, $H_1 = \{G_2, G_5\}$, $E_1 = \{e_1 = (G_2, G_5)\}$. Аналогично опишем другие подструктуры (ребра из множества $E = \{e_t\}_{t=1}^{s=7}$ не являются ориентированными, атрибуты им не заданы, т.е. $B = \emptyset$, они могут соединять не только вершины, но и подструктуры):

- $V_2 = \{v_1\}$, $H_2 = \{G_3\}$, $E_2 = \{e_2 = (v_1, G_3)\}$;
- $V_3 = \{v_2\}$, $H_3 = \{G_4\}$, $E_3 = \{e_3 = (v_2, G_4)\}$;
- $V_4 = \{v_3, v_4\}$, $H_4 = \emptyset$, $E_4 = \{e_4 = (v_3, v_4)\}$;
- $V_5 = \{v_5\}$, $H_5 = \{G_6\}$, $E_5 = \{e_5 = (v_5, G_6)\}$;
- $V_6 = \{v_6\}$, $H_6 = \{G_7\}$, $E_6 = \{e_6 = (v_6, G_7)\}$;
- $V_7 = \{v_7, v_8\}$, $H_7 = \emptyset$, $E_7 = \{e_7 = (v_7, v_8)\}$.

Поскольку все вершины и ребра находятся «внутри» той или иной подструктуры, множества $V_0 = E_0 = \emptyset$.

2.4 Темпоральная теоретико-графовая модель сказочного сюжета

Вторая теоретико-графовая модель возникает при исследовании сказочных сюжетов (например, из волшебных сказок А. М. Афанасьева [1]). Основоположником подобного структурного направления является В. Я. Пропп и его последователи. В текстах выделяются инварианты – действующие лица сказки [4], которых можно объединить в десять групп:

- герой (*H*);
- антигерой (антагонист, вредитель) (*A*);
- прорицатель (*P*);
- даритель (снабдитель) (*Д*);
- помощник (*П*);
- антипомощник (*V*);
- глупец (*G*);
- антидаритель (*W*);
- награда (*N*);
- препятствие (*R*).

Тело сказки в самом общем виде есть конечная последовательность встреч действующих лиц, связанных соединительными фразами (например, «долго-ли, коротко-ли шел он и наконец увидел…»). Встречи непосредственно связаны с их поступками: например, «Даритель даст Герою совет о том, как действовать дальше». Возможные встречи действующих лиц сказки представлены в таблице 2 [4].

Табл. 2. Возможные встречи действующих лиц сказки
Table 2. Possible meetings of characters in the tale

Действующее лицо	**С кем может встретиться**							
H	*A*	*P*	*Д*	*П*	*V*	*N*	*G*	*W*
A		*P*		*П*	*V*	*N*	*G*	*W*

Р				*П*		*N*	*G*	
Д				*П*		*N*		
П					*V*	*N*		*W*
V						*N*	*G*	
N							*G*	*W*
G								*W*

Построим теоретико-графовую модель, где вершинами являются действующие лица сказки, а ребра будут отражать их встречи, пронумерованные в соответствии с их появлением в теле сказки. Если встреч было несколько, то ребра будут кратными.

Табл. 3. Соответствие вершин/ребер и их групп
Table 3. Correspondence of vertices and their groups

№	Подмно-жества слов	Вершина или ребро	Группы	Функция принадлежности
1	W_1	v_1	$\alpha(v_1)=$“Герой (*H*)”	$\mu(v_1)=1$
2	W_2	v_2	$\alpha(v_2)=$“Награда (*N*)”	$\mu(v_2)=1$
3	W_3	v_3	$\alpha(v_3)=$“Даритель (*Д*)”	$\mu(v_3)=1$
4	W_4	v_4	$\alpha(v_4)=$“Антигерой (*A*)”	$\mu(v_4)=1$
5	W_5	v_5	$\alpha(v_5)=$“Препятствие (R)”	$\mu(v_5)=1$
6	W_6	e_1	$\beta(e_1)=$“*H-Д*”	$\mu(e_1)=1$
7	W_7	e_2	$\beta(e_2)=$“*H-Д*”	$\mu(e_2)=1$
8	W_8	e_3	$\beta(e_3)=$“*H-Д*”	$\mu(e_3)=1$
9	W_9	e_4	$\beta(e_4)=$“*H-R*”	$\mu(e_4)=1$
10	W_{10}	e_5	$\beta(e_5)=$“*H-R*”	$\mu(e_5)=1$
11	W_{11}	e_6	$\beta(e_6)=$“*H-R*”	$\mu(e_6)=1$
12	W_{12}	e_7	$\beta(e_7)=$“*H-A*”	$\mu(e_7)=1$
13	W_{13}	e_8	$\beta(e_8)=$“*H-N*”	$\mu(e_8)=1$

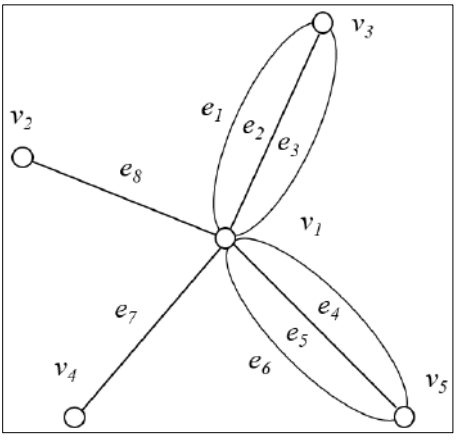

Рис. 2. Теоретико-графовая модель сказочного сюжета
Fig. 2. Graph-theoretic model of a fairy tale plot

Рассмотрим пример сказочного сюжета, в котором выделяются пять действующих лиц и восемь встреч [6]. Здесь не только каждому i–ому действующему лицу (вершине), но и j–й встрече (ребрам) будут соответствовать в тексте набор слов (словосочетаний) W_l, где $l = 1,2,\dots,L = n + s = 13$ (табл. 3). При этом подмножества W_l включают слова, которые могут повторяться и необязательно следовать друг за другом. Например, главный герой v_1 связан с набором слов, которые находятся в разных частях сказки (включая местоимения и синонимы). Ребра e_i, как правило, имеют отношение к цельным фрагментам, описывающим встречу действующих лиц. В данной модели отсутствует иерархия, поэтому $m = 0$ (рис. 2). Сравнивая между собой подобные графы можно посмотреть, как менялся сказочный сюжет с течением времени, какие были особенности у разных регионов и пр.

2.5 Нечеткая теоретико-графовая модель «сильных связей» грамматических связей

Третий вид теоретико-графовой модели основан на матрице частот парной встречаемости грамматических классов слов текста (биграмм). Подобные модели использовались, например, для атрибуции исторических текстов [9] и анонимных текстов из дореволюционных журналов «Время» (1861-1863), «Эпоха» (1864-1865) и еженедельника «Гражданин» (1873-1874), которые редактировал Ф. М. Достоевский [11]. Для получения такой матрицы необходимо:

- выбрать систему грамматических классов;
- перекодировать последовательность слов анализируемого текста в последовательность соответствующих обозначений грамматических классов;
- вычислить частоты парной встречаемости a_{ij} для каждой пары классов с учетом направления развертывания текста (слева направо).

Рассмотрим построение модели на примере текста Ф. М. Достоевского «Молодое перо. По поводу литературной подписи "Современник" № 1 и 2», опубликованной в журнале «Время» (1863 год. Разд. Современное обозрение. № 2. С. 221-226). Каждому i-му синтаксическому классу ($n = 8$ в примере на рис. 3) определяется вершина v_i (соответствующее подмножество слов W_i выбирается по принципу их принадлежности к этому классу, табл. 4).

Табл. 4. Соответствие вершин/ребер и их групп
Table 4. Correspondence of vertices and their groups

№	Подмно-жества слов	Вершина или ребро	Группы	Функция принадлежности
1	W_1	v_1	$\alpha(v_1) =$ “Существительное”	$\mu(v_1) = 1$
2	W_2	v_2	$\alpha(v_2) =$ “Прилагательное”	$\mu(v_2) = 1$
3	W_3	v_3	$\alpha(v_3) =$ “Местоимение”	$\mu(v_3) = 1$
4	W_4	v_4	$\alpha(v_4) =$ “Глагол”	$\mu(v_4) = 1$
5	W_5	v_5	$\alpha(v_5) =$ “Частица”	$\mu(v_5) = 1$
6	W_6	v_6	$\alpha(v_6) =$ “Предлог”	$\mu(v_6) = 1$
7	W_7	v_7	$\alpha(v_7) =$ “Союз”	$\mu(v_7) = 1$
8	W_8	v_8	$\alpha(v_8) =$ “Цитата”	$\mu(v_8) = 1$
9	-	e_1	$\beta(e_1) =$ “Существительное-Существительное”	$\mu(e_1) = 0{,}02041$
10	-	e_2	$\beta(e_2) =$ “Существительное-Местоимение”	$\mu(e_2) = 0{,}02099$

11	-	e_3	$\beta(e_3)$ = "Существительное-Глагол"	$\mu(e_3) = 0{,}02741$
12	-	e_4	$\beta(e_4)$ = "Существительное-Предлог"	$\mu(e_4) = 0{,}02041$
13	-	e_5	$\beta(e_5)$ = "Существительное-Союз"	$\mu(e_5) = 0{,}03032$
14	-	e_6	$\beta(e_6)$ = "Прилагательное-Существительное"	$\mu(e_6) = 0{,}05831$
15	-	e_7	$\beta(e_7)$ = "Местоимение-Существительное"	$\mu(e_7) = 0{,}02974$
16	-	e_8	$\beta(e_8)$ = "Местоимение-Глагол"	$\mu(e_8) = 0{,}02624$
17	-	e_9	$\beta(e_9)$ = "Глагол-Местоимение"	$\mu(e_9) = 0{,}02041$
18	-	e_{10}	$\beta(e_{10})$ = "Глагол-Союз"	$\mu(e_{10}) = 0{,}02332$
19	-	e_{11}	$\beta(e_{11})$ = "Частица-Глагол"	$\mu(e_{11}) = 0{,}02157$
20	-	e_{12}	$\beta(e_{12})$ = "Предлог-Существительное"	$\mu(e_{12}) = 0{,}03324$
21	-	e_{13}	$\beta(e_{13})$ = "Предлог-Местоимение"	$\mu(e_{13}) = 0{,}02507$
22	-	e_{14}	$\beta(e_{14})$ = "Союз-Местоимение"	$\mu(e_{14}) = 0{,}02741$
23	-	e_{15}	$\beta(e_{15})$ = "Цитата-Цитата"	$\mu(e_{15}) = 0{,}02857$

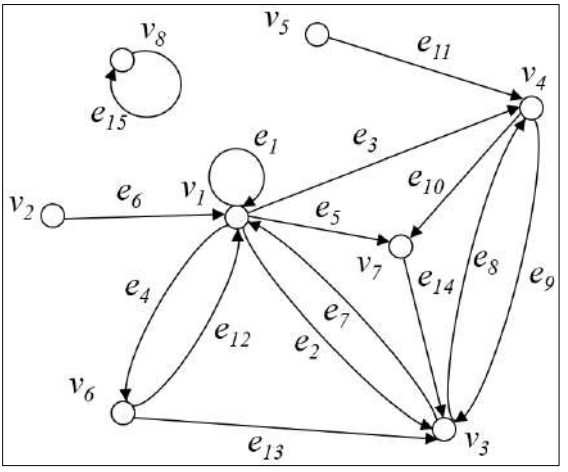

Рис. 3. Нечеткая модель «сильных» связей грамматических классов (текст Ф. М. Достоевского «Молодое перо», порог 0,02)
Fig. 3. Fuzzy model of «strong» connections of grammatical classes (text by F. M. Dostoevsky "Young Pen", threshold 0.02)

3. Формат представления теоретико-графовых моделей в информационной системе «Фольклор»

Приведенные примеры показывают, что в терминах обобщенной модели можно задавать структуры, обладающие разными свойствами. В [10] приводятся также другие виды

теоретико-графовых моделей, которые могут, например, обладать всеми тремя свойствами: иерархичность, нечеткость и темпоральность. Однако подобные формализации несомненно более сложны для описания и визуализации.

Возникает потребность в автоматической обработке текстов и их теоретико-графовых моделей, построенных на основе разных принципов. С этой целью в Петрозаводском государственном университете была разработана информационная система «Фольклор» [10]. Изначально она создавалась как проблемно-ориентированная система, предназначенная для сравнительного анализа одной коллекции бесёдных песен Заонежья конца XIX – начала XX века. Однако впоследствии программа была модифицирована таким образом, что позволила проводить исследование других коллекций на основе различных моделей.

Для хранения и последующего анализа текстов в информационной системе «Фольклор» был реализован формат SNG. Этот формат представляет собой текстовый файл, который можно легко редактировать. Рассмотрим на примере иерархической модели составляющих, как структурируются данные (табл. 5). Файл делится на пять частей: общие характеристики текста, слова, объекты, связи и матрица инцидентности (технически части разделены между собой одиночной строкой с комментариями, начинающиеся с символов //).

- Общие характеристики текста (1-11 строки). В первой строке указывается название текста, во второй строке – название группы, связанных между собой текстов (например, название теоретико-графовой модели), количество строк в тексте (если значение равно нулю, то граф строится без привязки к тексту), строки текста, количество характеристик текста, затем пары: название характеристики и ее значение.
- Слова текста (13-29 строки). В первой строке указывается количество слов, далее для каждого слова – номер строки, начало выделения, длина (параметр не является избыточным, т. к. в некоторых текстах возможно слияние слов), после дефиса – часть речи (*N* – существительное, *A* – прилагательное, *C* – местоимение, *O* – числительное, *V* – глагол, *E* – причастие, *G* – деепричастие, *D* – наречие, *S* - категория состояния, *P* – предлог, *L* – союз, *U* – частица, *I* – междометие). Теоретически в случае омонимии (например, слово «мой» может быть как местоимением, так и глаголом, но в данном случае из контекста понятно, что это местоимение) можно указать двойную часть речи (например, -*CV*). Если часть речи неизвестна или еще не определена, то ставится знак ‘?’.
- Объекты текста (31-62 строки). В первой строке указывается количество объектов, во второй строке – способ отображения вершин при визуализации, затем для каждого из них – название, уровень вложенности (если это обычная вершина, то указывается 0, если первый уровень, то – 1, если второй, то – 2, и т. д.), значение функции принадлежности, группа, количество слов объекта, номера слов, относящихся к объекту (если вершина фиктивная, то к ней не привязываются слова текста).
- Связи в тексте (64-75 строки). В первой строке указывается количество связей, затем - способ отображения ребер при визуализации, название связи, значение функции принадлежности, группа, количество слов связи, номера слов, относящихся к связи.
- Матрица связей (77-91 строки). Представляет собой матрицу инцидентности, где строки соответствуют объектам текста, а столбцы – связям. Если в столбце напротив вершины *i* указано -1, а напротив вершины *j* – 1, то это значит, что дуга идет из *i* в *j*.

Табл. 5. Представление иерархической теоретико-графовой модели составляющих фрагмента «Онегин, добрый мой приятель, родился на брегах Невы»
Table 5. Representation of a hierarchical graph-theoretic model of fragment components «Onegin, my good friend, was born on the banks of the Neva»

№	Строка	№	Строка
1	Онегин, добрый мой приятель,	45	брегах
	родился на брегах Невы	46	0 1 1 1 6
2	иерархическая модель составляющих	47	Невы

	///////////////	48	0 1 1 1 7
3	1	49	мой приятель
4	Онегин, добрый мой приятель,	50	1 1 1 2 2 3
5	родился на брегах Невы	51	добрый мой приятель
	///////////////	52	2 1 1 3 1 2 3
6	2	53	Онегин, добрый мой приятель
7	Автор	54	3 1 1 4 0 1 2 3
8	А.С. Пушкин	55	брегах Невы
9	Источник	56	1 1 1 2 6 7
10	Роман в стихах "Евгений Онегин"	57	на брегах Невы
11	///////////////	58	2 1 1 3 5 6 7
	8	59	родился на брегах Невы
12	0 0 6	60	3 1 1 4 4 5 6 7
13	-N	61	Онегин, добрый мой приятель, родился
14	0 8 6		на брегах Невы
15	-A	62	4 1 1 8 0 1 2 3 4 5 6 7
16	0 15 3	63	///////////////
17	-C	64	7
18	0 19 8	65	group4
19	-N	66	ребро 1
20	0 29 7	67	1 0 0 0
21	-V	68	ребро 2
22	0 37 2	69	1 0 0 0
23	-P	70	ребро 3
24	0 40 6	71	1 0 0 0
25	-N	72	ребро 4
26	0 47 4	73	1 0 0 0
27	-N	74	ребро 5
28	///////////////	75	1 0 0 0
29	15	76	///////////////
30	group2	77	0 -1 0 0 0 0 0
31	Онегин	78	0 0 -1 0 0 0 0
32	0 1 1 1 0	79	0 0 0 -1 0 0 0
33	добрый	80	0 0 0 1 0 0 0
34	0 1 1 1 1	81	0 0 0 0 -1 0 0
35	мой	82	0 0 0 0 0 -1 0
36	0 1 1 1 2	83	0 0 0 0 0 0 -1
37	приятель	84	0 0 0 0 0 0 1
38	0 1 1 1 3	85	0 0 1 0 0 0 0
39	родился	86	0 1 0 0 0 0 0
40	0 1 1 1 4	87	-1 0 0 0 0 0 0
41	на	88	0 0 0 0 0 1 0
42	0 1 1 1 5	89	0 0 0 0 1 0 0
43		90	1 0 0 0 0 0 0
44		91	0 0 0 0 0 0 0

4. Заключение

В данной статье предложена обобщенная контекстно-зависимая теоретико-графовая модель для атрибуции текстов. Модель обладает свойствами иерархичности, нечеткости и темпоральности. Она была апробирована на материале фольклорных (народные песни, сказки, духовные стихи, былины) и литературных текстов (за авторством Н. А. Клюева, А. К. Толстого, С. А. Есенина, А. С. Пушкина, П. А. Вяземского, Э. По, И. А. Бродского и др.). Примеры моделей представлены в монографиях [10, 11] и на электронном ресурсе СМАЛТ (“Статистические методы анализа литературных текстов”), который расположен по адресу

http://smalt.karelia.ru/. В работе рассмотрен формат хранения теоретико-графовой модели в информационной системе «Фольклор» и системе СМАЛТ.

Описание единого подхода к построению различных теоретико-графовых моделей текстов позволяет находить расстояния между ними, т.е. решать задачи классификации и кластеризации текстов. Это можно использовать, например, при решении задачи атрибуции. В качестве такого расстояния можно использовать меры на основе операций редактирования, на основе максимального общего подграфа, минимального общего надграфа, структурных спектров и т. п. [10].

Список литературы / References

[1] Афанасьев А.М. Народные русские сказки А. Н. Афанасьева: в 3 т. М., Государственное Издательство Художественной литературы (Гослитиздат), 1957 г. / Afanasyev A.M. Folk Russian fairy tales by A. N. Afanasyev: in 3 volumes. Moscow, State Publishing House of Fiction (Goslitizdat), 1957 (in Russian).

[2] Берштейн Л.С., Боженюк А.В. Использование темпоральных графов как моделей сложных систем. Известия ЮФУ. Технические науки, № 4 (105), 2010 г., стр. 198-203 / Bershtein L.S., Bozhenyuk A.V. The use of temporal graphs as models of complex systems. Izvestiya SFedU. Engineering Sciences, vol. 4 (105), 2010, pp. 198-203 (in Russian).

[3] Берштейн Л.С., Боженюк А.В. Нечеткие графы и гиперграфы. М., Научный мир, 2005 г., 256 стр. / Bershtein L.S., Bozhenyuk A.V. Fuzzy graphs and hypergraphs. Moscow, Scientific world, 2005, 256 p. (in Russian).

[4] Гаазе-Рапопорт М.Г. Поиск вариантов в сочинении сказок. Дополнение в книге Зарипов Р.Х. Машинный поиск вариантов при моделировании творческого процесса. М.: Наука, 1983 г., стр. 213-223. / Gaaze-Rapoport M.G. Search for variants in the composition of fairy tales. Supplement in Zaripov R.H. Machine search for variants in modeling the creative process. Moscow, Nauka, 1983, pp. 213-223 (in Russian).

[5] Гладкий А.В. Синтаксические структуры естественного языка. М., ЛКИ, 2007 г., 152 с. / Gladky A.V. Syntactic structures of natural language. Moscow, LKI, 2007, 152 p. (in Russian).

[6] Зубов А.В., Зубова И.И. Основы искусственного интеллекта для лингвистов. М., Университетская книга, Логос, 2007 г., 320 стр. / Zubov A.V., Zubova I.I. Fundamentals of artificial intelligence for linguists. Moscow, University book, Logos, 2007, 320 p. (in Russian).

[7] Ильвовский Д.А., Черняк Е.Л. Системы автоматической обработки текстов. Открытые системы. СУБД, no. 1, 2014 г., стр. 51-53 / Ilvovsky D.A., Chernyak E.L. Systems of automatic processing of texts. Open systems. DBMS, no. 1, 2014, pp. 51–53 (in Russian).

[8] Касьянов В.Н., Евстигнеев В.А. Графы в программировании: обработка, визуализация и применение. СПб., БХВ-Петербург, 2003 г., 1104 стр. / Kasyanov V.N., Evstigneev V.A. Graphs in programming: processing, visualization and application. St. Petersburg, BHV-Petersburg, 2003, 1104 p. (in Russian).

[9] Милов Л.В., Бородкин Л.И. и др. От Нестора до Фонвизина: Новые методы определения авторства. М., Прогресс, 1994 г., 445 стр. / Milov L.V., Borodkin L.I. et al. From Nestor to Fonvizin: New methods for determining authorship. Moscow, Progress, 1994, 445 p. (in Russian)

[10] Москин Н.Д. Теоретико-графовые модели фольклорных текстов и методы их анализа. Петрозаводск, Изд-во ПетрГУ, 2013 г., 148 стр. / Moskin N.D. Graph-theoretic models of folklore texts and methods of their analysis. Petrozavodsk, PetrGU Publishing House, 2013, 148 p. (in Russian)

[11] Рогов А.А., Абрамов Р.В. и др. Проблема атрибуции в журналах «Время», «Эпоха» и еженедельнике «Гражданин». Петрозаводск, Изд-во «Острова», 2021 г., 391 с. / Rogov A.A., Abramov R.V. et al. The problem of attribution in the magazines «Time», «Epoch» and the weekly «Citizen». Petrozavodsk: Publishing house «Islands», 2021, 391 p. (in Russian)

[12] Соколов И.А. Теория и практика применения методов искусственного интеллекта. Вестник Российской академии наук, том 89, вып. 4, 2019, стр. 365-370. / Sokolov I.A. Theory and practice of application of artificial intelligence methods. Bulletin of the Russian Academy of Sciences, vol. 89, issue 4, 2019, pp. 365-370. (in Russian)

[13] Хозяинов С.А. Атрибуция публицистических произведений, приписываемых А. С. Пушкину: тексты 1830-1836 гг. Санкт-Петербург, 2008 г., 24 с. / Hozyainov S.A. Attribution of publicistic works attributed to A. S. Pushkin: texts of 1830-1836. St. Petersburg, 2008, 24 p. (in Russian)

[14] Щеголева Л.В., Лебедев А.А., Москин Н.Д. Методы анализа данных в задаче разграничения фольклорных и авторских текстов. Вопросы языкознания, 2020 г., no. 2, стр. 61-74. / Shchegoleva L.V., Lebedev A.A., Moskin N.D. Methods of data analysis in the problem of distinguishing between folklore and author's texts. Questions of linguistics, 2020, no. 2, pp. 61-74. (in Russian)
[15] Calle-Martin J., Miranda-Garcia A. Stylometry and Authorship Attribution: Introduction to the Special Issue. English Studies, vol. 93, no. 3, 2012, pp. 251-258.
[16] Stamatatos E. A Survey of Modern Authorship Attribution Methods. Journal of the American Society for Information Science and Technology, vol. 60, no. 3, 2009, pp. 538-556.
[17] Vaswani A., Shazeer N. et al. Attention is all you need. In Proc. of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6000-6010.
[18] Zečević A. N-gram based text classification according to authorship. In Proc. of the Second Student Research Workshop associated with RANLP 2011, 2011, pp. 145-149.
[19] Zhou J., Cui G. et al. Graph neural networks: A review of methods and applications. AI Open, vol. 1, 2020, pp. 57-81.

Информация об авторах / Information about authors

Николай Дмитриевич МОСКИН – кандидат технических наук, доцент, доцент кафедры теории вероятностей и анализа данных. Сфера научных интересов: цифровые гуманитарные науки, теоретико-графовые модели, интеллектуальный анализ данных, компьютерная лингвистика, мультимедиа-технологии, компьютерная графика.

Nikolai Dmitrievich MOSKIN – Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of Probability Theory and Data Analysis. Research interests: digital humanities, graph-theoretic models, data mining, computational linguistics, multimedia technologies, computer graphics.

Александр Александрович РОГОВ – доктор технических наук, профессор, заведующий кафедрой теории вероятностей и анализа данных. Сфера научных интересов: математическое моделирование, прикладная статистика, математические методы распознавания образов, математические методы анализа литературных текстов.

Aleksandr Aleksandrovich ROGOV – Doctor of Technical Sciences, Professor, Head of the Department of Probability Theory and Data Analysis. Research interests: mathematical modeling, applied statistics, mathematical methods of pattern recognition, mathematical methods of analysis of literary texts.

Роман Владимирович ВОРОНОВ – доктор технических наук, профессор кафедры прикладной математики и кибернетики. Сфера научных интересов: математическое моделирование, задачи оптимизации, комбинаторные задачи на графах, математические методы и модели систем локального позиционирования мобильных объектов.

Roman Vladimirovich VORONOV – Doctor of Technical Sciences, Professor of the Department of Applied Mathematics and Cybernetics. Research interests: mathematical modeling, optimization problems, combinatorial problems on graphs, mathematical methods and models of mobile object local positioning systems.

DOI: 10.15514/ISPRAS-2022-34(1)-7

The Software Implementation of a Metagraph Processing System Based on the Big Data Approach

V.M. Chernenkiy, ORCID: 0000-0002-7684-3114 <chernen@bmstu.ru>
I.V. Dunin, ORCID: 0000-0003-1153-4561 <johnmoony@yandex.ru>
Yu.E. Gapanyuk, ORCID: 0000-0001-9005-8174 <gapyu@bmstu.ru>

Bauman Moscow State Technical University,
Baumanskaya 2nd, 5, Moscow, 105005, Russia

Abstract. The paper discusses the approach to solving the problem of processing metagraphs using Big Data technology. The formal definition of the metagraph data model and the metagraph agent model are given. The metagraph representation using the flat graph model is discussed. The flat graph and metagraph Big Data processing are described. The architecture of the system for processing data in metagraph is discussed. The metagraph processing using metagraph agents based on Big Data technology is discussed. The experiments result for parallel metagraph processing are given.

Keywords: Big Data; Big Graph; Flat Graph; Metagraph; Pregel; Signal-Collect; Gather Sum Apply

For citation: Chernenkiy V.M., Dunin I.V., Gapanyuk Yu.E. The Software Implementation of a Metagraph Processing System Based on the Big Data Approach. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 87-100. DOI: 10.15514/ISPRAS–2022–34(1)–7

Acknowledgements. The article is based on the materials of the report at the Seventh International Conference «Actual Problems of System and Software Engineering» (APSSE 2021).

Программная реализация системы обработки метаграфов на основе подхода Больших Данных

В.М. Черненький, ORCID: 0000-0002-7684-3114 <chernen@bmstu.ru>
И.В. Дунин, ORCID: 0000-0003-1153-4561 <johnmoony@yandex.ru>
Ю.Е. Гапанюк, ORCID: 0000-0001-9005-8174 <gapyu@bmstu.ru>

Московский государственный технический университет им. Н.Э. Баумана,
105005, Россия, Москва, 2-ая Бауманская, 5

Аннотация. В работе обсуждается подход к решению проблем обработки метаграфов с использованием технологий больших данных. Дается формальное определение метаграфовой модели данных и метаграфового агента. Обсуждается представление метаграфа через простой граф. Описывается обработка простых графов и метаграфов. Обсуждается архитектура системы по обработке метаграфов. Обсуждается обработка с использованием метаграфовых агентов на основе технологий Больших Данных. Демонстрируются результаты экспериментов.

Ключевые слова: большие данные; большие графы; графы; метаграфы; Pregel; Signal-Collect; Gather Sum Apply

Для цитирования: Черненький В.М., Дунин И.В., Гапанюк Ю.Е. Программная реализация системы обработки метаграфов на основе подхода Больших Данных. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 87-100. DOI: 10.15514/ISPRAS–2022–34(1)–7

Благодарности. Статья подготовлена по материалам доклада на Седьмой Международной конференции «Актуальные проблемы системной и программной инженерии» (АПСПИ 2021).

1. Introduction

Presently, Big Data frameworks are widely used to process information in various domains, from modern industrial enterprise [1] to social networks analysis [2]. Big data includes big datasets with numeric and categorical data, text information, images, and video data. Information in the form of graphs is traditionally used in computer science. In particular, thesauri and ontologies are used to process knowledge and texts, but such models cannot yet be classified as big data. Probably, it is appropriate to speak about the appearance of problems of processing Big Graphs, first of all, in connection with the appearance of knowledge graphs.

The problem of processing Big Graphs can be solved in various ways, for example, by creating specialized high-performance hardware for processing graphs [3, 4]. However, the most common way for Big Graphs processing is to use specialized Big Data frameworks for Big Graphs processing.

The dominant graph model in such frameworks is usually a flat graph model or property graph model (which is, in fact, the multigraph model). Here, by a flat graph, we mean a graph in which a directed or undirected edge connects exactly two vertices. However, the flat graph model is not flexible enough and may not be a convenient solution for modeling complex data domains with hierarchical relationships. For example, complex technical products' assembly sequence problem requires not a flat graph but a hypergraph model [5].

One of the existing extensions of the traditional graph model is the metagraph model. The metagraph model consists of a metagraph data model and a metagraph agent model aimed to process metagraph data. In this article, we present the implementation of the metagraph agent model using Big Data processing capabilities.

The article is organized as follows. In sections 2 and 3, the metagraph data model and the metagraph agent model are formally defined. In section 4, the metagraph representation using the flat graph model is discussed. In section 5, the flat graph and metagraph Big Data processing are described. In section 6, the metagraph processing using metagraph agents is discussed, including the experimental subsection.

2. The Metagraph Data Model

Metagraph is a kind of complex network model proposed by A. Basu and R. Blanning in their book [6] and then adapted for information systems description in our paper [7]. According to [7]:

$$MG = \langle V, MV, E \rangle, \tag{1}$$

where MG – metagraph; V – set of metagraph vertices; MV – set of metagraph metavertices; E – set of metagraph edges.

Metagraph vertex is described by a set of attributes:

$$v_i = \{atr_k\}, v_i \in V, \tag{2}$$

where v_i – metagraph vertex; atr_k – attribute.

Metagraph edge is described by a set of attributes, the source and destination vertices, and edge direction flag:

$$e_i = \langle v_S, v_E, eo, \{atr_k\} \rangle, e_i \in E, eo = true \mid false, \tag{3}$$

where e_i – metagraph edge; v_S – source vertex (metavertex) of the edge; v_E – destination vertex (metavertex) of the edge; eo – edge direction flag ($eo=true$ – directed edge, $eo=false$ – undirected edge); atr_k – attribute.

The metagraph fragment:

$$MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV), \tag{4}$$

where MG_i – metagraph fragment; ev_j – an element that belongs to the union of vertices, edges, and metavertices.

The metagraph metavertex:

$$mv_i = \langle \{atr_k\}, MG_j \rangle, mv_i \in MV, \quad (5)$$

where mv_i – metagraph metavertex belongs to a set of metagraph metavertices MV; atr_k – attribute, MG_j – metagraph fragment.

Thus, metavertex, in addition to the attributes, includes a fragment of the metagraph. The presence of private attributes and connections for metavertex is a distinguishing feature of metagraph. It makes the definition of metagraph holonic – metavertex may include a number of lower-level elements and, in turn, may be included in a number of higher-level elements.

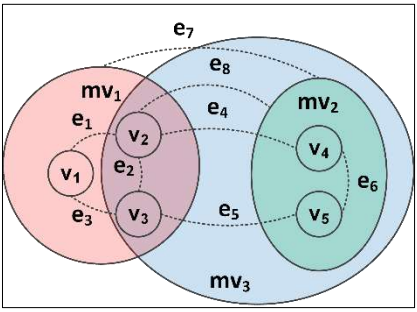

Fig. 1. The example of the metagraph data model

The example of the data metagraph (shown in fig. 1) contains three metavertices: mv_1, mv_2, and mv_3. Metavertex mv_1 contains vertices v_1, v_2, v_3 and connecting them edges e_1, e_2, e_3. Metavertex mv_2 contains vertices v_4, v_5, and connecting them edge e_6. Edges e_4, e_5 are examples of edges connecting vertices v_2-v_4 and v_3-v_5 that are contained in different metavertices mv_1 and mv_2. Edge e_7 is an example of the edge connecting metavertices mv_1 and mv_2. Edge e_8 is an example of the edge connecting vertex v_2 and metavertex mv_2. Metavertex mv_3 contains metavertex mv_2, vertices v_2, v_3, and edge e_2 from metavertex mv_1 and also edges e_4, e_5, e_8, showing the holonic nature of the metagraph structure.

3. The Metagraph Agent Model

The metagraph model is aimed for complex data description. But it is not aimed for data transformation. To solve this issue, the metagraph agent (ag^{MG}) aimed for data transformation is proposed. There are two kinds of metagraph agents: the metagraph function agent (ag^F) and the metagraph rule agent (ag^R). Thus $ag^{MG} = ag^F \mid ag^R$.

The metagraph function agent serves as a function with input and output parameters in the form of metagraph:

$$ag^F = \langle MG_{IN}, MG_{OUT}, AST \rangle, \quad (6)$$

where ag^F – metagraph function agent; MG_{IN} – input parameter metagraph; MG_{OUT} – output parameter metagraph; AST – abstract syntax tree of metagraph function agent in the form of metagraph.

The metagraph rule agent is rule-based:

$$ag^R = \langle MG, R, AG^{ST} \rangle, R = \{r_i\}, r_i : MG_j \rightarrow OP^{MG}, \quad (7)$$

where ag^R – metagraph rule agent; MG – working metagraph, a metagraph on the basis of which the rules of the agent are performed; R – set of rules r_i; AG^{ST} – start condition (metagraph fragment for start rule check or start rule); MG_j – a metagraph fragment on the basis of which the rule is performed; OP^{MG} – set of actions performed on metagraph.

The antecedent of the rule is a condition over a metagraph fragment. The consequent of a rule is a set of actions performed on metagraph. Rules can be divided into open and closed.

The consequent of the open rule is not permitted to change the metagraph fragment occurring in the rule antecedent. In this case, the input and output metagraph fragments may be separated. The open rule is similar to the template that generates the output metagraph based on the input metagraph.

The consequent of the closed rule is permitted to change the metagraph fragment occurring in the rule antecedent. The metagraph fragment changing in rule consequent causes to trigger the antecedents of other rules bound to the same metagraph fragment. But incorrectly designed closed rules system can cause an infinite loop of metagraph rule agents.

Thus, the metagraph rule agent can generate the output metagraph based on the input metagraph (using open rules) or can modify the single metagraph (using closed rules).

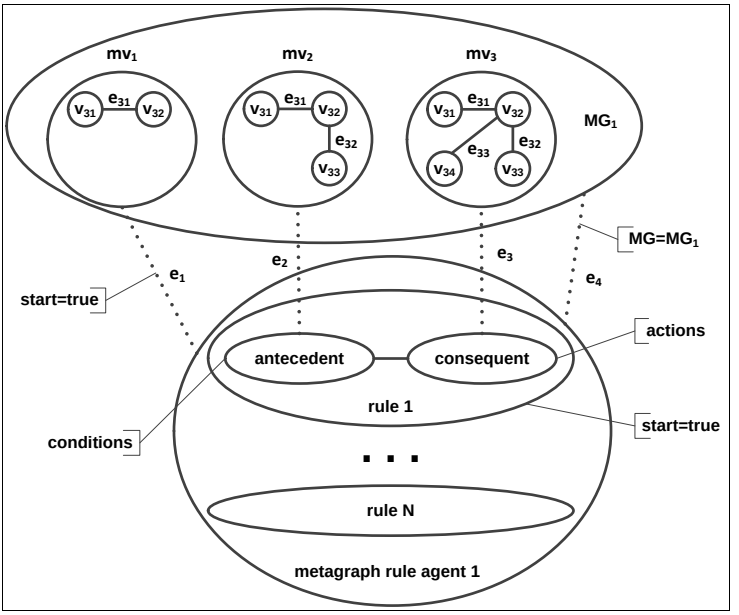

Fig. 2. The example of the metagraph rule agent

The example of a metagraph rule agent is shown in fig. 2. The metagraph rule agent «metagraph rule agent 1» is represented as metagraph metavertex. According to the definition, it is bound to the working metagraph MG_1 – a metagraph on the basis of which the rules of the agent are performed. This binding is shown with edge e_4.

The metagraph rule agent description contains inner metavertices corresponds to agent rules (rule 1 … rule N). Each rule metavertex contains antecedent and consequent inner vertices. For the given an example, mv_2 metavertex bound with the antecedent, which is shown with edge e_2, and mv_3 metavertex bound with consequent, which is shown with edge e_3. Antecedent conditions and consequent actions are defined in the form of attributes bound to antecedent and corresponding consequent vertices.

The start condition is given in the form of the attribute *«start=true.»* If the start condition is defined as a start metagraph fragment, then the edge bound start metagraph fragment to agent metavertex (edge e_1 in given an example) is annotated with the attribute *«start=true.»* If the start condition is defined as a start rule, then the rule metavertex is annotated with the attribute *«start=true»* (rule 1 in given an example). Fig. 2 shows both cases corresponding to the start metagraph fragment and to the start rule.

The distinguishing feature of a metagraph agent is its homoiconicity which means that it can be a data structure for itself. This is due to the fact that according to the definition, a metagraph agent

may be represented as a set of metagraph fragments, and this set can be combined in a single metagraph. Thus, a metagraph agent can change the structure of other metagraph agents.

4. The Metagraph Representation Using Flat Graph Model

To build systems using the metagraph model, it is necessary to determine the method of physical representation of the metagraph for persistent storage and for processing in Big Data platforms. We considered various options: relational representation, representation with multiple documents (nested or separate), representation via a flat graph. The term "flat graph" is also used for planar graphs. By flat graph we mean simple graph, as an opposite to graph with nested vertices (metagraph). We conducted a comparative analysis of different representation options. Features of different options and experiments to compare performance are described in our paper [8]. Experiments have shown that the representation of the metagraph via a flat graph is the most preferable.

In such representation, each vertex, metavertex, and the edge of the graph are represented with a separate vertex of the flat graph. The example is shown in fig. 3.

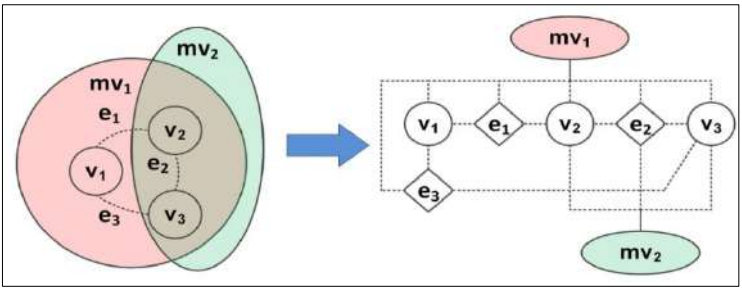

Fig. 3. The metagraph representation via a flat graph

Such a representation is, on the one hand, more intuitive. On the other hand, it allows us to efficiently process the metagraph using methods for working with flat graphs from graph DBMS (optimized graph queries) and Big Data graph platforms (distributed graph processing models).

5. The Flat Graph and Metagraph Big Data Processing

5.1 The Overview of a Metagraph Processing System

Consider a possible structure of a system for processing data in metagraph form. The system consists of two main components (shown in fig. 4): a storage module and a processing module.

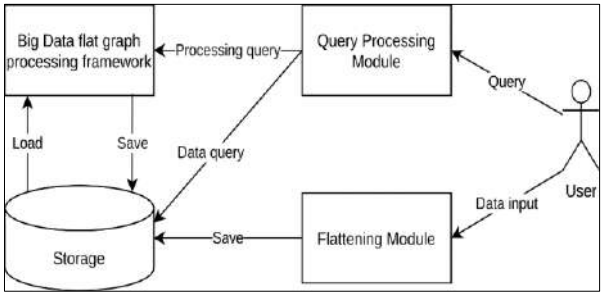

Fig. 4. The structure of the metagraph processing framework

Since we are using a graph representation, it is advisable to use a graph DBMS for storage. We assume that the conversion of the metagraph to a flat graph occurs at the writing stage to the storage in the flattening module. The processing is conducted on a Big Data platform running distributed across multiple machines.

The system works as follows. Firstly, a request comes from the user in the form of calling the system API or in some formal language. The query processing module determines if the query is a simple data query, and in this case, it returns data from storage. If the query requires calculation on metagraph, it occurs in the Big Data graph framework (which is shown in fig. 5). Metagraph is loaded from persistent storage (e.g., the Neo4j graph DBMS) into the Big Data processing platform (Apache Spark) into the RAM of the cluster nodes. Next, the request is processed with the tools of the graph processing platform. Then the problem is solved on a graph representation of the metagraph, and the result of a solution is displayed to the user.

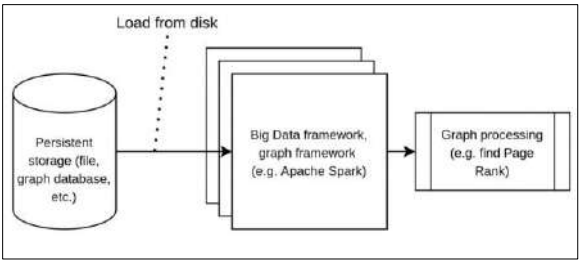

Fig. 5. The operation of the metagraph processing framework

5.2 Graph Processing Models

Modern Graph Big Data frameworks use different approaches to graph processing. Though these approaches share basic principles (like vertex-centric processing), but there are some differences between them. Below we briefly describe some popular models and consider their usage for metagraph processing.

5.2.1 Pregel model

Pregel model [9] was originally introduced by Google and is implemented in multiple graph frameworks (Apache Spark GraphX [10], Apache Giraph [11], Flink Gelly [12]). Pregel model (and other vertex-oriented approaches) were designed specifically to deal with graph data. Standard Big Data approaches, like Map Reduce, are not well suited for graph data structure and graph problems. Map Reduse requires passing the entire state of the graph from one stage to the next - in general requiring much more communication and associated serialization overhead. Abstraction from graph physical distribution (given by messaging system) also simplifies the algorithms. As we use graph representation of metagraph, it seems reasonable to use graph-specific approaches.

Graph processing in Pregel consists of a sequence of synchronous iterations. During each iteration, for each vertex of the graph, a user-defined function is executed, which modifies the value of the vertex in a certain way. Then the vertex sends messages to neighbors' vertices. The next iteration begins only when all vertices have been processed.

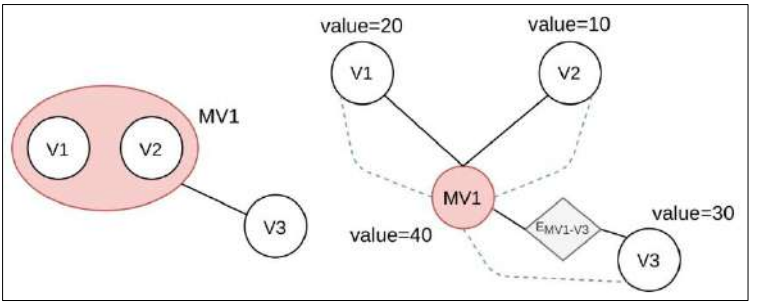

Fig. 6. The example metagraph for the Pregel model

This model is the most general and flexible, but it has a significant drawback in the restrictions on the synchronism of iterations. In real-world graphs (like social graphs), vertex degrees often follow

power-law distribution [13], which means that most of the graph vertices have a low degree, and there are a few vertices with a much higher degree. For graph representation of real-world metagraphs, we may expect the same distribution because graph metavertices have not only neighbors but also sub-vertices. If the graph has a significant number of high-degree vertices (which are metavertices of metagraph), Pregel iterations will be filled unevenly since the metavertices will receive and send more messages. For example, consider the simple metagraph in fig. 6 (on the left side, we show the original metagraph, on the right side corresponding flat graph representation).

During the Pregel step of some algorithms, vertices send values to each other. Note that we presume that edge-vertices like $E_{MV1\text{-}V3}$ are not processed as separate vertices and blindly transfer messages. Still, in general, it is up to the implementation of a specific algorithm. Metavertex MV_1 sends its value to neighbor V_3 and subvertices V_1 and V_2 and receives their values. It sends and receives three times more messages than other vertices. As the next iteration will start only when metavertex is processed, all other vertices stay idle. If a metagraph has relatively large metavertices, it may become a performance issue.

5.2.2 Signal-Collect / Scatter-Gather model

This model used by Apache Flink Gelly also assumes vertex-level iterative processing [14]. It consists of two phases: signal (sending messages to other vertices) and collect (receiving messages and updating the vertex).

Two modes of operation are possible. In synchronous mode, this model works similarly to the Pregel. In asynchronous mode, there is no barrier between iterations, and vertices function as independent actors. In this case, we do not get downtime for processing metavertices. However, such a model (depending on the problem being solved) may require local locks, which will complicate the algorithms. Consider the example in fig. 7.

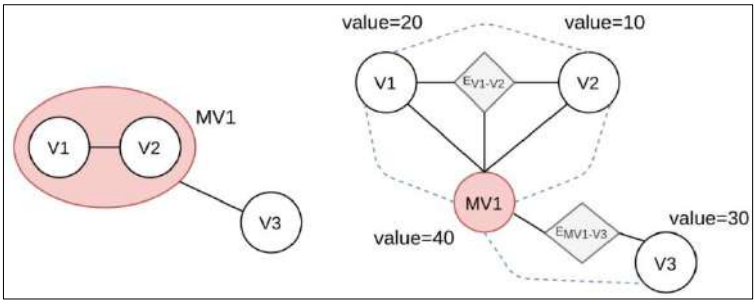

Fig. 7. The example metagraph for the Signal-Collect model

Vertex V_1 receives value from neighbor V_2 (through intermediate vertex representing edge V1-V2). With synchronous Pregel-like approach V_1 will process this value and send it further only in the next iteration, after MV_1 processes all its messages. In asynchronous mode, it will happen before MV_1 is processed. This approach is more error-prone to race condition-related problems, so algorithms should be adapted accordingly.

5.2.3 Gather Sum Apply and closely related Gather-Apply-Scatter [15] models

This model [15] used by PowerGraph [16], GraphLab [17], Apache Flink Gelly is also an iterative one. In this model, user code is launched not over the vertices but over the edges. It includes three phases (Gather, Sum, and Apply) which are shown in fig 8.

The Gather phase – execution of the algorithm relative to an edge and sending messages to the vertices. The Gather phase produces the partial result. Sum phase – each vertex performs aggregation of partial results from received messages. Apply phase – updating the vertex with the result value. For simplicity, we show gather messages only in one direction. In the case of an undirected graph, there are also reverse messages.

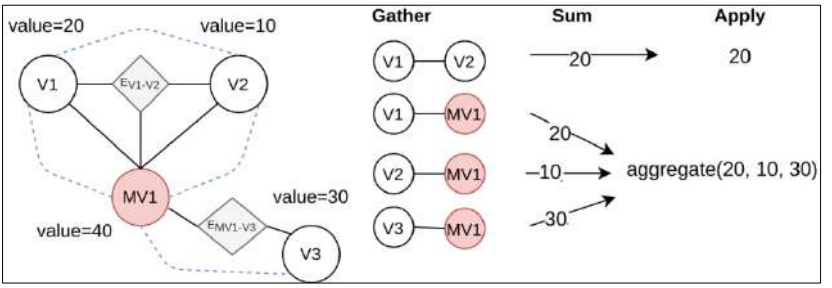

Fig. 8. The example metagraph for the Gather-Sum-Apply model

Since edges, unlike vertices, all have the same degree (equal to 2), they are processed equally. Thus, in this case, there is no downtime for processing metavertices. However, this model has limitations that can complicate the implementation of algorithms:

1) Updating a vertex requires the aggregation of messages from different edges. The update function must be associative and commutative.
2) There is no way to transfer messages between arbitrary vertices because messages are transmitted from edge to vertex.

Thus, the described models have both advantages and disadvantages. Pregel model is the most flexible, but it may not be most efficient if the metagraph contains some very high-degree metavertices. The Signal-Collect model can be more performant but requires more complicated concurrent algorithms. The Gather Sum Apply model also deals with high-degree metavertices, but it has limitations on program architecture. We can also expect that the model's effectiveness may depend on the metagraph which is being processed and the problem being solved.

How these three models are suitable for processing metagraphs is the subject of further detailed research. This article's experimental part uses the Pregel model as the most flexible of the three models.

6. The Metagraph Processing Using Metagraph Agents

6.1 Metagraph Agents in the Big Data Framework

In our experiments, we use the Apache GraphX framework, which is an extension of Apache Spark for graph processing. Below we will briefly describe how the metagraph agent approach can be applied while using frameworks like GraphX.

Metagraph agent is an instrument for transforming metagraphs. It includes information about whether it should apply to a metagraph fragment and information about how to transform a fragment.

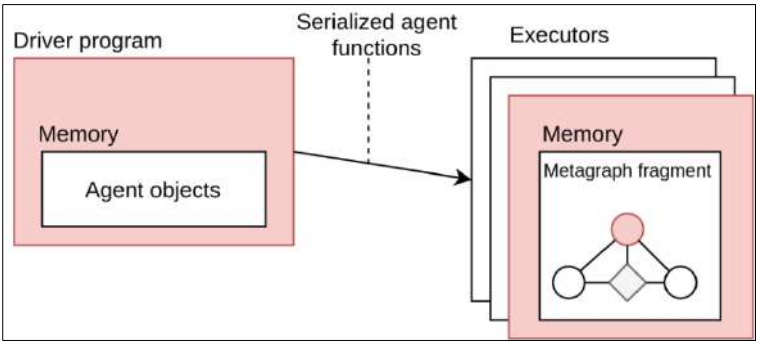

Fig. 9. Metagraph agents in processing framework

As was described in the previous section, in modern graph-oriented Big Data frameworks, usual execution models are vertex-oriented models. There are a driver program and multiple distributed

executors. In Apache Spark, the driver program transfers the required code as serialized objects or functions to executors. Executors run this code over parts of distributed datasets. In the case of graph framework, this code is run relatively to vertices of a graph. For example, in the Pregel model Spark GraphX driver program would distribute two functions: «vertex program» to update vertices and «message program» to send vertices.

From the software implementation point of view, the metagraph agent would be an object in the main memory of the driver program, with methods (functions) for updating the vertex based on received Pregel messages and for sending messages from vertex (which is shown in fig. 9). Agent functions also will include rule (condition) determining whether they should be applied to a specific vertex. Graph processing framework (like Spark GraphX) will serialize agent functions and transfer them to executors. Functions of metagraph agents will be executed (if allowed by agent condition) over vertices of a fragment of metagraph graph representation. As metagraph agents are objects, the driver program can organize them in a hierarchy and update them if necessary.

6.2 The Metagraph Single Source Shortest Path Problem

To illustrate metagraph processing, we conducted an experiment where we choose a Single Source Shortest Path (SSSP) problem as a typical graph processing algorithm. The experiment was conducted on the Spark GraphX framework, which implements the Pregel model.

When we apply the SSSP algorithm to a graph representation of metagraph, two main features occur. Firstly, we have a relation «sub vertex to metavertex.» For SSSP-problem, we can assume that the «parent» and «child» vertex can be accessed with some user-defined constant distance. For same-level vertices, distance is defined by edge length, as in a simple graph case. Secondly, the graph representation of the metagraph contains technical vertices representing edges (see vertices e_1, e_2, e_3 in fig. 3). They are to be excluded from the SSSP calculation.

The algorithm goes as follows. We initialize all vertices with an infinite distance value, except for starting vertex, which has zero distance. After that, we perform multiple iterations (Pregel steps). At each iteration, we run the same code over each vertex. Vertex sends messages over its outgoing edges unless finding an edge from a vertex representing the edge to its parent metavertex (see vertices e_1 and mv_1 in fig. 3). If the edge source vertex is a normal vertex, it sends the current distance value. If the edge source vertex is an edge-vertex, it adds the length. If the edge is an edge to parent metavertex, it adds user-defined distance between child and parent vertices.

At the beginning of each iteration, vertex updates its value with a minimum of received messages and current value. When no messages are sent during iteration, the algorithm converges.

In general, this algorithm resembles Pregel SSSP-algorithm for simple graphs with a few adaptations to metagraph physical representation.

Testing program is written is Scala, algorithm of determining Pregel messages at each iteration is presented below in Python-style pseudo-code:

```
messages = []
for edge in vertex.outgoingEdges:
  if edge.isEdgeToParent and vertex.isEdgeVertex:
    continue
  if edge.isEdgeToParent:
    distanceValue = distanceToParentVertex
  else if vertex.isEdgeVertex:
    distanceValue = vertex.distance + vertex.length
  else:
    distanceValue = vertex.distance
  messages.append((edge.dst.id, distanceValue)
```

In terms of metagraph agents, we can say that graph is processed by a single metagraph agent with a closed rule. Condition of the rule's antecedent allows any fragment to be processed (as the SSSP algorithm has to visit all edges of the graph). Rule consequent includes sending messages to neighbors and updating vertex with the minimum of received distances.

In our experiments we use synthetic generated datasets. Some real world examples for this problem may include processing social graph with hierarchy, where we need to know «distance» between people, but we have not only direct social connections, but also indirect connections based on hierarchy. This hierarchy would be represented by assigning people to nested metavertices of a metagraph, thus describing, for example, connections between people based on some organizational structure (possibly with overlapping metavertices). We could traverse such graph not only by direct «friendship» relation, but also up and down organizational structure.

6.3 Experiments

We generated multiple metagraphs and graphs with the same number of vertices and similar topology for the experiments. We use normal graph processing time as a baseline to show that processing of metagraph can be performed in comparable time (not varying by orders of magnitude, which would neglect benefits of the model). Metagraph consisted of N metavertices with two vertices in each, with each vertex also having one outgoing edge. A simple graph consisted of 3N vertices with three outgoing edges each. Each vertex has three relations in both cases, and a number of vertices were the same (not considering edge-vertices for metagraph). Graphs were generated by Python scripts as CSV files and imported into the Spark GraphX framework.

Experiments were conducted on Amazon EMR infrastructure with Spark 2.4.7 and Hadoop YARN 2.10.1. We used Amazon EC2 m5.xlarge instances with the following configuration (per instance): CPU - Intel Xeon Platinum 8175M 2.5 GHz, 16 GiB memory, 64 GiB EBS Storage. CSV files with initial data were imported into the HDFS filesystem.

The size of initial data (in Megabytes) for graph and metagraph with 100 000 to 10 000 000 vertices is shown in fig. 10.

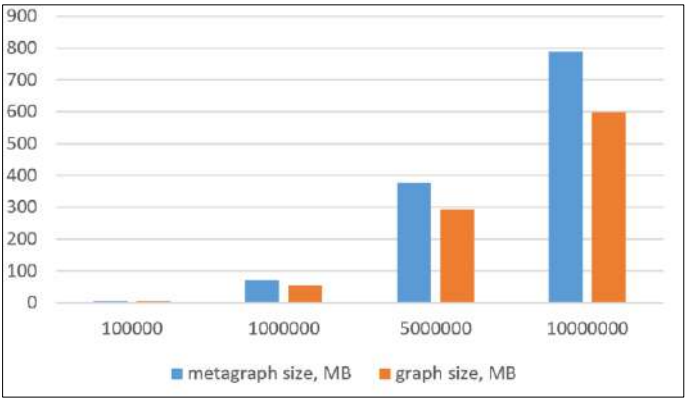

Fig. 10. Initial datasets size (MB) for different vertices counts

At first, we conducted an SSSP calculation with one master node and two worker nodes. SSSP calculation time (in seconds) for graphs and metagraphs of different sizes is shown in fig. 11.

As we can see, SSSP calculation time for metagraph is comparable to the time of processing a flat graph of the same size. Metagraph processing is obviously slower because graph representation of metagraph with N vertices and M edges will have N+M vertices in the flat graph.

We also conducted experiments to see how SSSP execution time in the case of metagraph processing is affected by the level of parallelism. At first, we conducted SSSP execution with a single worker node and a different number of Apache Spark executors. Results for processing metagraphs with

1 000 000 vertices are shown in fig. 12. Simple graph processing time is also included as the reference point.

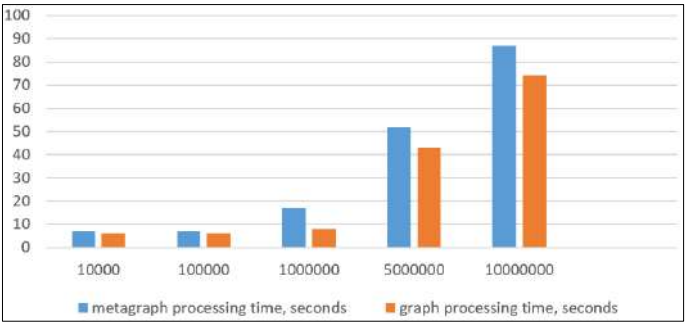

Fig. 11. SSSP calculation time (seconds) for a metagraph and a flat graph depending on the number of vertices in the graph (metagraph)

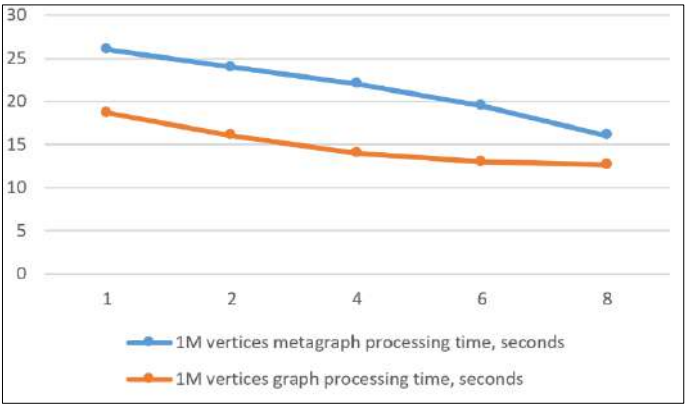

Fig. 12. SSSP calculation time (seconds) for metagraph and graph with 1 000 000 vertices depending on the number of executors

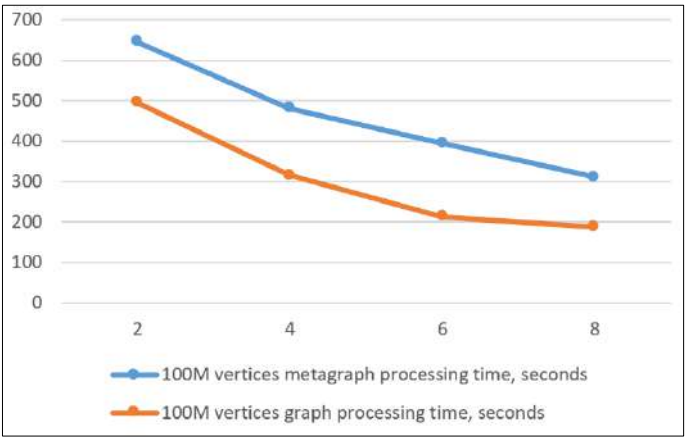

Fig. 13. SSSP calculation time (seconds) for metagraph and graph with 100 000 000 vertices depending on the number of worker nodes

Then we conducted an experiment with a larger dataset (metargraph and graph with 100 000 000 vertices) and with two to eight worker nodes in a cluster (see Fig. 13). As we can see from the charts, with the addition of new Spark executors and with the addition of worker nodes to the cluster, SSSP calculation time decreases similarly for metagraph and simple graph. Thus, metagraph processing may be effectively parallelized using Apache Spark.

7. Future Work

At the moment, we have implementation of persistent metagraph storage in graph database (Neo4j, ArangoDB). We also processed graph representation of metagraph in Big Data graph framework (Apache GraphX). Our next step will be creating a system which includes both persistent storage and processing framework and provides API for working with metagraph without manual processing of its graph representation.

8. Conclusions

Nowadays, Big Data frameworks are widely used for processing numeric and categorical data, text information, images, video data, and graph information.

The dominant graph model in such frameworks is usually a flat graph model or property graph model (which is, in fact, the multigraph model). However, to describe complex situations, flat graph models may not be enough, and we propose using a metagraph model.

The critical element of the metagraph model is metavertex. From the general system theory point of view, a metavertex is a particular case of the manifestation of the emergence principle, which means that a metavertex with its private attributes and connections becomes a whole that cannot be separated into its component parts.

The metagraph may be represented as a multipartite flat graph.

The system for processing data in metagraph form consists of two main components: the storage module and the processing module, where the processing module is based on the Big Data processing platform.

The three most popular graph processing models are the Pregel model, Signal-Collect / Scatter-Gather model, and Gather Sum Apply model. How these three models are suitable for processing metagraphs is the subject of further detailed research. In the experimental part, we use the Pregel model as the most flexible of the three models.

The experimental part is based on the Single Source Shortest Path (SSSP) problem. The SSSP calculation time for the metagraph is comparable to processing a flat graph of the same size. The metagraph processing may be effectively parallelized using Apache Spark.

The described metagraph approach with flat graph representation in graph processing frameworks seems to be an effective instrument for solving problems with processing complex data.

References

[1] Reut D., Falko S., Postnikova E. About scaling of controlling information system of industrial complex by streamlining of big data arrays in compliance with hierarchy of the present lifeworlds. International Journal of Mathematical, Engineering and Management Sciences, vol. 4, issue 5, 2019, pp. 1127-1139.

[2] Chesnokov V. Overlapping community detection in social networks with node attributes by neighborhood influence. In Proc. of the 6th International Conference on Network Analysis, Springer Proceedings in Mathematics & Statistics, vol. 197, 2017, pp. 187-203.

[3] Rasheed B., Popov A.Yu. Network graph datastore using DiSc processor. In Proc. of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, 2019, pp. 1582-1587.

[4] Abdymanapov C., Popov A.Yu. Motion planning algorithms using DISC. In: Proc. of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, 2019, pp. 1844-1847.

[5] Bozhko A.N. Hypergraph model for assembly sequence problem. In IOP Conference Series: Materials Science and Engineering, vol. 560, Issue 1, 2019.

[6] Basu A., Blanning R. Metagraphs and Their Applications. Springer, 2007, 174 p.

[7] Chernenkiy V.M., Gapanyuk Yu.E. et al. The Hybrid Multidimensional-Ontological Data Model Based on Metagraph Approach. Lecture Notes in Computer Science, vol. 10742, 2018, pp. 72-87.

[8] Chernenkiy V.M., Gapanyuk Yu.E. et al. The Principles and the Conceptual Architecture of the Metagraph Storage System. Communications in Computer and Information Science, vol. 1003, 2018, pp. 73-87.

[9] Malewicz G., Austern M.H. et al. Pregel: A System for Large-scale Graph Processing. In Proc. of the 2010 ACM SIGMOD International Conference on Management of Data, 2010, pp. 135-146.

[10] Xin R.S., Crankshaw D. et al. GraphX: Unifying Data-Parallel and Graph-Parallel Analytics. arXiv preprint arXiv:1402.2394, 2014.
[11] Shaposhnik R., Martella C., Logothetis D. Practical Graph Analytics with Apache Giraph. Apress, 2015, 334 p.
[12] Introducing Gelly: Graph Processing with Apache Flink, URL:https://flink.apache.org/news/2015/08/24/introducing-flink-gelly.html, accessed 2021/04/09.
[13] Faloutsos M., Faloutsos P., Faloutsos C. On power-law relationships of the internet topology. ACM SIGCOMM Computer Communication Review, vol. 29, issue 4, 1999, pp. 251-262.
[14] Signal-Collect programming model, URL: https://uzh.github.io/signal-collect/documentation.html, accessed 2021/04/09.
[15] Han M., Daudjee K. et al. An Experimental Comparison of Pregel-like Graph Processing Systems. Proceedings of the VLDB Endowment, vol. 7, issue 12, 2014, pp. 1047-1058.
[16] Low Y., Gonzalez J. et al. Distributed GraphLab: A Framework for Machine Learning in the Cloud. arXiv preprint arXiv:1204.6078, 2012.
[17] Gonzalez J.E., Low Y. et al. PowerGraph: Distributed graph-parallel computation on natural graphs. In Proc. of the 10th USENIX conference on Operating Systems Design and Implementation, 2012, pp. 17-30.

Information about authors / Информация об авторах

Valeriy Mikhailovich CHERNENKIY – doctor of technical sciences, professor. Research interests: parallel-sequential process theory, imitation modelling, automated organization management system design.

Валерий Михайлович ЧЕРНЕНЬКИЙ – доктор технический наук, профессор. Сфера научных интересов: теория описания параллельно-последовательных процессов, имитационное моделирование информационных систем, проектирование автоматизированных систем организационного управления.

Ivan Vladimirovich DUNIN – postgraduate student. Research interests: graph processing, graph databases, distributed processing.

Иван Владимирович ДУНИН – аспирант. Сфера научных интересов: обработка графов, графовые базы данных, распределенная обработка.

Yuriy Evegenievich GAPANYUK – candidate of technical sciences, associate professor. Research interests: automated systems design, hybrid intelligent information systems, complex graph models.

Юрий Евгеньевич ГАПАНЮК – кандидат технических наук, доцент. Сфера научных интересов: проектирование автоматизированных систем, проектирование гибридных интеллектуальных информационных систем, сложные графовые модели.

DOI: 10.15514/ISPRAS-2022-34(1)-8

Анализ регулярности матриц

И.Б. Бурдонов, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>
А.А. Карнов, ORCID: 0000-0002-2066-9946 <karnov@ispras.ru>
Институт системного программирования РАН им. В.П. Иванникова,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В статье исследуется задача анализа регулярности многомерных матриц, основанной на повторении значимых (не пустых) символов в ячейках матрицы. Такое повторение означает, что при сдвиге матрицы по одной или нескольким её координатам некоторые значимые символы сохраняются. Для каждого сдвига, повторяющегося *r* раз, вводится число регулярности как произведение *rs*, где *s* – число значимых символов, сохраняющихся при всех *r* повторениях сдвига. Вводятся две числовые характеристики регулярности матрицы: сумма регулярности и коэффициент регулярности. Сумма регулярности определяется как сумма чисел регулярности при всех возможных сдвигах матрицы и позволяет сравнивать регулярность матриц одной формы, т.е. одной размерности и одного размера с одинаковым расположением непустых символов. Коэффициент регулярности позволяет сравнивать регулярность произвольных матриц и определяется как процентное отношение суммы регулярности матрицы к сумме регулярности «самой регулярной» матрицы (все значимые символы которой одинаковы) той же формы. Предложены алгоритмы вычисления суммы и коэффициента регулярности матрицы, которые были реализованы в компьютерных программах. В качестве прикладной области в статье используется анализ регулярной структуры стихотворений древнекитайского «Канона стихов» (*Ши цзин*). Стихотворение представляется четырёхмерной матрицей, её координаты – это строфа, строка в строфе, стих в строке и иероглиф в стихе; пустые символы выравнивают размеры стихов, строк и строф. В статье приводятся обобщающие результаты компьютерных экспериментов со всеми 305 стихотворениями *Ши цзина*.

Ключевые слова: многомерные матрицы; регулярность; повторение значимых символов; Канон стихов; *Ши цзин*; 詩經; параллелизм в стихах

Для цитирования: Бурдонов И.Б., Карнов А.А. Анализ регулярности матриц. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 101-122. DOI: 10.15514/ISPRAS–2022–34(1)–8

Matrix regularity analysis

I.B, Burdonov, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>
A.A. Karnov, ORCID: 0000-0002-2066-9946 <karnov@ispras.ru>
Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. The paper investigates the problem of analyzing the regularity of multidimensional matrices based on the repetition of significant (non-empty) characters in the matrix cells. Such a repetition means that when the matrix is shifted along one or more of its coordinates, some significant characters are preserved. For each shift repeated *r* times, the regularity number is entered as the product of *rs*, where *s* is the number of significant symbols that persist for all *r* repetitions of the shift. Two numerical characteristics of matrix regularity are introduced: the regularity sum and the regularity coefficient. The regularity sum is defined as the sum of the regularity numbers for all possible matrix shifts and allows you to compare the regularity of matrices of the same form, i.e. the same dimension and the same size with the same arrangement of non-empty characters. The regularity coefficient allows you to compare the regularity of arbitrary matrices and is defined as the percentage of the sum of the regularity of a matrix to the sum of the regularity of the «most regular» matrix (all significant

symbols of which are the same) of the same form. Algorithms for calculating the sum and regularity coefficient of a matrix are proposed and implemented in computer programs. As an applied area, the article uses the analysis of the regular structure of the poems of the ancient Chinese «Canon of Poems» (*Shih-ching*). The poem is represented by a four-dimensional matrix, its coordinates are a stanza, a line in a stanza, a verse in a line, and a hieroglyph in a verse; blank characters equalize the sizes of verses, lines and stanzas. The article presents generalizing results of computer experiments with all 305 poems of *Shih-ching*.

Keywords: multidimensional matrices; regularity; repetition of meaningful symbols; Classic of Poetry; *Shih-ching*; 詩經; parallelism in poems

For citation: Burdonov I.B., Karnov A.A. Matrix regularity analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 101-122 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–8

1. Введение

Статья посвящена исследованию регулярной структуры многомерных матриц, основанной на повторении символов, расположенных в ячейках матрицы. Кроме значимых символов, ячейки матрицы могут содержать пустые символы; размерность и размер матрицы вместе с расположением пустых символов определяют класс матриц одинаковой формы. Целью было определение числовой характеристики регулярности, которая позволяла бы сравнивать регулярность матриц одинаковой формы, а также числовой характеристики, позволяющей сравнивать регулярность произвольных матриц, в том числе, имеющих разную форму.

Эта задача возникла при структурном анализе стихотворений древнекитайского «Канона стихов» – *Ши цзин* 詩經 (другой, распространённый, но не точный, перевод – «Книга Песен»). Известно, что в этих стихотворениях помимо рифм (все стихи *Ши Цзина* рифмованные) много параллелизмов, в том числе построенных на повторении иероглифов и групп иероглифов, которое мы будем называть регулярностью. Задача заключается в выявлении регулярной структуры стихотворения, введении числовой меры регулярности и сравнении разных стихотворений по этой мере. Регулярность такого рода в стихах *Ши цзина* (и не только в них) эстетически значима, подобно рифмовке, поскольку создаёт определённый ритмический рисунок стихотворения. Это особенно верно для китайских стихов, которые представляют собой не только звуковую (стихи слушают), но и графическую (стихи смотрят) картину, что особенно ярко проявляется в каллиграфии.

Более того, структурный анализ древнекитайских (и средневековых) китайских текстов, особенно, канонов (*цзин* 經), со 2-й половины XX в. составил содержание целого направления в российской (и не только) синологии: «структурная или структурно-семантическая методология в изучении китайской классики, тесно связанная с поисками аутентичной методологии у самих китайских классиков» [1]. «Чрезвычайно возросший среди китайских ученых с началом идеологической перестройки в конце 1970-х – начале 1980-х интерес к методологическим аспектам собственной классической философии вдохновил и опередивших их российских исследователей, среди которых изучение этой проблематики на рубеже 1950-х – 1960-х было начато В.С. Спириным (1929-2002), в середине 1970-х развито А.М. Карапетьянцем и А.И. Кобзевым, с середины 1980-х продолжено В.Е. Еремеевым, С.В. Зининым, М.В. Исаевой, В.В. Лихтман/Дорофеевой, А.А. Крушинским и др.» [там же].

Данное направление берёт на вооружение формальный подход к тексту, отвлекаясь от его содержания: без этого, по Спирину, текстология оказывается не столько наукой, сколько разновидностью эссеистики, в которой «главную роль играют интуиция и эрудиция, а иногда и просто личный авторитет и другие случайные обстоятельства» [2]. «Обращение к форме как к первому и важнейшему фактору понимания содержания является одной из основных особенностей структурного анализа» [там же]. Задачей такого формального подхода в первую очередь становится «изучение параллелизмов в древнекитайских текстах» [там же]. Спирин вводит понятие «универсального параллелизма», в котором, с одной стороны, 1) достаточно широко трактуется совпадение параллельных мест текста: они не обязательно

тождественны, но «в чём-то тождественны», а с другой стороны, 2) сам текст понимается не обязательно (и не столько) как линейная последовательность символов, а, скорее, как многомерная структура.

Первое ведёт к введению тех или иных математических отношений эквивалентности на отрезках текста, а второе – к математическому понятию многомерной матрицы. В данной работе мы ограничиваемся вторым, а эквивалентность будем понимать предельно узко – как равенство, совпадение. С учётом этого наше понятие регулярности, фактически, совпадает с понятием универсального параллелизма у Спирина.

Выбор в качестве предметной области именно древнекитайской поэзии объясняется ещё и особенностями китайской иероглифической письменности, в которой иероглифы представляет собой удобные «атомы», из которых строятся «молекулы» стихотворений и вообще текстов. В китайском языке (особенно древнем) слову, как правило, соответствует один иероглиф, который не меняется при изменении рода и числа, не склоняется по падежам, что значительно облегчает анализ регулярностей, построенных на повторении элементов текста. И хотя сами параллелизмы (в широком смысле) и основанная на них ритмика характерны для многих древнекитайских текстов (особенно, канонов), именно в стихах это особенно значимо с эстетической точки зрения.

Структурный анализ *Ши цзина* проводился ранее либо на макроуровне канона в целом, без исследования внутренней структуры самих стихотворений[3], [4], [5], [6], [7], [8], [9], [10], либо как анализ отдельных стихотворений [11], [12], [13]. Можно указать и другие работы по структурному анализу текста, не только *Ши цзина*:[11], [12], [13], [14], [15]. В нашей работе впервые проводится систематический анализ внутренней формальной структуры всех стихотворений *Ши цзина*, основанной (это нужно ещё раз подчеркнуть) на буквальном повторении некоторых иероглифов или групп иероглифов в пределах стихотворения.

В данной статье задача анализа регулярности многомерности матриц ставится и решается в общем виде, а иллюстрируется анализом регулярности стихотворений *Ши цзина*, полученным с помощью компьютерных экспериментов. Разд. 2 вводит представление китайских стихотворений как четырёхмерных матриц, а также для наглядности в виде двумерных матриц на плоскости. В разд. 3 определяются основные используемые далее понятия вектора, матрицы и повторяющейся фигуры в матрице. В разд. 4 устанавливается связь повторения фигур матрицы со сдвигами матрицы по одной или нескольким координатам, и на этой основе вводится число регулярности повторяющейся фигуры. В разд. 5 вводится сумма регулярности матрицы как числовая характеристика, пригодная для сравнения регулярности матриц одной формы, и предлагается алгоритм её вычисления. В разд. 6 решается проблема сравнения регулярности матриц разной формы с помощью введения коэффициента регулярности как процентного отношения суммы регулярности матрицы к сумме регулярности «самой регулярной» матрицы той же формы, и предлагается алгоритм вычисления коэффициента регулярности. В разд. 7 предлагается наглядный способ представления регулярной структуры двумерного представления четырёхмерной матрицы как разбиение на простые многоугольники повторяющихся фигур. Разд. 8 содержит обобщающие результаты, полученные в компьютерных экспериментах на всех 305 стихотворениях *Ши цзина*. Разд. 9 затрагивает проблему параллелизма стихотворений *Ши цзина* при их переводе на другие языки. В Заключении подводятся итоги исследования и намечаются направления дальнейших исследований.

2. Представление стихотворения Ши цзина в виде матрицы

Стихотворение *Ши цзина* состоит из строф, строфа состоит их групп стихов– *n*-стиший (в основном, двустиший), стих состоит из некоторого числа (в основном четыре) иероглифов. Традиционно китайский текст записывался по столбцам сверху вниз и справа налево без знаков препинания. В современной записи иероглифы записываются по горизонтали слева направо, строфы разделяются пустой строкой, *n*-стишие соответствует строке (разделяются

переводом строки), стихи в строке разделяются знаками препинания (пробелы, запятые, точки и т.п.). Такое стихотворение мы будем представлять в виде четырёхмерной матрицы, в ячейке которой находится иероглиф, четыре координаты соответствуют строфам, строкам в строфах, стихам в строках и иероглифам в стихах. Знаки препинания опускаются. Поскольку стихи могут состоять из разного числа иероглифов, строки – из разного числа стихов, а строфы – из разного числа строк, будем подравнивать их по самым длинным стихам, строкам и строфам, соответственно. Для этого в конце короткого стиха добавляются пустые символы, в конце коротких строк – пустые стихи (состоящие из пустых символов), в конце коротких строф с пустые строки (состоящие из пустых стихов).

В примерах при изображении на плоскости будем использовать двумерную матрицу, в которой строка матрицы соответствует строке стихотворения, а столбец – позиции иероглифа в стихе и стиха в строке. Ячейки, добавленные для выравнивания стихов, строк и строф, перечёркиваются двумя диагоналями. Разделители стихов и строф – двойные линии. Такое представление четырёхмерной матрицы на плоскости предназначено только для наглядности примеров и не используется в предлагаемом формализме и алгоритме. Эти алгоритмы применимы к матрицам любых размерностей, но результаты для четырёхмерной матрицы и её двумерного представления будут разными.

Из такого представления стихотворения традиционная запись получается транспонированием матрицы, затем отражением слева направо и затем удалением пустых ячеек со сдвигом вверх. На рис. 1 дан пример стихотворения в этих трёх видах записи: современная, в виде матрицы и традиционная. На этом рисунке видны основные виды регулярности: повторяющиеся иероглифы, несколько одинаковых иероглифов в строке или в столбце, а также другие повторяющиеся «фигуры» иероглифов.

Рис. 1. Стихотворение № 5 (1.1.5) 螽斯 чжун сы – Саранча
Fig. 1. Poem No. 5 (1.1.5) 螽斯 Jung si - Locust

Абстрагируясь от иероглифов и китайских стихотворений, задача формулируется как анализ регулярности в t-мерной матрице, где $t \geq 1$,введении числовой меры регулярности и сравнении разных матриц по этой мере. Мы будем рассматривать регулярности, основанных на повторении «фигуры», когда имеется два одинаковых набора символов с тем же относительным расположением друг относительно друга. Одна фигура получается из другой сдвигом по одной или нескольким координатам матрицы.

3. Вектор, матрица и фигура

Рассматриваются вектора $x = (x_1, x_2, \ldots, x_k)$ размерности $k = 1, 2, \ldots$, содержащие целые числа; i-ую компоненту вектора x будем обозначать $x(i) = x_i$. Для векторов одной размерности k определим:

Покомпонентную сумму векторов $(x + y)(i) = x(i) + y(i)$.

Покомпонентную разность векторов $(x - y)(i) = x(i) - y(i)$.

Противоположный вектор $(-x)(i) = -x(i)$.

Модуль вектора $|x|(i) = |x(i)|$.

Вектор $[n]$, все компоненты которого равны числу n: $[n](i) = n$. Если нужно указать размерность t вектора $[n]$, будем писать $[n]_t$.

Покомпонентное отношение частичного порядка:

$y > x \Leftrightarrow x < y \Leftrightarrow \forall\ i = 1, \ldots, k\ (\ x(i) < y(i)\ );\ y \geq x \Leftrightarrow x \leq y \Leftrightarrow x = y \vee x < y.$

Линейный порядок векторов:

$x <_l y \Leftrightarrow \exists\ i\ (\ 1 \leq i \leq k\ \&\ x(i) < y(i)\ \&\ (\ \forall\ j\ (\ 1 \leq j < i \Rightarrow x(j) = y(j)\ )\ )\ );$

$y \geq_l x \Leftrightarrow x \leq_l y \Leftrightarrow x = y \vee x <_l y.$

Очевидно, $-x = [0] - x$ и $x - y = x + (-y)$.

Также рассматриваются матрицы размерности $t \geq 1$ в алфавите L, в каждой ячейке которых находится символ из алфавита L, причём некоторые символы могут повторяться. Размер матрицы задаётся вектором n размерности t, где $n(i) \geq 1$, $i = 1, \ldots, t$.

Координаты ячейки матрицы задаются вектором x размерности t, где $1 \leq x(i) \leq n(i)$, $i = 1, \ldots, t$. В ячейке с координатами x матрицы A находится символ $A(x)$. Матрицу будем просматривать в линейном порядке «$<_l$» векторов, задающих координаты ячеек. Для одномерной матрицы (вектора) это будет просмотр слева направо. Для двумерной матрицы это будет просмотр слева направо сверху вниз.

Пустой символ – выделенный символ ε алфавита L. *Пустой ячейкой* будем называть ячейку, в которой записан пустой символ. В примерах пустая ячейка – это ячейка с белым фоном и отсутствием символа в ней.

Маской будем называть матрицу M размерности t и размера m, в которой каждая ячейка содержит либо пустой символ ε, либо специальный *прозрачный* символ τ, с дополнительным требованием наличия символов τ на «периферии» матрицы M: для $i = 1, \ldots, t$ хотя бы в одной ячейке матрицы M с координатами x, где $x(i) = 0$ или $x(i) = m(i)$, имеется символ τ, т.е. $M(x) = \tau$. Для одномерной матрицы (вектора) это означает наличие символа τ в первой и последней компоненте. Для двумерной матрицы это означает наличие символа τ в первой и последней строках, а также в первом и последнем столбце.

Будем говорить, что фигура F размерности t и размера m определяется в матрице A размерности t и размера n координатами z (в матрице A) по маске M размерности t и размером m, где $z + m - [1] \leq n$, если выполняется условие $(\ M(x) = \tau\ \&\ F(x) \neq \varepsilon\ \&\ F(x) = A(x + z - [1])\ ) \vee (\ M(x) = \varepsilon\ \&\ F(x) = \varepsilon\ )$, и, кроме того, хотя бы одна ячейка $F(x)$ не пуста. Фигура F является матрицей размерности t и размера m, координаты z определяют «начало» фигуры F в матрице A, прозрачные символы маски M – те ячейки, которые должны быть непустыми в матрице и которые становятся ячейками фигуры F, а пустые символы маски M – те ячейки, которые становятся пустыми в F независимо от того, какими они были в матрице. Непустые символы фигуры составляют «содержание» фигуры и располагаются в t-мерном пространстве, а пустые символы заполняют «пропуски» между непустыми символами, если такие пропуски есть. Такую фигуру F будем обозначать $A(z, M)$.

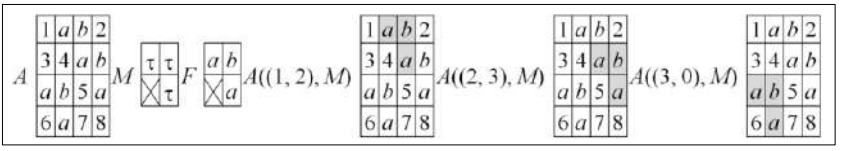

Рис. 2. Пример повторяющейся фигуры
Fig. 2. Example of a repeating figure

Повторяющейся фигурой назовём фигуру F, имеющую более одного вхождения в матрицу A, т.е. существуют маска M и хотя бы две координаты z_1 и z_2 такие, что $F = A(z_1, M) = A(z_2, M)$. На рис. 2 пример двумерной матрицы A, маски M, фигуры F и места фигуры в матрице с указанием координат вхождений.

4. Сдвиг матрицы и число регулярности фигуры

Сдвигом матрицы A размерности t и размера n будем называть её преобразование, задаваемое вектором d размерности t, результатом которого является матрица $d(A)$ размерности t и размера n, определяемая равенствами $d(A)(x) = A(x)$, если $[1] \le x - d \le n$ и $A(x) = A(x - d)$, и $d(A)(x) = \varepsilon$ в противном случае. Иными словами, каждая ячейка матрицы с координатами x сдвигается на вектор d, т.е. по каждой координате $i = 1, \ldots, t$ сдвигается на $d(i)$ ячеек: i-ая компонента координаты $x(i)$ увеличивается на $d(i)$ (уменьшается на $|d(i)|$, если $d(i) < 0$). Если ячейка не выходит за пределы матрицы и $A(x) = A(x + d)$, то после сдвига в ячейке с координатами $x + d$ будет символ $A(x + d)$, который там и был в A, в противном случае в ячейке будет пустой символ. Ячейки, выходящие за пределы матрицы, исчезают, а освободившиеся ячейки (в которые не перемещаются другие ячейки) заполняются пустым символом. Также пустыми будут ячейки с координатами $x + d$, если $A(x) \ne A(x + d)$. Примеры на рис. 3.

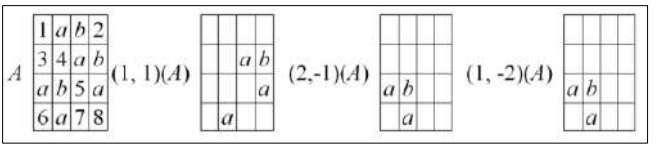

Рис. 3.Примеры сдвига двумерной матрицы
Fig. 3. Examples of two-dimensional matrix shift

Выбираем сдвиги, которые хотя бы одну ячейку матрицы не выводят за её пределы. После этого из двух противоположно направленных сдвигов d и $-d$ выбираем один: тот, который дальше в линейном порядке сдвигов, т.е. d, если $-d <_l d$, и $-d$, если $d <_l -d$. Нулевой сдвиг [0], не меняющий любую матрицу, не выбираем. Множество всех *выбранных* сдвигов обозначим D. Их число, очевидно, равно $((2n(1)-1)*(2n(2)-1)*\ldots*(2n(t)-1) - 1) / 2$. Имеем:

$-(n(i) - 1) \le d(i) \le (n(i) - 1)$ для $i = 1, \ldots, t$;

$d_1 \ge 0$;

если $d_1 = 0$, то $d_2 \ge 0$;

если $d_1 = d_2 = 0$, то $d_3 \ge 0$;

…

если $d_1 = d_2 = \ldots = d_{t-1} = 0$, то $d_t \ge 0$.

Утверждение 1. Если в матрице A размерности t и размера n есть два вхождения одной фигуры F с координатами z_1 и z_2 по маске M размерности t и размера m, т.е. $F = A(z_1, M) = A(z_2, M)$, и $z_1 <_l z_2$, то второе вхождение может быть получено из первого выбранным сдвигом матрицы на величину $(z_2 - z_1)$ и затем выделением фигуры с координатами z_2 по маске M: $(z_2 - z_1) \in D$ и $F = ((z_2 - z_1)(A))(z_2, M)$.

Доказательство: Из того, что z_1 и z_2 координаты в A и условия $z_1 <_l z_2$ непосредственно следует, что $(z_2 - z_1) \in D$. По определению сдвига $((z_2 - z_1)(A))(x + z_2 - [1]) = A(x + z_2 - [1] - (z_2 - z_1)) = A(x + z_1 - [1])$ при условии, что 1) $[1] \le x + z_1 - [1] \le n$, где x координаты в маске M, т.е. $[1] \le x \le m$, и 2) $A(x + z_2 - [1]) = A(x + z_1 - [1])$. Из существования вхождения $A(z_1, M)$ следует выполнение условия 1, а из равенства $A(z_1, M) = A(z_2, M)$ следует выполнение условия 2. Утверждение доказано. □

Опираясь на 0, будем говорить, что фигура F *повторяющаяся при сдвиге* d в матрице A, если она есть в сдвинутой матрице $d(A)$ и, следовательно, есть в A. Фигура F может многократно повторяться при последовательных сдвигах d, т.е. иметь вхождение в каждую из матриц, получающихся последовательным применением сдвига d, т.е. в матрицах $d(A)$, $d(d(A)) = d^2(A)$, … $d(d(\ldots(d(A))\ldots)) = d^r(A)$. Максимальное число $r > 1$ будем называть *числом вхождений* фигуры F в матрицу A *при сдвиге* d, т.е. фигура F входит в каждую матрицу $d^i(A)$, где $i = 1, 2, \ldots, r$, но не входит в матрицу $d^{i+1}(A)$.

Числом регулярности фигуры F по сдвигу d в матрице A назовём произведение sr, где $s \geq 1$ число непустых ячеек в F, а $r > 1$ число вхождений фигуры F в матрицу A при сдвиге d. На рис. 4 пример двумерной матрицы A и двух матриц $d(A)$ и $d^2(A)$ для сдвига $d = (1, 1)$ с указанием местоположения в них двух фигур с числом регулярности $3 = 3*1$ и $2 = 1*2$. Заметим, что в матрице $d(A)$ показано местоположение фигуры из трёх непустых ячеек, которая не максимальна в том смысле, что, добавив ещё одну ячейку с символом a, можно получить фигуру с четырьмя непустыми ячейками.

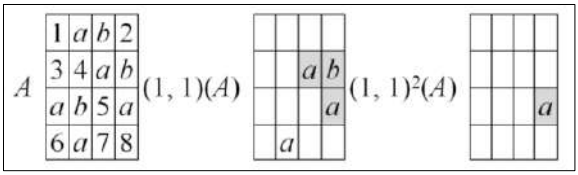

Рис. 4.Примеры фигур, повторяющихся при сдвигах
Fig. 4. Examples of figures repeated during shifts

Повторяющуюся фигуру в матрице A будем называть *максимальной* при сдвиге d, если она имеет максимальное число непустых ячеек среди фигур, повторяющихся при сдвиге d. В примере на рис. 4 при однократном сдвиге (1, 1) максимальная фигура образуется из всех четырёх непустых ячеек матрицы $(1, 1)(A)$.

Поскольку фигура также является матрицей, можно определить вложенность фигур следующим образом: будем говорить, что фигура B *вложена* в фигуру A, если B является фигурой в матрице A.

Утверждение 2. 1) Фигура F максимальная в матрице A при сдвиге d получается из сдвинутой матрицы $d(A)$ удалением пустой «периферии», т.е. повторением, пока возможно, следующей процедуры: если для некоторого $i = 1, \ldots, t$ все ячейки матрицы $d(A)$ с координатами x, где $x(i) = 0$, пусты, то все они удаляются из матрицы. 2) Каждая фигура в матрице A, повторяющаяся при сдвиге d, вложена в фигуру F. 3) Число непустых ячеек фигуры F равно числу непустых ячеек матрицы $d(A)$.

Доказательство. Если ячейка с координатами x матрицы $d(A)$ не пуста, то, во-первых, в этой ячейке в матрице A находится тот же символ, а, во-вторых, этот же символ в матрице A находится в ячейке с координатами $x - d$. Поскольку любая фигура матрицы A, повторяющаяся при сдвиге d, состоит из таких ячеек, отсюда непосредственно следует доказываемое утверждение. □

5. Сумма регулярности матрицы и алгоритм её вычисления

Суммой регулярности матрицы A будем называть сумму $S(A)$ чисел регулярности всех её максимальных фигур, повторяющихся при сдвигах, по всем сдвигам. Из 0 следует, что сумма регулярности матрицы A равна сумме произведений sr по всем d и r, где d выбранный сдвиг, r кратность сдвига d, т.е. число применений одного и того же сдвига d для получения сдвинутой матрицы $d^r(A)$, а s число непустых ячеек в сдвинутой матрице $d^r(A)$. Ниже приведён алгоритм вычисления суммы регулярности матрицы и необходимые вспомогательные процедуры.

Алгоритм_1 (A, t, n) /* Вычисление суммы регулярности матрицы */
Input: матрица A размерности t размера n.
Output: сумма регулярности $S(A)$.

```
S = 0;                          /* нулевая сумма регулярности */
d = [0]t;                       /* нулевой сдвиг */
while d<n - [1]t do             /* цикл по сдвигам */
|   d = Nextd(d, t, n);              /* вычисление следующего сдвига */
|   (B, s) = Сдвиг(A, t, n, d);      /* сдвиг матрицы */
```

```
    r = 1;                                  /* кратность сдвига равна 1 */
    while s > 0 do                          /* цикл повторения одного сдвига */
        S = s*r;                            /* вычисление числа регулярности */
        (B, s) = Сдвиг (B, t, n, d);        /* дальнейший сдвиг матрицы */
        r++;                                /* увеличение на 1 кратности сдвига */
    return S;
Сдвиг (C, t, n, d)      /* сдвиг матрицы */
    Input: матрица C размерности t размера n, сдвиг d.
    Output: сдвинутая матрица D = d(C) и число s непустых ячеек в ней.
    x = [1]t;                               /* координаты начальной ячейки матрицы */
    s = 0;                                  /* начальное число s непустых ячеек равно нулю */
    while x <l n do                         /* цикл по координатам ячеек матрицы */
        if [1]t ≤ x - d ≤ n then            /* если в ячейку сдвигается ячейка матрицы */
            if C(x) = C(x-d) then           /* если при сдвиге символ не меняется */
                D(x) = C(x);                /* то этот символ – в ячейке сдвинутой матрицы */
                if D(x) ≠ ε then            /* если это не пустой символ */
                    s++;                    /* то число s непустых ячеек увеличивается на 1 */
            else                            /* если при сдвиге символ меняется */
                D(x) = ε;                   /* то в ячейке пустой символ */
        else                                /* если в ячейка освобождается */
            D(x) = ε;                       /* то в ячейке пустой символ */
        x = Nextx(x, t, n);                 /* вычисление координат следующей ячейки */
    return (D, s);                          /* когда переберём все ячейки, конец вычисления */
Nextd(d, t, n)      /* вычисление следующего сдвига */
    Input: сдвиг d для матрицы размерности t размера n, [0]t ≤l d <l n - [1]t.
    Output: выбранный сдвиг, следующий за d в линейном порядке сдвигов.
    i = t;                                  /* начинаем с последней t-й координаты */
    while do                                /* цикл по координатам i = t, t - 1, ..., 1 */
        d(i)++;                             /* увеличение на 1 сдвига по текущей i-й координате */
        if d(i) > n(i) then                 /* если выходим за границы матрицы */
            d(i) = -(n(i) - 1);             /* то минимальный сдвиг по i-й координате */
            i--;                            /* и переходим к предыдущей (i - 1)-й координате */
        else                                /* если не выходим за границы матрицы */
            return d;                       /* то конец вычисления */
Nextx(x, t, n)      /* вычисление координат следующей ячейки */
    Input: координаты x для матрицы размерности t размера n, [1]t ≤l x <l n.
    Output: координаты, следующие за x в линейном порядке координат.
    i = t;                                  /* начинаем с последней t-й координаты */
    whiledo                                 /* цикл по координатам i = t, t - 1, ..., 1 */
        x(i)++;                             /* увеличение на 1 текущей i-й координаты */
        if x(i) > n(i) then                 /* если выходим за границы матрицы */
            x(i) = 1;                       /* то начальное значение текущей i-й координаты */
            i--;                            /* и переходим к предыдущей (i - 1)-й координате */
        else                                /* если не выходим за границы матрицы */
            return x;                       /* то конец вычисления */
```

Утверждение 3. Алгоритм вычисления суммы регулярности матрицы имеет сложность $\boldsymbol{O}(N^2 n_{max})$, где N число ячеек матрицы, а $n_{max} = max\{n(1), \ldots, n(t)\}$ максимальный размер матрицы по одной из координат.

Доказательство. Число ячеек матрицы $N = n(1) * \ldots * n(t)$. Число выбранных сдвигов d равно $((2n(1)-1)*\ldots*(2n(t)-1) - 1) / 2 = \boldsymbol{O}(N)$. Кратность r сдвига не превышает $n_{max} - 1 = \boldsymbol{O}(n_{max})$. Для каждого сдвига d и его кратности r вычисление сдвинутой матрицы $d^r(A)$ и вычисление числа непустых её ячеек делается за один просмотр ячеек матрицы, т.е. за время $\boldsymbol{O}(N)$. Таким образом, сложность алгоритма $\boldsymbol{O}(N^2 n_{max})$. □

Замечание: Число сдвигов вида d^r, где d выбранный сдвиг, оставляющих в пределах матрицы хотя бы одну ячейку, может быть меньше $n_{max} - 1$, поскольку кратность $n_{max} - 1$ имеет только такой сдвиг d, в котором $|d(i)| = 1$, если $n_i = n_{max}$, и $d(i) = 0$, если $n_i < n_{max}$, $i = 1, \ldots, t$.

6. Коэффициент регулярности матрицы

Сумма регулярности матрицы позволяет корректно сравнивать матрицы одной формы, т.е. имеющих одинаковые размерность, размер и расположение пустых символов. В частности, если выполнить *расширение матрицы*, увеличив максимальные значения некоторых координат $n(i)$, $i = 1, \ldots, t$ (в двумерной матрице это значит добавить строки после последней строки и/или столбцы после последнего столбца), и заполнить новые ячейки пустыми и/или уникальными символами, то сумма регулярности не изменится.

Для сравнения регулярности матриц разного размера и/или с разным числом и/или расположением пустых символов, введём *коэффициент регулярности матрицы* $K(A)$, который определим как процентное отношение суммы регулярности матрицы A к сумме регулярности *самой регулярной* матрицы $R(A)$ того же размера и с тем же числом и расположением пустых символов, что матрица A. Такой самой регулярной матрицей естественно считать матрицу с наибольшей суммой регулярности. Такая наибольшая сумма регулярности будет у матрицы, получаемой из исходной матрицы A заменой всех непустых символов на один и тот же непустой символ. Заметим, что если матрицу A расширить до матрицы B, добавив хотя бы одну непустую ячейку, то сумма регулярности самой регулярной матрицы вырастет: $S(R(B)) > S(R(A))$. Коэффициент регулярности определяется как $K(A) = 100 * S(A) / S(R(A))$.

В качестве примера рассмотрим семь матриц одного размера 3 × 3 на рис. 5, где матрица G самая регулярная: $G = R(A) = \ldots = R(F)$.

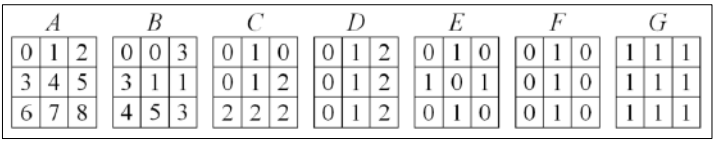

Рис. 5. Пример матриц для вычисления коэффициента регулярности
Fig. 5. Example of matrices for calculating the regularity coefficient

Табл. 1. Подсчёт суммы регулярности и коэффициента регулярности матриц
Table 1. Calculation of the sum of regularity and the coefficient of regularity of matrices

матрица	r	сдвиг d												S	K%
		(0, 1)	(0, 2)	(1, 2)	(1, -1)	(1, 0)	(1, 1)	(1, 2)	(2, -2)	(2, -1)	(2, 0)	(2, 1)	(2, 2)		
A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2														
B	1	2	0	1	0	0	0	1	0	0	1	0	0	5	10
	2	0													
C	1	2	2	2	1	3	0	0	0	0	0	0	0	12	23
	2	1			0	0									
D	1	0	0	0	0	6	0	0	0	0	3	0	0	15	29
	2					3									
E	1	0	3	0	4	0	4	0	1	0	3	0	1	20	38
	2				1		1								
F	1	0	3	2	0	6	0	2	1	0	3	0	1	24	46
	2					3									

G	1	6	3	2	4	6	4	2	1	2	3	2	1	52	100
	2	3			1	3	1								

В табл. 1 приведён подсчёт суммы регулярности и коэффициента регулярности всех этих матриц. Обозначения в табл. 1: r – кратность сдвига, S – сумма регулярности, K – коэффициент регулярности. В ячейке таблицы указано число s непустых ячеек в сдвинутой матрице $d^r(X)$, $X = A, \ldots, G$. Ячейка с серым фоном соответствует сдвигу d^r, выводящему все ячейки за пределы матрицы. Ячейка с заштрихованным фоном соответствует сдвигу d^r, оставляющему некоторые ячейки в пределах матрицы, но такому, что в матрице $d^{r-1}(X)$ и, следовательно, в матрице $d^r(X)$, все ячейки пустые.

Алгоритм_2 (*A, t, n*)/* Вычисление коэффициента регулярности матрицы */

```
Input: матрица A размерности t размера n.
Output: коэффициент регулярности K(A) = 100S(A) / S(R(A)).
S = Алгоритм_1 (A, t, n);      /* сумма регулярности матрицы A*/
                                 /* вычисление самой регулярной матрицы R(A)*/
x = [1]t;                        /* координаты начальной ячейки матрицы */
while x<In do                    /* цикл по координатам ячеек матрицы */
|   if A(x) ≠ε then              /* если в A ячейка не пуста */
|   |   R(x) = 1                 /* то пишем в ячейку 1 */
|   else                        /* если в A ячейка пуста */
|   |   R(x) = ε;                /* то пишем в ячейку пустой символ */
|   x = Nextx(x, t, n);          /* вычисление координат следующей ячейки */
SR = Алгоритм_1 (R, t, n);     /* сумма регулярности матрицы R(A)*/
return ( 100 * S / SR );
```

7. Разбиение двумерного представления четырёхмерной матрицы на простые многоугольники повторяющихся фигур

Сумма регулярности и коэффициент регулярности являются обобщёнными числовыми характеристиками регулярности матрицы. Полное описание должно было бы содержать перечисление максимальных фигур, повторяющихся при тех или иных выбранных сдвигах d с указанием числа кратности r сдвига, т.е. троек (сдвиг d, кратность r, максимальная фигура). Однако такое описание не наглядно.

Для наглядного изображения на плоскости повторяющихся фигур четырёхмерной матрицы стихотворения можно использовать описанное в разделе 2 двумерное представление, в котором выделим повторяющиеся фигуры следующим образом. Для каждой ячейки, содержащей иероглиф, определим множество выбранных сдвигов, при которых этот иероглиф сохраняется. Далее удалим границу между каждой парой соседних ячеек (левая-правая или верхняя-нижняя), если эти ячейки 1) содержат иероглифы с одинаковыми множествами сдвигов (в частности, оба иероглифа неповторяющиеся, т.е. с пустыми множествами выбранных сдвигов) или 2) содержат пустые символы.

В результате мы получим разбиение двумерной матрицы на плоские фигуры, ограниченными простыми многоугольниками. Эти фигуры могут быть трёх типов: 1) в ячейках фигуры находятся повторяющиеся (в матрице) иероглифы с одинаковыми множествами сдвигов, 2) в ячейках фигуры находятся уникальные иероглифы, 3) в ячейках фигуры находятся пустые символы. После этого присвоим фигурам первого типа (т.е. содержащим повторяющиеся в матрице иероглифы), номера таким образом, чтобы фигуры с одинаковыми множествами выбранных сдвигов получили одинаковый номер. Очевидно, каждая фигура первого типа содержит попарно различные иероглифы, которые повторяются во всех фигурах с тем же номером и только в них.

Наконец, заменим каждый повторяющийся в матрице иероглиф в ячейке на номер фигуры, в которую входит эта ячейка. Границу стихов и строф по-прежнему будем показывать двойной линией там, где эта граница совпадает с границей многоугольников, и пунктирной линией там, где не совпадает, т.е. проходит внутри многоугольника. На рис. 6 показан пример матрицы стихотворения и её разбиения. В этом примере все иероглифы повторяющиеся, поэтому все они заменены номерами соответствующих фигур в разбиении. В других случаях, в частности, рассматриваемых ниже, это не обязательно так.

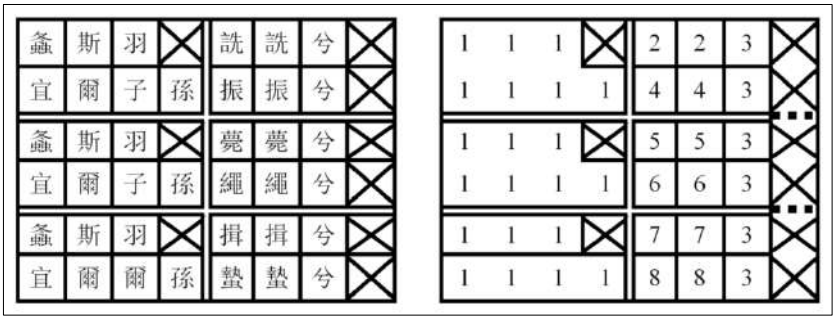

Рис. 6. Стихотворение № 5 (1.1.5): матрица и её разбиение
Fig. 6. Poem No. 5 (1.1.5): the matrix and its partition

8. Результаты компьютерных экспериментов

Для компьютерных экспериментов была составлена программа, реализующая описанные выше алгоритмы. На её вход подавался текст *Ши цзина*, взятый с сайта «Chinese Text Project» [19] в кодировке utf-8 и преобразованный в следующий формат в виде регулярного выражения (не совсем формально, но в интуитивно очевидной форме).

Литералы изображаются жирным шрифтом (иероглифы шрифтом SimSun), *метасимволы* изображаются курсивом (TimesNewRoman). Пробелы и переводы строк используются только для наглядности представления и не являются символами текста. Кроме иероглифов, файл может содержать следующие литералы: **0 1 2 3 4 5 6 7 8 9 ( ) = . ABCDE.**

Файл_с_текстом_Ши_цзин= Стихотворение_1...Стихотворение_305 **E**

Стихотворение_N= Заголовок_стихотворения_N Строфа...Строфа ***D***

Заголовок_стихотворения_N= ***(i , j , k) =*** *N* ***.*** *Z****.***

где:

i – номер раздела: от 1 до 4,

j – номер подраздела в разделе: от 1 до 15,

k – номер стихотворения в подразделе: от 1 до 21,

N – глобальный *номер* стихотворения: от 1 до 305,

номер – натуральное число в десятеричной системе счисления:

номер = цифра...цифра, цифра = **0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9,**

Z – текст названия стихотворения, содержащий только иероглифы:

Z = Иероглиф... Иероглиф.

Пример заголовка стихотворения: **(1,1,1)=1.關雎.**

Строфа= Строка... Строка **C**

Строка=Стих...Стих **B**

Стих= Иероглиф...Иероглиф **A**

Пример строфы:

(1.1.3)=3.卷耳.采采卷耳A不盈頃筐AB嗟我懷人A寘彼周行ABC陟彼崔嵬A我馬虺隤AB我姑酌彼金罍A維以不永懷ABC陟彼高岡A我馬玄黃AB我姑酌彼兕觥A維以不永傷ABC陟彼砠矣A我馬瘏矣A我僕痛矣A云何吁矣ABCD

Файл заканчивается символами **ABCDE**.

С помощью компьютерной программы были посчитаны суммы регулярности и коэффициенты регулярности всех 305 стихотворений *Ши цзина*. В этом разделе дан предварительный анализ полученных результатов, в том числе, в сравнении с оценками параллелизма в стихах *Ши цзина* по трём статьям китайских авторов[20], [21], [22].

Ши цзин подразделяется на 4 раздела: 1) *Гофэн*國風 (Нравы царств), 2) *Сяо я* 小雅 (Малые оды), 3) *Да я* 大雅 (Великие оды) и 4) *Сун* 頌 (Гимны). На рис. 7 (внизу) показан график среднего значения коэффициента регулярности K_{xyzt} (с учётом всех сдвигов) по этим разделам *Ши цзина*, а также его среднее значение 2,71 по всем стихотворениям *Ши цзина*. Он наглядно показывает, как ожидаемо падает регулярность стихотворений от первого к последнему разделам, что обратно росту пафоса этих стихов от народных песен до храмовых песнопений.

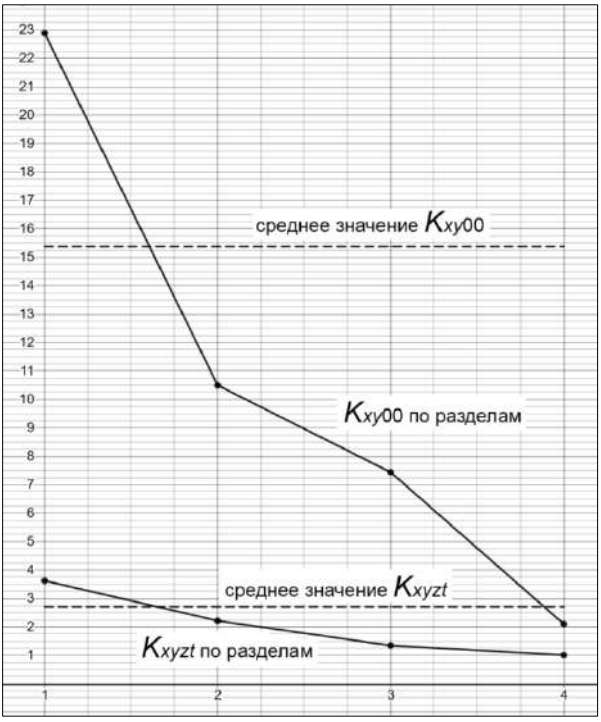

Рис. 7. График коэффициентов регулярности как функция от раздела Ши цзина.
Fig. 7. Graph of regularity coefficients as a function of the Shih-ching section.

В табл. 2 приведены значения коэффициента регулярности (округлённого до второго знака после запятой) для всех 305 стихотворений *Ши цзина*. На рис. 8 (внизу) это же показано в виде графика, где через K_{xyzt} обозначен коэффициент регулярности с учётом сдвигов матрицы стихотворения по всем четырём координатам (строфа, строка в строфе, стих в строке, иероглиф в стихе).

Обратим внимание на пик значения K_{xyzt} = 14,18, который достигается для стихотворения № 8. Это стихотворение упоминается в [22] как экстремально регулярное среди перечисляемых там стихотворений № 4, 6, 7 и 8. Табл. 2 и нижний график на рис. 8 показывают, что оно экстремально среди всех стихотворений *Ши цзина*.

Табл. 2. Коэффициенты регулярности 305 стихотворений Ши цзин
Table 2. Regularity coefficients of 305 Shih-ching poems

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		2,51	1,06	2,60	6,19	7,26	6,74	5,64	14,18	4,18	1,10	10,19	8,31	4,55	3,44	3,29	7,19	2,65	5,64	4,86
20	5,37	1,93	6,02	1,49	2,59	2,83	1,21	4,77	2,50	2,95	1,85	1,06	1,33	0,90	1,12	0,95	6,03	2,97	1,26	0,98
40	2,82	4,08	1,28	2,90	3,62	2,23	8,04	2,64	5,34	6,34	0,62	1,16	6,43	6,60	1,28	3,61	4,70	0,60	0,66	1,08
60	2,98	3,44	1,16	6,90	7,41	4,07	2,40	2,35	4,93	5,50	4,49	4,73	7,96	1,96	5,60	7,42	3,76	4,62	2,61	2,08
80	3,29	3,65	3,05	2,43	3,08	8,33	4,43	3,23	4,17	1,09	4,86	3,36	2,80	1,72	2,12	2,32	1,64	6,40	6,18	5,94
100	2,82	2,57	2,66	7,26	5,96	3,82	6,19	1,21	4,17	2,50	3,23	4,60	3,90	3,82	3,54	3,26	2,10	3,69	4,53	2,54
120	2,97	3,15	3,49	2,59	2,01	4,96	2,40	0,63	0,44	3,31	2,01	2,40	3,57	4,82	1,99	2,72	3,44	1,02	3,21	6,19
140	2,90	1,65	2,90	8,54	2,87	2,96	3,45	4,54	6,34	2,37	4,47	2,66	4,96	3,87	1,10	1,63	1,87	4,49	2,17	1,38
160	3,08	2,33	1,59	4,65	0,94	0,92	1,56	1,14	0,84	1,64	4,60	4,88	6,59	2,70	2,41	4,83	4,42	0,59	1,27	0,74
180	0,79	1,94	3,99	1,66	1,69	3,93	1,89	3,17	3,00	1,10	1,49	0,68	0,71	0,64	0,49	1,09	0,43	1,04	0,98	1,25
200	1,12	3,00	1,62	0,95	0,35	1,33	5,17	0,80	2,69	0,56	0,78	0,90	0,98	3,10	3,21	1,46	4,77	1,54	0,80	4,21
220	0,79	8,54	1,79	0,57	2,14	2,08	1,25	2,18	3,22	0,94	5,18	8,68	3,51	1,28	1,80	0,95	0,71	0,72	0,90	1,43
240	0,63	0,86	1,21	1,10	2,88	0,91	0,80	1,39	4,77	0,97	1,26	4,03	1,54	3,03	1,01	2,03	0,63	0,46	0,97	1,33
260	1,02	0,67	0,72	1,04	0,96	0,84	0,80	2,12	1,06	1,53	2,19	0,51	1,73	0,42	0,75	0,27	0,50	0,61	0,61	0,33
280	0,42	1,63	0,47	0,61	2,48	0,61	0,87	0,34	0,51	0,25	1,26	0,69	0,59	0,74	0,63	1,28	2,04	3,98	3,27	0,74
300	0,47	0,75	0,42	0,57	0,68	0,52														

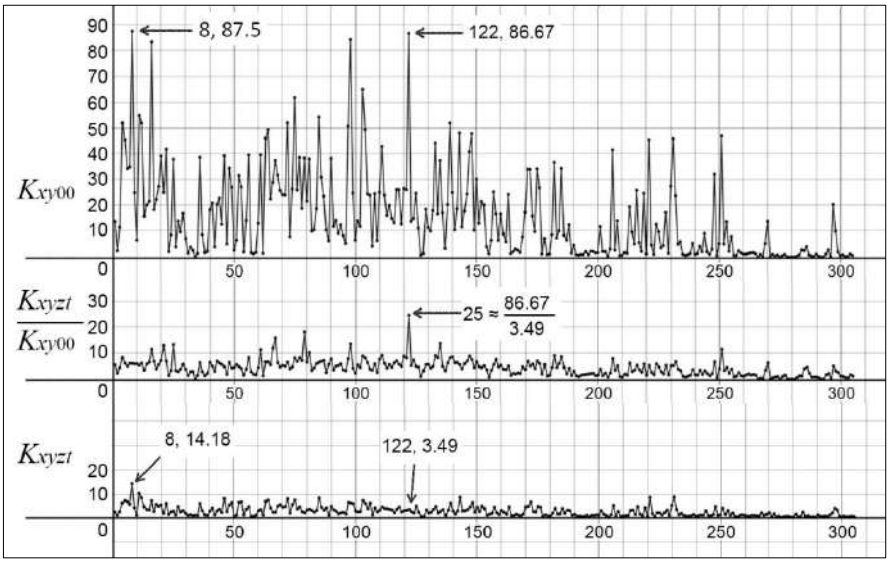

Рис. 8. График двух коэффициентов регулярности и их отношения для 305 стихотворений Ши цзина
Fig. 8. Graph of two regularity coefficients and their relations for 305 Shih-ching poems

На рис. 9 показано разбиение матрицы стихотворения № 8 и перевод А.А. Штукина. Мы видим, что все строки стихотворения № 8 одинаковы, за исключением 3-го иероглифа 2-го стиха. Все эти 6 иероглифов – глаголы, причём 5 из них уникальны. Одинаковость строк означает, что при (ненулевых) сдвигах по вертикали, т.е. по строфам и/или строкам, сохраняются 7 из 8 иероглифов каждой строки. Перевод Штукина довольно точно передаёт этот параллелизм: первые стихи строк одинаковы и содержат по 4 слова, правда, во вторых стихах строк число слов колеблется от 2 до 4 и меняется не только глагол. Но требовать

большего от перевода на русский, наверное, и нельзя: например, в русском языке нет глаголов, которые точно передавали бы значение китайского глагола *цзе* 袺 – поднять полу платья, положить в полу платья, и глагола *се* 襭 – брать (что-л.) в подол, заложив полу платья за пояс. Поэтому приходится использовать несколько слов: в подол набрала, в подоле понесла.

1	1	2	2	2	2	1	3	Рву да рву подорожник// Всё срываю его.
1	1	2	2	2	2	有	3	Рву да рву подорожник// Собираю его.
1	1	2	2	2	2	掇	3	Рву да рву подорожник// Рву всё время его.
1	1	2	2	2	2	捋	3	Рву да рву подорожник// Чищу семя его.
1	1	2	2	2	2	袺	3	Рву да рву подорожник// Вот в подол набрала.
1	1	2	2	2	2	襭	3	Рву да рву подорожник// В подоле понесла.

Рис. 9. Разбиение матрицы стихотворения № 8 (1.1.8) Фу и 芣苢 (Подорожник)
Fig. 9. Partition of the matrix of poem No. 8 (1.1.8) Fou yi 芣苢 (Plantain)

Заметим, что при горизонтальных (ненулевых) сдвигах, т.е. по стихам в строках и/или иероглифам в стихах, в стихотворении № 8 не сохраняется ни один иероглиф. Если при вычислении коэффициента регулярности (обозначенного K_{xy00}) учитывать только вертикальные сдвиги, т.е. по строфам и/или строкам, у этого стихотворения № 8 тоже будет максимальный коэффициент регулярности, равный 87,5. График среднего значения K_{xy00} по разделам *Ши цзина* показан на рис. 7 вверху, а по всем 305 стихотврениям – на рис. 8 (вверху), там же (в середине) показан график отношения K_{xy00} / K_{xyzt}. Пик этого отношения достигается для стихотворения № 122 и равен примерно 25 (рис. 10).

1	1	1	2	七	3	4	4	4	4	2		5	5	吉	3	
1	1	1	2	六	3	4	4	4	4	2		5	5	燠	3	

Разве можно сказать, что я сам не имею одежды? Семь различных нарядов теперь у меня; Только нынешний дар твой, вот эти одежды Будут много удобней и лучше, поверь, для меня.
Разве можно сказать, что я сам не имею одежды? Но различных одежд было шесть у меня. Только нынешний дар твой, вот эти одежды И теплей, и удобней нарядов, что есть у меня.

Рис. 10. Разбиение матрицы стихотворения № 122 (1.10.9) Цзю му 無衣 (Разве можно сказать)
*Fig. 10. Partition of the matrix of poem No. 122 (1.10.9) Wu yi **無衣** (How can it be said)*

Стихотворение № 122 имеет K_{xyzt} = 3,49, что выше среднего значения 2,71, но далеко от максимального значения 14,18, это 46-е по величине значение K_{xyzt}. А вот при сдвигах только по вертикали имеем второе после максимального (87,5) значение K_{xy00} = 86,67. В китайских статьях [20], [21], [22] это стихотворение не упоминается.

Тот факт, что K_{xy00} существенно (в несколько раз) больше K_{xyzt} говорит о том, что повторение иероглифов и фигур происходит в основном при вертикальных сдвигах, т.е. по строфам и строкам в строфах, и значительно меньше при горизонтальных сдвигах, т.е. по стихам в строках и иероглифам в стихах. Это подтверждается табл. 3, где указаны средние, максимальные и минимальные значения коэффициентов регулярности для всех 15 видов сдвигов, отличающихся тем, по каким координатам сдвиг не происходит (они обозначены нулями в индексах). Столбцы упорядочены слева направо по убыванию среднего коэффициента регулярности.

Табл. 3. Коэффициенты регулярности(K) при разных видах сдвигов
Table 3. Regularity coefficients (K) for different types of shifts

K	*x*,0, 0,0	*x*,0, *z*,0	*x*,*y*, 0,0	*x*,*y*, *z*,0	*x*,0, 0,*t*	0,*y*, 0,0	0,0, *z*,0	*x*,0, *z*,*t*	0,*y*, *z*,0	*x*,*y*, 0,*t*	*x*,*y*, *z*,*t*	0,*y*, *z*,*t*	0,*y*, 0,*t*	0,0, *z*,*t*	0,0, 0,*t*
average	32,74	17,25	15,38	8,78	6,47	4,91	4,41	4,40	4,26	3,97	2,71	1,59	1,59	1,55	1,25
max	92,82	74,29	87,50	44,93	22,19	87,50	33,33	18,59	29,17	24,14	14,18	8,33	13,10	6,09	7,14
min	0	0	0	0	0	0	0	0	0	0	0,25	0	0	0	0
ст.№8	87,50	46,05	87,50	44,93	21,36	87,50	0	13,43	29,17	24,14	14,18	8,33	13,10	3,17	3,85
ст.№127	1,25	1,32	0,66	0,68	0,49	0	0	1	0	0,29	0,63	0,68	0	1,19	0

В частности, для сдвигов по одной координате имеет место следующее соотношение средних коэффициентов регулярности: $K_{x000} > K_{0y00} > K_{00z0} > K_{000t}$, т.е. повторы случаются чаще всего при сдвиге по строфам, затем по строкам в строфах, затем по стихам в строках и, на последнем месте, по иероглифам в стихах. Эта же тенденция видна из соотношения средних значений коэффициентов регулярности при сдвигах по трём из четырёх координат: $K_{xyz0} > K_{x0zt} > K_{xy0t} > K_{0yzt}$.Наконец, даже минимальный из средних коэффициентов регулярности по вертикальным сдвигам оказывается больше максимального из средних коэффициентов регулярности по горизонтальным сдвигам: $min\{ K_{x000}, K_{xy00}, K_{0y00} \} > max\{ K_{00z0}, K_{00zt}, K_{000t} \}$. Хотя, конечно, для отдельных стихотворений соотношения могут быть другими, в частности, для стихотворения № 8имеем $K_{00z0} = 0 < K_{000t} = 3{,}85$ и$K_{x0zt} = 13{,}43 < K_{xy0t} = 24{,}14$.

Полученные результаты компьютерных экспериментов позволяют также прояснить связь между регулярностью стихотворений и их размером (в числе иероглифов). На рис. 11 показаны коэффициенты регулярности (K_{xyzt}) в зависимости от размера стихотворения, а также средний коэффициент регулярности K_{xyzt}, вычисленный в каждом диапазоне [*x*-50, *x*), как функция числа *x* иероглифов в стихотворении, округлённого до кратности 50. Можно отметить два момента. 1) как и следовало ожидать, средний коэффициент регулярности падает с ростом размера стихотворения, 2) коэффициенты регулярности стихотворений одного размера распределены между минимальным и максимальным значениями более-менее равномерно – им соответствуют столбцы точек на рис. 11.

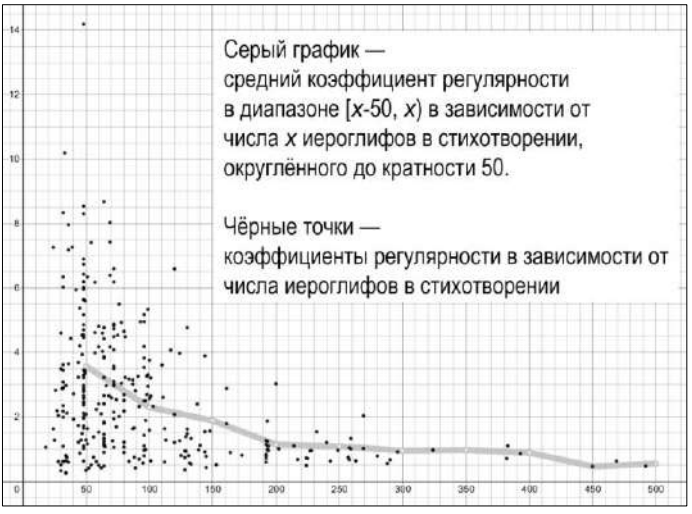

Рис. 11. Коэффициенты регулярности как функция размера стихотворения и график среднего коэффициента регулярности
Fig. 11. Regularity coefficients as a function of the size of the poem and the graph of the average regularity coefficient

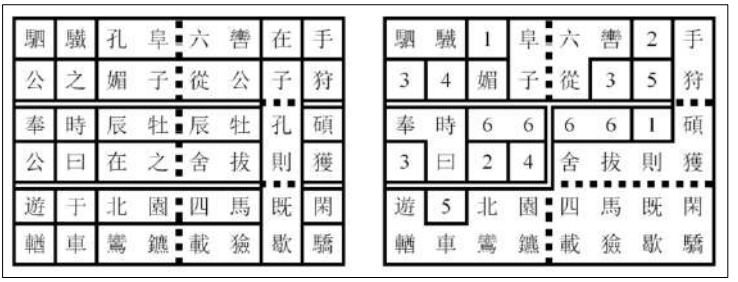

Рис. 12. Стихотворение № 127 (1.11.2) Сы те 駟驖(Князь на охоте): матрица и её разбиение
Fig. 12. Poem No. 127 (1.11.2) Si tie 駟驖 (Prince on the hunt): the matrix and its partition

Для примера можно посмотреть коэффициенты регулярности стихотворений с 48 иероглифами. В *Ши цзине* таких стихотворений больше всего – 40 штук. Максимальный коэффициент регулярности K_{xyzt} = 14,18 имеет отмеченное выше стихотворение № 8, а минимальный коэффициент регулярности K_{xyzt} = 0,63 – стихотворение № 127. Коэффициенты регулярности при всех 15 видах сдвигов для обоих стихотворений приведены внизу табл. 3. Матрица стихотворения № 127 и её разбиение на многоугольники приведены на рис. 12. Это разбиение существенно отличается от разбиения матрицы стихотворения № 8 (рис. 9), прежде всего, обилием уникальных иероглифов.

9. Параллелизмы в оригинале и в переводе

В предыдущем разделе мы уже говорили о сохранении регулярной структуры стихотворения на примере перевода А.А. Штукиным стихотворения № 8 (ПОДОРОЖНИК). Аналогичную структуру имеет и классический перевод на английский Дж. Легга (J. Legge) [19]:

We gather and gather the plantains; // Now we may gather them.
We gather and gather the plantains; // Now we have got them.
We gather and gather the plantains; // Now we pluck the ears.
We gather and gather the plantains; // Now we rub out the seeds.
We gather and gather the plantains; // Now we place the seeds in our skirts.
We gather and gather the plantains; // Now we tuck out skirts under our girdles.

А вот в переводе стихотворения № 122 Дж. Легг в большей мере, чем Штукин, сохранил структуру повторов.

How can it be said that he is without robes? // He has those of the seven orders;
But it is better that he get those robes from you. // That will secure tranquillity and good fortune.
How can it be said that he is without robes ? // He has those of the six orders;
But it is better that he get those robes from you. // That will secure tranquillity and permanence.

Серым фоном выделены уникальные слова, точно соответствующие уникальным иероглифам на рис. 10. Кроме, видимо, неизбежного изменения «теперь – было» во 2 и 6 строках перевода Штукина, появляется отсутствующее в оригинале «Но», а также меняется «Было много – И» и «поверь – что есть». Отчасти это вызвано, наверное, необходимостью соблюсти размер.

При всех нюансах можно отметить стремление обоих переводчиков сохранить регулярную структуру стихотворений, основанную на буквальном повторении знаков (иероглифов – слов). Так не всегда бывает. Но зачем вообще в стихах *Ши цзина* так много повторов? Для чего они нужны? И нужно ли их сохранять при переводе?

В [22] китайский автор пишет о *Ши цзине*: «Если бы эта работа появилась в более позднюю эпоху, когда метрическая поэзия стала популярна, я боюсь, что многие люди посмеялись бы над ней. Написание метрических стихотворений всегда избегало семантических повторений. [В стихосложении] существует так называемая болезнь *хэчжан* 合掌 (болезнь «сложенных

ладоней»), в данном случае болезнь парных иероглифов в стихах». И далее: «По сравнению с прозой и романами, длина стихотворений слишком мала. Чтобы максимально расширить смысл текста, поэты не могут не дорожить чернилами, как золотом… Повторение равносильно пустой трате места в стихотворении». И специально по поводу стихотворения № 8 (ПОДОРОЖНИК): «Поэт династии Цин Юань Мэй 袁枚 (1716-1798) посмеялся над этим и сказал: Триста стихов *Ши цзина* подобны множеству подорожников, которые рвут и рвут. Они не предназначены для подражания будущим поколениям». Но далее автор статьи пишет об этой шутке Юань Мэя: «На самом деле он критикует не *Ши цзин*, а тех, кто его читает в цветных очках уставных (пяти- и семисловных) стихов, и это тенденциозный подход». Здесь китайская идиома «цветные очки» – это то же самое, что русские «шоры».

В китайской энциклопедии [23] говорится: «Повторения в строфах и строках относятся к способу выражения в литературных произведениях, в которых одни и те же или похожие предложения неоднократно появляются в одной и той же позиции в разных абзацах. Их роль состоит в том, чтобы углубить впечатление, передать атмосферу, углубить тему стихотворения, усилить музыкальность и чувство ритма стихотворения, чтобы чувства могли быть выражены в полной мере». Профессор Ли Юйлян в [21] добавляет: «Параллелизмы нужны не просто для повторов, а для получения своеобразного художественного эффекта, который заставляет эмоциональное восприятие стихотворения подниматься по спирали. Согласно исследованиям, большинство стихотворений в *Ши цзине* изначально были народными песнями, их пели. При пении эти стихи нужно петь трижды, передавая людям волнующие ощущения и углубляя эмоциональное выражение».

Почему трижды? Во-первых, потому что в *Ши цзине* больше всего стихотворений с тремя строфами (113 из 305, т.е. 37%). Среди стихотворений самого распространённого размера (48 иероглифов – 40 стихотворений) также больше всего трехстрофных (32, т.е. 67%). Во-вторых, три – это важнейшее нумерологическое число, которому должен подчиняться *Ши цзин*, как и любой китайский канон-*цзин*.

Автор статьи [22] поясняет: «Подорожник» – это трудовая песня, которую пели во время сбора диких овощей. И далее сравнивает её с песней Бо Цзюй-и «Вечная печаль», которую пела певичка династии Тан, держа в руках пипу. Эта песня является шедевром с великолепной риторикой и сложным ритмом. Но «Вечная печаль» не подходит для сцены труда по сбору диких овощей, и в этом ценность и рациональность «Подорожника». Тут можно вспомнить о «принципиальной утилитарности китайского искусства» [23], стр. 20-24.

Аналогично стихотворение № 7 (1.1.7) *Ту цзюй* 兔罝 (Охотник) – это песня, которую распевает солдат во время занятий боевыми искусствами. Стихотворения № 5 (1.1.5) *Чжун сы* 螽斯 (Саранча), № 4 (1.1.4) *Цзю му* 樛木 (На юге у дерева долу склоняются ветви) и № 6 (1.1.6) *Тао яо* 桃夭 (Песнь о невесте) – это другой тип сценария: песни для благословения свадьбы. Автор справедливо замечает: это ведь то же самое, что бесконечное “Happy birthday to you” на праздновании дня рождения.

«Нужно ли при переводе сохранять эту риторическую форму параллелизма?» – задаётся вопросом профессор Ли Юйлян в [21] и отвечает: «Это зависит от того, интерпретирует ли переводчик стихи как литературные произведения и признаёт ли он ценность этого художественного приёма». Далее он разбирает пример стихотворения № 143 (1.12.8) *Юэ чу* 月出 (Вышла на небо луна) в его переводах на английский Легга (J. Legge), Дженнингса (W. Jennings), Аллена (C.F.R. Allen) и Эзры Паунда (E. Pound), а также на современный китайский Сюй Юаньчуна (许渊冲).

Эзра Паунд полностью отказался от этой формы риторики. Ли Юйлян цитирует свои собственные слова из статьи «Принцип “реальности” и поэтика имажизма в переводе *Ши цзина* Паунда»: «Принцип имажизма придает его переводу большое художественное очарование имажистской поэзии, и это имеет большое значение для дальнейшего развития и совершенствования его имажистской поэтики». Но далее замечает: «Конечно, с точки зрения

развития самой американской поэзии в таком подходе нет ничего плохого; но с точки зрения культурной коммуникации такой перевод *Ши цзина* очень неэффективен, и он даже вводит читателей в заблуждение и помешает их пониманию классической китайской культуры».

Позиция Аллена демонстрирует, как пишет профессор Ли Юйлян, «другой менталитет, своего рода культурное высокомерие и крайний культурный консерватизм». И приводит высказывание Аллена о переводе Легга: «В глазах современных китайцев перевод Легга очень хорошо соответствует оригинальному тексту и совершенен, но как английское стихотворение он не имеет никакой ценности вообще» ([24],p. XXI). Сам Аллен делал перевод, стараясь, прежде всего, следовать традициям английской поэзии, прежде всего, музыкальности в духе Суинберна. Поэтому он пишет: «Когда произведение состоит из одного предложения, выраженного три или четыре раза с наименьшими возможными вариациями, я часто сжимал его целиком в одну строфу» [там же, p. XXV].

Профессор Ли Юйлян замечает: «Согласно экспертизе, Аллен действительно переписал все строфы с параллелизмами в соответствии с эстетическими стандартами английской поэзии». И далее делает далеко идущие выводы: «Такого рода чрезмерный акцент в переводе на филологических нормах и навязывании таких норм переведенным произведениям является пренебрежением и незнанием литературного разнообразия; ставит литературу перед определенным метафизическим барьером, заставляет ее подчиняться определенному фиксированному шаблону и стирает природу оригинальной культуры. Это уже не литературный акт по своей природе, а акт культурной манипуляции и даже культурный и политический акт. Такого рода культурное высокомерие не редкость. В силу этого, хотя перевод может более или менее вобрать в себя некоторые сильные стороны экзотических культур, в долгосрочной перспективе его культурный консерватизм и узость неизбежно ограничат обмен между двумя культурами и искусствами, препятствуя нормальному развитию собственной культуры. Со второй половины прошлого века положение британской нации продолжало ухудшаться. Может быть, в этом причина?».

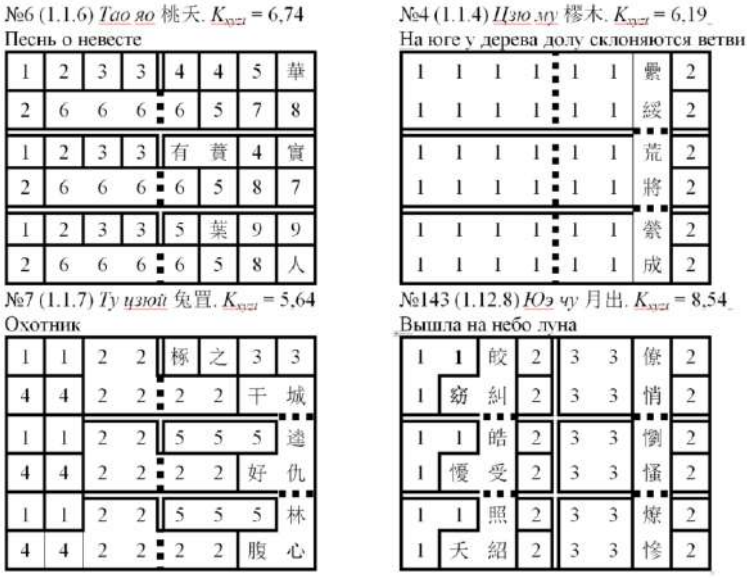

Рис. 13. Разбиения матриц четырёх стихотворений Ши цзина
Fig. 13. Matrix partitions of four Shih-ching poems

На рис. 13 мы приводим разбиения матриц упоминаемых выше стихотворений *Ши цзина* (кроме тех, что приведены выше) для того, чтобы можно было наглядно увидеть степень их регулярности.

10. Заключение

В статье рассматриваются такие регулярности матрицы как повторение фигур, т.е. одинаковые наборы символов с одинаковым относительным расположением. Предложена числовая характеристика регулярности матриц (сумма регулярности), которая позволяет сравнивать матрицы одного размера с одинаковым расположением пустых символов. Для сравнения регулярности матриц разного размера и/или с различным расположением пустых символов предложен коэффициент регулярности как процентное отношение суммы регулярности данной матрицы к сумме регулярности «самой регулярной» матрицы, в которой все непустые ячейки заполнены одним и тем же непустым символом. В статье приведены алгоритмы вычисления суммы и коэффициента регулярности матрицы. По этим алгоритмам реализованы соответствующие компьютерные программы.

Сумма и коэффициент регулярности являются обобщёнными характеристиками регулярности матрицы. Для наглядного представления полной структуры регулярности предложено разбиение матрицы на плоские фигуры, ограниченные простыми многоугольниками. Повторяющиеся фигуры нумеруются так, что вхождения одной повторяющейся фигуры получают одинаковые номера и не содержат одинаковых символов. Другие фигуры состоят либо из сохраняемых уникальных символов, либо из пустых символов.

Компьютерные программы были апробированы для вычисления суммы и коэффициента регулярности всех 305 стихотворений древнекитайского «Канона стихов» – *Ши цзина*. Стихотворение представлялось в виде четырёхмерной матрицы, координаты которой соответствуют строфам, строкам в строфах, стихам в строках и иероглифам в стихах. Некоторые обобщающие результаты этих компьютерных экспериментов приведены в статье, так же, как и примеры разбиения матриц некоторых стихотворений, в том числе, упоминаемых в статьях китайских авторов, посвящённых повторам в стихах *Ши цзина*. Статья также включает раздел, содержащий краткий обзор (по материалам китайских статей) проблемы повторов в *Ши цзине*: предназначение повторов и сохранение или не сохранение их при переводах на другие языки.

Можно предложить следующие направления дальнейших исследований.

1) В статье исследовались повторы при сдвигах матрицы по координатам. Можно провести исследование регулярности при других преобразованиях. Например, совпадение фигур при вращении на кратное 90 число градусов, зеркальная или осевая симметрия и т.п.
2) В статье регулярности основаны только на равенстве символов и фигур символов при повторении. Перспективным представляется рассмотрение других отношений эквивалентности на алфавите символов матриц и/или множестве фигур символов так, что при повторении символ и/или фигура могут меняться, но в пределах того же класса эквивалентности. В частности, это важно для исследования регулярности матричных текстов на русском и других языках, где слова меняются при изменении рода, числа, падежа, времени и т.п.
3) С практической точки зрения было бы интересно проанализировать регулярность матриц других китайских (и не только китайских) стихотворений и других матричных текстов.

Список литературы / References

[1] Рыков С.Ю. Методология науки и философии. Энциклопедия «Духовная культура Китая» в 5 т. М., Восточная литература., 2006–2010 гг. Том 5, Наука, техническая и военная мысль, здравоохранение и образование, 2009 г., стр. 662-668 / Rykov S.Yu. Methodology of science and philosophy. In Encyclopedia «Spiritual Culture of China» in 5 volumes. M., Vostochnaya literatura, 2006-2010. Vol. 5, Science, technical and military thought, health care and education, 2009, pp. 662-668 (in Russian).

[2] Спирин В.С. Построение древнекитайских текстов. М., Наука, 2-е изд., 1976 г., 231 стр. / Spirin V.S. Construction of ancient Chinese texts. M., Nauka, 2nd ed., 1976, 231 p. (in Russian).

[3] Карапетьянц А.М. У истоков китайской словесности. М., Восточная литература, 2010 г., 480 стр. / Karapipants A.M. At the origins of Chinese literature. M., Vostochnaya literatura, 2010, 280 p. (in Russian).

[4] Кобзев А.И. Учение о символах и числах в китайской классической философии. М., 1994 Восточная литература, 432 стр. / Kobzev A. I. The doctrine of symbols and numbers in Chinese classical philosophy. M., Vostochnaya literatura, 1994, 432 p. (in Russian).

[5] Кобзев А.И. Сань у. Энциклопедия «Духовная культура Китая» в 5 т. М., Восточная литература., 2006–2010 гг. Том 5, Наука, техническая и военная мысль, здравоохранение и образование, 2009 г., стр. 803–825. / Kobzev A.I. San U. In Encyclopedia «Spiritual Culture of China» in 5 volumes. M., Vostochnaya literatura, 2006-2010. Vol. 5, Science, technical and military thought, health care and education, 2009, pp. 803–825 (in Russian).

[6] Лихтман В.В. Функция древнекитайских текстов, построенных на десятеричных структурах. В сб. «Дальний Восток и Центральная Азия», М., Наука, 1985 г. / Lichtman V.V. The function of ancient Chinese texts built on ten-meter structures. In Far East and Central Asia, M., Nauka, 1985 (in Russian).

[7] Лихтман В.В. 13-ричные текстологические структуры как план (ту). Труды XVII научной конференции «Общество и государство в Китае», ч. 1, I986, стр. 72-81 / Lichtman V.V. 13-richety textological structures as a plan (TU). In Proc. of the XVII Scientific Conference «Society and the State in China», Part 1, pp. 71-82 (in Russian).

[8] Лихтман В.В. Пространственные текстологические структуры («Шань хай цзина» и «Ши цзина»). Труды XVIII научной конференции «Общество и государство в Китае», ч. 1, I987, стр. 151-154. / Lichtman V.V. Spatial textological structures («Shan Hai Jing» and «Shih-ching»). In Proc. of the XVIII Scientific Conference «Society and the State in China», Part 1, pp. 151-154 (in Russian).

[9] Дорофеева В.В. [Лихтман В.В.] «Ши цзин» как исторический источник для реконструкции пространственных представлений в древнем Китае. Автореферат диссертации на соискание ученой степени кандидата исторических наук, МГУ им. М. В. Ломоносова, 1992 г., 28 стр. / Dorofeeva V.V. [Likhtman V.V.] "Shih-ching" as a historical source for the reconstruction of spatial representations in ancient China. Abstract of the dissertation for the degree of Candidate of Historical Sciences, Lomonosov Moscow State University, 1992, 28 p. (in Russian).

[10] Бурдонов И.Б. Горы и воды: алгебра стихов и гармония перемен. Сборник материалов 21-й конференции из цикла «Григорьевских чтений», 2019 г., стр. 33-58. Также: Геопоэтика гор и вод Канона Стихов. Тезисы докладов 3-й Международной конференции по геопоэтике, 2018. Полный текст: URL: http://burdonov.ru/SHI_ZIN/Geopoetika_gor_i_vod_Shi_Jing/1.html / Burdonov I.B. Mountains and waters: algebra of verses and harmony of changes. Collection of materials of the 21st conference from the cycle of "Grigorievsky Readings", 2019, pp. 33-58. See also: Geopoetics of mountains and waters of the Canon of Poems. Abstracts of the 3rd International Conference on Geopoetics, 2018. Full text: URL: http://burdonov.ru/SHI_ZIN/Geopoetika_gor_i_vod_Shi_Jing/1.html (in Russian).

[11] Кобзев А.И. Старые проблемы и новый перевод «Ши цзина». Общество и государство в Китае, том XLVIII, ч. 2, 2018 г., стр. 261-331 / Kobzev A.I. Old Problems and a New Translation of the Shi Jing. Society and the State in China, vol. XLVIII, part 2, 2018, pp. 261-331 (in Russian).

[12] Кобзев А. И. Юбилей русского перевода «Ши цзина» и нерешённые проблемы. Сборник статей и докладов участников X Международной научно-практической конференции «Россия– Китай: история и культура», 2017 г., стр. 265-274 / Kobzev A. I. The anniversary of the Russian translation "Shih-ching" and unresolved problems. A collection of articles and reports of participants in the X International Scientific and Practical Conference «Russia-China: History and Culture», 2017, pp. 265-274 (in Russian).

[13] Бурдонов И.Б. «Центральное» стихотворение Ши-цзина. Общество и государство в Китае, том XLIX, ч. 2, 2019 г., стр. 311-328 / Burdonov I.B. «Central» poem by Shih-ching. Society and the state in China, vol. XLIX, part 2, 2019, pp. 311-328 (in Russian).

[14] Спирин В.С. Каноны конфуцианства и школы имен. «Великое учение» (Да-сюэ) и «Мудрец Дэн Си» (Дэн Си-цзы). В 2-х томах. М., Восточная литература, 2014 г. Том 2. «Дэн Си-цзы» как логико-гносеологическое произведение. 325 стр. / [14] Spirin V.S. Canons of Confucianism and schools of names. «Great Teaching» (Da-xue) and «Sage Deng Xi» (Deng Xi-tzu). In 2 volumes. M., Vostochnaya literatura, 2014. Vol. 2. «Deng Xi-tzu» as a logical and epistemological work. 325 pp. (in Russian).

[15] Карапетьянц А.М. Раннекитайская системология. М., Восточная литература, 2015 г., 565 стр. / Karapetyants A.M. Early Chinese systemology. M., Vostochnaya literatura, 2015, 565 p. (in Russian).

[16] Кобзев А. И., Орлова Н. А. Поэтика «оборванных строк» (цзюэ-цзюй) и поэтический перевод ста четверостиший Бо Цзюй-и. Общество и государство в Китае, том L, ч. 1, 2020 г., стр. 390–495 /

Kobzev A.I., Orlova N. A. Poetics “quatrain” (jué-jù) and a poetic translation of a hundred quatrains of Bo Zyu-i. Society and the state in China, vol. L, part 1, 2020, pp, 390–495 (in Russian).

[17] Еремеев В.Е. «Книга перемен» и исчисление смыслов. М., Восточная литература, 2013 г., 583 стр. / Yeremeyev V.E. «Book of Changes» and Working out of Meanings. M., Vostochnaya literatura, 2013 (in Russian).

[18] Сторожук А. Г. Слово и цифра: знаковая структура уставных стихов. Вестник СПГУ. Серия 13, вып. 1, 2009 г., стр. 43–67. / Storozhuk A. G. Word and digit: the sign structure of the regular verse. Vestnik of Saint Petersburg University. Series 13, issue 1, 2009, pp. 43–67 (in Russian).

[19] 《詩經 - Book of Poetry》. Chinese Text Project. URL: https://ctext.org/book-of-poetry (дата обращения 06.04.2022).

[20] Чун чжан де чан 重章叠唱 (Параллелизмы в строфах и стихах). Китайская интернет-энциклопедия Байду (Бай-ду бай-кэ 百度百科) / Chong Zhang de Chan 重章 唱 唱 唱 (parallelisms in stanzas and verses). Chinese Internet encyclopedia bydu (bai-do bai-ke). URL: https://baike.baidu.com/item/重章 唱唱/5093143, accessed 06.04.2022 (in Chinese).

[21] Ли Юйлян 李玉良. «Ши цзин» чжун дэ чун чжан де чан цзици фаньи《诗经》中的重章叠唱及其翻译 (Параллелизмы в строфах и стихах «Канона стихов» и перевод). Из книги «Ши цзин фаньи фаньвэй»《<诗经>翻译探微》(Исследование переводов «Канона Стихов»). / Lee Yulyan 李玉良.《诗经》 中 的 叠唱 及 其 翻译 翻译 (parallelisms in stanzas and verses of the “Classic of Poetry” and translation). From the book《<诗经> 翻译》》 (Study of translations of “Classic of Poetry”). Network discos 调色 盘网. URL: https://www.tspweb.com/key/ 重章 是 什么 手法.html, accessed 06.04.2022 (in Chinese).

[22] Цзинь Гунцзы 晋公子. «Ши цзин» гай цзэньме ду?《诗经》该怎么读？(Как читать «Канон стихов»?). Китайская интернет-энциклопедия Байду (Бай-ду бай-кэ 百度百科) / Jin Gunzi 晋公子《诗经》 该 怎么 读？ (How to read the “Classic of Poetry”?). Chinese Internet encyclopedia by Baydu 百度百科 URL: https://baike.baidu.com/tashuo/browse/content?id=52088533ECA6B5EC3AC55555B39 accessed 06.04.2022 (in Chinese).

[23] Энциклопедия «Духовная культура Китая» в 5 т. М., Восточная литература., 2006–2010 гг. Том 6. Дополнительный. Искусство. 1126 стр. / In Encyclopedia «Spiritual Culture of China» in 5 volumes. M., Vostochnaya literatura, 2006-2010. Vol. 6. Supplementary. Art. 1126 pp. (in Russian).

[24] Allen C.F.R. The Book of Chinese Poetry: Being the Collection of Ballads, Sagas, Hymns, and Other Pieces Known as the Shih Ching or Classic of Poetry. L., K. Paul, Trench, Trübner, 1891, 528 p.

Информация об авторах / Information about authors

Игорь Борисович БУРДОНОВ – доктор физико-математических наук, г.н.с. Научные интересы: формальные спецификации, генерация тестов, технология компиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработки программного обеспечения.

Igor Borisovich BURDONOV – Doctor of Physical and Mathematical Sciences, Chief Scientist. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.

Алексей Александрович КАРНОВ – аспирант. Научные интересы: формальные спецификации, верификация и тестирование, криптографические интернет-протоколы.

Aleksei Aleksandrovich KARNOV – PhD student. Research interests: formal specifications, verification and testing, cryptographic Internet protocols.

DOI: 10.15514/ISPRAS-2022-34(1)-9

Алгоритмы планирования вычислений с учетом избыточности и неопределенности

[1] *А.Г. Феоктистов, ORCID: 0000-0002-9127-6162 <agf65@icc.ru>*
[1] *Р.О. Костромин, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>*
[1] *С.А. Горский, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>*
[1] *И.В. Бычков, ORCID: 0000-0002-1765-0769 <bychkov@icc.ru>*
[2,3,4] *А.Н. Черных, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[1,5] *О.Ю. Башарина, ORCID: 0000-0002-7151-782X <basharinaolga@mail.ru>*

[1] *Институт динамики систем и теории управления им. В.М. Матросова СО РАН, 664033, Россия, Иркутск, ул. Лермонтова, 134, а/я 29*
[2] *Центр научных исследований и высшего образования Мексика, 22860, Нижняя Калифорния, Энсенада, ш. Тихуана-Энсенада, 3918*
[3] *Южно-Уральский государственный университет, 454080, Россия, Челябинск, проспект Ленина, 76*
[4] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, Москва, ул. А. Солженицына, 25*
[5] *Иркутский государственный университет, 664003, Иркутск, ул. Карла Маркса, 1, Россия*

Аннотация. В статье предложены новая модель и алгоритмы планирования вычислений в пакетах программ. Их отличительной особенностью является использование времени выполнения модулей пакетов на ресурсах среды в процессе их непрерывной интеграции, доставки и развертывания. Предложенные модель и алгоритмы позволяют строить избыточные схемы решения задачи. На практике избыточность схемы позволяет адаптировать ее к изменяющимся характеристикам ресурсов среды и повысить надежность вычислений. Построение схем решения задач показано на модельных примерах.

Ключевые слова: распределенные вычисления; научные приложения; схема решения задачи; планирование вычислений; избыточность; неопределенность

Для цитирования: Феоктистов А.Г., Костромин Р.О., Горский С.А., Бычков И.В., Черных А.Н., Башарина О.Ю. Алгоритмы планирования вычислений с учетом избыточности и неопределенности. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 123-140. DOI: 10.15514/ISPRAS–2022–34(1)–9

Благодарности: Исследование выполнено при поддержке РФФИ, проект № 19-07-00097-а. Имитационная модель среды разработана при частичной поддержке Министерства науки и высшего образования Российской Федерации, проект «Технологии разработки и анализа предметно-ориентированных интеллектуальных систем группового управления в недетерминированных распределенных средах».

Redundancy and Uncertainty-Based Algorithms for Computation Planning

[1] *A.G. Feoktistov, ORCID: 0000-0002-9127-6162 <agf@icc.ru>*
[1] *R.O. Kostromin, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>*
[1] *S.A. Gorsky, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>*
[1] *I.V. Bychkov, ORCID: 0000-0002-1765-0769 <bychkov@icc.ru>*
[2,3,4] *A.N. Tchernykh, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[1,5] *O.Yu. Basharina, ORCID: 0000-0002-7151-782X <basharinaolga@mail.ru>*

[1] *Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of the RAS, 134, Lermontov st., Irkutsk, postbox 292, 664033, Russia*
[2] *Centro de Investigación Científica y de Educación Superior, 3918, Ensenada-Tijuana Highway, Ensenada, 22860, Mexico*
[3] *South Ural State University, Chelyabinsk, 76, Lenin prospekt, Chelyabinsk, 454080, Russia*
[4] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[5] *Irkutsk State University, 1, KarlMarks St., Irkutsk, 664003, Russia*

Abstract. Nowadays, the development and use of workflow-based applications (distributed applied software packages) are some of the key challenges in terms of preparing and carrying out large-scale scientific experiments in distributed environments with heterogeneous computing resources. The environment resources can be represented by clusters of personal computers, supercomputers, and private or public cloud platforms and differ in their computational characteristics. Moreover, the composition and characteristics of resources change in dynamics. Therefore, computations planning and resource allocation in the considered environments are important problems. In this regard, we propose new algorithms for computation planning taking into account redundancy and uncertainty in such distributed applied software packages. Compared to other algorithms of a similar purpose, the proposed algorithms use evaluations of workflow execution makespan obtained in the process of continuous integration, delivery, and deployment of applied software. The proposed algorithms provide the construction of redundant problem-solving schemes that allow us to adapt them to the dynamic characteristics of computational resources and improve distributed computing reliability. The algorithms are based on a theory of conceptual modeling computational processes. We demonstrate the process of constructing problem-solving schemes on model examples. In addition, we show the utility in using redundancy for increasing the distributed computing reliability In comparison with some traditional meta-schedulers.

Keywords: distributed computing; scientific applications; workflow; computation planning; redundancy; uncertainty

For citation: Feoktistov A.G., Kostromin R.O., Gorsky S.A., Bychkov I.V., Tchernykh A.N., Basharina O.Yu. Redundancy and Uncertainty-Based Algorithms for Computation Planning. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 123-140 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–9

Acknowledgements. The study is supported by the Russian Foundation of Basic Research, project no. 19-07-00097. Developing the simulation model of the computing environment was supported in part by the Ministry of Science and Higher Education of the Russian Federation, project «Technologies for the development and analysis of subject-oriented intelligent group control systems in non-deterministic distributed environments».

1. Введение

Распределенные вычисления представляют собой эффективный подход к решению ресурсоемких вычислительных задач посредством использования множества вычислительных устройств, которые в общем случае являются разнородными по своим вычислительным и программно-аппаратным характеристикам. При этом распределенные вычисления можно считать высокопроизводительными, если они существенно ускоряют процесс решения задачи. В данном случае, термин *распределенные вычисления* определяет

весь спектр вопросов, связанных с планированием и распределением ресурсов вычислительной среды, а также управлением ими с целью достижения максимальной эффективности решения научных и прикладных задач.

Для таких сред мы разрабатываем масштабируемые научные и прикладные приложения – распределенные пакеты прикладных программ (РППП), модули программного обеспечения (ПО) которых могут выполняться на ресурсах среды с различными вычислительными характеристиками. Как правило, такие пакеты создаются на основе модульного подхода. Модули представляют собой прикладные программы или сервисы, снабженные спецификациями, которые отражают информацию о назначении, способах применения, форматах входных и выходных параметров, а также другие необходимые для вычислительного процесса сведения. Процесс решения задачи в РППП основывается на проведении многовариантных расчетов [1].

В рамках проблемы управления распределенными вычислениями выделяются две взаимосвязанные задачи [2]:

- планирование вычислений (построение схемы решения задачи, определяющей порядок выполнения модулей пакета);
- управление ими на ресурсах вычислительной среды, назначенных для выполнения прикладного программного обеспечения (модулей).

Статья посвящена вопросам, относящимся к решению первой задачи. Разработка и применение подхода к решению второй задачи детально рассмотрены в [3]. Средства для решения обеих задач реализуются в рамках инструментального комплекса Orlando Tools (OT) [4].

2. Обзор связанных работ

Автоматическое планирование взаимодействия компонентов научного рабочего процесса является сложной задачей, поскольку трудно определить их функциональность и семантику используемых ими данных [5]. В частности, необходимы разработка и применение специальных средств описания и использования алгоритмических и схемных знаний о предметной области научного рабочего процесса. Например, в качестве фундаментальной основы представления таких знаний может быть задействовано концептуальное моделирование [6].

Поэтому в системах поддержки научных рабочих процессов (workflow) наиболее распространены подходы, основанные на процедурной форме построения таких процессов [7]. Среди них Askalon [8], Condor DAGMan [9], GridAnt [10], GridWay [11], Kepler [12], Taverna [13], Triana [14] и UNICORE [15].

Наиболее популярными из них являются следующие подходы:

- применение универсальных языков разметки, таких, как XML [16], и также их расширений (например, XML Process Definition Language [17] для описания рабочих процессов, относящихся к любым сферам деятельности, и Business Process Modeling Language [18] для описания рабочих процессов различного типа);
- использование специальных форматов описания заданий по выполнению научных рабочих процессов (например, форматы заданий в Condor DAGMan и GridWay);
- графическое представление научных рабочих процессов в виде ориентированных циклических или ациклических графов (например, сети Петри [19] и UML диаграммы [20]).

Однако при увеличении числа компонентов научных рабочих процессов и их параметров восприятие таких рабочих процессов с помощью вышеупомянутых подходов существенно затрудняется. Это обусловлено ограниченностью области экрана компьютера, на котором отражается описание научного рабочего процесса. Во многих случаях разработчики научных

рабочих процессов и их конечные пользователи видят только отдельные фрагменты описаний и зачастую не могут проследить связи с другими фрагментами, находящимися за пределами экрана. Отчасти это нивелируется использованием иерархических форм представления научных рабочих процессов.

В этой связи наиболее продвинутые системы поддержки научных рабочих процессов (например, OT и Pegasus [21]) включают средства автоматического планирования взаимодействия компонентов приложений. Такие системы более предпочтительны для рабочих процессов, которые характеризуются большим числом своих компонентов и взаимосвязей между ними [22].

Системы GridAnt и UNICORE поддерживают конкретные научные рабочие процессы. Для таких научных рабочих процессов используемое прикладное ПО их компонентов и источники данных полностью или частично предопределены. Это зачастую обусловлено необходимостью использования конкретных сенсоров, управляющего оборудования, непереносимого ПО и др.

В сравнении с ними системы Askalon, Condor DAGMan, GridWay, Kepler, Pegasus и Triana ориентированы на абстрактные научные рабочие процессы. Компоненты таких научных рабочих процессов не связаны с конкретными источниками данных, прикладным ПО и ресурсами. Они могут запускаться на ресурсах, назначаемых для их выполнения статически или динамически локальными менеджерами ресурсов или метапланировщиками среды.

Дополнительным преимуществом систем OT и Taverna является то, что они поддерживают оба типа научных рабочих процессов. Это обеспечивает гибкость в выборе средств построения научных рабочих процессов.

Система поддержки научного рабочего процесса является частным случаем пакета прикладных программ. В отличие от научного рабочего процесса пакет реализует решение целого класса задач в конкретной предметной области. Решение задач осуществляется на модели предметной области (вычислительной модели). Такая модель формализует описание исследуемой системы, ее объектов, их взаимосвязей, а также происходящих в ней событий и процессов. В рамках направления исследований, связанного с разработкой и применением пакетов, разработан ряд известных подходов к автоматическому синтезу абстрактных программ на вычислительных моделях.

Работа [23] посвящена дедуктивному методу синтеза программ. Этот метод используется на статическом этапе построения программы для композиционного программирования в целом. Вопросы, связанные с избыточностью и неопределенностью, а также назначением ресурсов, требуют дополнительного развития.

Новый подход к построению параллельных асинхронных абстрактных программ требуемой длины представлен в [24]. Система булевых ограничений используется в качестве постановки задачи. В том числе учитываются ограничения на число доступных процессоров однородных узлов вычислительной среды и модельные временные задержки при выполнении модулей абстрактной программы.

В [25] рассмотрена проблема синтеза параллельных рекурсивных программ по непроцедурной постановке задачи. Предложено специальное исчисление для ее решения.

Система фрагментированного программирования LuNA, предназначенная для разработки библиотек параллельных численных программ, представлена в [26]. Она базируется на использовании вычислительной модели и ориентируется на высокопроизводительный однородный кластер [27].

В отличие от рассмотренных выше работ, использующих вычислительные модели, мы предлагаем модель и алгоритмы планирования вычислений в разнородной распределенной среде с учетом реального времени выполнения модулей, полученного во время их тестирования на ресурсах среды [28]. Представлены специализированные структуры данных,

необходимые для реализации данных моделей и алгоритмов. Приведены модельные примеры, иллюстрирующие применение предложенных моделей и алгоритмов.

3. Модель

Структура модели. Представим вычислительную модель РППП следующей структурой:

$$M_{DASP} = \langle PAR, O, M, N, R^{in}, R^{out}, \tilde{R}, \tilde{\tilde{R}}, PR, NPPF, NPPF', S \rangle, \quad (1)$$

где PAR – множество параметров пакета (обрабатываемых данных), O – множество вычислительных операций над полем параметров, M – множество модулей, являющихся конкретными программными реализациями абстрактных операций, N – множество разнородных ресурсов распределенной вычислительной среды, на которых размещены модули. Каждая операция $o_i \in O$ интерпретируется как абстрактный процесс вычисления множества ее выходных параметров $PAR_i^{out} \subset PAR$ по множеству ее входных параметров $PAR_i^{in} \subset PAR$, $PAR_i^{in} \cap PAR_i^{out} = \emptyset$.

Связь между параметрами и операциями задается соответственно булевыми матрицами R^{in} и R^{out} размерности $n_o \times n_p$. Элемент матрицы $r_{ij}^{in} = 1$ ($r_{ij}^{in} = 0$) указывает, что j-й параметр является (не является) входным для i-й операции, $i = \overline{1, n_o}$, $j = \overline{1, n_p}$. Аналогично, элемент матрицы $r_{ij}^{out} = 1$ ($r_{ij}^{out} = 0$) указывает, что j-й параметр является (не является) выходным для i-й операции.

Связь между модулями и операциями представлена булевой матрицей $\tilde{R}$ размерности $n_m \times n_o$. Элемент матрицы $\tilde{r}_{ij} = 1$ ($\tilde{r}_{ij} = 0$) определяет, что i-й модуль реализует (не реализует) j-ю операцию, $i = \overline{1, n_m}$, $j = \overline{1, n_o}$. Эта связь имеет тип «один-ко-многим». Поэтому на нее накладывается следующее ограничение целостности:

$$\bigvee_{j=1}^{n_o}(f_j \vee g_j) = 0, \quad f_j = \bigvee_{i=1}^{n_m - 1} \bigvee_{k=i+1}^{n_m}(\tilde{r}_{ij} \wedge \tilde{r}_{kj}), \quad g_j = \bigwedge_{i=1}^{n_m} \bar{\tilde{r}}_{ij}. \quad (2)$$

Ограничение (2) определяет, что одна операция может быть реализована только одним модулем. В тоже самое время, один модуль может реализовывать несколько операций.

Связь между ресурсами и модулями представлена булевой матрицей $\tilde{\tilde{R}}$ размерности $n_n \times n_m$. Элемент матрицы $\tilde{\tilde{r}}_{ij} = 1$ ($\tilde{\tilde{r}}_{ij} = 0$) определяет, что j-й модуль может быть выполнен на i-м ресурсе, $i = \overline{1, n_n}$, $j = \overline{1, n_m}$. Эта связь имеет тип «многие-ко-многим». Она означает, что один модуль может выполняться на разных ресурсах. В тоже самое время, несколько разных модулей могут быть размещены на одном и том же ресурсе.

Схема решения задачи. Для представления схемы решения задачи (ярусно-параллельная форма алгоритма решения задачи) будем использовать булеву матрицу S размерности $n_t \times n_o$, где n_t – это число ярусов схемы. Элемент матрицы $s_{ij} = 1$ ($s_{ij} = 0$) означает, что j-я операция размещена (не размещена) на i-м ярусе схемы, $i = \overline{1, n_t}$, $j = \overline{1, n_o}$. Матрица U размерности $n_o \times n_o$ представляет сведения о предшествовании операций при выполнении схемы S. Элемент $u_{ij} = 1$ ($u_{ij} = 0$) означает, что завершение операции o_j предшествует (не предшествует) выполнению операции o_i и $PAR_i^{in} \cap PAR_j^{out} \neq \emptyset$.

Непроцедурная постановка задачи. Схема S строится по непроцедурной постановке задачи $NPPF$. В рамках такой постановки задачи задаются множество $S^{in} \subset PAR$ исходных параметров, значения которых известны, и множество $S^{out} \subset PAR$ целевых параметров, значения которых должны быть вычислены. На основе связи между параметрами и операциями требуется определить множество $S^o \subset O$ операций, которые нужно выполнить, чтобы вычислить значения целевых параметров. Таким образом, непроцедурную постановку задачи $NPPF$ можно определить следующим образом:

$$NPPF: S^{in}, S^{out} \rightarrow S^o?, \quad (3)$$

где символ ‘?’ означает, что соответствующее множество не определено.

Назначение ресурсов. Как было отмечено выше, в данном контексте понятие схемы решения задачи соответствует абстрактному научному рабочему процессу. Назначение ресурсов для выполнения модулей, реализующих операции схемы S, осуществляется в процессе распределения вычислительных заданий с помощью мультиагентной системы (МАС) в рамках тендера вычислительных работ, детально рассмотренного в [29-31].

Таким образом, в результате проведения вышеупомянутого тендера мы получаем номер ресурса j для каждой i-й операции схемы S. На основе данной привязки определяются оценки времени (E_S^{asynch}), стоимости (E_S^{cost}) и надежности (E_S^r) выполнения операций схемы решения задачи.

Оценка времени выполнения схемы. Для оценки времени выполнения заданий мы используем модели, рассмотренные в [32]. Здесь мы приводим формулу вычисления такой оценки, обобщенную и уточненную для асинхронного режима выполнения схемы S с учетом возможных запусков (удалений) виртуальных машин и возникновений отказов узлов, системного и прикладного ПО.

Пусть матрица D размерности $n_s \times n_s$ отражает оценки размера данных, передаваемых между операциями, n_s – число операций схемы. Элемент матрицы $d_{ij} \geq 0$ показывает размер данных, передаваемых операцией o_i операции o_j. Матрица W размерности $n_s \times n_s$ представляет пропускную способность интерконекта между ресурсами, на которых будут запущены модули, реализующие операции. Элемент матрицы $w_{ij} \geq 0$ показывает пропускную способность интерконекта между ресурсами с модулями, реализующими операции o_i и o_j.

Оценка E_S^{asynch} времени выполнения схемы в асинхронном режиме вычисляется по следующей формуле:

$$E_S^{\text{asynch}} = \max_{i=\overline{1,n_s}} e_i^{\tau},$$

$$e_i^{\tau} = e_i^{q} + e_i^{vml} + e_i^{run} + e_i^{res} + e_i^{vmr} + \max_{\forall j \in \overline{1,n_s}: u_{ij}=1, i \neq j} \left(e_j^{\tau} + e^{dtr}\right),$$

$$e^{dtr} = \frac{d_{ji}}{c_{ji}(t) w_{ji}},$$

где переменные интерпретируются следующим образом:

- e_i^{τ} (e_j^{τ}) – оценка периода времени τ с начала выполнения схемы и до завершения операции o_i (o_j);
- $e_i^{q} \geq 0$ – оценка времени ожидания модуля, реализующего операцию o_i в очереди локального менеджера ресурсов;
- $e_i^{run} > 0$ – оценка времени выполнения модуля до его успешного завершения или отказа;
- $e_i^{res} \geq 0$ – оценка времени на рестарт модуля в случае возникновения отказа какого-либо типа;
- $e_i^{vml} \geq 0$ и $e_i^{vmr} \geq 0$ – экспериментально определяемые оценки времени, необходимого на запуск и завершение виртуальных машин;
- $0 < c_{ji}(t) \leq 1$ – коэффициент уменьшения пропускной способности интерконнекта в момент времени t между ресурсами, связанными с выполнением операций o_i и o_j.

Оценка стоимости выполнения задания. Данная оценка определяется по следующей формуле:

$$E_S^{cost} = \sum_{i=1}^{n_s} \pi_i e_i^{run} + \pi_i^*,$$

где $\pi_i \geq 0$ – себестоимость единицы процессорного времени ресурса, $\pi_i^* \geq 0$ – торговая наценка, закладываемая провайдером данного ресурса. В частности, для пользователей

суперкомпьютерных центров коллективного пользования или приватных облачных платформ как π_i, так и π_i^* могут быть равны 0.

Оценка надежности. Получение оценки надежности выполнения схемы решения задачи базируется на использовании методов логико-вероятностного анализа, предложенных И.А. Рябининым применительно к исследованию сложных технических систем [33]. Введем следующие обозначения:

- x – это набор булевых переменных (параметров) x_{ij_i}, отражающих события завершения ($x_{ij_i} = 1$) или незавершения ($x_{ij_i} = 0$) операции o_i на j_i-м ресурсе, $i \in \overline{1,n_o}$, $j_i \in \overline{1,n_r}$;
- y_i – булева функция, определяющая надежность выполнения i-й операции на j_i-м ресурсе;
- $p_{ij_i}(t)$ – вероятность выполнения операции o_i на j_i-м ресурсе в момент времени t (стационарный коэффициент готовности, характеризующий j_i-й ресурс в произвольный момент времени t по статистическим результатам его работы).

Функция y_i определяется следующей формулой:

$$y_i = \begin{cases} x_{ij_i}, & \text{если } \exists k: u_{ik} = 1,\ i \neq k, k \in \overline{1,n_o}, \\ x_{ij_i} \wedge z_i, & \text{в противном случае,} \end{cases} \qquad z_i = \bigwedge_{\forall k: w_{ik}=1, i \neq k} y_k. \tag{4}$$

В результате выполнения всех подстановок в соответствии с (3) и учетом информации о предшествовании операций, представленной матрицей U, мы получим булеву функцию

$$q_s = \bigwedge_{i:s_{li}=1} x_{ij_i},\ \ l \in \overline{1,n_l}, \tag{5}$$

определяющую надежность выполнения схемы S. Для получения вероятностного показателя надежности схемы S производится переход от функции (5) к вероятностной функции

$$P(t) = \prod_{i:s_{li}=1} p_{ij_i}(t) \tag{6}$$

с помощью соответствующих правил преобразования [34]. Функция (6) вычисляет вероятность (оценку надежности E_S^r) выполнения схемы S без резервирования ресурсов в момент времени t в процессе планирования. Определение вероятности выполнения схемы S с резервированием ресурсов рассмотрено в [34]. Однако резервирование зачастую приводит к снижению эффективности использования ресурсов. Поэтому в данной работе предложен подход к определению вероятности выполнения схемы S без резервирования дополнительных ресурсов.

Избыточность. В модели (1) может наблюдаться избыточность относительно вычисления одних и тех же параметров разными операциями, т.е. могут существовать такие i и j, что $PAR_i^{out} \cap PAR_j^{out} \neq \emptyset,\ \ i,j \in \overline{1,n_l}$. Поэтому схема S в общем случае может содержать несколько альтернативных сценариев решения задачи.

Модель (1) не имеет избыточности относительно вычисления одних и тех же параметров разными операциями, если выполняется следующее условие:

$$\bigvee_{j=1}^{n_p}(f_j \vee g_j) = 0,\ \ f_j = \bigvee_{i=1}^{n_o-1}\bigvee_{k=i+1}^{n_o}\left(r_{ij}^{out} \wedge r_{kj}^{out}\right),\ \ g_j = \bigwedge_{i=1}^{n_o} r_{ij}^{out}. \tag{7}$$

С целью обеспечения возможности устранения избыточности в процессе построения схемы решения задачи операции $o_1, o_2, \dots, o_{n_o} \in O$ снабжаются приоритетами $pr_{1j}, pr_{2j}, \dots, pr_{n_oj}$ относительно данного j-го параметра. Приоритеты операций автоматически определяются следующим образом:

$$pr_{ij} = \begin{cases} 0, & i \in \overline{1,n_o},\quad j \in \overline{1,n_p}: r_{ij}^{out} = 0, \\ \overline{i}, & \forall i \in \overline{1,n_o},\ \ j \in \overline{1,n_p}: r_{ij}^{out} = 1. \end{cases} \tag{8}$$

Очевидно, что чем меньше номер операции i, тем больше ее приоритет относительно j-го параметра в (8). Формируемая таким образом вещественная матрица приоритетов PR может быть изменена в дальнейшем разработчиком или конечным пользователем пакета вручную путем изменения мест параметров и операций в соответствующих списках параметров и операций. Такие изменения осуществляются с учетом экспертного опыта разработчиков и конечных пользователей пакета, обладающих алгоритмическими и схемными знаниями относительно операций пакета.

Новая непроцедурная постановка задачи. Для устранения избыточности мы используем следующую новую постановку задачи:

$$NPPF': S^{in}, S^{out}, PR_{j_1}, PR_{j_2}, \dots, PR_{j_l} \to S^o?, \quad (9)$$

где $j_1, j_2, \dots, j_l \in \overline{1, n_p}$.

При планировании на основе такой постановки задачи в модели (1) остаются операции, для которых выполняются условия $r^{out}_{i_k j_k} = 1$, $r^{out}_{i j_k} = 0$, где $i_k = \underset{\mu=1,|PR_k|}{\operatorname{argmax}} pr_\mu$, $i_k \in \overline{1, n_o}$, $j_k \in \overline{1, n_p}$, $i = \overline{1, n_o}$, $i \neq i_k$, $k = \overline{1, l}$, l – число параметров, значения которых вычисляются разными операциями.

Эти операции имеют наивысшие приоритеты относительно параметров $par_{j_1}, par_{j_2}, \dots, par_{j_l} \in PAR$. При этом остальные операции с меньшими приоритетами становятся недопустимыми и временно скрываются в модели (1) в процессе планирования. Тем самым матрицы S^{in} и S^{out} специализируются в соответствии с постановкой задачи. Таким образом, обеспечивается выполнение условия (7) и построение схемы с единственным сценарием решения задачи по ее постановке (9).

В случае, когда требуется построение схемы, включающей несколько сценариев решения задачи, применяется постановка задачи (3).

В целом модель (1) является развитием вычислительной модели, представленной в [35].

4. Алгоритмы планирования вычислений

Рассмотрим алгоритмы планирования схем с одним или несколькими сценариями решения задачи в РППП на рассмотренной выше модели (1).

Псевдокод алгоритма $A_1(O, S^{in}, S^{out}, n_o \to S, n_l)$ построения схемы с несколькими альтернативными сценариями решения задачи по ее постановке (3) представлен на рис. 1. Предполагается, что для каждой j-й операции из O известны PAR_j^{in} и PAR_j^{out}. Булевы переменные C_f и A_f используются в качестве флагов. Флаг $C_f = 1$ ($C_f = 0$) означает, что схема решения задачи построена (не построена). Флаг $A_f = 1$ ($A_f = 0$) означает, что на очередной итерации планирования, хотя бы одна операция была включена (ни одна операция не была включена) в схему.

После инициализации вспомогательных переменных в цикле выполняется проверка на возможность включения каждой j-й операции в схему S. После завершения цикла оцениваются текущие результаты планирования. Проверяем, разрешима проблема или неразрешима. В первом случае, мы готовим данные для следующей итерации планирования и приступаем к ней. Во втором случае, алгоритм завершает свою работу.

Алгоритм A_1 имеет вычислительную сложность $O(n_o)$.

В общем случае в результате выполнения алгоритма A_1 на модели (1), для которой условие (7) не выполняется, мы получим схему с несколькими сценариями решения задачи, которая может также включать ненужные вычисления. С целью обеспечения возможности редукции ненужных вычислений разработан алгоритм $A_2(O, S, S^{out}, n_o, n_l \to S)$, псевдокод которого показан на рис. 2. Предполагается, что для каждой j-й операции из O известны PAR_j^{in} и PAR_j^{out}.

В процессе работы алгоритма A_2 на каждом уровне схемы для каждой j-й операции происходит проверка вычисления ею значений параметров из целевого множества S^{out*}. Если проверка выполняется, то вспомогательное множество S^{in*} объединяется со множеством P_j^{in} входных параметров операции. В противном случае, операция удаляется из схемы решения задачи. После проверки всех операций i-го уровня схемы S^{out*} объединяется с S^{in*} $\forall i > 1$.

```
1  // Инициализация вспомогательных переменных
2  O* = O; S^in* = S^in; S^out* = ∅;
3  i = 1;
4  for j = 1..C step 1 do
5    s_ij = 0;
6  end do
7  C_f = 0; A_f = 0;
8  // Проверка возможности включения операций в схему
9  for j = 1..n_o step 1 do
10   if (o_j^* = 1) ∧ (PAR_j^in = ∅ ∨ PAR_j^in ⊆ S^in*) then
11     s_ij = 1; A_f = 1; S^out* = S^out* ∪ PAR_j^out; o_j^* = 0;
12   end if
13 end do
14 // Проверка достижения целевого множества параметров
15 S^in* = S^in* ∪ S^out*;
16 if S^out ⊆ S^in* then
17 // Задача разрешима
18   C_f = 1;
19 end if
20 // Проверка разрешимости задачи для очередной итерации планирования
21 if A_f ∨ C_f = 0 then
22 // Задача неразрешима
23   return NULL;
24 else
25   if A_f ∨ C̄_f = 0 then
26 // Схема построена
27     n_l = i;
28     return < S, n_l >;
29   else
30 // Подготовка данных для очередной итерации планирования
31     S^out* = ∅;
32     i = i + 1;
33     for j = 1..n_o step 1 do
34       s_ij = 0;
35     end do
36     A_f = 0;
37 // Переход на очередную итерацию планирования
38     goto 9;
39   end if
40 end if
```

Рис. 1. Псевдокод алгоритма A_1
Fig. 1. Pseudocode of the algorithm A_1

На заключительной стадии осуществляется удаление всех i-х строк матрицы S, для которых выполняется условие $\bigvee_{j=1}^{n_o} s_{ij} = 0, n_l = n_l - n_0$, где n_0 – это число удаленных строк. Удаление строк выполняется с помощью вспомогательной подпрограммы $Delete\_rows()$.

Алгоритм A_2 имеет вычислительную сложность $O(n_l \times n_o)$.

```
1  // Инициализация переменных
2  S^{in*} = ∅; S^{out*} = S^{out};
3  // Редукция ненужных вычислений
4  for i = n_l..1 step −1 do
5    for j = 1..n_o do
6      if s_ij = 1 step 1 then
7        if   PAR_j^{out} ∩ S^{out*} ≠ ∅ then
8          S^{in*} = S^{in*} ∪ PAR_j^{in};
9        else
10         s_ij = 0;
11       end if
12     end if
13   end do
14   if i > 1 then
15     S^{out*} = S^{out*} ∪ S^{in*}; S^{in*} = ∅;
16   end if
17 end do
18 // Удаление i-й строки матрицы S: ⋁_{j=1}^{n_o} s_ij = 0
19 Delete_rows(S, n_l → S, n_l);
20 return S;
```

Рис. 2. Псевдокод алгоритма A_2
Fig. 2. Pseudocode of the algorithm A_2

```
1  // Инициализация вспомогательных переменных
2  O* = O;
3  // Специализация множества O*
4  for k = 1..l step 1 do
5    i_k = argmax_{μ=1,|PR_k|} pr_μ;
6    for i = 1..n_o step 1 do
7      if (r_{ij_k}^{out} = 1) ∧ (i ≠ i_k) then
8        o_i^* = 0;
9      end if
10   end do
11 end do
12 // Специализация множеств R^{in} и R^{out}
13 Specialize(O*, R^{in}, R^{out} → R^{in}, R^{out});
14 // Применение алгоритма A_1
15 A_1(O*, S^{in}, S^{out}, n_o → S, n_l);
16 if S ≠ NULL then
17 // Схема построена. Применение алгоритма A_2
18   A_2(O*, S, S^{out}, n_o, n_l → S);
19   return S;
20 end if
21 // Задача неразрешима
22 return NULL;
```

Рис. 3. Псевдокод алгоритма A_3
Fig. 3. Pseudocode of the algorithm A_3

Для построения схемы с единственным сценарием решения задачи разработан алгоритм $A_3\left(O, R^{in}, R^{out}, S^{in}, S^{out}, n_o, PR_{j_1}, PR_{j_2}, \dots, PR_{j_l}, l \to S, n_l\right)$. Его псевдокод приведен на рис. 3.

После инициализации вспомогательного множества O^* осуществляется его специализация в соответствии с заданными приоритетами операций. Затем специализируются множества R^{in} и R^{out} входных и выходных параметров операций с помощью вспомогательной подпрограммы $Specialize()$. Применяется алгоритм A_1. Если схема S построена, то после применения алгоритма A_2 алгоритм A_3 завершает свою работу. В противном случае задача неразрешима. Разработчик или конечный пользователь РППП могут изменить условия постановки задачи ($S^{in}, S^{out}, PR_{j_1}, PR_{j_2}, \dots, PR_{j_l}$) и вновь запустить процесс планирования.

Алгоритм A_3 имеет вычислительную сложность $O(n_0 \times \max\{l, n_l\})$.

5. Пример

В данном разделе мы демонстрируем аспекты планирования вычисления на простом модельном примере. Пусть множества PR, O и M модели (1) заданы следующим образом: $PR = \{pr_1, pr_2, pr_3, pr_4, pr_5, pr_6\}$, $O = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$, $M = \{m_1, m_2, m_3, m_4, m_5, m_6\}$. Мы специфицируем операцию следующим образом:

$$operation\_name(a\ set\ of\ inputs \to a\ set\ of\ ouputs\ ).$$

Таким образом, операции имеют следующие спецификации: $o_1(pr_1 \to pr_2)$, $o_2(pr_2 \to pr_3)$, $o_3(pr_2 \to pr_3)$, $o_4(pr_2 \to pr_3)$, $o_5(pr_3 \to pr_4)$, $o_6(pr_4 \to pr_5)$, $o_7(pr_2 \to pr_6)$. Операции o_2, o_3 и o_4 вычисляют значение одного и того же параметра pr_3. Для этих операций автоматически формируется множество $PR_3 = \{1.50, 1.00, 0.75\}$ их приоритетов относительно параметра pr_3. Очевидно, что в модели имеется вычислительная избыточность. Модуль m_2 реализует две операции o_2 и o_7. На практике, это распространенная ситуация, когда в зависимости от качества или структуры значения параметра над ним выполняются концептуально разные операции, реализуемые одной и той же программой. В этом случае, с концептуальной точки зрения, значения выходных параметров могут иметь разный смысл.

Связи между объектами вышеупомянутых множеств представлены матрицами R^{in}, R^{out}, U и $\tilde{R}$:

$$R^{in} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&1&0&0&0&0 \end{bmatrix},\ R^{out} = \begin{bmatrix} 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{bmatrix},\ U = \begin{bmatrix} 0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 0&1&1&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&0 \end{bmatrix},\ \tilde{R} = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1 \\ 0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \end{bmatrix}.$$

Рассмотрим случай постановки задачи

$$NPPF: S^{in}, S^{out} \to S^o?$$

где $S^{in} = \{pr_1\}$ и $S^{out} = \{pr_4\}$.

Стадии планирования вычислений и построения схемы S представлены в табл. 1, где n – номер стадии (нулевая стадия – это подготовка исходных структур данных), x – случайное событие применения операции при выполнении схемы, $\hat{p}_1(x)$–$\hat{p}_7(x)$ – округленные вероятностные оценки включения операций o_1–o_7 в схему S, $\hat{E}$ – показатель неопределенности процесса планирования, представленный информационной энтропией:

$$\hat{E} = \sum_{i=1}^{m_j} \hat{p}_i(x) E_i(x),$$

$$E_i(x) = \begin{cases} 0, & \text{if } \hat{p}_i(\mathrm{x}) = 0, \\ -log_2 \hat{p}_i(x) & \text{otherwise}, \end{cases}$$

m_j – число операций – кандидатов на применение при выполнении схемы на j-й стадии планирования. Стадии 1-4 осуществляются в рамках метода прямой волны, реализуемого

алгоритмом A_1. Стадии 5-8 выполняются в рамках метода обратной волны, реализуемого алгоритмом A_2. На заключительной стадии работы алгоритма A_2 удаляются все строки матрицы S, все элементы которых равны 0.

Табл. 1. Планирование на основе $NPPF: S^{in}, S^{out} \rightarrow S^o?$

Table 1. Planning on the base of $NPPF: S^{in}, S^{out} \rightarrow S^o?$

n	S	S^o	$\hat{p}_1$	$\hat{p}_2$	$\hat{p}_3$	$\hat{p}_4$	$\hat{p}_5$	$\hat{p}_6$	$\hat{p}_7$	$\hat{E}$
0	[0 0 0 0 0 0 0]	$\emptyset$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.81
1	[1 0 0 0 0 0 0]	$\{o_1\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
2	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_7\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
3	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_7\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
4	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
5	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_7\}$	0.16	0.16	0.16	0.16	0.16	–	0.16	2.54
6	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_7\}$	0.20	0.20	0.20	0.20	+	–	0.20	2.32
7	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5\}$	0.25	0.25	0.25	0.25	+	–	–	2.00
8	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&0\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5\}$	+	0.33	0.33	0.33	+	–	–	1.58

В табл. 1 символы ‘+’ и ‘–‘ означают соответственно включение операции в схему и ее исключение из кандидатов на включение. В результате построена схема с тремя сценариями решения задачи, определяемыми применением альтернативных операций o_2, o_3 или o_4. Поэтому в схеме остается неопределенность, которая устраняется в дальнейшем в рамках тендера вычислительных работ.

Теперь рассмотрим случай постановки задачи

$$NPPF: S^{in}, S^{out}, PR_3 \rightarrow S^o?$$

Стадии планирования вычислений и построения схемы S представлены в табл. 2. Схема S строится с помощью алгоритма A_3. В рамках нулевой стадии происходит специализация множества O. Затем из алгоритма A_3 вызываются алгоритмы A_1 и A_2, которые соответственно реализуют методы прямой и обратной волн (этапы 1-4 и 5-8). В результате построена схема с единственным сценарием решения задачи. В этом случае неопределенность полностью устранена.

Табл. 2. Планирование на основе $NPPF: S^{in}, S^{out}, PR_3 \rightarrow S^o?$

Table 2. Planning on the base of $NPPF: S^{in}, S^{out}, PR_3 \rightarrow S^o?$

n	S	S^o	$\hat{p}_1$	$\hat{p}_2$	$\hat{p}_3$	$\hat{p}_4$	$\hat{p}_5$	$\hat{p}_6$	$\hat{p}_7$	$\hat{E}$
0	[0 0 0 0 0 0 0]	$\emptyset$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
1	[1 0 0 0 0 0 0]	$\{o_1\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32

2	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\end{bmatrix}$	$\{o_1, o_2, o_7\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
3	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_7\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
4	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_6, o_7\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
5	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_7\}$	0.25	0.25	–	–	0.25	–	0.25	2.00
6	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_7\}$	0.33	0.33	–	–	+	–	0.33	1.58
7	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_5\}$	1.00	+	–	–	+	–	–	0.00
8	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_5\}$	+	+	–	–	+	–	–	–

6. Повышение надежности вычислений путем применения схемы с несколькими сценариями решения задач

Избыточные вычисления могут успешно использоваться в том случае, когда происходят отказы определенных типов или заранее назначенные ресурсы недоступны. Например, это возможно при отказе или перегрузке необходимого ресурса.

Пусть схема решения задачи содержит альтернативный сценарий выполнения фрагмента операций, модули которого могут выполняться на другом ресурсе, возможно имеющем другие вычислительные характеристики. Например, он работает под управлением другой операционной системы. В этом случае мы можем выполнять остальные операции схемы, используя этот ресурс.

Для определения фрагмента схемы решения задачи, операции которого выполняются или ожидают своего запуска, введем следующие обозначения:

- $O_s \subseteq O$ – множество операций схемы S;
- $O_e \subset O_s$ – множество операций, которые выполнены, $n_e = |O_e|$ – число таких операций;
- $O_u = O_s \backslash O_e$ – множество операций, выполняющихся или ожидающих своего запуска, $n_u = |O_e|$ – число таких операций.

Схема S_r, включающая операции из O_u, является остаточной схемой решения задачи. $S_r^{in} = S^{in} \cup \left(\bigcup_{k=1}^{n_e} P_{i_k}^{out}\right)$ и $S_r^{out} = S^{out} \cap \overline{\left(S^{in} \cup \left(\bigcup_{k=1}^{n_e} P_{i_k}^{out}\right)\right)}$ – это множества ее входных и выходных параметров, где $i_k \in \overline{1, n_o}: o_{i_k} \in O_e$. Множество S_r^{in} включает промежуточные результаты вычислений.

Для определения остаточной схемы S_r мы используем понятие, введенное А.П. Ершовым [36]. Как правило, остаточная схема может включать несколько сценариев решения проблемы. Мы можем редуцировать схему, удалив избыточные операции относительно операции с выбранным приоритетом. Схемы, полученные таким образом для разных приоритетов операций, являются эквивалентными.

Будем считать, что две схемы S_r^1 и S_r^2 являются эквивалентными, если они удовлетворяют следующему условию:

$$S_r^{out} \subseteq \left(S_r^{in1} \cup \left(\cup_{k=1}^{n_1} P_{i_k}^{out}\right)\right) \cap \left(S_r^{in2} \cup \left(\cup_{l=1}^{n_2} P_{l_k}^{out}\right)\right),$$

где $O_s^1 \neq O_s^2$, $O_s^1, O_s^2 \subseteq O_s$, $o_{i_k} \in O_s^1$, $o_{i_l} \in O_s^2$, $S_r^{in1}, S_r^{in2} \subseteq S^{in}$, n_1 (n_2) – это число операций в множестве O_s^1 (O_s^2), $0 < k, l \leq n_u$.

Таким образом, в отличие от [37], в рамках управления распределенными вычислениями мы переходим от конкретных модулей к абстрактным операциям, используя приоритеты. Кроме того, мы уточняем логику принятия решения при выборе сценария дальнейшего решения задачи (рис. 4).

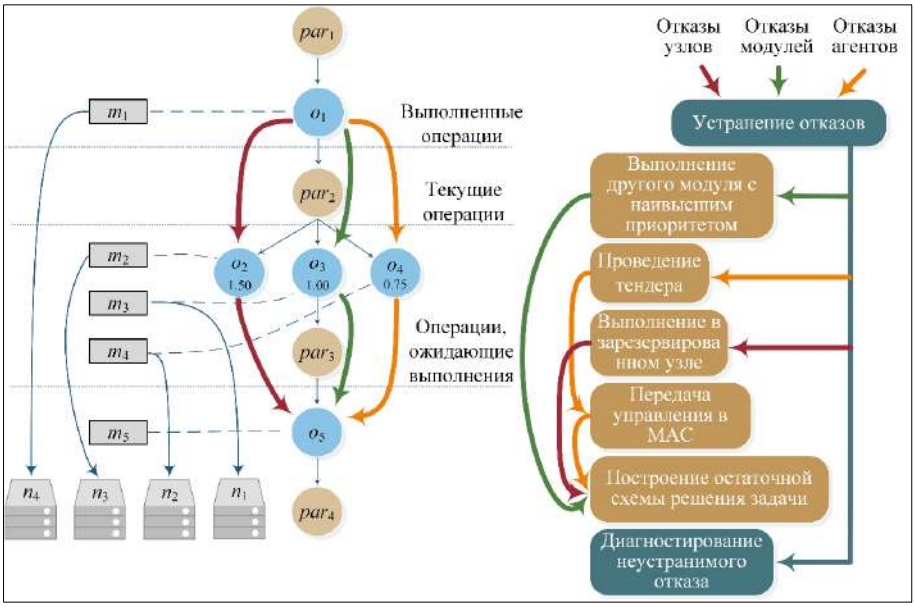

Рис. 4. Блок-схема выбора сценария
Fig. 4. Block diagram for selecting a scenario

На этой блок-схеме идентифицируется отказ. Если прерванная операция предыдущего сценария решения задачи не может быть выполнена, то выбирается сценарий с альтернативной операцией с наивысшим приоритетом.

Если имеются такая операция и доступный ресурс для ее выполнения, то осуществляется переключение на соответствующий сценарий и формируется остаточная схема решения задачи. Затем проводится тендер вычислительных работ с использованием рассмотренных выше оценок времени, стоимости и надежности схемы. После назначения ресурсов для выполнения всех операций схемы, начинается ее выполнение.

Мы сравниваем несколько ключевых характеристик MAC в отношении отказоустойчивости при выполнении схемы решения задачи с двумя традиционными метапланировщиками Condor DAGMan и GridWay. Среди них – поддержка отказоустойчивости в случае отказа ресурсов, системы и прикладного программного обеспечения, а также средняя продолжительность перезапуска. Мы моделируем выполнение потоков заданий, сгенерированных с использованием истории вычислений реальных потоков заданий для широкого спектра практических задач [38]. Мы используем имитационную модель гетерогенной распределенной вычислительной среды [39]. Она была адаптирована к модели (1).

Результаты сравнения представлены в табл. 3. Эксперименты показали, что все сравниваемые системы успешно устраняют проблемы в случае отказов ресурсов или системы. Процент не устраненных отказов этих типов очень низок.

Табл. 3. Результаты сравнения
Table 3. Comparison results

Метапланировщик / Meta-scheduler	**Поддержка отказоустойчивости при отказе ресурса / Fault-tolerance support in case of resource failure**	**Поддержка отказоустойчивости при отказе системы / Fault-tolerance support in case of system failure**	**Поддержка отказоустойчивости при отказе прикладного ПО / Fault-tolerance support in case of applied software failure**	**Средняя продолжительность перезапуска, сек. / Average restart makespan, s**
Condor DAGMAn	+/–	+/–	–	8.05
GridWay	+/–	+/–	–	8.11
MAC	+/–	+/–	+/–	6.23

В то же время Condor DAGMan и GridWay не имеют инструментов для обработки отказов прикладного ПО. Кроме того, такие метапланировщики не поддерживают избыточные обработку заданий с вычислительной избыточностью. Поэтому решение задачи под их контролем может быть прекращено в результате отказа ресурса.

В отличие от таких метапланировщиков, MAC может в некоторых случаях перезапускать вычисления за счет использования вычислительной избыточности. Наличие приоритетов операций значительно ускоряет принятие решений и перезапуск вычислительного процесса.

7. Заключение

Статья посвящена проблемам планирования вычисления в РППП. Исследование базируется на концептуальном моделировании вычислительных процессов.

В данной работе модифицирована и дополнена новыми понятиями предложенная ранее концептуальная модель РППП. В частности, введены приоритеты выполнения операций, которые автоматически формируются и пересчитываются при изменении спецификации модели. Предложена новая постановка задачи, в рамках которой используются такие приоритеты. При избыточности модели приоритеты обеспечивают возможность установить порядок выполнения операций, тем самым смягчая неопределенность при попытке построить схему с единственным сценарием решения задачи. Разработаны специализированные алгоритмы построения схем решения задач на модели с избыточностью и неопределенностью. Показана полезность избыточности модели для повышения надежности вычислений по сравнению с традиционными метапланировщиками.

В рамках будущих исследований предполагается реализовать автоматическое определение приоритетов операций на основе дополнительной информации о модулях, реализующих операции (оценках времени выполнения и размера передаваемых данных, статистики по отказам и др.). В настоящее время такая информацию уже собирается в процессе непрерывной интеграции программного обеспечения распределенных пакетов прикладных программ.

Список литературы / References

[1] Casanova H., Legrand A. et al. Heuristics for scheduling parameter sweep applications in grid environments. In Proc. of the 9th Heterogeneous Computing Workshop, 2000, pp. 349-363.

[2] Casavant T.L., Kuhl J.G. A Taxonomy of Scheduling in General–Purpose Distributed Computing Systems. IEEE Transactions on Software Engineering, vol. 14, issue 2, 1988, pp. 141-154.

[3] Черных А.Н., Бычков И.В. и др. Смягчение неопределенности при разработке научных приложений в интегрированной среде. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 151-171 / Tchernykh A., Bychkov I.V. et al. Mitigating Uncertainty in Developing Scientific Applications in Integrated Environment. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 151-172 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–11.

[4] Feoktistov A., Gorsky S. et al. Collaborative Development and Use of Scientific Applications in Orlando Tools: Integration, Delivery, and Deployment. Communications in Computer and Information Science, vol. 1087, 2020, pp. 18-32.

[5] Cardoso J, Sheth A. Semantic E-Workflow Composition. Journal of Intelligent Information Systems, vol. 21, issue 3, 2003, pp.191-225.

[6] Mineau G.W., Missaoui R., Godinx R. Conceptual modeling for data and knowledge management. Data & Knowledge Engineering, vol. 33, issue 2, 2000, pp. 137-168.

[7] Yu J., Buyya R. A taxonomy of workflow management systems for grid computing. Journal of Grid computing, vol. 3, issue 3-4, 2005, pp. 171-200.

[8] Fahringer T., Prodan R. et al. ASKALON: a Grid application development and computing environment. In Proc. of The 6th IEEE/ACM International Workshop on Grid Computing, 2005, pp. 1-10.

[9] Tschager T., Schmidt H.A. Condor, DAGwoman: enabling DAGMan-like workflows on non-Condor platforms. In Proc. of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, 2012, article no. 3, pp. 1-6.

[10] Amin K., Laszewski G. et al. GridAnt: a client-controllable grid workflow system. In. Proc. of the 37th Annual Hawaii International Conference on System Science, 2004, pp. 1-10.

[11] Carrion I.M., Huedo E., Llorente I.M. Interoperating Grid infrastructures with the GridWay metascheduler. Concurrency Computation, vol. 27, issue. 9, 2015, pp. 2278-2290.

[12] Barseghian D., Altintas I. et al. Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis. Ecological Informatics, vol. 5, issue 1, 2010, pp. 42-50.

[13] Missier P., Soiland-Reyes S. et al. Taverna, Reloaded. Lecture Notes in Computer Science, vol. 6187, 2010, pp. 471-481.

[14] Vahi K., Harvey I. et al. A General Approach to Real-Time Workflow Monitoring. In Proc. of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, 2012, pp. 108-118.

[15] Benedyczak K., Bala P. et al. Key aspects of the UNICORE 6 security model. Future Generation Computer Systems, vol. 27, issue 2, 2011, pp. 195-201.

[16] Extensible Markup Language (XML). Available at: https://www.w3.org/XML, accessed 13.07.2021.

[17] XML Process Definition Language. Available at: https://www.w3.org/TR/xmlschema-0, accessed 13.07.2021.

[18] Guizania K., Ghannouchia S.A. An approach for selecting a business process modeling language that best meets the requirements of a modeler. Procedia Computer Science, vol. 181, 2021, pp. 843-851.

[19] Mo'Minov B.B., Eshankulov, K. Modelling Asynchronous Parallel Process with Petri Net. International Journal of Engineering Advanced Technology, vol. 8, issue 5S3, 2019, pp. 400-405.

[20] Unified Modeling Language (UML) Diagrams. Available at: https://www.uml.org, accessed 13.07.2021.

[21] Deelman E., Vahi K. et al. Pegasus, a workflow management system for science automation. Future Generation Computer Systems, vol. 46, 2015, pp. 17-35.

[22] Blythe J., Deelman E. et al. The Role of Planning in Grid Computing. In Proc. Of the Thirteenth International Conference on Automated Planning and Scheduling, 2003, pp. 153–163.

[23] Matskin M., Tyugu E. Strategies of structural synthesis of programs and its extensions. Computing and Informatics, vol. 20, issue 1, 2001, pp. 1-26.

[24] Опарин Г.А., Новопашин А.П. Булевы модели и методы планирования параллельных абстрактных программ. Автоматика и телемеханика, вып. 8, 2008 г., стр. 166-175 / Oparin G.A., Novopashin A.P. Boolean models and planning methods for parallel abstract programs. Automation and Remote Control, vol. 69, no. 8, 2008, pp. 1423-1432.

[25] Новосельцев В.Б. Синтез параллельных рекурсивных программ в структурных функциональных моделях. Программирование, том 33, вып. 5, 2007 г., стр. 75-81 / Novoseltsev V.B. Synthesis of parallel recursive programs in structural functional models. Programming and Computer Software, vol. 33, no. 5, pp. 293-298.

[26] Malyshkin V.E., Perepelkin V.A. LuNA fragmented programming system, main functions and peculiarities of run-time subsystem. Lecture Notes in Computer Science, vol. 6873, 2011, pp. 53-61.

[27] Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. Наука. Сибирское отделение, 1988 г., 128 стр. / Valkovsky V.A., Malyshkin V.E. Synthesis of parallel programs and systems on computational models. Nauka. Siberian branch, 1988, 128 p. (in Russian).

[28] Gorsky S., Kostromin R. et al. Orlando Tools: Supporting High-performance Computing in Distributed Environments. In Proc. of the 6th International Conference on Information Technology and Nanotechnology, 2020, pp. 1-6.

[29] Feoktistov A., Tchernych A. et al. Knowledge Elicitation in Multi-Agent System for Distributed Computing Management. In Proc. of the 40th International Convention on information and communication technology, electronics and microelectronics, 2017, pp. 1350-1355.

[30] Bychkov I., Feoktistov A. et al. Machine Learning in a Multi-Agent System for Distributed Computing Management. In Proc. of the International Conference Information Technology and Nanotechnology. Session Data Science, CEUR-WS Proceedings, vol. 2212, 2018, pp. 89-97.

[31] Feoktistov A., Kostromin R., Tchernykh A. Agent Behavior Model for Distributed Computing Management in the Environment with Virtualized Resources. In Proc. of the 41st International Convention on information and communication technology, electronics and microelectronics, 2018, pp. 1153-1158.

[32] Feoktistov A.G., Basharina O.Yu. Predicting runtime of computational jobs in distributed computing environment. In Proc. of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments, CEUR-WS Proceedings, 2020, vol. 2638, pp. 109-117.

[33] Ryabinin I.A. Logical probabilistic analysis and its history. International Journal of Risk Assessment and Management, vol. 18, issue 3-4, 2015, pp. 256-265.

[34] Feoktistov A.G., Sidorov I.A. Logical-Probabilistic Analysis of Distributed Computing Reliability. In Proc. of the 39th International Convention on information and communication technology, electronics and microelectronics, 2016, pp. 247-252.

[35] Bychkov I., Oparin G. et al. Conceptual Model of Problem-Oriented Heterogeneous Distributed Computing Environment with Multi-Agent Management. Procedia Computer Science, vol. 103, 2017, pp. 162-167.

[36] Ershov A.P. On Mixed Computation: Informal Account of the Strict and Polyvariant Computation Schemes. In Control Flow and Data Flow: Concepts of Distributed Programming. Springer Study Edition, vol. 14, 1985, pp. 107-120.

[37] Feoktistov A., Kostromin R. et al. Multi-Agent Algorithm for Re-Allocating Grid-Resources and Improving Fault-Tolerance of Problem-Solving Processes. Procedia Computer Science, 2019, vol. 150, pp. 171-178.

[38] Tchernykh A., Feoktistov A. et al. Orlando Tools: Development, Training, and Use of Scalable Applications in Heterogeneous Distributed Computing Environments. Communications in Computer and Information Science, vol. 979, 2019, pp. 265-279.

[39] Bychkov I.V., Oparin G.A. et al. Multiagent control of computational systems on the basis of meta-monitoring and imitational simulation. Optoelectronics, Instrumentation and Data Processing, vol. 52, issue 2, 2016, pp. 107-112.

Информация об авторах / Information about authors

Александр Геннадьевич ФЕОКТИСТОВ – кандидат технических наук, доцент, заведующий лабораторией параллельных и распределенных вычислительных систем. Области исследования: распределенные пакеты программ, мультиагентные технологии, имитационное моделирование, складская логистика.

Alexander Gennadevich FEOKTISTOV – Ph.D., Associate Professor, Head of the Laboratory of Parallel and Distributed Computing Systems. Fields of research: distributed software packages, multi-agent technologies, simulation modeling and warehouse logistics

Роман Олегович КОСТРОМИН – кандидат технических наук, младший научный сотрудник. Область исследования: мультиагентные средства управления распределенными вычислениями.

Roman Olegovich KOSTROMIN – Ph.D., Junior Researcher. Field of research: multi-agent tools for distributed computing management.

Сергей Алексеевич ГОРСКИЙ – кандидат технических наук, старший научный сотрудник. Сфера научных интересов: параллельные и распределенные вычисления, математическое моделирование и сервис-ориентированное программирование.

Sergei Alexeevich GORSKY – Ph.D., Senior Researcher. Research interests: parallel and distributed computing, mathematical modeling, and service-oriented programming.

Игорь Вячеславович БЫЧКОВ – академик РАН, доктор технических наук, профессор, директор Института динамики систем и теории управления им. В.М. Матросова СО РАН. Области исследования: искусственный интеллект, геоинформационные системы, веб-технологии, математическое моделирование и облачные вычисления.

Igor Vyacheslavovich BYCHKOV – Academician of RAS, Ph.D., Professor, Director of the Matrosov Institute for System Dynamics and Control Theory of SB RAS. Fields of research: artificial intelligence, geoinformation systems, web-technologies, mathematical modeling, and cloud computing.

Андрей Николаевич ЧЕРНЫХ получил степень кандидата наук в Институте точной механики и вычислительной техники РАН. Он является профессором Центра научных исследований и высшего образования в Энсенаде, Нижняя Калифорния, Мексика. В научном плане его интересуют многоцелевая оптимизация распределения ресурсов в облачной среде, проблемы безопасности, планирования, эвристики и метаэвристики, интернет вещей и т.д.

Andrei Nikolaevitch TCHERNYKH received his PhD degree at the Institute of Precision Mechanics and Computer Engineering of the Russian Academy of Sciences. He is holding a full professor position in computer science at CICESE Research Center, Ensenada, Baja California, Mexico. He is interesting in grid and cloud research addressing multiobjective resource optimization, both, theoretical and experimental, security, uncertainty, scheduling, heuristics and meta-heuristics, adaptive resource allocation, and Internet of Things.

Ольга Юрьевна БАШАРИНА – кандидат технических наук, доцент, научный сотрудник ИДСТУ СО РАН, доцент Иркутского государственного университета, Сфера научных интересов: исследование операций, имитационное моделирование и распределенные вычисления.

Olga Yurevna BASHARINA – Ph.D., Research Officer of ISDCT SB RAS, Associate Professor of the Irkutsk State University. Research interests: operations research, simulation modeling and distributed computing.

DOI: 10.15514/ISPRAS-2022-34(1)-10

Алгоритмы обработки естественного языка для понимания семантики текста

Д.О. Жаксыбаев, ORCID: 0000-0001-6355-5431 <darhan.03.92@mail.ru>
Г.Н. Мизамова, ORCID: 0000-0002-1012-9700 <mizamgul@mail.ru>
Западно-Казахстанский аграрно-технический университет имени Жангир хана
090009, Республика Казахстан, город Уральск, улица Жангир-хана, 51

Аннотация. Векторное представление слов используется для различных задач автоматической обработки естественного языка. Множество методов существует для представления векторов слов, включая методы нейронных сетей Word2Vec и GloVe, а также классический метод латентно-семантического анализа LSA. Цель данной работа посвящена исследованию эффективности использования сетевых векторных методов LSTM для неклассической классификации высоты тона в текстах на русском и английском языках. Описаны характеристики векторных методов классификации слов (LSA, Word2Vec, GloVe), описана архитектура нейросетевого классификатора слов на основе LSTM и взвешены методы векторной классификации слов, представлены результаты экспериментов, вычислительных средств и их обсуждение. Лучшей моделью векторного представления слов является модель Word2Vec, учитывая скорость обучения, меньший размер корпуса слов для обучения, большую точность и скорость обучения нейросетевого классификатора.

Ключевые слова: обработка теста; ключевые слова; процедура отбора; вектор слов

Для цитирования: Жаксыбаев Д.О., Мизамова Г.Н. Алгоритмы обработки естественного языка для понимания семантики текста. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 141-150. DOI: 10.15514/ISPRAS–2022–34(1)–10

Natural Language Processing Algorithms for Understanding the Semantics of Text

D.O. Zhaxybayev, ORCID: 0000-0001-6355-5431 <darhan.03.92@mail.ru>
G.N. Mizamova, ORCID: 0000-0002-1012-9700 <mizamgul@mail.ru>
West Kazakhstan Agrarian and Technical University named after Zhangir Khan
51 Zhangir Khan Street, Uralsk, Republic of Kazakhstan, 090009

Abstract. Representation of vector words is used for various tasks of automatic processing of natural language. Many methods exist for the representation of vector words, including methods of neural networks Word2Vec and GloVe, as well as the classical method of latent semantic analysis LSA. The purpose of this paper is to investigate the effectiveness of using network vector methods LSTM for non-classical pitch classification in Russian and English texts. The characteristics of vector methods of word classification (LSA, Word2Vec, GloVe) are described, the architecture of neural network classifier based on LSTM is described and vector methods of word classification are weighted, the results of experiments, computational tools and their discussion are presented. The best model for vector word representation is Word2Vec model given the training speed, smaller word corpus size for training, greater accuracy and training speed of neural network classifier.

Keywords: test processing; keywords; selection procedure; word vector

For citation: Zhaxybayev D.O., Mizamova G.N. Natural Language Processing Algorithms for Understanding the Semantics of Text. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 141-150 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–10

1. Введение

Обработка текста на естественном языке приобретает все большее значение и вызывает большой интерес в социологии, маркетинге, лингвистике, психологии и других областях человеческой деятельности. Быстрое распространение Интернета является одной из основных движущих сил этой тенденции.

Автоматизация обработки текстовой информации требует представления текста в виде числовой модели, которая представляет текст как вектор его характеристик – описание его атрибутов. чтобы получить

Метод представления символов (называемый текстовым индексированием) основан на двух подходах: представлении текста в виде «мешка слов» и представлении текста в виде последовательности слов, представленных в виде векторов [1].

Представляя текст в виде «мешка слов», текст можно представить в виде вектора словарного запаса обучающей выборки. Каждый элемент вектора представляет вес соответствующего слова в словаре. Весом может быть частота слова в тексте или какой-то другой более сложный показатель. Это представление не учитывает порядок слов в тексте, что является одним из основных недостатков этого метода.

Многообещающей тенденцией в обработке естественного языка является векторное представление слов, которое позволяет представить текст в виде последовательности векторов, соответствующих словам в тексте.

Простейшим способом преобразования слова в вектор является одиночное кодирование [2], то есть кодирование каждого слова вектором длины, равной размеру словаря обучающей выборки. Каждый из этих векторов состоит из нуля и единицы, соответствующей позиции слова в словаре. Это представление малоэффективно с точки зрения запоминания и, прежде всего, не дает никакого объяснения смыслового значения слов и не позволяет сравнивать слова по смысловой близости, что очень затрудняет классификацию.

Существует несколько методов получения векторных представлений слов, позволяющих создать низкоразмерное векторное представление для каждого слова в корпусе (наборе текстов) и сохранить контекстуальное сходство слов. Во-первых, известны два метода нейронных сетей: Word2Vec [3, 4], разработанный Google, и Global Vectors (GloVe) [5], разработанный в Стэнфордском университете.

Было показано, что эти два метода превосходно справляются с задачами обработки естественного языка, такими как распознавание похожих слов и т. д. Однако также было показано [5], что классические методы, в частности метод латентного семантического анализа LSA [6], могут быть более полезными, чем, например, Word2Vec. Это связано с тем, что Word2Vec с самого начала учится на низкоразмерных векторах и не использует всю информацию из учебного корпуса слов. Было показано [7], что метод LSA более надежен и существенно не зависит от размера корпуса.

Цель данной работы является изучение эффективности использования этих методов при автоматической обработке текстов на русском и английском языках. Обнаружение тона – это специфическая классификационная задача, которая включает в себя количественную оценку и категоризацию мнений, выраженных в тексте, с целью определения того, является ли отношение автора к той или иной теме, продукту положительным, отрицательным или нейтральным [5]. Искусственные нейронные сети (ИНС) оказались хорошим решением таких проблем. Архитектура используемых ИНС может различаться. ИНС прямого распространения [6] являются контекстно-зависимыми, то есть они могут учитывать только определенное количество слов вокруг данного слова, чтобы определить его значение повторно.

2. Методология исследования

В данной статье сравниваются три метода: LSA – классический метод, Word2Vec как наиболее популярный метод в русскоязычном сообществе среди программистов и GloVe как популярная альтернатива, мало описанная в русскоязычных источниках. Далее «слово» заменяется словом «терм», подчеркивая, что это текстовый элемент с определенной смысловой нагрузкой.

2.1 Семантический анализ

Метод латентного семантического анализа (LSA) основан на принципах факторного анализа [6], который позволяет, в том числе, выявить латентные (скрытые) отношения между терминами и текстами (документами), определяющие присущую документам тематику и термины. Каждая тема характеризуется своим значением в смысловом отношении документов и терминов. Размерность векторного представления уменьшена за счет исключения из лингвистической модели тем с наименьшей смысловой нагрузкой.

Первым шагом является преобразование основного текста (документов) в массив терминов. Элементами этой матрицы обычно являются веса TF-IDF [6].

TF (частотный термин) – это отношение nt (количества вхождений слова t) к общему количеству слов в документе:

$$\boldsymbol{TF(t,d) = \frac{n_t}{\sum_k n_k}.}$$

Кроме того, согласно теореме о методах вычитания сингулярных разложений [7], полученную вещественную прямоугольную матрицу можно вычесть как произведение трех матриц:

$$A = USV^t,$$

где матрицы U и V ортогональны, а S – диагональная матрица, диагональ которой содержит сингулярные значения матрицы A.

Если в матрице S оставляем только наибольшие сингулярные значения k и матрицы U и V – только столбцы, соответствующие этим значениям, то произведение $\hat{S}$ и $\hat{V}$ матриц T есть наилучшее приближение исходной матрицы A к матрице $\hat{A}$ с рангом k.

Эта матрица представляет собой структуру ассоциативной зависимости, которая лежит в основе исходной матрицы, и каждый терм (строка в матрице $\hat{U}$) и каждый документ (строка в матрице $\hat{V}$) представлены в виде векторов в общем пространстве с размерностью k (пространство гипотез) (см. рис. 1). k подбирается опытным путем и зависит от количества исходных документов. Векторные термины можно использовать как векторные слова.

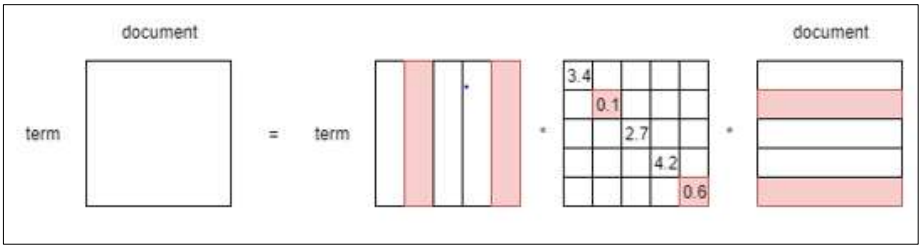

Рис. 1. Векторная модель LSA
Fig. 1. LSA vector model

2.2 Word2Vec

Word2Vec [3] –метод, позволяющий предсказывать контекст слова по заданному слову (метод Skip-Gram) или наоборот предсказывать слово по заданному контексту (метод CBOW). В этом случае скрытый слой нейронной сети проецирует слова на небольшой вектор.

Общая структура нейронной сети (см. рис. 2):

- входной слой, который получает векторы контекстных слов в случае CBOW или векторы одномерных кодирующих слов v в случае Skip-Gram;
- скрытый слой с несколькими нейронами k;
- выходной слой с иерархической функцией активации Softmax [4] или отрицательной выборкой [4], выход которой сходится во время обучения к векторам слов в случае CBOW или к векторам контекстных слов в случае Skip-Gram, в одноразмерное кодирование v.

После обучения выходные данные скрытого слоя используются для получения вектора слов размера k.

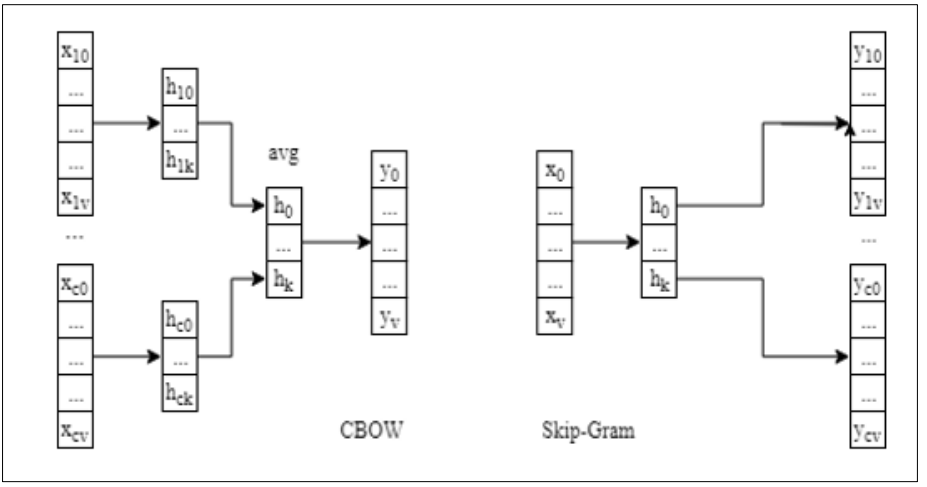

Рис. 2. Метод Word2Vec
Fig. 2. Word2Vec Method

Хотя этот метод охватывает только статистические свойства текстов, кажется, что обученная модель Word2Vec охватывает некоторые семантические свойства слов.

2.3 GloVe

GloVe [5] представляет собой модель, сочетающую в себе свойства метода уменьшения сингулярности и метода Word2Vec.

Первым шагом является создание матрицы состава X для тренировочного датасета. Значение элемента X_{ij} показывает, как часто слово j появляется в контексте слова i. Семантическая близость слов i и j оценивается с помощью отношения вероятности их совместного появления в контексте k:

$$F(w_i, w_j, \widehat{w}_k) = \frac{P_{ik}}{P_{jk}} = \frac{X_{ik}/\sum_m X_{im}}{X_{jk}/\sum_n X_{jn}},$$

где w_i, w_j – векторы слов, $\widehat{w}_k$ – вектор контекста.

Семантическая близость этих векторов определяется их скалярным произведением. С помощью преобразований и допущений в [5] показано, что целью формирования GloVe является изучение векторов таким образом, чтобы их скалярное произведение было близко к логарифму вероятности совпадения слов в выборке формирования. Чтобы уменьшить важность совпадений редких (менее информативных, шумных и ненадежных) или ненадежных слов, а также уменьшить важность очень распространенных (например, «есть») совпадений, Pennington и др.

В [5] в качестве целевой функции обучения (функции потерь) используется модель взвешенной регрессии по методу наименьших квадратов (1), где w_i – вектор

существительного, $\widehat{w}_j$ – вектор контекста, b_i и b_j – масштабирование значений отклонения слов существительного и контекста соответственно, V - размер словаря:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \hat{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

2.4 Классификация текстов

Основными этапами разработки и формирования текстового классификатора являются:

- предварительная обработка текста,
- индексация текста,
- разработка и формирование классификатора,
- оценка качества классификации.

Предварительная обработка текста обязательно предполагает токенизацию, т. е. выделение в тексте языковых единиц, так называемых лексем, слов или терминов.

Обработка текста может также включать

- сопоставление слов в верхнем и нижнем регистре, чтобы избежать семантических различий между похожими словами в разных регистрах;
- исключение таких слов, как союзы, предлоги, артикли, т.е. слова не похожие на слово, имеющие одинаковое значение, например, к ним относятся устранение семантически нейтральных слов (стоп-слов);
- удаление или замена цифр их текстовыми эквивалентами;
- устранение знаков препинания и лишних пробелов;
- нормализация слов: стемминг или лемматизация, замена всех слов стандартизированной формой.

Возможны и другие методы предварительной обработки слов. Выбор сценария предварительной обработки зависит от типа решаемой задачи и характеристик выборки. В результате выделяются все значимые слова, которые собственно и определяют смысловое значение текста.

Классификация текста требует индексации текста, т.е. описания каждого свойства текста, подлежащего классификации.

Всеобъемлющие и точные измерения, а также конкретные тестовые образцы могут использоваться для оценки качества классификации.

2.5 Определение тональности

Ставится задача сравнить эффективность методов представления векторов слов, рассмотренных в предыдущем разделе, при определении тональности текста.

Структура сети показана на рис. 3 и содержит следующие уровни.

1) Входной слой, который получает документ в виде последовательности лексемно-термальных индексов (метки).
2) Слой векторного словаря с фиксированными весами, где веса слоя последовательности определяются матрицей, строка которой i является векторным представлением термина i; используется для выбора соответствующего вектора термов; матрица строится из векторов, сгенерированных моделью векторного словаря.
3) Корректирующий слой, который изменяет определенный процент выходных значений предыдущего слоя, чтобы избежать чрезмерного обучения.
4) Сверточная сеть – это последовательность сверточных слоев и подвыборок с функцией максимума; он используется для уменьшения количества входных параметров

следующего слоя и для повышения скорости обучения.

5) Слой LSTM как основной классификатор.
6) Выходной слой с сигмоидальной функцией активации.

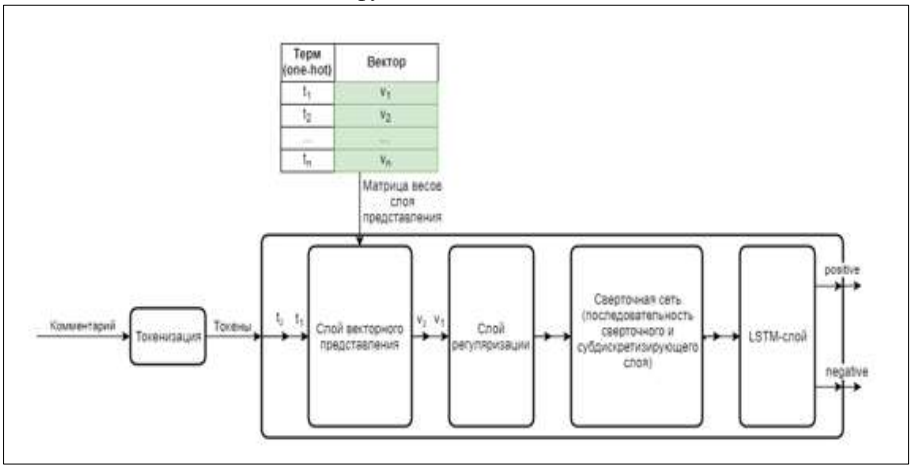

Рис. 3. Тональность текста
Fig. 3. Sentiment of the text

3. Результаты и обсуждения

В данной работе мы используем корпус обзоров социальных сетей на русском языке [4] и корпус обзоров фильмов на английском языке на IMDB [5].

Для каждого корпуса были проведены эксперименты по обучению разного количества текстов и с разными сочетаниями параметров вектора и шаблона слова:

– количество обучающих текстов – 1000, 10000, 25000;
– размерность векторного представления – 50, 150, 300;
– количество периодов или итераций обучения LSA – 2, 8, 15.

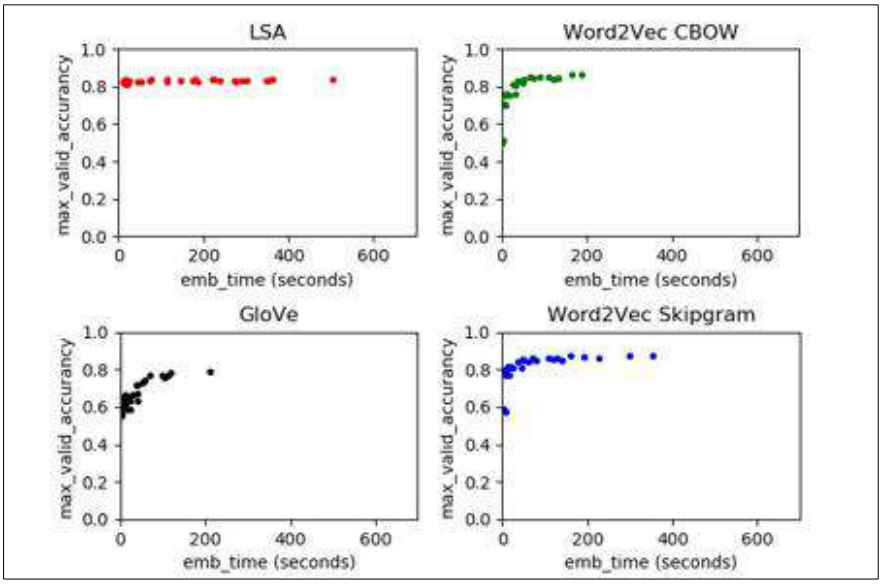

Рис. 4. Диаграммы рассеяния для англоязычного текста
Fig. 4. Scattering charts for English text

Для Word2Vec и GloVe использовался размер окна контекста: 2 слоя левее и 2 слова правее основного слова. Для Word2Vec использовался отрицательный размер выборки. Для каждого теста фиксировалось время обучения векторной модели слов (на рис. 4, 5 показан график по оси абсцисс). Чтобы проверить эффективность сети LSTM в определении тона текста, она была обучена 13 000 аннотаций. Проверки правдоподобия были выполнены для 3250 аннотаций. Поэтому максимальная точность классификации, полученная для тестовой выборки за 10 эпох обучения, была определена как максимально допустимая точность классификации (ось ординат на рис. 4, 5).

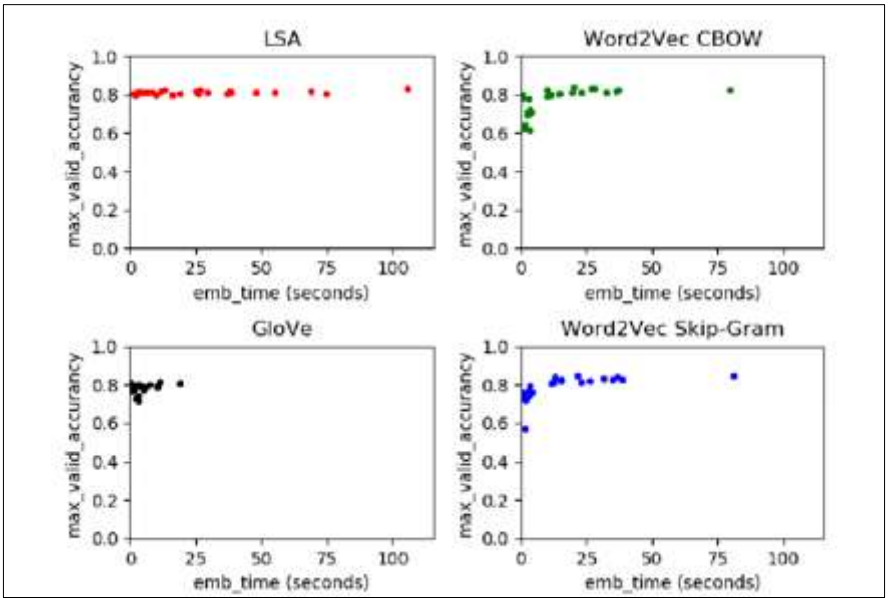

Рис. 4. Диаграммы рассеяния для русскоязычного текста
Fig. 4. Scattering charts for Russian text

У английских текстовых классификаторов были самые высокие средние значения точности (за исключением тестов, в которых использовался GloVe), но это может быть связано и с тем, что они были длиннее – обучающих данных было больше.

LSA является наиболее надежной моделью с точки зрения переменных параметров точности классификатора, так как все эксперименты дали примерно одинаковую точность классификации. LSA дал лучшие результаты по точности классификации (в среднем 81,2 % для русских текстов и 82,9 % для английских текстов), но в то же время самые плохие результаты по времени классификации как для русского, так и для английского языков. Широко используемые методы Word2Vec Skip-Gram и Word2Vec CBOW вели себя примерно одинаково для обоих корпусов: точность классификации снижалась при небольшом количестве учебных текстов. Скорость их обучения ниже, чем у моделей LSA, но значительно ниже, чем у моделей GloVe.

4. Заключение

Метод Word2Vec показал более высокую среднюю точность классификации и меньшую чувствительность к размеру обучающей выборки, несмотря на меньшую длину текста в русском корпусе. Эксперимент, в котором использовался этот метод, показал наивысшую точность. В случае с английским корпусом этот метод показал лучшие результаты по точности классификации и наименьшую чувствительность к размеру обучающей выборки.

Согласно [5], эксперты обычно сходятся во мнении о точности того или иного текста в 79% случаев. Эффективным считается классификатор, определяющий тональность текста с

точностью более 70%. В случае русского корпуса все классификаторы на основе LSA и GloVe могут быть использованы на практике по этому критерию (см. рис. 5). Как показывают вычислительные эксперименты, Word2Vec сложно применять к небольшим корпусам, но в других случаях он работает хорошо. В случае английского корпуса на практике можно использовать все классификаторы на основе LSA (см. рис. 4). Word2Vec также подходит для небольших корпусов, но в большинстве случаев работает хорошо. Большинство экспериментов с использованием модели GloVe показали неприемлемые результаты.

Обратите внимание, что для каждого набора параметров шаблонов представления вектора слов был выполнен только один тест, поэтому результаты велики.

Точность классификатора на основе сети LSTM также зависит от метода токенизации и обработки токенизации текста (обрезка текста или лемматизация, удаление стоп-слов, сокращение одиночного слова и т. д.), параметров векторного представления модели, а также структуру и параметры самой нейронной сети. Исследования, описанные в этой статье, были выполнены с ограниченным числом значений параметров для моделей представления векторов слов. Поэтому стоит рассмотреть различные способы обработки текстов и проверить эффективность моделей векторного представления с другими параметрами и другими сетевыми архитектурами LSTM, чтобы улучшить результаты. Стоит обратить внимание и исследовать эффективность GloVe.

Заслуживает внимания эффективность модели Word2Vec в других российских корпусах, так как она показывает очень хорошие результаты при анализе тонов. Однако примеров применения этой системы в русскоязычной литературе не обнаружено.

Выбор лучшей текстовой векторной модели также основывается на критериях классификации. Их критерии и приоритеты неодинаковы для разных задач, поэтому модель необходимо выбирать в соответствии с ограничениями, которым должна соответствовать выбранная модель.

Список литературы / References

[1] Chilakapati A. Word Bags vs Word Sequences for Text Classification. URL: https://towardsdatascience.com/word-bags-vs-word-sequences-for-text- classification-e0222c21d2ec, accessed 01.02.2022.

[2] Brownlee J. How to One Hot Encode Sequence Data in Python. URL: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python accessed 05.02.2022.

[3] Le Q., Mikolov T. Distributed Representations of Sentences and Documents. In Proc. of the 31st International Conference on Machine Learning, 2014, pp. 1188-1196.

[4] Mikolov T., Chen K. et al. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781, 2013, 12p.

[5] Pennington J., Socher R., Manning C. GloVe: Global Vectors for Word Representation. In Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532-1543.

[6] Landauer T.K., Foltz P.W., Laham D. Introduction to Latent Semantic Analysis. Discourse Processes, vol. 25, issue 2-3, 1998, pp. 259-284.

[7] Altszyler E., Sigman M., Slezak D.F. Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. arXiv preprint arXiv:1610.01520, 14 p.

Информация об авторах/ Information about authors

Дархан Оракбаевич ЖАКСЫБАЕВ – магистр педагогических наук, окончил аспирантуру по специальности "Информационные системы" в Евразийском национальном университете имени Л. Гумилева, преподаватель кафедры информационных систем.

Darkhan Orakbayevich ZHAXYBAYEV – Master of Pedagogical Sciences, graduated from the PhD studies by the specialty "Information Systems" at the Eurasian National University named after L. Gumilyov, Lecturer of the Department of Information Systems.

Гулбаршын Нурлановна МИЗАМОВА – магистр технических наук, окончила аспирантуру по специальности "Информационные технологии и телекоммуникации", преподаватель кафедры информационных систем.

Gulbarshyn Nurlanovna MIZAMOVA – Master of Technical Sciences, graduated from the post-graduated studies in the specialty "Information technology and telecommunications" (PhD), Lecturer of the Department of Information.

DOI: 10.15514/ISPRAS-2022-34(1)-11

О комбинированном алгоритме обнаружения заимствований в текстовых документах

[1] *К.Ф. Сафин,* ORCID: 0000-0001-9891-5513 <*kamil.safin@phystech.edu*>
[2,3] *Ю.В. Чехович,* ORCID: 0000-0002-5204-5484 <*chehovich@ap-team.ru*>
[1] *Московский физико-технический институт,*
141701, Россия, Московская область, г. Долгопрудный, Институтский переулок, д.9
[2] *Федеральный исследовательский центр "Информатика и управление" РАН*
119333, Россия, Москва, Вавилова, д.44, кор.2
[3] *Компания "Антиплагиат"*
121205, Россия, Москва, тер. Сколково инновационного центра, Большой б-р, д. 42 стр. 1

Аннотация. Поиск заимствований в текстовом документе по отношению к обширной коллекции потенциальных источников является вычислительно тяжелой задачей. При этом существуют так называемые внутренние методы поиска заимствований, которые не используют внешний корпус, а анализируют исключительно проверяемый документ. Эти методы не отличаются точностью, но обеспечивают довольно высокую производительность. В работе предложен комбинированный подход к обнаружению текстовых заимствований, основанный на использовании внутренних методов для выявления высокооригинальных документов, проверка которых по внешней коллекции не требуется. Предлагаемый алгоритм призван разгрузить систему поиска заимствований по внешней коллекции, отфильтровывая документы с высокой степенью оригинальности. В работе предлагается алгоритм поиска внутренних заимствований, описываются результаты вычислительных экспериментов.

Ключевые слова: обработка естественного языка; обнаружение заимствований; внутренние заимствования; поиск выбросов в статистике; антиплагиат

Для цитирования: Сафин К.Ф., Чехович Ю.В. О комбинированном алгоритме обнаружения заимствований в текстовых документах. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 151-160. DOI: 10.15514/ISPRAS–2022–34(1)–11

Combined method for plagiarism detection in text documents

[1] *K.F. Safin,* ORCID: 0000-0001-9891-5513 <*kamil.safin@phystech.edu*>
[2,3] *Y.V. Chehovich,* ORCID: 0000-0002-5204-5484 <*chehovich@ap-team.ru*>[1]
[1] *Moscow Institute of Physics and Technology,*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia
[2] *Federal Research Center "Computer Science and Control" RAS*
44 Vavilova St., Moscow, 119333, Russia
[3] *"Antiplagiat" Company*
42 Bol'shoy Blvd., Moscow, 121205, Russia

Abstract. There are two global approaches to the problem of searching plagiarism in the text: external and intrinsic search. The first approach implies search through an external collection of documents that could have been used for text reuse. The second approach, on the contrary, does not use any external data, but analyzes the text by itself. It is proposed to combine these two approaches to speed up the search for text plagiarism. With a large flow of documents that need to be checked, the outer corpus search system processes each document and finds plagiarised blocks in each document, if there are any. However, intrinsic search could be used to determine the fact of plagiarism. Thus, it is possible to reduce the number of documents for the expensive

procedure for searching for plagiarism by the outer corpus. Moreover, in an isolated analysis of a single document, there is no need to try to find specific blocks of plagiarism, this procedure is considered as a unique indicator of the originality of the document. If the overall originality is at a low level, then this document should be sent for a more detailed and accurate check. The proposed method allows to filter texts with a high rate of originality that do not need additional verification.

Keywords: natural language processing; plagiarism detection; intrinsic plagiarism; outliers detection; antiplagiat

For citation: Safin K.F., Chehovich Y.V. Combined method for plagiarism detection in text documents. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 151-160 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–11

1. Введение

1.1 Обзор литературы

Поиск заимствований в текстовых документах является сложной, но в то же время востребованной задачей, особенно в академической и студенческой средах [1, 2, 3].

Можно выделить два глобальных подхода к задаче поиска заимствований в тексте: поиск внешних заимствований (external plagiarism detection) и поиск внутренних заимствований (intrinsic plagiarism detection). Поиск внешних заимствований представляет собой поиск по внешней коллекции документов, которые могли быть использованы в качестве источника заимствования. Такой подход в том или ином виде сводится к попарному сравнению исследуемого документа с каждым документом из коллекции.

Коллекция текстовых документов, по которой происходит поиск внешних заимствований, как правило, довольно большая, а значит и поиск по ней является тяжелой вычислительной задачей. Как правило, тексты представляют в виде перекрывающихся словесных n-грамм (т.н. шинглов), которые впоследствии сравнивают с n-граммами анализируемого документа [4]. Промышленные инструменты, работающие на таком принципе сравнения документов, показывают высокую точность при поиске заимствований в текстовых документах [5]. Такой метод работает только в случае дословного заимствования фрагмента текста. Однако существуют методы обфускации (маскирования) заимствованных фрагментов, например, перефразирование или перевод текстового фрагмента из документа на другом языке. Конечно, системы поиска заимствований умеют находить и перефразирования [6] и переводные заимствования [7], однако это требует дополнительных расходов. Во-первых, требуется больше времени и вычислительных ресурсов на проверку одного документа, а во-вторых, необходимо значительно расширять текстовую коллекцию потенциальных источников.

Поиск внутренних заимствований же, наоборот, не использует внешнюю коллекцию потенциальных источников, а анализирует текст изолированно. При поиске анализируются различные стилистические, синтаксические, орфографические особенности текста.

Поиск внутренних заимствований обычно рассматривается как полноценный инструмент обнаружения текстовых заимствований. То есть, в результате работы алгоритма должны быть указаны конкретные фрагменты текста, которые были заимствованы [8]. Анализируемый текст при таком подходе, как правило, разбивается на отдельные сегменты. Например, текст делится на предложения [9], или определяется некоторая ширина шага, в соответствии с которой текст разделяется на сегменты одинаковой длины [10]. Полученные сегменты сравниваются со всем текстом и делается вывод о заимствовании для каждого сегмента. Для сравнения сегментов используются различные признаки, например, частота символьных n-грамм, из которых состоит текст [11, 12], или грамматические [13] и синтаксические признаки [14]. Иногда используются векторные представления, полученные с помощью нейронных сетей [15]. Довольно часто решается более общая задача диаризации авторов, в рамках

которой нужно определить авторство для каждого фрагмента текста [16, 17]. Методы поиска внутренних заимствований, в силу ограничения на анализ только исследуемого текста, не отличаются высокими показателями точности [18].

1.2 Актуальность работы

Сравнивая эти два подхода, можно сделать вывод, что методы поиска заимствований по внешней коллекции являются точными, но ресурсоемкими, а методы поиска внутренних заимствований – гораздо менее точными, но не сильно требовательными к ресурсам. При этом, в периоды пиковой нагрузки (например, во время сессии у студентов), система поиска по внешней коллекции может перестать справляться со входящим потоком документов для проверки, что приведет либо к сильной задержке ответа, либо к отказу от проверки. Оба случаи крайне нежелательны со стороны системы проверки. Самый простой способ ускорить работу заключается в уменьшении количества проверок (например, отказ от поиска переводных заимствований) или в сокращении коллекции потенциальных источников заимствований. И то и другое сильно скажется на качестве поиска заимствований в каждом рассматриваемом документе.

В такой ситуации кажется логичным не упрощать работу точной, но ресурсоемкой системы, а каким-то образом сократить поток входящих документов. Так как основной целью работы системы является выявление документов с высоким процентом заимствований, то было бы логично сокращать поток за счет высокооригинальных (т.е. с малой долей заимствований) документов. Для этой цели предлагается использовать подход по поиску внутренних заимствований. Как было сказано, в качестве самостоятельного инструмента, такой подход имеет очень низкое качество работы. Но его можно использовать как грубый фильтр перед более точной проверкой, который будет отсеивать документы, которым не нужна детальная экспертиза.

Причем при изолированном анализе отдельно взятого документа не нужно пытаться найти конкретные блоки заимствований, эта процедура рассматривается как своеобразный показатель оригинальности документа. В случае, если общая оригинальность на низком уровне, то этот документ стоит отправить на более детальную и точную проверку. Если же документ имеет высокую степень оригинальности, его можно пропустить и не отдавать на детальную проверку.

2. Постановка задачи

Пусть D – коллекция текстовых документов:

$$D = \{d_i\}_{i=1}^{N}.$$

Каждый документ d_i этой коллекции нужно отнести к одному из двух классов:

- класс 0 – класс высокооригинальных документов,
- класс 1 – класс документов с заимствованиями.

Под высокооригинальным документом понимается документ, содержащий малое количество заимствований из любых других текстов (или не содержащий их вовсе). Соответственно, под документом с заимствованиями понимается текст, содержащий большое число вставок из других текстов.

2.1 Критерии качества

Основная цель предлагаемого алгоритма – отфильтровывать высокооригинальные документы, не пропуская при этом документы с заимствованиями. Поэтому, при оценке качества алгоритма важны, прежде всего, следующие показатели:

- полнота класса документов с заимствованиями;

- количество отфильтрованных высокооригинальных документов.

Под полнотой класса (recall) понимается следующая величина:

$$Recall = \frac{TP}{TP + FN},$$

где TP, FN – true-positive и false-negative объекты. То есть это тексты, которые верно отнесены алгоритмом к нужному классу и, наоборот, тексты, которые неверно отнесены алгоритмом к противоположному классу, соответственно. Таким образом, полнота класса – это доля верно найденных алгоритмом документов из всех документов этого класса.

Второй показатель (количество высокооригинальных документов) важен, так как можно привести пример крайнего случая, когда все документы относятся к классу документов с заимствованиями. Тогда полнота класса документов с заимствованиями будет равна 1, но ни одного документа отфильтровано не будет, и нагрузка на систему поиска внешних заимствований не уменьшится.

Количество отфильтрованных документов можно рассматривать как полноту класса высокооригинальных документов. То есть, нам важна полнота обоих классов, однако полнота класса документов с заимствованиями важнее. Можно ввести вспомогательную метрику, которую будем использовать при настройке гиперпараметров алгоритма:

$$Recall_\beta = \beta \cdot Recall_1 + Recall_0, \quad (1)$$

где $Recall_1$ и $Recall_0$ – полнота класса документов с заимствованиями и высокооригинальных документов соответственно. Для того, чтобы полнота класса 1 была в приоритете, весовой коэффициент берется β больше единицы

3. Описание алгоритма

Общую логику работы предлагаемого алгоритма можно представить в виде псевдокода:

Алгоритм определения факта заимствования в тексте

```
Require: text
     statsList ← [ ]
     text ← preprocess(text)
     segmentsList ← getSegments(text)
     for segment in segmentsList do
         segmentVector = vectorize(segment)
         stat ← calcStat(vector)
         statsList.append(stat)
     end for
     outliersCount ← 0
     for stat in statsList do
         if stat > statT hreshold then
             outliersCount+ = 1
         end if
     end for
     if outliersCount > outliersThreshold then
         return 'text is not original'
     else
         return 'text is original'
     end if
```

Алгоритм состоит из следующих основных этапов:

- предобработка текста;
- сегментация текста;
- векторизация сегментов;
- расчет статистик для сегментов;
- обнаружение выбросов в ряде статистик.

3.1 Предобработка текста

Сначала текст проходит процедуру предобработки. Используются стандартные техники обработки естественного языка: удаление редких символов, удаление стоп-слов (слов, не несущих смысловую нагрузку), приведение слов к начальной форме. Конкретные техники предобработки и их параметры подбираются при настройке алгоритма на конкретном корпусе документов.

3.2 Сегментация текста

Под сегментацией текста понимается процедура разбиения текста d на сегменты s_j:

$$d = \bigcup_{j=1}^{m} s_j.$$

При этом сегменты могут быть пересекающимися и, наоборот, иметь нулевое пересечение. Для каждого сегмента затем будет рассчитываться некоторая статистика, поэтому сегментирование должно удовлетворять некоторым условиям.

Разбиение на сегменты должно быть достаточно мелким, чтобы можно было детектировать выброс в ряде статистик, и значение статистики на некотором сегменте сильно отличалось от остальных значений. С другой стороны, размер отдельного сегмента должен быть достаточно велик, чтобы можно было посчитать адекватную статистику.

Самые популярные стратегии сегментации, которые применимы в данном случае:

- разбиение по параграфам;
- разбиение окном с фиксированным шагом.

В процессе настройки гиперпараметров алгоритма, выбирается стратегия сегментации, а также ее аргументы (ширина окна и размер шага).

3.3 Векторизация сегментов

Каждый сегмент текста подвергается процедуре векторизации. Для построения векторного представления сегмента используются частоты словесных и символьных *n*-грамм.

Под символьной или словесной *n*-граммой понимается последовательность из n символов или слов в тексте. Каждой *n*-грамме ω в тексте d ставится в соответствие число:

$$freq_\omega = \frac{cnt(\omega)}{\sum_{\omega'} cnt(\omega')} \log\left(\frac{m}{seq(m)}\right), \quad (2)$$

где $cnt(\omega)$ – число вхождений ω в текст d, m – число сегментов в тексте, $seq(m)$ – число сегментов, содержащих ω.

Вектор сегмента формируется из рассчитанных величин (2) для всех уникальных n-грамм в тексте. Если *n*-грамма есть в тексте, но отсутствует в сегменте, то значение (2) равно 0.

Тип рассматриваемых *n*-грамм (символьные или словесные) выбирается при настройке гиперпараметров алгоритма на конкретном корпусе документов.

3.4 Подсчет статистик и нахождение аномалий

Для рассматриваемого текста строится ряд статистик путем подсчета некоторой статистики для каждого сегмента текста. В качестве статистики выбрано расстояние от вектора сегмента s_j до усредненного вектора всех сегментов $\bar{s}$:

$$s = \frac{1}{m}\sum_{j=1}^{m} s_j.$$

Тип расстояния между векторами также выбирается при настройке алгоритма. Выбор происходит из следующих вариантов:

- евклидово расстояние;
- косинусное расстояние;
- нормированное евклидово расстояние.

Полученный ряд статистик сглаживается скользящим средним фиксированной ширины.

В полученном ряде статистик выполняется поиск выбросов. Под выбросом подразумевается значение статистики, которое превышает некоторый заданный порог (который подбирается при настройке гиперпараметров). По количеству выбросов в тексте принимается решение об оригинальности документа.

4. Вычислительный эксперимент

Для проверки качества предложенного алгоритма было проведено два вычислительных эксперимента. Первый эксперимент был проведен на корпусе англоязычных документов, подготовленных в рамках конкурса по обнаружению текстовых заимствований PAN-2020. Для проведения второго эксперимента был использован корпус русскоязычных текстов Paraplag [19], специально составленный для проверки алгоритмов поиска текстовых заимствований.

В рамках каждого эксперимента производилась настройка гиперпараметров описанного алгоритма. Для настройки, данные разбивались на обучающую и тестовую выборки, в размерах 70% и 30% от всего корпуса соответственно. Настройка гиперпараметров производилась с помощью кроссвалидации на трех разбиениях обучающей выборки. В качестве целевой метрики использовалась предложенная метрика (1).

4.1 Описание данных

В первой части вычислительного эксперимента используется корпус текстов, подготовленных и размеченных в рамках конкурса PAN-2020 [20]. Корпус содержит документы на английском языке. Каждый документ может содержать от 0 до 10 вставок текста другого авторства.

Корпус состоит из двух частей. Первая часть представляет из себя узкоспециализированный набор документов – все документы в ней посвящены теме технологий. Вторая часть корпуса является набором текстов различной тематики (путешествия, философия, экономика, история и т.д.). Это сделано для того, чтобы была возможность протестировать предлагаемые алгоритмы на устойчивость к смене тематики при работе с документами. Количество документов с заимствованиями примерно равно количеству высокооригинальных документов.

Для второй части эксперимента был использован русскоязычный корпус тексов, содержащий документы с заимствованиями Paraplag [19]. Корпус представляет из себя набор текстов (эссе), в которые авторы намеренно добавляли заимствования из других документов. В качестве высокооригинальных документов представлены источники этих заимствований

(статьи из энциклопедий). Доля документов с заимствованиями относительно всего корпуса около 15%.

4.2 Результаты

Для оценки итогового качества полученного алгоритма, была рассчитана полнота каждого из целевых классов на тестовой выборке. Результаты экспериментов приведены в табл. 1.

Табл..1. Результаты эксперимента
Table 1. Experiment results

Название корпуса	Язык	Доля класса 1 в корпусе	Полнота класса 0	Полнота класса 1
PAN-2020	английский	50%	10%	94%
Paraplag	русский	15%	32%	97%

Видно, что на обоих корпусах полнота класса 1 близка к 100%. Это значит, что малая часть документов, которым необходима детальная проверка с помощью системы поиска внешних заимствований, будет неправомерно отфильтрована. При этом, часть высокооригинальных документов будет правильно отсеяна, что снизит нагрузку на систему.

Примеры работы алгоритма на конкретных текстах приведены на рис. 1 и рис. 2. Синими линиями представлены значения рядов статистик для каждого текста, красной горизонтальной – верхний порог статистики, выше которого она считается выбросом.

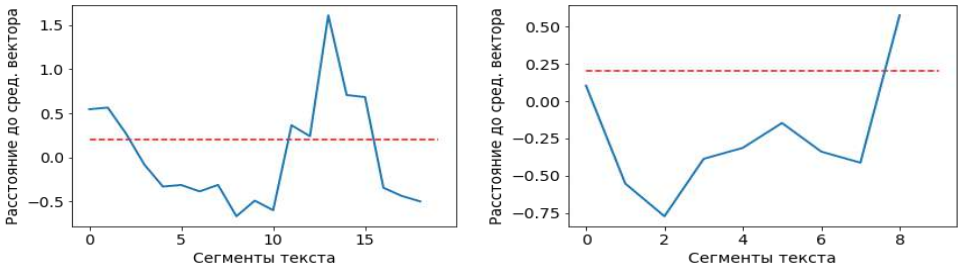

Рис. 1. Пример работы на тексте с заимствованиями (слева) и на высокооригинальном тексте (справа). Английский корпус документов
Fig. 1. Algorithm results on plagiarised text (left) and on original one (right). English corpus

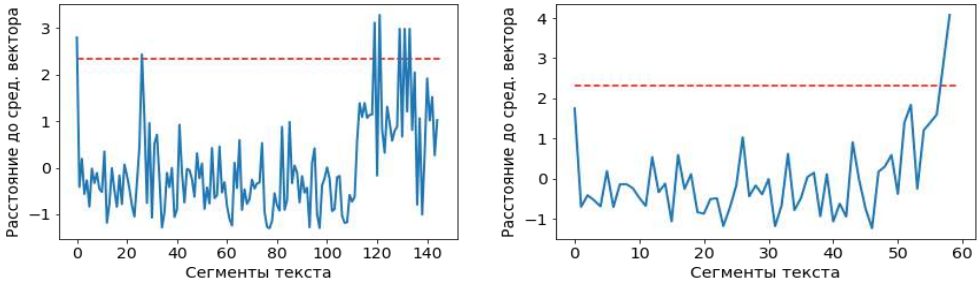

Рис. 2. Пример работы на тексте с заимствованиями (слева) и на высокооригинальном тексте (справа). Русский корпус документов
Fig. 2. Algorithm results on plagiarised text (left) and on original one (right). Russian corpus

5. Заключение

В статье предложен алгоритм по обнаружению факта заимствований в тексте. При этом алгоритм анализирует текст изолированно, не используя внешнюю коллекцию возможных источников заимствований. Для установления факта заимствования, текст сегментируется, и для каждого сегмента рассчитывается статистика, основанная на частотах распределения n-грамм. В полученном ряде статистик происходит поиск выбросов, и по их количеству делается вывод о степени оригинальности текста.

Предложенный алгоритм был настроен и протестирован на корпусах английских и русских текстов. В обоих случаях алгоритм корректно отбирает высокооригинальные документы, оставляя при этом документы с заимствованиями для дальнейшей проверки. Таким образом, алгоритм удовлетворяет выдвинутым к нему требованиям по качеству и может быть использован в качестве первичного отбора документов при использовании высоконагруженной системы поиска заимствований по внешнему корпусу.

Как было сказано, основная цель предложенного алгоритма заключается в фильтрации высокооригинальных документов, для которых не требуется детальная проверка. Конечно, в идеальном случае все поступающие на проверку документы должны проходить полноценную проверку. Однако реальность такова, что при высокой нагрузке, система поиска внешних заимствований может сильно задерживать ответ или пропускать документы. В такой постановке кажется логичным пожертвовать малым (частью документов, с низкой долей заимствований), чтобы сохранить работоспособность системы.

Предложенный алгоритм как раз и осуществляет такую логику. При сравнительно небольшом количестве пропущенных документов с заимствованиями (около 3% для русского корпуса) удалось сократить поток документов для обработки почти на треть.

Список литературы / References

[1] Никитов А.В., Орчаков О.А., Чехович Ю.В. Плагиат в работах студентов и аспирантов: проблема и методы противодействия. Университетское управление: практика и анализ, no. 5, 2012 г., стр. 61-68 / Nikitov A.V., Orchakov O.A., Chehovich Yu.V. Plagiarism in works of undergraduate and graduate students: problem and methods of counteraction. University Management: Practice and Analysis, no. 5, 2012, pp. 61-68 (in Russian).

[2] Stein B., Koppel M., Stamatatos E. Plagiarism analysis, authorship identification, and near-duplicate detection PAN'07. SIGIR Forum, vol. 41, no. 2, 2007, pp. 68–71.

[3] Chekhovich Y.V., Khazov A.V. Analysis of duplicated publications in Russian journals. Journal of Informetrics, vol.16, issue 1, 2022, article no. 101246.

[4] Зеленков И.В., Сегалович И.В. Сравнительный анализ методов определения нечетких дубликатов для Web-документов. Труды 9-ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» (RCDL'2007), 2007 г., стр. 166-174 / Zelenkov I.V., Segalovich I.V. Comparative analysis of methods for determining fuzzy duplicates for Web documents. In Proc. of the 9th All-Russian Scientific Conference «Digital Libraries: Advanced Methods and Technologies, Digital Collections» (RCDL'2007), 2007, pp. 166-174 (in Russian).

[5] Журавлев Ю.И., Рудаков К.В. и др. Система распознавания интеллектуальных заимствований «Антиплагиат». Математические методы распознавания образов, том 12, no. 1, 2005 г., стр. 329-332 / Zhuravlev Yu.I., Rudakov K.V. et al. The system of recognition of intellectual borrowings «Anti-plagiarism». Mathematical methods of pattern recognition, vol. 12, no. 1, 2005, pp. 329-332 (in Russian).

[6] Socher R., Huang E.H.-C. et al. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In Proc. of the 24th International Conference on Neural Information Processing Systems, 2011, pp. 801-809.

[7] Кузнецова Р.В., Бахтеев О.Ю., Чехович Ю.В. Методы обнаружения переводных заимствований в больших текстовых коллекциях. Информатика и её применения, том 15, no. 1, 2021 г., стр. 30–41 / Kuznetsova R.V., Bakhteev O.Yu., Chekhovich Yu.V. Methods of cross-lingual text reuse detection in large textual collections. Informatics and Applications, vol. 15, no. 1, 2021, pp. 30-41 (in Russian).

[8] Meier zu Eissen S., Stein B. Intrinsic Plagiarism Detection. Lecture Notes in Computer Science, vol. 3936, 2006, pp. 565-569.

[9] Zechner M., Muhr M. et al. External and Intrinsic Plagiarism Detection Using Vector Space Models. In Proc. of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse, CEUR Workshop Proceedings, vol. 502, 2009, pp. 47-55.
[10] Oberreuter G., L'Huillier G. et al. Outlier-Based Approaches for Intrinsic and External Plagiarism Detection. Lecture Notes in Computer Science, vol. 6882, 2011, pp. 11-20.
[11] Stamatatos E. Intrinsic Plagiarism Detection Using Character n-gram Profiles. In Proc. of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse, CEUR Workshop Proceedings, vol. 502, 2009, pp. 38–46.
[12] Bensalem I., Rosso P., Chikhi S. Intrinsic Plagiarism Detection using Ngram Classes. In Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1459-1464.
[13] Tschuggnall M., Specht G. Countering plagiarism by exposing irregularities in authors grammars. In Proc. of the European Intelligence and Security Informatics Conference, 2013, pp. 15-22.
[14] Романов А.С., Мещеряков Р.В., Резанова З.И. Методика проверки однородности текста и выявления плагиата на основе метода опорных векторов и фильтра быстрой корреляции. Доклады Томского государственного университета систем управления и радиоэлектроники, no. 2(32), 2014 г., стр. 264-269 / Romanov A.S., Mescheryakov R.V., Rezanova Z.I. Plagiarism detection and text homogeneity checking technique based on one-class support machine and fast correlation-based filter. Proceedings of TUSUR University, no. 2(32), 2014, pp. 264-269.
[15] Safin K., Kuznetsova R. Style Breach Detection with Neural Sentence Embeddings. In Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 1866, 2017, 7 p.
[16] Kuznetsov M., Motrenko A., Kuznetsova R., Strijov V Methods for intrinsic plagiarism detection and author diarization. In Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, CEUR Workshop Proceedings, vol. 1609, 2016, 8 p.
[17] Gillam L., Vartapetiance A. Quite Simple Approaches for Authorship Attribution, Intrinsic Plagiarism Detection and Sexual Predator Identification. In Working Notes for CLEF 2012 Conference, CEUR Workshop Proceedings, vol. 1178, 2012, 12 p.
[18] Potthast M., Eiselt A. et al. Overview of the 3rd International Competition on Plagiarism Detection. Working Notes for CLEF 2011 Conference, CEUR Workshop Proceedings, vol. 1171, 2011, 10 p.
[19] Sochenkov I.V., Zubarev D.V., Smirnov I.V. The ParaPlag:: Russian dataset for paraphrased plagiarism detection. In Proc. of the International Conference "Dialogue 2017", 2017, 13 p.
[20] Zangerle E., Mayerl, M. et al. PAN20 Authorship Analysis: Style Change Detection. Available at: https://doi.org/10.5281/zenodo.3660984.

Информация об авторах / Information about authors

Камиль Фанисович САФИН – аспирант. Сфера научных интересов: прикладной анализ данных, обработка естественного языка, поиск текстовых заимствований, методы оптимизации.

Kamil Fanisovich SAFIN – PhD student. Research interests: applied data analysis, natural language processing, text plagiarism detection, optimization methods.

Юрий Викторович ЧЕХОВИЧ – к.ф.-м.н., заведующий отделом Федерального исследовательского центра "Информатика и управление" РАН, исполнительный директор компании "Антиплагиат". Сфера научных интересов: прикладные задачи анализа данных, алгоритмы и системы обнаружения заимствований в текстовых документах, моделирование транспортных систем.

Yury Victorovich CHEHOVICH – Head of Department, Federal Research Center for Informatics and Control, Russian Academy of Sciences, CEO at Antiplagiat Company. Research interests: applied problems of data analysis, algorithms and systems for text plagiarism detection, transport systems modeling.

DOI: 10.15514/ISPRAS-2022-34(1)-12

Решение проблемы хранения траекторий молекулярной динамики в СУБД

И.В. Лихачев, ORCID: 0000-0002-4926-5654 <ilya_lihachev@mail.ru>
Институт математических проблем биологии РАН – филиал ИПМ им. М.В. Келдыша РАН
142290, Московская область, г. Пущино, ул. проф. Виткевича, д.1

Аннотация. В работе решается проблема хранения траекторий молекулярной динамики в реляционных и нереляционных базах данных. Традиционный подход организации структуры реляционных таблиц не подходит для хранения траекторий ввиду появления большого числа записей в одной таблице. Описано, каким образом можно структурировать данные в СУБД класса NoSQL. Затем эти идеи переносятся на реляционную СУБД MySQL.

Ключевые слова: СУБД; SQL; NoSQL; MongoDB; MySQL; моделирование молекулярной динамики; траектории молекулярной динамики; TAMD; анализатор траекторий молекулярной динамики; большие данные

Для цитирования: Лихачев И.В. Решение проблемы хранения траекторий молекулярной динамики в СУБД. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 161-172. DOI: 10.15514/ISPRAS–2022–34(1)–12

Благодарности: Автор благодарит Балабаева Н.К. за критические замечания, а также Сычугова А.А. и Сулимову В.В. за то, что доверили вести курс «Технология хранения и обработки больших данных» в Тульском государственном университете.

Solving the problem of storing trajectories of molecular dynamics in a DBMS

I.V. Likhachev, ORCID: 0000-0002-4926-5654 <ilya_lihachev@mail.ru>
The Institute of Mathematical Problems of Biology RAS - the Branch of Keldysh IAM of RAS
1, Professor Vitkevich St., 142290, Pushchino, Moscow Region, Russia

Abstract. The paper solves the problem of storing molecular dynamics trajectories in relational and non-relational databases. The traditional approach to organizing the structure of relational tables is not suitable for storing trajectories due to the appearance of a large number of records in one table. It is described how best to place data in the NoSQL class DBMS. These ideas are then transferred to the MySQL relational DBMS.

Keywords: DBMS; SQL; NoSQL; MongoDB; MySQL; molecular dynamics simulation; molecular dynamics trajectories; TAMD; Trajectory Analyzer of Molecular Dynamics; Big Data

For citation: Likhachev I.V. Solving of the problem of storing trajectories of molecular dynamics in a DBMS. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 161-172 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–12

Acknowledgements. The author thanks Balabaev N.K. for critical remarks, as well as Sychugov A.A. and Sulimova V.V. for being entrusted to conduct the course "Technology of storage and processing of big data" at Tula State University.

1. Введение

Метод моделирования молекулярной динамики (МД) является общепринятым методом исследования биологических структур. Результатами МД расчётов служат траектории молекулярной динамики – записи координат атомов через определённые, обычное одинаковые, промежутки времени. К настоящему моменту времени во многих лабораториях накоплено большое количество файлов траекторий. С ними работают при помощи программ анализа траекторий, таких как VMD [1], [2], TAMD [3-5], PyMol [6], [7]. Программы анализа траекторий, как правило, способны воспроизводить траекторию в виде молекулярного кино, а также рассчитывать некоторые характеристики вдоль траектории. Характеристики могу быть простыми, такие как расстояние между атомами, так и более сложными, например, вычисления матрицы ковариации.

К настоящему моменту времени ощущается нужда в средстве централизованного хранения и анализа траекторий. Это естественный этап развития вида программного обеспечения. Многие в нынешнем известные как гигантские распределенные, сетевые, клиент-серверные (в данном контексте это синонимы) программные комплексы изначально создавались как приложения для одного компьютера. И лишь с развитием локальных вычислительных сетей программные продукты стали поддерживать сетевой режим работы.

Когда речь идёт о большом количестве информации, которую необходимо сделать доступной для большого числа пользователей, разумно задуматься в первую очередь о хранении этой информации в системе управления базами данных (СУБД). Настоящая статья посвящена именно проблеме хранения траекторий молекулярной динамики в различных СУБД.

Предполагается, что программы анализа траекторий МД – не являющиеся предметом данной работы – могут подключаться напрямую к СУБД по сети в случае десктоп-приложений, либо быть разделены на клиент и сервер, который, в свою очередь, будет иметь подключение к серверу баз данных.

2. Представление данных в реляционной СУБД

Реляционные СУБД завоевали свою популярность ещё с конца 1970-х годов. Среди самых популярных стоит отметить Oracle [8], MySQL [9], [10], Microsoft SQL Server [11], PostgreSQL [12] и даже Microsoft Access. Все эти СУБД поддерживают размещение данных в связанных таблицах, доступ к которым может быть получен с применением языка SQL (Structured Query Language, структурированный язык запросов).

Попытка хранить данные в реляционной СУБД изначально выглядит естественной. Многие программисты владеют навыками работы именно с реляционными базами данных. При применении языка SQL доступ к этим базам в первом приближении выглядит практически одинаково. Под доступом понимаются операции CRUD, как их сейчас принято называть (create, read, update, delete – создание, выборка/чтение, обновление, удаление).

Табл. 1. Структура таблицы Names
Table 1. The structure of table Names

Поле	**Тип**	**Описание**
Id	Int, Первичный ключ	Идентификатор имени
Name	строка	Название эксперимента
Natoms	Int	Количество атомов
Nframes	Int	Количество кадров
Typename	char	Тип элемента: dir – каталог exp – данные об эксперименте (траектория молекулярной динамики)
Parent	Int	Ссылка на родителя

Мы будем использовать СУБД MySQL [9], [10] для иллюстрации. Выбор обусловлен бесплатной лицензией и широкой распространённостью данной СУБД.

Итак, траектория молекулярной динамики – это прежде всего файл – именованная область данных. Предлагается использовать отдельную таблицу Names (табл. 1) для хранения сведений о самой траектории. Остальные таблицы будут ссылаться на траекторию по её первичному ключу – уникальному идентификатору id.

Здесь следует рассказать о траектории более подробно. Траектория – это информация о координатах каждого атома в конкретный момент времени. Атомов фиксированное количество *natoms*. У каждого атома три координаты *x, y, z*. Моменты времени в траектории фиксируются только определённые, с необходимой частотой, заданной в МД-эксперименте. Зафиксированный в траектории момент времени будем называть кадром (фреймом). Всего кадров *nframes*.

Разрабатываемая система Trajectory Commander поддерживает хранение траекторий в древовидном каталоге, подобном организации файлов в файловой системе. Для отделения траекторий от каталогов служит поле *typename*, которое принимает значение *dir* в случае каталога, либо *exp* в случае хранения траектории. Поле *parent* ссылается на идентификатор (поле *id*) родительского каталога. В случае корневого каталога используется значение 0 в качестве родителя.

Таблица *Trj* (табл. 2) будет содержать данные непосредственно самих траекторий. Для данной траектории *id_name*, в заданном кадре *nframe*, у атома под номером *natom* будут храниться координаты *x, y, z*.

Табл. 2. Структура таблицы Trj
Table 2. The structure of table Trj

Поле	**Тип**	**Описание**
id_name	Int, внешний ключ	Идентификатор имени
Nframe	int	Номер кадра
Natom	int	Номер атома
X	float	координаты
Y		
Z		

SQL-запрос получения координат второго кадра определенной траектории может выглядеть как:

```
SELECT x, y, z,
FROM Trj
WHERE id_name=$trj_id AND nframe=2
ORDER BY natom;
```

Для выбора координат только двух атомов *atom1* и *atom2* можно написать запрос:

```
SELECT x, y, z,
FROM Trj
WHERE id_name=$trj_id AND (natom=$atom1 OR natom=$atom2)
ORDER BY nframe, natom;
```

Сортировка обязательна, чтобы кадры и атомы шли строго по порядку.

Как правило, траектории молекулярной динамики не хранят сведения о типах атомов, аминокислот и прочую метаинформацию. Она будет идентична от кадра к кадру. Вместо того в директориях с МД-расчетами всегда присутствуют PDB-файлы – файлы Protein Data Bank в одноимённом формате. Здесь мы не будет рассматривать формат PDB. Хранение его в СУБД не вызывает трудностей. Приведём лишь соответствующую таблицу (табл. 3), каждая строчка которой должна ссылаться своим внешним ключом на траекторию.

Табл. 3. Структура таблицы PDB
Table 3. The structure of table PDB

Поле	**Тип**	**Описание**
id_name	Int, внешний ключ	Идентификатор имени
Natom	int	Номер атома
Atomstring	Str	Строчка из файла без координат

Хранение информации о PDB-файлах необходимо для программ анализа траекторий. Будем считать, что такой файл удобно подавать этим программам в неизменном виде.

2.1. Проблема выбора реальных данных

Объемы траекторий весьма разные. Различно как количество атомов, так и количество кадров. Так при количестве атомов 1000 и 1000 кадров (1 нс при частоте записи 1 кадр на 1 пс, либо 10 нс при частоте записи каждые 10 пс) мы получаем 1 миллион строк. Стоит загрузить 10 таких траекторий и размер таблицы Trj возрастает до 10 миллионов строк. А если хранить 100 траекторий по 1000 кадров, по 10 000 атомов в кадре, то получаем 1 миллиард строк.

Заметим, что приведённые числа служат скорее прикидкой сверху, чем снизу. Многогигабайтные файлы траекторий – это норма, равно как и системы из десятков тысяч атомов.

В тестовой базе порядка 24 млн строк таблицы *Trj*. Запрос выборки одного кадра без использования индексов занимает порядка 2 минут. При включении индексации по всем полям, что учувствуют в запросе это время сокращается до 5 секунд, что говорит о грамотном применении индексов. Однако для воспроизведения молекулярного кино это недостаточно. 1 кадр в секунду неприемлем. Особенно учитывая тот факт, то 24 млн строк – это совсем небольшая база.

Возможно, проблему можно решить, прибегнув к NoSQL базам данных?

3. СУБД класса NoSQL

Сам термин NoSQL можно воспринимать по-разному. Это и базы данных, не поддерживающие реляционную модель, и не поддерживающие язык запросов SQL. Нам больше нравится определение Not only SQL, подчёркивающее, что этот класс СУБД предлагают нечто иное, чем реляционные СУБД [13].

Стоит также отметить, что СУБД начала 1990-ых годов чем-то напоминали этот новый класс СУБД. Язык SQL не был в то время распространён. СУБД того времени поддерживали многие операции, доступ к которым программисты получали при помощи оригинального API, своего собственного языка, применяемого именно с конкретной СУБД. Яркий тому пример – FoxPro для DOS, поддерживающий и мульти индексные файлы, показывающий прекрасное быстродействие на вычислительных машинах класса Pentium. Эта СУБД (или язык, если угодно) перешел в версию Visual FoxPro for Windows. Стал поддерживать SQL, но потерял свою популярность, не выдержал конкуренции. Факт наличия статей в 2020-2021 годах [14], [15] вызывает удивление. Данный абзац лишь отражает одно из мнений авторов, а не четкие определения: к СУБД можно напрямую обращаться при помощи API, без использования SQL-запросов. Несомненно, это быстрее, т.к. не нужно тратить время на синтаксический разбор операторов.

Согласно рейтингу Top-5 СУБД [16], самая распространённая СУБД класса NoSQL – это MongoDB [17], [18]. Позиционируется как документно-ориентированная СУБД. Под документом понимается нечто, не поддающееся строгой классификации. Точнее, имеется в виду документ в формате JSON (JavaScript Object Notation), хранящийся в формате BSON (Binary JavaScript Object Notation), т.е. в бинарном виде для экономии дискового

пространства. Несмотря на своё название, данный формат может использоваться практически в любом языке программирования как ассоциативный массив.

БД для хранения траекторий может состоять из двух таблиц (двух коллекций, переходя на терминологию MongoDB):

Коллекция Names

```
/* Элемент директория */
{
    "_id" : ObjectId("610129f9ba76f8e36dc5da7d"),
    "name" : "dir",
    "Parent" : "0",
    "Type" : "dir",
    "PDB" : "",
    "aim" : 0,
    "NFrames" : 0
}

/* Элемент МД-эксперимент */
{
    "_id" : ObjectId("61012a04ba76f8e36dc5da7e"),
    "name" : "123",
    "Parent" : "610129f9ba76f8e36dc5da7d",
    "Type" : "exp",
    "PDB" : "ATOM      1   CT  Ac     1     -13.432  -5.484   ",
// содержимое поля целиком не приводится
    "aim" : 44689,
    "NFrames" : 145
}
```

Здесь и далее по тексту в каждой коллекции будут использоваться те же поля, как в реляционных СУБД. Это не требуется для работы с MongoDB, но необходимо для индексации.

Табл. 4. Описание полей коллекции Names
Table 4. The description of Names collection

Поле	Тип	Описание
_id_	ObjectId	Идентификатор имени
Name	String	Название эксперимента
PDB	String	Содержимое PDB-файла
Aim	Int	Количество атомов
NFrames	Int	Количество кадров
Parent	String	Ссылка на родителя

Коллекция *Names* аналогично таблице *Names* реляционной БД с тем различием, что поле *id* в MySQL числовое, а в MongoDB мы использовали внутренний идентификатор, который есть у каждого документа. Также таблица *Names* содержит строковое поле PDB, в котором целиком помещается содержимое PDB-файла. Как говорилось ранее, программы анализа траекторий ориентированы на чтение одного PDB файла, относящегося к траектории для распознавания типов атомов и первичной структуры белка. Координаты при этом берутся исключительно из траекторных файлов.

Коллекция Trj:

```
{
    "_id" : ObjectId("5eb2dd86895408ca51dd9b4e"),
// собственный идентификатор
    "trj_id" : ObjectId("5eb2dd84895408ca51dd9b4d"),
// ссылка на имя траектории
```

```
    "frame" : 0,
    "data" : [
        25.80421,
...
}
```

Табл. 5. Описание полей коллекции Trj
Table 5. The description of Trj collection

Поле	**Тип**	**Описание**
_id_	ObjectId	Собственный уникальный идентификатор кадра
trj_id	ObjectId	Идентификатор имени, ссылка на имя траектории
Frame	int	Номер кадра
Data	Array	Массив координат в формате (*x1, y1, z1, x2, y2, z2, …, xn, yn, zn*)

Как видно из описания полей, одна таблица коллекции (таблицы) *Trj* хранит кадр целиком, а не координату каждого атома в каждой строке. Существование Поля *_id_* является требованием СУБД, а не продиктовано архитектурой конкретной базы.

В итоге количество строк в таблице (коллекции) уменьшается на 3-4 порядка. А запрос на выборку одного кадра занимает доли секунды (0,15-0,2 с).

Мы достигли своей цели – получили кадр траектории за приемлемое время. Надо заметить, что при этом мы возможности выбора координат конкретных атомов. Большая ли это плата за быстродействие? С одной стороны, программы анализа траекторий молекулярной динамики читают кадр целиком, даже когда нужно построить кривую расстояния между двумя атомами. С другой стороны, многие СУБД, и MongoDB в частности, поддерживают хранимые процедуры. Это небольшие программы, исполняемы на серверной стороне. Именно от такой программы можно потребовать выборку конкретных координат из массива координат текущего кадра (поля *data* таблицы *Trj*). Либо же можно написать функцию, которая вернёт сразу расстояние между заданными номерами атомов.

3.1 Подсказка от NoSQL: реализация хранения кадра в одном поле реляционной СУБД

Выдвинем гипотезу о том, что успех СУБД класса NoSQL связан не с самим классом баз данных, а со способом хранения кадра целиком в одном поле.

Начиная с версии 5.7.8, СУБД, MySQL поддерживает тип данных JSON (JavaScript Object Notation). В таком поле удобно будет хранить линеаризованный массив координат одного кадра траектории. Структура таблицы хранения траектории примет следующий вид:

Табл. 6. Новая структура таблицы Trj
Table 6. The new structure of Trj table

Поле	Тип	Описание
id_name	Int, внешний ключ	Идентификатор имени
Nframe	int	Номер кадра
Data	JSON	Координаты кадра в формате [*x1, y1, z1, x2, y2, z2, …]*

Индексируемыми полями будут *id_name* как ссылка на название траектории и *nframe* для быстрой выборки номера кадра. При этом экономится место на поле, отвечающего за номер атома в кадре. Причем как в таблице, так и в индексе.

В новой структуре теряется возможность обращения к конкретному номеру атому в кадре. Возможно только получение кадра целиком. Однако получать доступ к конкретному кадру и

не требуется, если мы используем Буфер случайного доступа к траектории Анализатора траекторий молекулярной динамики. Анализатор всегда считывает кадр целиком. Программы, такие как VMD и PyMol считывают траекторию целиком, что не экономично использует оперативную память.

4. Программная реализация – приложение Trajectory Commander как часть Анализатора траекторий молекулярной динамики

В рамках Анализатора траекторий молекулярной динамики (TAMD) было создано приложение в стиле двухпанельного менеджера, позволяющего оперировать траекториями на диске и сразу в двух базах данных – MySQL и MongoDB.

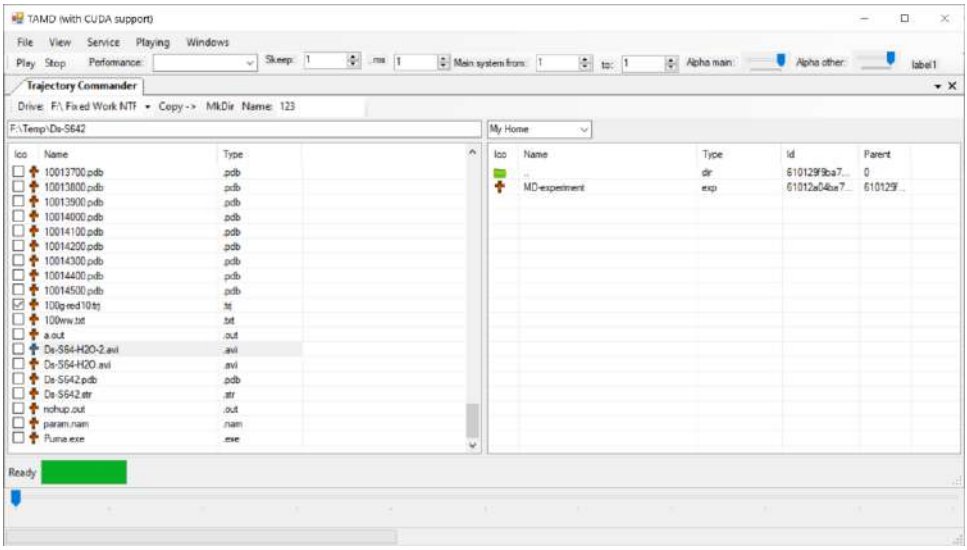

Рис. 1. Trajectory Commander в Анализаторе траекторий молекулярной динамики (TAMD)
Fig. 2. Trajectory Commander as part of Trajectory analyzer of molecular dynamics (TAMD)

На данном этапе работы основной целью было сравнение NoSQL и SQL подходов к хранению траекторий. Поэтому был реализован следующий список возможностей:

- отображение и позиционирование по локальной файловой системы на левой панели;
- хранение учетных записей подключений к различным СУБД;
- отображение и позиционирование по древовидной структуре траекторий, хранящихся в СУБД;
- создание каталогов в СУБД;
- копирование PDB и траекторных файлов из локальной файловой системы в СУБД;
- открытие и работа с траекториями молекулярной динамики напрямую из СУБД средствами Анализатора траекторий.

Анализатор траекторий молекулярной динамики изначально разрабатывался для работы с текстовыми траекториями формата PUMA, близкого к формату XYZ. Для чтения траекторий был разработан буфер случайного доступа к траектории, который хранит последние прочитанные файлы.

Для чтения траекторий напрямую из СУБД были добавлены функции чтения кадров не из текстового файла, а из базы данных. Таким образом сохранена возможность работы с траекторией через буфер случайного доступа к траекториям. Все возможности TAMD

сохранили работоспособность благодаря тому, что они обрабатывают координаты исключительно из модуля чтения траекторий.

4.1 Сравнение выбранных решений

Для сравнения выбранных решений была использована траектория, которая в текстовом формате PUMA занимает 535 МБ дискового пространства.

Табл. 7. Объемы информации
Table 7. Volumes of information

СУБД	Объем данных	Объем индексов	Общий объем данных
Текстовый файл	535 МБ		535 МБ
MySQL	324 МБ (61%)	410 МБ (+127%)	734 МБ (137%)
MySQL (JSON)	246 МБ (46%)	32 КБ (+0,013%)	246 МБ (46%)
MongoDB	281 МБ (53%)	45 КБ (+0,016%)	281 МБ (53%)

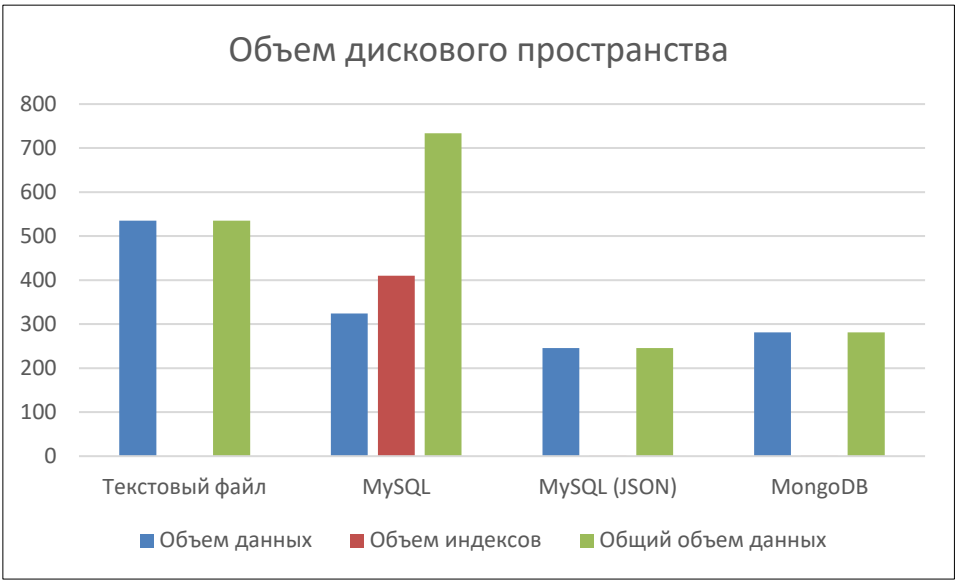

Рис. 3. Объем дискового пространства (меньше – лучше)
Fig. 4. Volume of disk space (less is better)

В столбце «объём данных» и в столбце «общий объем данных» процент приведен от размера исходного текстового траекторного файла, в столбце «объем индексов» от размера данных в соответствующей СУБД.

Из табл. 7 видно, что хранение траекторий в СУБД практически в 2 раза выгоднее по сравнению с текстовым форматом. Однако для эффективного доступа к траектории необходима индексация полей. В MongoDB индексация идёт по идентификатору траектории и кадру. В MySQL к индексам ещё добавляется номер атома в кадре. Размер записей в реляционной базе данных увеличивается на количества атомов в системе (на три порядка при размере системы 1000 атомов). Именно этим объясняется столь существенный объем индексов.

Согласно табл. 8, по скорости считывания кадра лидирует MongoDB. Хранение одного кадра целиком в одной строке реляционной СУБД даёт десятикратный прирост в скорости выполнения запроса.

Табл. 8. Скорость чтения кадров траектории
Table 8. Trajectory frame reading speed

СУБД	**Время чтения одного кадра**
Текстовый файл	55 мс
MySQL	300 мс
MySQL (JSON)	30 мс
MongoDB	4 мс

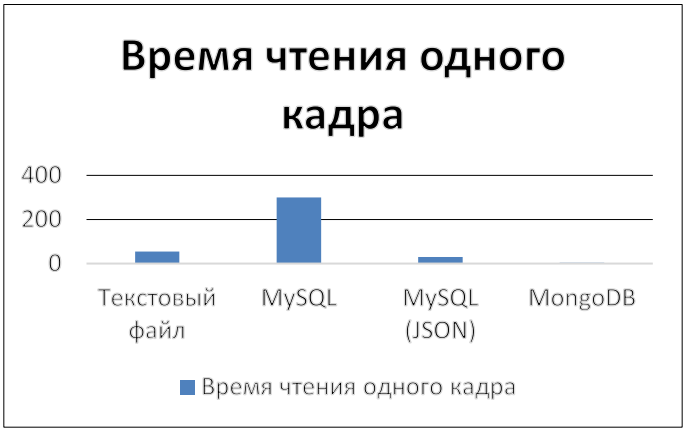

Рис. 5. Сравнение быстродействия при чтении траектории (меньше – лучше)
Fig. 6. Comparison of performance of reading the trajectory (less is better)

5. Обсуждение

В рамках рабочего программного продукта, активно используемого в НИИ для работы с траектория молекулярной динамики (Анализатора траекторий молекулярной динамики TAMD) были испытаны способы хранения и доступа к траекториям в СУБД различных видов.

Ввиду массового распространения нереляционных СУБД в последние годы важным и разумным казалось сравнивать двух представителей СУБД: реляционную СУБД MySQL и нереляционную, так называемого класса NoSQL, документо-ориентированную СУБД MongoDB. Обе СУБД полностью бесплатны, с открытым исходным кодом (open source), являются широко известными представителями СУБД в своём классе. Вместо конкретных СУБД можно было бы выбрать их конкурентов. Но задачей статьи является сравнение подходов, а не конкретных программных продуктов.

При построении **структуры данных** существенная разница проявилась в способе хранения одного кадра траектории. В реляционной СУБД естественным кажется хранение трёхмерных координат каждого атома в отдельной строчке. Количество строк соответствует количеству атомов в системе. В NoSQL подходе все координаты кадра хранятся в линеаризованном массиве, в одной ячейке.

В реляционной СУБД **дисковое пространство** уходит как на хранение самих данных, так и на хранение индексов. При хранении большого количества траекторий быстродействие реляционной СУБД ещё больше падает ввиду большого количества строк в одной таблице.

В работе с траекториями предполагается, что **кадр необходимо читать полностью**. Разумеется, есть множество характеристик, таких как расстояние между двумя отдельными атомами или валентные и торсионные углы, которые требуют выборки только конкретных номеров атомов. Но архитектура Анализатора траекторий такова, что кадр читается полностью. Так удобнее было реализовать Буфер случайного доступа к траектории, который

хранит в оперативной памяти последние кадры. Они могут понадобится как для нахождения характеристик, так и для визуализации кадра. В последнем случае необходимо иметь полный набор кадра.

Сравнивая характеристики MySQL- и MongoDB-подходов, выбор становится очевидным. MongoDB существенно выигрывает и в занимаемом дисковом пространстве, и в скорости доступа к траектории.

Встаёт вопрос, а можно ли перенести нереляционный подход на реляционную СУБД? Подход заключается лишь в способе хранения одного кадра траектории. Все координаты одного кадра разумно хранить в виде линейного массива.

Необходимо выбрать тип данных, поддерживаемый MySQL для хранения массива чисел. Можно было бы использовать бинарный формат, но MySQL начиная с версии 5.7, поддерживает тип JSON, который выглядит предпочтительнее ввиду простоты интерпретации данных человеком при поиске ошибок.

Выбранный подход в таблицах и рисунках изображен как MySQL (JSON). При хранении дискового пространства требуется даже меньше, чем для MongoDB. Выборка данных всё же занимает большего количества времени, чем в MongoDB. По всей вероятности, сказывается устройство реляционной СУБД в целом, начиная с анализа запросов SQL, нежели структура. Однако выбранный подход выглядит гораздо более привлекательным, нежели классическое реляционное представление с некоторой избыточностью в виде доступа к конкретному атому кадра. При этом не тратится существенных дисковых ресурсов на хранение индексов. 30 мс на чтение кадра кажется вполне приемлемыми. И это время не будет стремительно расти, как в традиционном подходе с большим количеством строк в таблице.

В итоге делается вывод, что компактное хранение MySQL (JSON) может конкурировать с СУБД MongoDB. MongoDB продолжает выигрывать в скорости, но скорость MySQL уже приемлема. При этом MySQL выигрывает в расходе дискового пространства, не смотря даже на применение текстового формата JSON против бинарного BSON у MongoDB.

При разработке решения на языке C# MySQL выглядит более документированным, а сам API более привычной. При работе с MongoDB создатели отмечают в плюсах отсутствие схемы, что должно ускорить разработку. Автору работы эта возможность показалась и положительной, когда не надо создавать схему под новую таблицу, и отрицательной, т.к. нет под глазами четкой схемы данных. При использовании сторонних продуктов-утилит для работы с СУБД (таких как текстовая консоль, phpmyadmin, HeidiSQL, Robo 3T, MogoDBCompass) отмечается больших функционал у MySQL, отсутствие явных ошибок. При работе с MongoDB MogoDBCompass не смогла отображать данные коллекции с координатами кадров за приемлемое время (требовались десятки секунд).

6. Выводы

Построение классической структуры реляционной СУБД для хранения траекторий молекулярной динамики не подходит для хранения существенных объемов информации. Основная причина – возникновение большого количества строк в одной таблице. Следствие – низкая скорость выполнения запросов, существенный объем индексных файлов.

NoSQL-подход на базе MongoDB даёт лучше результаты по быстродействию. Кадр траектории хранится в одном поле. Подход пригоден для построения больших хранилищ траекторий молекулярной динамики.

Для MySQL возможно компромиссная ситуация, когда один кадр также хранится в одном поле. В работе для этого использован тип данных JSON, но для этой цели может использоваться и другой тип, пригодный для хранения массивов. Подход MySQL+JSON показывает приемлемые результаты по скорости доступа, нет быстрого падения производительности при увеличении объемов информации. Среди преимуществ отмечается наименьший объем дискового пространства.

Несмотря на очевидное преимущество MongoDB-подхода по скорости, при построении хранилища выбор может быть сделан в пользу MySQL+JSON подхода ввиду лучшего знакомства с этой СУБД, лучшей документированности и большего количество API и полезных утилит.

Табл. 9. Преимущества и недостатки
Table 9. Advantages and disadvantages

	MySQL	**MySQL (JSON)**	**MongoDB**
Использование дискового пространства	137% от исходного файла	46% от исходного файла	53% от исходного файла
Скорость чтения одного кадра	Не справляется с большим объёмом данных	Приемлемое при работе больших объемов данных	Быстрое при работе больших объемов данных
Рекомендуется использовать для организации хранилища траекторий молекулярной динамики	Нет. Ввиду большого количества записей в одной таблице	Да	Да
Легкость разработки	Одинаково легко		Изучение новых API
Выбор инструментальных средств	Широкий		Ограниченный

Список литературы / References

[1] W. Humphrey, A. Dalke, and K. Schulten. VMD: Visual molecular dynamics. Journal of Molecular Graphics, vol. 14, issue 1, 1996, pp. 33-38.

[2] J. Hsin, A. Arkhipov et al. Using VMD: an introductory tutorial. Current Protocols in Bioinformatics, issue Suppl. 24, 2008, pp. 5.7.1-5.7.48.

[3] I.V. Likhachev, N.K. Balabaev, and O.V. Galzitskaya. Available Instruments for Analyzing Molecular Dynamics Trajectories. The Open Biochemistry Journal, vol. 10, 2016, pp.1-11.

[4] I.V. Likhachev, N.K. Balabaev. Trajectory analyzer of molecular dynamics. Mathematical Biology and Bioinformatics, vol. 2, issue 1, 2007, pp. 120-129.

[5] I.V. Likhachev, N.K. Balabaev. Construction of Extended Dynamical Contact Maps by Molecular-Dynamics Simulation Data. Mathematical Biology and Bioinformatics, vol. 4, issue 1, 2009, pp. 36-45.

[6] The PyMol Molecular Graphics System, Version 2.0. Schrödinger, 2015.

[7] R.E. Rigsby, A.B. Parker. Using the PyMOL application to reinforce visual understanding of protein structure. Biochemistry and Molecular Biology Education, vol. 44, issue 5, 2016, pp. 433-437.

[8] [8]Т. Кайт, Д. Кун. Oracle для профессионалов. Архитектура, методики программирования и основные особенности. Вильямс, 2020 г., 960 стр. / T. Kyte, D. Kuhn. Expert Oracle Database Architecture 3rd ed. Edition. Apress, 2014, 857 p.

[9] Р. Шелдон, Д. Мойе, MySQL. Базовый курс. Диалектика, 2007 г., 880 стр. / R. Sheldon, G. Moes. Beginning MySQL. Wrox, 2005, 864 p.

[10] Ш. Чаллавала, Дж. Лакхатария и др. MySQL 8 для больших данных. ДМК-Пресс, 2019 г., 226 стр. / Sh. Challawala, J. Lakhatariya et al. MySQL 8 for Big Data. Packt, 2017, 266 p.

[11] Р.М. Михеев. MS SQL Server 2005 для администраторов. BHV, 2007 г., 534 стр. / R.M. Mikheev. MS SQL Server 2005 for administrators. BHV, 2007, 534 p. (in Russian).

[12] Г.-Ю. Шениг. PostgreSQL 11. Мастерство разработки. ДМК-Пресс, 2019 г., 352 стр. / H.-J. Schönig. Mastering PostgreSQL 11. Packt Publishing, 2018, 446 p.

[13] А.А. Зоткина, И.А. Подопригора, Е.И. Маркин. Сравнительный анализ NoSQL баз данных. Вестник современных исследований, вып. 9.1 (24), 2018 г., стр. 146-148 / A.A. Zotkina, I.A. Podoprigora, and E.I. Markin. Comparative analysis of NoSQL databases. Bulletin of Modern Research, issue. 9.1 (24), 2018, pp. 146-148 (in Russian).

[14] О.Н. Чопоров, О.Е. Работкина, А.Е. Капишников, Разработка баз данных в среде Субд Ms Access и Ms Visual FoxPro. Учебное пособие. Воронежский государственный технический университет, 2004 г., 171 стр. / O.N. Choporov, O.E. Rabotkina, A.E. Kapishnikov, Development of databases in the environment of DBMS Ms Access and Ms Visual FoxPro. Tutorial. Voronezh State Technical University, 2004, 171 p. (in Russian).

[15] Д.А. Шорохов. Создание меню и отчётности для информационной системы учёта игроков и тренеров футбольного клуба „Калининград“ в среде разработки Microsoft Visual Foxpro 9.0. Актуальные научные исследования в современном мире, вып. 7–1 (63), 2020 г., стр. 227-232 / D.A. Shorokhov. Creating a menu and reporting for the information system of accounting of players and trainers of the football club “Kaliningrad” in the environment of Microsoft Visual Foxpro 9.0 development. Actual scientific research in the modern world, vol. 7–1 (63), 2020, pp. 227-232 (in Russian).

[16] Top 5 NoSQL databases for Data Scientists in 2020. URL: https://content.techgig.com/top-5-nosql-databases-for-data-scientists-in-2020/articleshow/78330888.cms, accessed 29.09.2021.

[17] Ю.С. Порохненко, П.Н. Полежаев. Сравнительный нализ NoSQL баз данных. Актуальные направления научных исследований XXI века: теория и практика, т. 5, вып. 9 (35), 2017 г., стр. 25-30 / Yu.S. Porokhnenko, P.N. Polezhaev. Comparative analysis of NoSQL database. Actual Directions of Scientific Research of the 21st Century: Theory and Practice, vol. 5, no. 9 (35), 2017, pp. 25-30 (in Russian).

[18] К. Бэнкер. MongoDB в действии. ДМК Пресс, 2017 г., 394 стр. / K. Banker. MongoDB in Action. Manning Publications, 2011, 312 p.

Информация об авторе / Information about the author

Илья Вячеславович ЛИХАЧЕВ – кандидат физико-математических наук, старший научный сотрудник. Сфера научных интересов: моделирование молекулярной динамики, параллельное программирование, в т.ч. графических процессоров по технологии CUDA, большие данные, СУБД, NoSQL.

Ilya Vyacheslavovich LIKHACHEV – PhD in Physics and Mathematics, Senior Researcher Research interests: modeling of molecular dynamics, parallel programming, incl. GPUs using CUDA technology, Big Data, DBMS, NoSQL.