

Сибирское отделение
Российской академии наук

ISSN 2073-0667

ПРОБЛЕМЫ ИНФОРМАТИКИ

ПРОБЛЕМЫ ИНФОРМАТИКИ № 1 2026



ТЕОРЕТИЧЕСКАЯ И СИСТЕМНАЯ ИНФОРМАТИКА

ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

1
—
2026

ПРОБЛЕМЫ ИНФОРМАТИКИ № 1 (70) 2026 г.

Журнал выходит ежеквартально, издается с 2008 г.

Учредитель журнала — Институт вычислительной математики и математической геофизики СО РАН.

Редакционный совет

Председатель — акад. НАН РК М. Н. Калимолдаев,
акад. РАН А. Л. Асеев, проф. В. А. Васенин, акад. РАН С. Н. Васильев,
проф. В. М. Вишневецкий, акад. РАН С. С. Гончаров, акад. РАН Н. А. Колчанов,
акад. РАН Н. А. Кузнецов, акад. РАН А. П. Кулешов, проф. РАН М. А. Марченко,
проф. А. Г. Марчук, А. Ю. Пальянов, проф. Б. Я. Рябко, акад. РАН И. А. Соколов,
проф. А. Н. Сотников, чл.-кор. РАН Ю. А. Флеров.

Редколлегия

Главный редактор — проф. В. Э. Малышкин,
Д. А. Афонников, Д. Ж. Ахмед-Заки, А. Г. Вострецов, Б. С. Гольдштейн, В. И. Гужов,
Ю. А. Загорулько, В. А. Иванисенко, С. Д. Каракозов, В. Н. Касьянов, О. В. Кибис,
В. В. Корнеев, И. В. Котенко, И. М. Куликов, М. Г. Курносов, С. А. Лашин, Т. П. Любимова,
А. Н. Ляхов, Ю. Г. Матушкин, В. В. Окольников, Б. В. Поллер, М. П. Пономаренко,
А. С. Родионов (зам. гл. редактора), А. Н. Савостьянов, М. А. Сонькин, М. С. Хайретдинов,
И. Г. Черных, Moonseong Kim (Korea), V. D. Nguyen (Vietnam), Michele Pagano (Italy).

Редакция: отв. секретарь М. С. Делидович, системный администратор В. А. Перепелкин,
верстка Д. В. Лазуткин, логист Л. В. Трофимова.

Адрес редакции, издателя: 630090, г. Новосибирск, просп. Академика Лаврентьева, д. 6,
ИВМ и МГ СО РАН

тел. (383) 330-96-43; e-mail: problem-info@sscc.ru, <http://www.problem-info.sccc.ru>.

Журнал зарегистрирован в Федеральной службе по надзору в сфере массовых коммуникаций, связи и охраны культурного наследия. Свидетельство ПИ № ФС77-32088 от 27 мая 2008 г.
Журнал распространяется по подписке. Оформление подписки на сайте «Объединенного каталога „Пресса России“» https://www.ppressa-rf.ru/cat/1/edition/y_e69980/, подписной индекс 69980, и через интернет-магазин «Пресса по подписке» https://www.akc.ru/itm/problemu_i-informatiki/. Цена свободная. Журнал распространяется на территории России.

Журнал включен в Перечень ведущих рецензируемых научных журналов, рекомендованных для публикаций Высшей аттестационной комиссией. Входит в «Белый список» Единого государственного перечня научных изданий (ЕГПНИ).

Все права авторов сохранены. Использование материалов журнала возможно только с разрешения редакции и авторов.

Отпечатано в типографии «Омега Принт» ИП Ипатова А. Е.

Адрес: 630090, Новосибирск, пр-т Академика Лаврентьева, 6, оф. 3-022,

e-mail: omegapr@yandex.ru.

Формат 60 × 84 1/8. Усл. печ. л. 9,3. Печать офсетная.

Тираж 50 экз. Заказ № 28. Подписано в печать 26.03.2026 г. Выход в свет 30.03.2026 г.

© Институт вычислительной математики и математической геофизики СО РАН, 2026

JOURNAL “PROBLEMS OF INFORMATICS”. No. 1 (70) 2026

Publisher: Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of Russian Academy of Sciences.

Editorial Council

Chairman Academician of the National Academy of Sciences of the Republic of Kazakhstan

M. N. Kalimoldayev

Full Member of the RAS A. L. Aseev, Professor V. A. Vasenin, Full Member of RAS C. N. Vassilyev,

Professor V. M. Vishnevsky, Full Member of RAS S. S. Goncharov, Full Member of RAS

N. A. Kolchanov, Full Member of RAS N. A. Kuznetsov, Full Member of RAS A. P. Kuleshov,

Professor of RAS M. A. Marchenko, Professor A. G. Marchuk, A. YU. Palyanov, Professor

B. Y. Ryabko, Full Member of RAS I. A. Sokolov, Professor A. N. Sotnikov, Corr. Member RAS

Y. A. Flerov.

Editorial board

The Editor-in-Chief Professor V. E. Malyshkin.

Associate Editors-in-Chief: A. S. Rodionov, D. A. Afonnikov D. Zh. Akhmed-Zaki, A. G. Vostretsov,

B. S. Goldstein, V. I. Guzhov, Y. A. Zagorulko, V. A. Ivanisenko, S. D. Karakozov, V. N. Kasyanov,

O. V. Kibis, V. V. Korneev, I. V. Kotenko, I. M. Kulikov, M. G. Kurnosov, S. A. Lashin,

T. P. Lyubimova, A. I. Lyakhov, V. V. Okolnishnikov, B. V. Poller, M. P. Ponomarenko,

A. N. Savostyanov, M. A. Sonkin, M. S. Khairatdinov, I. G. Chernykh, Moonseong Kim (Korea),

Van Duc Nguyen (Vietnam), Michele Pagano (Italy).

Editorial staff: Managing Editor M. S. Delidovich, System Administrator V. A. Perepelkin,
Maker-up D. V. Lazutkin, Logistician L. V. Trofimova.

Address of the editorial office: 630090, pr. Lavrentieva, 6, Novosibirsk, Russia, Institute of Computational Mathematics and Mathematical Geophysics of SB RAS.

Phone: +7 (383) 330-96-43; e-mail: problem-info@sscc.ru, <http://www.problem-info.sbcc.ru>.

The journal has been registered in accordance with Legislation of the Russian Federation. Certificate of Mass Media Registration: ПИ № ФС77-32088, of 27 May, 2008, ISSN 2073-0667. The journal is distributed in Russia.

The journal “Problems of Informatics” is in the List of Peer-Reviewed Scientific Journals for publication of scientific results of Ph.D. and Dr. of Sci.

All rights reserved. The journal contents may only be used by the permission of editors and authors.

СОДЕРЖАНИЕ

Теоретическая и системная информатика

<i>Алеева В. Н., Соколов М. П.</i> Формирование представлений алгоритмов для программных систем на основе концепции Q -детерминанта	5
<i>Власенко А. Ю., Городничев М. А., Курбатов М. А.</i> Система отладки простых вычислительных моделей в графовом представлении	24
<i>Герб А. Р., Девярых Е. Е., Омарова Г. А.</i> Оптимизации алгоритма поиска k кратчайших путей	40

Прикладные информационные технологии

<i>Кенжаев С. С., Ахатов А. Р., Тоджиев М. Р.</i> Гибридный комбинационный подход к количественной оценке чувствительности при выборе функции принадлежности в процессе фаззификации параметров сервера	50
<i>Перепелкин В. А., Арыков С. Б.</i> Совместное применение концепции активных знаний и методов машинного обучения	65

Правила представления и подготовки рукописей для публикации в журнале «ПРОБЛЕМЫ ИНФОРМАТИКИ»	79
---	----

На сайте «Объединенного каталога "Пресса России"» www.pressa-rf.ru
можно оформить подписку на печатную версию журнала
«Проблемы информатики» по подписному индексу 69980,
а также подписаться через интернет-магазин
«Пресса по подписке» www.akc.ru

CONTENTS

Theoretical informatics

<i>Aleeva V. N., Sokolov M. P.</i> Formation of Representations of Algorithms for Software Systems Based on the Concept of a Q -Determinant	5
<i>Vlasenko A. Yu., Gorodnichev M. A., Kurbatov M. A.</i> Debugging System of Simple Computational Models in Graph Representation	24
<i>Gerb A. R., Deviatykh E. E., Omarova G. A.</i> Optimization of the k -shortest Paths Algorithm	40

Applied information technologies

<i>Kenjaev S. S., Akhatov A. R., Tojiev M. R.</i> A Hybrid Combinational Approach to Sensitivity Quantification in Membership Function Selection during Fuzzification of Server Parameters	50
<i>Perepelkin V. A., Arykov S. B.</i> On Combining Active Knowledge Concept and Machine Learning	65

Rules of Presentation and Preparation of Manuscripts Offered for Publication	79
--	----

FORMATION OF REPRESENTATIONS OF ALGORITHMS FOR SOFTWARE SYSTEMS BASED ON THE CONCEPT OF A Q -DETERMINANT

V. N. Aleeva, M. P. Sokolov

South Ural State University (National Research University),
454080, Chelyabinsk, Russia

DOI: 10.24412/2073-0667-2026-1-5-23

EDN: EZZXDL

The paper continues the research based on the concept of the Q -determinant, which is the author's approach to parallelization of numerical algorithms. There are described the following concepts of the Q -determinant concept used in research.

An algorithmic problem has the form $\bar{y} = F(N, B)$, where $N = \{n_1, \dots, n_k\}$ is the set of parameters of the algorithmic problem dimension or N is the empty set, B is the set of input data, $\bar{y} = \{y_1, \dots, y_m\}$ is the set of output data. Here n_i ($i \in \{1, \dots, k\}$) is any positive integer. If $N = \{n_1, \dots, n_k\}$, then by $\bar{N} = \{\bar{n}_1, \dots, \bar{n}_k\}$ we denote a set of k positive integers, where $\bar{n}_i$ is some given value of the parameter n_i for each $i \in \{1, \dots, k\}$. We denote the set of all possible k -tuples $\bar{N}$ by $\{\bar{N}\}$. An algorithm for solving an algorithmic problem is denoted by $\mathcal{A}$, and the set of operations used by algorithm $\mathcal{A}$ is denoted by Q .

An expression over B and Q is a term in the standard sense of mathematical logic. A chain of length n is the result of applying some associative operation from Q to n expressions. If $N = \emptyset$, then any expression w over B and Q is an unconditional Q -term. If $N \neq \emptyset$ and V is the set of all expressions over B and Q , then any mapping $w : \{\bar{N}\} \rightarrow V \cup \emptyset$ is also called an unconditional Q -term. $w(\bar{N}) = \emptyset$ means that the value of $w(\bar{N})$ is undefined. If $N = \emptyset$ and the expression w over B and Q has a value of logical type under any interpretation of the variables B , then we call the unconditional Q -term w an unconditional logical Q -term. Let $N \neq \emptyset$ and w be an unconditional Q -term. Suppose also that the expression $w(\bar{N})$ for each $\bar{N} \in \{\bar{N}\}$ has a value of logical type under any interpretation of the variables B . In this case, we call the unconditional Q -term w an unconditional logical Q -term. Suppose that $u_1, \dots, u_l$ are unconditional logical Q -terms, $w_1, \dots, w_l$ are unconditional Q -terms. Then the set l of pairs $(\hat{u}, \hat{w}) = \{(u_i, w_i)\}_{i \in \{1, \dots, l\}}$ is a conditional Q -term of length l . Let $(\hat{u}, \hat{w}) = \{(u_i, w_i)\}_{i=1, 2, \dots}$ be a countable set of pairs of unconditional Q -terms. Suppose that $\{(u_i, w_i)\}_{i \in \{1, \dots, l\}}$ is a conditional Q -term of length l for any $l < \infty$. Then we call $(\hat{u}, \hat{w})$ a conditional infinite Q -term. Q -terms can be calculated.

Suppose that algorithm $\mathcal{A}$ consists in finding for each $i \in \{1, \dots, m\}$ the value y_i by computing the value of the Q -term f_i . Then the set of Q -terms $\{f_i \mid i \in \{1, \dots, m\}\}$ we call the Q -determinant of the algorithm $\mathcal{A}$. Moreover, we call the system of equations $\{y_i = f_i \mid i \in \{1, \dots, m\}\}$ a representation of the algorithm $\mathcal{A}$ in the form of a Q -determinant.

Previously, the concept of the Q -determinant was applied to improve the efficiency of parallel computing by exploring and exploiting the parallelism resources of numerical algorithms. To this end, a software Q -system for studying parallelism resources of algorithms and a program design method for the effective implementation of numerical algorithms have been developed. The Q -system and method can be used for any numerical algorithms, including those solving practically significant problems.

The method was applied to algorithms for matrix multiplication, solving systems of linear algebraic equations, finding the maximum element in an array of numbers, sorting, training neural networks, Dijkstra's algorithm, and other algorithms.

The method of designing programs for the efficient implementation of numerical algorithms is very useful, but its application requires the development of programs by programmers. There is a question about possibility of creating software system that develops and executes effective programs using a generally accepted description of algorithms without the use of programmers. To answer this question, several published papers have demonstrated the possibility of creating a unified software system for the automated design and execution of programs for the efficient implementation of numerical algorithms. These papers describe different approaches to designing such systems and outline general principles for their organization. The application of the developed software systems was demonstrated on problems with small values of the parameters of the algorithmic problem dimension. It is now important to explore their application to practically significant problems.

Software systems for the automated design and execution of programs for the efficient implementation of numerical algorithms use representations of algorithms in the form of Q -determinants to describe algorithms. These representations are formed using a subsystem of the Q -system for analyzing the parallelism resources of algorithms. The same representations of algorithms in the form of Q -determinants are also used by the subsystem for analyzing the characteristics of the parallelism resources of the Q -system. In this regard, the formation of representations of algorithms in the form of Q -determinants is an important task for the use of software systems based on the concept of Q -determinant.

The parameters of the algorithmic problems dimension can reach large values when solving practical problems. The number of iterations when implementing iterative algorithms can also be large. As a result, the Q -terms that make up the Q -determinants of algorithms can be very long, and the files containing the algorithm representations in the form of Q -determinants can be large. Therefore, problems can arise when forming and using algorithm representations in the form of Q -determinants. In this case, we will refer to Q -determinants as large, and the resulting problems as of large Q -determinant problems. Solving of large Q -determinant problems is extremely important for the practical application of software systems based on the Q -determinant concept.

In this paper, the problems of large Q -determinants are described and solved. More precisely, the problems that arise during the formation and use of representations of algorithms in the form of Q -determinants are formulated and solved. Furthermore, the results of experimental studies confirming the validity of the proposed solutions are presented. The solutions proposed in the paper enable the generation of representations of algorithms in the form of Q -determinants for large values of the algorithmic problem dimension parameters and a large number of iterations of iterative algorithms. Furthermore, these solutions reduce the size of files containing representations of algorithms in the form of Q -determinants.

The paper consists of three sections and a conclusion. Section 1 introduces the concepts of the Q -determinant concept used in this article. Section 2 describes the generation of representations of algorithms in the form of Q -determinants using algorithm flowcharts. Section 3 presents the problems of large Q -determinants and proposes solutions to them. The conclusion summarizes the results of the study and formulates the direction of further research.

Key words: Q -determinant of algorithm, representation of algorithm in form of Q -determinant, Q -effective implementation of algorithm, parallelism resource of algorithm, Q -effective program, computer-aided design of effective programs.

References

1. Aleeva V. N. Analiz parallel'ny'x chislenny'x algoritmov. Preprint N 590. Novosibirsk: VC SO AN SSSR, 1985. 23 s. (in Russian).
2. Valentina Aleeva, Rifkhat Aleev. Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. Vol. 10. N 2, Article number 8. P. 1–64. DOI: 10.1145/3583755.
3. Aleeva V. N. Podxod k e'ffektivnoj realizacii chislenny'x algoritmov // Problemy' informatiki. 2025. № 1. S. 29–44. DOI: 10.24412/2073-0667-2025-1-29-44 (in Russian).
4. Aleeva V. N., Kuzneczov E. K. E'ffektivnaya realizaciya algoritmov sortirovki s pomoshh'yu koncepcii Q -determinanta // Vestnik NGU. Seriya: Informacionny'e tekhnologii. 2025. T. 23, № 2. S. 5–17. DOI: 10.25205/1818-7900-2025-23-2-5-17 (in Russian).
5. Aleeva V. N., Sapozhnikov A. S. E'ffektivnaya realizaciya algoritmov obucheniya nejronny'x setej s pomoshh'yu koncepcii Q -determinanta // Problemy' informatiki. 2025. № 3. S. 5–16. DOI: 10.24412/2073-0667-2025-3-5-16 (in Russian).
6. Aleeva V. N. Avtomatizirovannoe proektirovanie i ispolnenie e'ffektivny'x programm dlya chislenny'x algoritmov // Vestnik YuUrGU. Seriya: Vy'chislitel'naya matematika i informatika. 2023. T. 12, № 3. S. 31–49. DOI: 10.14529/cmse230303 (in Russian).
7. Manatin P., Aleeva V. Efficient Implementation of Numerical Algorithms Based on a Lexical Analyzer // Parallel Computational Technologies. PCT 2024. Communications in Computer and Information Science. 2024. Vol. 2241. P. 107–121. DOI: 10.1007/978-3-031-73372-7_8.
8. Aleeva V. N. Razrabotka programmy'x sistem avtomatizirovannogo proektirovaniya i ispolneniya programm dlya e'ffektivnoj realizacii chislenny'x algoritmov na osnove koncepcii Q -determinanta // Vestnik NGU. Seriya: Informacionny'e tekhnologii. 2025. T. 23, № 1. S. 5–18. DOI: 10.25205/1818-7900-2025-23-1-5-18 (in Russian).
9. Ershov Yu. L., Palyutin E. A. Matematicheskaya logika. M.: Nauka, 1987. 336 s (in Russian).
10. Aleeva V. N., Zotova P. S., Skleznev D. S. Rasshirenie vozmozhnostej issledovaniya resursa parallelizma chislenny'x algoritmov s pomoshh'yu programmnoj Q -sistemy' // Vestnik YuUrGU. Seriya: Vy'chislitel'naya matematika i informatika. 2021. T. 10, № 2. S. 66–81. DOI: 10.14529/cmse210205 (in Russian).
11. Aleeva V. N. Improving Parallel Computing Efficiency // Proceedings — 2020 Global Smart Industry Conference, GloSIC 2020. IEEE. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.
12. Aleeva V., Bogatyreva E., Skleznev A., et al. Software Q -system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. 2019. Vol. 1129. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52.
13. Sokolov M. P. Razrabotka programmnogo obespecheniya dlya formirovaniya Q -determinantov chislenny'x algoritmov s Q -termami bol'shoj dliny' // Vy'p. kvalif. rabota bakalavra po napravleniyu "Fundamental'naya informatika i informacionny'e tekhnologii": 02.03.02 / Yuzhno-Ural'skij gosudarstvenny'j universitet. Chelyabinsk. 2020. 32 l. [Electron. Res.]: http://omega.sp.susu.ac.ru/publications/bachelorthesis/2020_402_sokolovmp.pdf. (Data obrashheniya: 15 dekabrya 2025 goda) (in Russian).
14. Superkomp'yuter "Tornado YuUrGU". [Electron. Res.]: <http://supercomputer.susu.ru/computers/tornado/> (Data obrashheniya: 15 dekabrya 2025 goda) (in Russian).
15. Sokolov M. P. Formirovanie predstavlenij algoritmov v forme Q -determinantov na klasterny'x vy'chislitel'ny'x sistemax // Vy'p. kvalif. rabota magistra po napravleniyu "Fundamental'naya informatika i informacionny'e tekhnologii": 02.04.02 / Yuzhno-Ural'skij gosudarstvenny'j universitet.

Chelyabinsk. 2022. 40 l. [Electron. Res.]: http://omega.sp.susu.ru/publications/masterthesis/2022_220_sokolovmp.pdf (Data obrashheniya: 15 dekabrya 2025 goda) (in Russian).

16. Oseledets I., Tyrtyshnikov E. TT-cross approximation for multidimensional arrays // Linear Algebra and its Applications, 2010. Vol. 432. N 1, P. 70–88. DOI: 10.1016/j.laa.2009.07.024.

17. Oseledets I. V. Tensor-Train Decomposition // SIAM Journal on Scientific Computing (SISC), 2011. Vol. 33. N 5, P. 2295–2317. DOI: 10.1137/090752286.

ФОРМИРОВАНИЕ ПРЕДСТАВЛЕНИЙ АЛГОРИТМОВ ДЛЯ ПРОГРАММНЫХ СИСТЕМ НА ОСНОВЕ КОНЦЕПЦИИ Q -ДЕТЕРМИНАНТА

В. Н. Алеева, М. П. Соколов

Южно-Уральский государственный университет (НИУ),
454080, Челябинск, Россия

УДК 004.021, 004.032.24, 004.051, 004.272

DOI: 10.24412/2073-0667-2026-1-5-23

EDN: EZZXDL

Концепция Q -детерминанта является одним из подходов к распараллеливанию численных алгоритмов. На ее основе можно повысить эффективность параллельных вычислений с помощью выявления ресурсов параллелизма алгоритмов с применением программной системы, а затем и их использования. Для этого была разработана технология параллельного программирования. Исследования с применением концепции Q -детерминанта доказали возможность создания программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Для описания алгоритмов программные системы должны использовать представления алгоритмов в форме Q -детерминантов. При решении практических задач файлы, содержащие представления алгоритмов в форме Q -детерминантов, в настоящее время могут иметь большие размеры. В связи с этим при формировании и использовании представлений алгоритмов в форме Q -детерминантов могут возникать проблемы. В этом случае Q -детерминанты называются большими, а возникающие проблемы — проблемами больших Q -детерминантов. В статье описаны проблемы больших Q -детерминантов и предложены их решения.

Ключевые слова: Q -детерминант алгоритма, представление алгоритма в форме Q -детерминанта, Q -эффективная реализация алгоритма, ресурс параллелизма алгоритма, Q -эффективная программа, автоматизированное проектирование эффективных программ.

Введение. Статья продолжает исследования на основе концепции Q -детерминанта — авторского подхода к распараллеливанию численных алгоритмов [1]. Ранее с его помощью было предложено повышение эффективности параллельных вычислений за счет исследования и использования ресурсов параллелизма численных алгоритмов [2]. Для этого, в частности, разработан метод проектирования программ для эффективной реализации численных алгоритмов, называемых Q -эффективными. Метод может быть использован для любых численных алгоритмов, в том числе для решения практически значимых задач. Обзор по применению данного метода для различных алгоритмов приведен в работе [3]. Его дополняет работа [4] по использованию метода для алгоритмов сортировки и работа [5] по использованию метода для алгоритмов обучения нейронных сетей.

Метод проектирования программ для эффективной реализации численных алгоритмов очень полезен, однако его применение предполагает разработку программ специалистами. А можно ли создать программную систему, которая без использования специалистов

разрабатывает и исполняет эффективные программы, используя общепринятое описание алгоритмов? Чтобы ответить на этот вопрос, была показана возможность создания единой для численных алгоритмов программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов [6–8]. В работах [6–7] описаны разные подходы к проектированию таких систем, а в [8] изложены общие принципы их организации. Применение разработанных систем было продемонстрировано на задачах с небольшими значениями параметров размерности. Теперь важно исследовать их работоспособность на практически значимых задачах.

Программные системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов используют для описания алгоритмов представления алгоритмов в форме Q -детерминантов, формируемые с помощью подсистемы программной Q -системы для исследования ресурсов параллелизма алгоритмов [2]. Такие же представления алгоритмов в форме Q -детерминантов использует и подсистема для анализа характеристик ресурсов параллелизма алгоритмов программной Q -системы. В связи с этим подготовка представлений алгоритмов в форме Q -детерминантов является важной задачей для использования программных систем на основе концепции Q -детерминанта.

При решении практических задач параметры размерности алгоритмической проблемы могут достигать больших значений, количество итераций при реализации итерационных алгоритмов также может быть большим. В результате Q -термы, составляющие Q -детерминанты алгоритмов, могут иметь большую длину, а файлы, содержащие представления алгоритмов в форме Q -детерминантов, большие размеры. В связи с этим при формировании и использовании представлений алгоритмов в форме Q -детерминантов могут возникать проблемы. В этом случае Q -детерминанты будем называть большими, а возникающие проблемы — проблемами больших Q -детерминантов. Решение этих проблем чрезвычайно важно для практического применения программных систем на основе концепции Q -детерминанта.

Цель данного исследования — решение проблем больших Q -детерминантов. Для достижения цели необходимо решить следующие задачи.

- 1) Описание и решение проблем, возникающих при формировании и использовании представлений алгоритмов в форме Q -детерминантов.
- 2) Описание экспериментальных исследований, подтверждающих правильность предложенных решений.

Статья организована следующим образом. Раздел 1 содержит некоторые основные понятия концепции Q -детерминанта, используемые в данной работе. В разделе 2 описано формирование представлений алгоритмов в форме Q -детерминантов с помощью их блок-схем. В разделе 3 приведены проблемы больших Q -детерминантов, предложены их решения и описаны проведенные экспериментальные исследования. В заключении подводятся итоги исследования и формулируется направление дальнейших исследований.

1. Некоторые основные понятия концепции Q -детерминанта. Концепция Q -детерминанта в настоящее время наиболее полно представлена в работе [2]. Кратко опишем понятия концепции Q -детерминанта, используемые в данной работе.

Алгоритмическую проблему опишем в виде $\bar{y} = F(N, B)$, где $N = \{n_1, \dots, n_k\}$ — множество параметров размерности проблемы или N — пустое множество, B — множество входных данных, $\bar{y} = \{y_1, \dots, y_m\}$ — множество выходных данных, при этом целое число m является либо константой, либо значением вычислимой функции параметров N при усло-

вии, что $N \neq \emptyset$. Здесь n_i ($i \in \{1, \dots, k\}$) равно любому положительному целому числу. Если $N = \{n_1, \dots, n_k\}$, то через $\bar{N} = \{\bar{n}_1, \dots, \bar{n}_k\}$ обозначим набор из k положительных целых чисел, где $\bar{n}_i$ — некоторое заданное значение параметра n_i для каждого $i \in \{1, \dots, k\}$. Через $\{\bar{N}\}$ обозначим множество всех возможных k -наборов $\bar{N}$. Пусть $\mathcal{A}$ — алгоритм для решения алгоритмической проблемы, Q — набор операций, используемых алгоритмом $\mathcal{A}$.

Определение 1. Под выражением над B и Q будем понимать терм в стандартном смысле математической логики [9].

Определение 2. Мы называем выражение цепочкой длины n , если оно является результатом применения некоторой ассоциативной операции из Q к n выражениям.

Определение 3. Если $N = \emptyset$, то любое выражение w над B и Q мы называем безусловным Q -термом. Пусть $N \neq \emptyset$ и V — множество всех выражений над B и Q . Тогда любое отображение $w : \{\bar{N}\} \rightarrow V \cup \emptyset$ также называется безусловным Q -термом.

Определение 4. Пусть $N = \emptyset$ и w — безусловный Q -терм. Предположим, что выражение w над B и Q имеет значение логического типа при любой интерпретации переменных B . Тогда безусловный Q -терм w называется безусловным логическим Q -термом. Пусть $N \neq \emptyset$ и w — безусловный Q -терм. Если выражение $w(\bar{N})$ для каждого $\bar{N} \in \{\bar{N}\}$ имеет значение логического типа при любой интерпретации переменных B , то безусловный Q -терм w называется безусловным логическим Q -термом.

Определение 5. Пусть $u_1, \dots, u_l$ — безусловные логические Q -термы, $w_1, \dots, w_l$ — безусловные Q -термы. Тогда множество l пар

$$(\hat{u}, \hat{w}) = \{(u_i, w_i)\}_{i \in \{1, \dots, l\}}$$

называется условным Q -термом длины l .

Определение 6. Пусть $(\hat{u}, \hat{w}) = \{(u_i, w_i)\}_{i=1,2,\dots}$ — счетное множество пар безусловных Q -термов. Предположим, что $\{(u_i, w_i)\}_{i \in \{1, \dots, l\}}$ — условный Q -терм для любого $l < \infty$. Тогда мы называем $(\hat{u}, \hat{w})$ условным бесконечным Q -термом.

Опишем нахождение значения безусловного Q -терма w при интерпретации переменных B . Если $N = \emptyset$, то нахождение значения выражения w означает нахождение значения безусловного Q -терма w при любой интерпретации переменных B . Если $N \neq \emptyset$ и $w(\bar{N}) \neq \emptyset$, то $w(\bar{N})$ является выражением над B и Q . Можно найти значение выражения $w(\bar{N})$ при любой интерпретации переменных B . Конечно, мы опускаем значение $w(\bar{N}) = \emptyset$. Следовательно, мы находим значение безусловного Q -терма w при любой интерпретации переменных B . Теперь опишем нахождение значения условного Q -терма $(\hat{u}, \hat{w})$ при интерпретации переменных B . Пусть $N = \emptyset$. Находим значения выражений u_i, w_i для $i \in \{i = 1, \dots, l\}$. При нахождении значений мы можем найти пару u_{i_0}, w_{i_0} такую, что u_{i_0} имеет значение **true**. Следовательно, мы можем найти значение w_{i_0} . Тогда считаем, что $(\hat{u}, \hat{w})$ имеет значение w_{i_0} . В противном случае считаем, что значение $(\hat{u}, \hat{w})$ при интерпретации переменных B не определено. Пусть $N \neq \emptyset$ и $\bar{N} \in \{\bar{N}\}$. Находим выражения $u_i(\bar{N}), w_i(\bar{N})$ для $i \in \{i = 1, \dots, l\}$. При нахождении значений мы можем найти пару $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ такую, что $u_{i_0}(\bar{N})$ имеет значение **true**. Следовательно, мы можем найти значение $w_{i_0}(\bar{N})$. Тогда мы считаем, что $(\hat{u}, \hat{w})$ имеет значение $w_{i_0}(\bar{N})$. В противном случае считаем, что значение $(\hat{u}, \hat{w})$ для $\bar{N}$ и при такой интерпретации переменных B не определено. Аналогично можно определить значение условного бесконечного Q -терма.

Определение 7. Пусть $M = \{1, \dots, m\}$. Предположим, что алгоритм $\mathcal{A}$ состоит в нахождении для каждого $i \in M$ значения y_i путем вычисления значения Q -терма f_i . Тогда

набор Q -термов $\{f_i \mid i \in M\}$ называется Q -детерминантом алгоритма $\mathcal{A}$. Система уравнений $\{y_i = f_i \mid i \in M\}$ называется представлением алгоритма $\mathcal{A}$ в форме Q -детерминанта.

Определение 8. Процесс вычисления Q -термов $\{f_i \mid i \in M\}$ алгоритма $\mathcal{A}$ называется реализацией алгоритма $\mathcal{A}$. Реализация алгоритма $\mathcal{A}$ называется параллельной, если существуют операции, которые выполняются одновременно.

Для дальнейшего изложения будем использовать некоторое разбиение множества M . Предположим, что U , C и I образуют разбиение множества $M = \{1, \dots, m\}$ с пустыми членами, то есть: $U \cup C \cup I = M$; $U \cap C = U \cap I = C \cap I = \emptyset$; кроме того, одно или два подмножества U , C и I могут быть пустыми. Предположим также, что подмножества U , C и I можно связать с подмножествами множества Q -термов $\{f_i \mid i \in M\}$ таким образом: 1) для каждого $i \in U$ существует Q -терм f_i , который является безусловным, и $f_i = w^i$; 2) для каждого $i \in C$ существует Q -терм f_i , который является условным, и $f_i = \{(u_j^i, w_j^i)\}_{j \in \{1, \dots, l(i)\}}$, где $l(i)$ — либо константа, либо значение вычисляемой функции от N , если $N \neq \emptyset$; 3) для каждого $i \in I$ существует Q -терм f_i , который является условным бесконечным, и $f_i = \{(u_j^i, w_j^i)\}_{j \in \{1, 2, \dots\}}$.

Определение 9. Опишем реализацию алгоритма $\mathcal{A}$, называемую Q -эффективной. Пусть $N = \emptyset$. Зададим интерпретацию переменных B . Будем вычислять выражения

$$W = \{w^i(i \in U); u_j^i, w_j^i(i \in C, j \in \{1, \dots, l(i)\}); u_j^i, w_j^i(i \in I, j \in \{1, 2, \dots\})\}$$

одновременно, параллельно. Мы говорим, что операция готова к выполнению, если вычислены значения всех ее операндов. При вычислении каждого из выражений W мы выполняем операции, как только они готовы к выполнению. Если несколько операций цепочки готовы к выполнению, то они выполняются по схеме сдваивания. Например, вычисление цепочки $a_1 + a_2 + a_3 + a_4$ по схеме сдваивания выполняется так: сначала вычисляем $b_1 = a_1 + a_2$ и $b_2 = a_3 + a_4$ одновременно, затем $c = b_1 + b_2$. Если для $i \in C \cup I$ и $j \in \{1, 2, \dots\}$ u_j^i имеет значение **false**, то вычисление соответствующего w_j^i прекращается. Если для $i \in C \cup I$ и $j \in \{1, 2, \dots\}$ вычисление пары (u_j^i, w_j^i) приводит к тому, что значение одного выражения не определено, то вычисление второго выражения прекращается. Если для $i \in C \cup I$ вычисление некоторой пары $(u_{j_0}^i, w_{j_0}^i)$ приводит к определению их значений и значение $u_{j_0}^i$ **true**, то вычисление выражений u_j^i, w_j^i прекращается для любого $j \neq j_0$.

Теперь пусть $N \neq \emptyset$. Зададим интерпретацию переменных B и $\bar{N} \in \{\bar{N}\}$. Получаем множество выражений

$$W(\bar{N}) = \{w^i(\bar{N})(i \in U); u_j^i(\bar{N}), w_j^i(\bar{N})(i \in C, j \in \{1, \dots, l(i)\}); u_j^i(\bar{N}), w_j^i(\bar{N})(i \in I, j \in \{1, 2, \dots\})\}.$$

Выражения $W(\bar{N})$ вычисляются по аналогии с выражениями W .

Замечание. Определение Q -эффективной реализации показывает, что она полностью использует ресурс параллелизма алгоритма.

Ресурс параллелизма алгоритма характеризуют его высота и ширина. Эти понятия рассматриваются в работах [2, 10–12]. В [2, 10, 12] описана программная Q -система, разработанная для автоматизированного исследования ресурса параллелизма алгоритмов, а также сравнения ресурсов параллелизма алгоритмов, решающих одну и ту же алгоритмическую проблему.

Определение 10. Реализация алгоритма $\mathcal{A}$ называется выполнимой, если одновременно должно выполняться конечное (непустое) множество операций.

Замечание. Существуют алгоритмы, Q -эффективная реализация которых невыполнима. Пример такого алгоритма приведен в работе [10].

2. Использование блок-схем алгоритмов для формирования их представлений в форме Q -детерминантов. Опишем метод формирования представлений алгоритмов в форме Q -детерминантов, реализованный в подсистеме для генерации Q -детерминантов численных алгоритмов программной Q -системы [12].

Представления алгоритмов в форме Q -детерминантов можно формировать, используя их блок-схемы. Блок-схемы алгоритмов хорошо известны и часто используются. Опишем ограничения и дадим пояснения для блок-схем, применяемых в исследованиях на основе концепции Q -детерминанта. Блок-схема имеет два терминальных блока, один из которых «Начало» означает начало алгоритма, а другой «Конец» — конец алгоритма. Блок «Начало» имеет одну исходящую линию потока и не имеет входящей линии потока. Блок «Конец» имеет одну входящую линию потока и не имеет исходящей линии потока. Блок «Решение» имеет две исходящие потоковые линии, одна из которых соответствует передаче управления, если условие блока имеет значение true, а другая, если его значение false. Условие блока использует одну операцию сравнения. Его операнды не содержат операций. Блок «Процесс» содержит одно присвоение значения переменной. Справа от знака присваивания находится не более одной операции. Для групп операций используются цепочки блоков. Блоки «Данные» делятся на блоки «Входные данные» и «Выходные данные».

Для формирования представления алгоритма в форме Q -детерминанта блок-схема алгоритма анализируется при фиксированных значениях параметров размерности алгоритмической проблемы $\bar{N} = \{\bar{n}_1, \dots, \bar{n}_k\}$, если $N \neq \emptyset$. При этом исследуются особенности алгоритмов, Q -детерминанты которых содержат различные типы Q -термов.

Если Q -детерминант алгоритма имеет безусловные Q -термы, то проход по блок-схеме должен осуществляться последовательно в соответствии с выполнением алгоритма. Для каждого $i \in U$ выражения w^i , если $N = \emptyset$, и $w^i(\bar{N})$, если $N \neq \emptyset$, получаются, как значения безусловного Q -терма f_i . Они формируются с использованием содержимого блоков «Процесс», участвующих в вычислении f_i для каждого $i \in U$.

Предположим, что Q -детерминант содержит условные Q -термы. Проход по блоку «Решение» с условием, содержащим входные данные, порождает ветвление. Затем каждая ветвь обрабатывается отдельно. В результате обработки одной ветви для каждого $i \in C$ и соответствующего ветви $j \in \{1, \dots, l(i)\}$ получаем пары выражений (u_j^i, w_j^i) , если $N = \emptyset$ и $(u_j^i(\bar{N}), w_j^i(\bar{N}))$, если $N \neq \emptyset$. Выражения u_j^i , если $N = \emptyset$ и $u_j^i(\bar{N})$, если $N \neq \emptyset$ формируются из условий блоков «Решение», содержащих входные данные. Выражения w_j^i , если $N = \emptyset$ и $w_j^i(\bar{N})$, если $N \neq \emptyset$ формируются с использованием содержимого блоков «Процесс». Как только первая ветвь обработана, обработчик блок-схемы возвращается к ближайшему блоку, где произошло ветвление, и продолжает обработку с противоположным условием. Следующая ветвь будет получена путем возврата к ближайшему блоку, где ветвление выполнялось один раз. Если таких блоков не осталось, то перебор ветвей заканчивается. После обработки всех ветвей для всех $i \in C$ и $j \in \{1, \dots, l(i)\}$ получаем пары выражений (u_j^i, w_j^i) , если $N = \emptyset$ и $(u_j^i(\bar{N}), w_j^i(\bar{N}))$, если $N \neq \emptyset$ для всех условных Q -термов f_i .

Условные бесконечные Q -термы содержатся в Q -детерминантах итерационных численных алгоритмов. Ограничивая количество итераций, можно свести случай Q -детерминанта с условными бесконечными Q -термами к случаю Q -детерминанта с условными Q -термами. Будем использовать параметр L для количества итераций. Этот параметр

может принимать любые целые положительные значения. Обозначим значение параметра L через $\bar{L}$. Значение $\bar{L}$ определяет длину Q -термов f_i для каждого $i \in I$.

Опишем формат файлов, содержащих представления алгоритмов в форме Q -детерминантов, формируемых с помощью подсистемы для генерации Q -детерминантов численных алгоритмов программной Q -системы. Представления алгоритмов в форме Q -детерминантов имеют фиксированные значения параметров размерности N и ограниченное число итераций L .

Для каждого $i \in C$ условный Q -терм f_i определяет $l(i)$ строк файла, где $l(i)$ — длина Q -терма. Каждая из этих строк содержит идентификатор выходной переменной, вычисляемой с использованием данного Q -терма, знак равенства и одну пару выражений (u_j^i, w_j^i) , если $N = \emptyset$ или $(u_j^i(\bar{N}), w_j^i(\bar{N}))$, если $N \neq \emptyset$ для некоторого $j \in \{1, \dots, l(i)\}$. Здесь для каждого $j \in \{1, \dots, l(i)\}$ Q -термы таковы, что u_j^i является безусловным логическим Q -термом, а w_j^i безусловным Q -термом. Выражения (u_j^i, w_j^i) , если $N = \emptyset$, или $(u_j^i(\bar{N}), w_j^i(\bar{N}))$, если $N \neq \emptyset$, описываются в формате JSON и разделяются точкой с запятой. Описание условного бесконечного Q -терма аналогично описанию условного Q -терма, поскольку длина условного бесконечного Q -терма ограничена значением параметра L .

Для каждого $i \in U$ безусловный Q -терм f_i определяет одну строку файла. Безусловные Q -термы также описываются в формате JSON. В этом случае нет логического Q -терма, поэтому вместо него используется пробел.

Для описания выражений в формате JSON применяются следующие обозначения: op — унарная или бинарная операция, od — операнд унарной операции, fO — первый операнд бинарной операции, sO — второй операнд бинарной операции, скобки $\{$ и $\}$ определяют порядок выполнения операций.

Пример. Пусть матрица A размера 2×2 имеет ненулевой определитель. Предположим, что $\vec{X} = (X(1), X(2))^T$ и $\vec{B} = (\bar{A}(1,3), \bar{A}(2,3))^T$ — векторы-столбцы, где $\bar{A} = [A(i,j)]_{2 \times 3}$ — расширенная матрица системы $A\vec{X} = \vec{B}$. На рис. 1 показано содержимое файла с представлением в форме Q -детерминанта алгоритма решения системы линейных уравнений второго порядка методом Гаусса–Жордана.

3. Проблемы больших Q -детерминантов и их решения.

3.1 Формирование представлений алгоритмов в форме Q -детерминантов с использованием персонального компьютера. Подсистема для генерации Q -детерминантов численных алгоритмов сначала использовалась на персональном компьютере. В результате было сформировано множество представлений алгоритмов в форме Q -детерминантов для алгоритмов с различными структурами Q -детерминантов, с различными значениями параметров размерности, а для итерационных алгоритмов с различными значениями количества итераций. Это множество представлений алгоритмов в форме Q -детерминантов было использовано для разработки и тестирования подсистемы Q -системы для вычисления ресурсов параллелизма численных алгоритмов, а также для разработки и тестирования программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов.

Эксплуатация подсистемы для генерации Q -детерминантов численных алгоритмов выявила, что при формировании Q -термов некоторых алгоритмов возникает исключение `OutOfMemoryException`, что делает невозможным в этом случае формирование представлений алгоритмов в форме Q -детерминантов. Например, исключение возникало при формировании представления в форме Q -детерминанта алгоритма для решения системы линейных уравнений методом Гаусса–Зейделя при задании количества линейных уравнений

Рис. 1. Представление в форме Q -детерминанта алгоритма решения СЛАУ второго порядка методом Гаусса-Жордана

Когда была доказана возможность создания на основе концепции Q -детерминанта программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов, стало понятно, что вычисления представлений алгоритмов в форме Q -детерминантов на персональном компьютере выполнили свою задачу, поэтому было принято решение использовать для этих вычислений суперкомпьютер «Торнадо ЮУрГУ» [14].

3.2. *Формирование представлений алгоритмов в форме Q -детерминантов с использованием суперкомпьютера.* Для формирования представлений алгоритмов в форме Q -

Таблица 1

Результаты вычислительных экспериментов для алгоритма
умножения двух плотных квадратных матриц

Размер матриц	Размер выходного файла, МБ
150	209
160	254
170	306
180	364
190	429
200	502

детерминантов с помощью кластерной вычислительной системы «Торнадо ЮУрГУ» было разработано новое программное обеспечение [15], использующее метод формирования представлений алгоритмов в форме Q -детерминантов, который применялся в подсистеме для генерации Q -детерминантов численных алгоритмов программной Q -системы (см. раздел 2). Для организации распределенных вычислений на кластерной вычислительной системе разработанное программное обеспечение использует технологию MPI, с помощью которой осуществляется распределение проходов по блок-схеме алгоритма по вычислительным узлам системы. В результате использования технологии MPI существенно сократилось время формирования выходного файла, а также стало возможным получить представления алгоритмов в форме Q -детерминантов, которые было невозможно получить ранее, используя последовательную программу на персональном компьютере.

Приведем некоторые результаты, полученные с помощью вычислительных экспериментов с использованием разработанного программного обеспечения. Результаты вычислительных экспериментов, полученные при формировании представлений в форме Q -детерминанта алгоритма умножения двух плотных квадратных матриц, приведены в табл. 1. В табл. 2 представлены результаты вычислительных экспериментов, показывающие зависимость размера выходного файла от длины массива при формировании представлений алгоритма поиска максимального элемента в массиве чисел в форме Q -детерминанта. Кроме того, были получены результаты вычислительных экспериментов, в которых использовались итерационные алгоритмы. В этом случае формирование представлений алгоритмов в форме Q -детерминантов выполняется при фиксированных значениях параметров размерности и числа итераций. Табл. 3 содержит результаты вычислительных экспериментов для алгоритма, реализующего метод Якоби, а табл. 4 для алгоритма, реализующего метод Гаусса-Зейделя для решения систем линейных уравнений (СЛАУ).

Приведенные результаты показывают, что с ростом параметров размерности и количества итераций файл, содержащий представление алгоритма в форме Q -детерминанта, может достигать довольно большого размера, затрудняющего его формирование, хранение и использование. Следует отметить, что представления алгоритмов в форме Q -детерминантов могут быть заранее сформированы и сохранены для дальнейшего использования, поэтому время их формирования не имеет такого значения, как размер содержащих их файлов. Несмотря на это, быстроедействие программного обеспечения для формирования представлений алгоритмов в форме Q -детерминантов планируется повысить за счет имеющегося у него потенциала распараллеливания, что будет способствовать решению практически значимых задач.

Таблица 2

Результаты вычислительных экспериментов для алгоритма поиска максимального элемента в массиве чисел

Длина массива	Размер выходного файла, МБ
24	10784
25	22528
26	46972
27	97779
28	203227
29	421789
30	874250

Таблица 3

Результаты вычислительных экспериментов для алгоритма, реализующего метод Якоби для решения СЛАУ

Размер матрицы	Количество итераций	Размер выходного файла, МБ
3	19	2483
4	13	6093
5	11	17487
6	9	9501
7	8	9674
8	6	801
8	7	5608
9	6	2096
9	7	16772

3.3. *Подход к сокращению размеров файлов с представлениями алгоритмов в форме Q -детерминантов.* Опишем решения, позволяющие сократить размеры файлов, содержащих представления алгоритмов в форме Q -детерминантов, формируемые с помощью подсистемы Q -системы для генерации Q -детерминантов численных алгоритмов. Эти решения основаны на особенностях структуры представлений алгоритмов в форме Q -детерминантов.

В разделе 2 показано, что после обработки всех ветвей блок-схемы алгоритма для всех $i \in C$ и $j \in \{1, \dots, l(i)\}$ будут получены пары выражений (u_j^i, w_j^i) , если $N = \emptyset$ и $(u_j^i(\bar{N}), w_j^i(\bar{N}))$, если $N \neq \emptyset$ для всех условных Q -термов f_i . При этом безусловные логические Q -термы с одинаковыми значениями j совпадают, так как они формируются из условий блоков «Решение», содержащих входные данные, при одном проходе по блок-схеме. Рис. 1 подтверждает это утверждение. Таким образом, в случае условных Q -термов значительную часть файла, содержащего представление алгоритма в форме Q -детерминанта, могут занимать повторяющиеся безусловные логические Q -термы. Целесообразно сократить размер файла за счет однократного использования безусловных логических Q -термов.

Предлагается также формировать, хранить и использовать представления алгоритмов в форме Q -детерминантов, разделив их согласно разработанному стандарту на некото-

Таблица 4

Результаты вычислительных экспериментов для алгоритма, реализующего метод Гаусса-Зейделя для решения СЛАУ

Размер матрицы	Количество итераций	Размер выходного файла, МБ
3	10	3854
4	6	3904
10	2	6554
20	1	25200

рое количество частей, размещаемых в отдельных архивированных файлах. Опишем этот стандарт.

Если представление алгоритма в форме Q -детерминанта содержит безусловные Q -термы, то его строки должны быть последовательно распределены по файлам с именами k_w , где k — порядковый номер файла. Количество файлов выбирается, исходя из структуры Q -детерминанта. В каждой строке после идентификатора выходной переменной и знака равенства сразу следует безусловный Q -терм, то есть исключаются пробел, используемый вместо безусловного логического Q -терма, и точка с запятой, разделяющая безусловный логический и безусловный Q -термы.

Если представление алгоритма в форме Q -детерминанта содержит условные Q -термы, то предлагается три варианта согласно разработанному стандарту: с разделением по логическим Q -термам, с разделением безусловных логических и безусловных Q -термов, с разделением по безусловным Q -термам. Следует выбирать оптимальный вариант в зависимости от структуры Q -детерминанта. Опишем эти варианты.

Первый вариант представления алгоритма в форме Q -детерминанта, содержащего условные Q -термы, с разделением по логическим Q -термам состоит в следующем. Строки представления алгоритма в форме Q -детерминанта должны быть распределены по файлам с именами k_uw , где k — порядковый номер файла. Каждый из этих файлов содержит либо описание одного подмножества пар условных Q -термов, имеющих одинаковые безусловные логические Q -термы, либо нескольких подмножеств. Описание подмножества пар условных Q -термов состоит в том, что первая строка содержит безусловный логический Q -терм, а в каждой из следующих строк находится идентификатор выходной переменной, знак равенства и безусловный Q -терм. Количество строк в файле и количество файлов выбираются, исходя из структуры Q -детерминанта.

Второй вариант представления алгоритма в форме Q -детерминанта, содержащего условные Q -термы, с разделением безусловных логических и безусловных Q -термов заключается в следующем. Пусть Q -детерминант состоит из условных Q -термов длины M . Q -термы должны быть распределены по файлам с именами

$$1_u1, 2_w1, 3_u2, 4_w2, 5_u3, 6_w3, \dots, (K-1)_uM, K_wM,$$

где K — количество файлов. Пары условных Q -термов, имеющие порядковый номер m , описываются следующим образом: безусловный логический Q -терм содержится в файле с именем k_um , где k — порядковый номер файла, а безусловные Q -термы в файле с именем

$(k + 1)_{wm}$, каждая из строк которого содержит идентификатор выходной переменной, знак равенства и один безусловный Q -терм.

Третий вариант представления алгоритма в форме Q -детерминанта, содержащего условные Q -термы, заключающийся в разделении по безусловным Q -термам, отличается от предыдущего построчным последовательным распределением содержимого файлов с именами k_{wm} по файлам с именами

$$k_{wm}, k_{wmp1}, k_{wmp2}, k_{wmp3}, \dots$$

Количество Q -термов, содержащееся в каждом таком файле, и количество файлов выбираются исходя из структуры Q -детерминанта.

По-видимому, можно провести аналогию между описанным подходом к сокращению размеров файлов с представлениями алгоритмов в форме Q -детерминантов и подходом, использующим понятие тензорного продукта (Tensor Train) — формы разложения многомерных массивов (тензоров), которая не страдает от «проклятия размерности» [16–17]. Учитывая это, представление алгоритма в форме Q -детерминанта, полученное в соответствии с предложенным стандартом, можно назвать Q -детерминантным поездом (Q -determinant Train).

Внедрение описанного подхода позволило значительно сократить размер выходных данных подсистемы для генерации Q -детерминантов численных алгоритмов. В зависимости от структуры Q -детерминанта величина сокращения размера разная. Для подтверждения этого опишем проведенные вычислительные эксперименты.

Так как пользователи суперкомпьютера имеют ограничение на использование дискового пространства, было невозможно сформировать представления в форме Q -детерминанта алгоритма поиска максимального элемента в массиве чисел длины более 30. Использование нового формата и архивации по мере формирования файлов, содержащих выходные данные, позволило сформировать такие представления. В данном случае Q -детерминант алгоритма состоит из одного условного Q -терма длины $2^{(l-1)}$, где l — длина массива. Пары Q -терма таковы, что их безусловные логические Q -термы не повторяются, а безусловные Q -термы представляют из себя элементы массива. С учетом структуры Q -детерминанта был выбран первый вариант представления алгоритма в форме Q -детерминанта, содержащего условные Q -термы, с разделением по логическим Q -термам. С использованием нового формата были сформированы как представления в форме Q -детерминанта алгоритма, которые могут быть получены в исходном формате, так и представления, которые в исходном формате получить было невозможно. Результаты экспериментального исследования представлены в табл. 5.

В целях равномерного и универсального распределения пар условного Q -терма по файлам было принято решение размещать в каждом файле число пар, являющееся степенью числа 2. Тогда количество файлов также будет числом, являющимся степенью числа 2. Для организации параллельного формирования файлов было выбрано максимально доступное пользователю суперкомпьютера число MPI-процессов, являющееся степенью числа 2, а именно 256. Также при выборе количества информации в одном файле необходимо учесть, что суммарный размер параллельно формируемых файлов не должен превышать доступное дисковое пространство. Рассмотрим пример, когда массив чисел имеет длину 30. В этом случае число пар условного Q -терма равно $2^{29} = 536870912$. При формировании представления алгоритма в форме Q -детерминанта было получено $2^{12} = 4096$ архивированных файла, каждый из которых содержал $2^{17} = 131072$ пар условного Q -термов. Каж-

Таблица 5

Результаты экспериментального исследования алгоритма
поиска максимального элемента в массиве чисел

Длина массива	Объем выходных данных в исходном формате, МБ	Объем выходных данных в новом формате, МБ
24	10784	102,48
25	22528	213,92
26	46972	445,76
27	97779	927,36
28	203227	1926,40
29	421789	3996,16
30	874250	8279,04
31	—	17131,52
32	—	35409,92
33	—	73113,60
34	—	150814,70
35	—	310804,50

Таблица 6

Результаты экспериментального исследования алгоритма, реализующего метод Якоби для решения пятиточечных разностных уравнений

Количество итераций	Объем выходных данных в исходном формате, МБ	Объем выходных данных в новом формате, МБ
3	687581	4,6
4	2761375	14,9
5	11009442	51,6

дая из пар была описана в виде двух строк: первая содержит безусловный логический Q -терм, вторая идентификатор выходной переменной «max», знак равенства и безусловный Q -терм. В результате использования нового формата и архивации по мере формирования файлов, содержащих выходные данные, объем выходных данных сократился на 99 %.

В табл. 6 представлены результаты экспериментального исследования алгоритма, реализующего метод Якоби для решения пятиточечных разностных уравнений на сетке размера 100×100 , при этом используется представление алгоритма в форме Q -детерминанта, которое содержит только последнюю итерацию. В этом случае безусловные логические Q -термы одинаковы и имеют значительные размеры, а безусловные Q -термы небольшие, поэтому был применен второй вариант представления алгоритма в форме Q -детерминанта, содержащего условные Q -термы, и получено два файла, первый из которых содержит безусловный логический Q -терм, а второй все безусловные Q -термы. По мере формирования файлы были архивированы. В результате объем выходных данных сократился на 99,99 %.

Из-за ограничения на использование дискового пространства представления алгоритмов в форме Q -детерминанта для данного алгоритма, размер которых превышает ограничение, в исходном формате сформировать было невозможно, например, для количества итераций 4 и 5. Поэтому для табл. 6 объемы выходных данных в исходном формате для количества итераций 4 и 5 оценены аналитически. Для этого был проведен анализ струк-

Таблица 7

Результаты вычислительных экспериментов для алгоритма умножения двух плотных квадратных матриц с использованием предлагаемого стандарта

Размер матриц	Размер представления, ГБ	Количество файлов
3000	143	4800
3500	228	7200
4000	340	8400
4500	485	10800
5000	664	12600
5500	886	15600
6000	1149	18000
6500	1463	21150

туры представлений алгоритма в форме Q -детерминанта и выявлена зависимость объема выходных данных в исходном формате от параметров размерности и количества итераций. Неповторяемость безусловного логического Q -терма и архивирование выходных данных по частям позволили сформировать представления алгоритмов в форме Q -детерминанта для количества итераций 4 и 5 в новом формате.

Также были проведены вычислительные эксперименты, результаты которых могут быть использованы для решения практически значимых задач. Эти эксперименты заключаются в формировании представлений в форме Q -детерминанта алгоритма умножения двух плотных квадратных матриц с использованием схемы сдваивания для различных размеров матриц в соответствии с предлагаемым стандартом. Так как в данном случае представление алгоритма в форме Q -детерминанта содержит безусловные Q -термы, то его строки, согласно разработанному стандарту, были последовательно распределены по файлам. Следует отметить, что сформировать полученные представления без использования предлагаемого стандарта было бы невозможно, начиная с размера матриц 3500 включительно, из-за лимита дискового пространства при проведении экспериментов. Некоторые результаты экспериментов приведены в табл. 7.

Заключение. Параметры размерности алгоритмической проблемы при решении задач, применяемых на практике, могут иметь большие значения, кроме того, количество итераций при реализации итерационных алгоритмов может быть значительным. Это приводит к Q -термам большой длины и к представлениям алгоритмов в форме Q -детерминантов больших размеров. В этом случае Q -детерминанты мы называем большими, а возникающие проблемы при формировании и использовании представлений алгоритмов в форме Q -детерминантов проблемами больших Q -детерминантов. В статье описаны проблемы больших Q -детерминантов, предложены их решения и приведены экспериментальные исследования, подтверждающие правильность этих решений.

Использование предложенных решений делает возможной реализацию практических задач с помощью систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Одно из направлений дальнейших исследований, основанных на концепции Q -детерминанта, будет заключаться в исследовании этой возможности. Также полученные результаты можно использовать для повышения точности вычислений ресурсов параллелизма алгоритмов с помощью программной Q -системы. Кроме того, могут возникать и другие направления исследований, использующие решения, рассмотренные в данной статье.

Список литературы

1. Алеева В. Н. Анализ параллельных численных алгоритмов. Препринт № 590. Новосибирск: ВЦ СО АН СССР, 1985. 23 с.
2. Valentina Aleeva, Rifkhat Aleev. Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. Vol. 10. N 2, Article number 8. P. 1–64. DOI: 10.1145/3583755.
3. Алеева В. Н. Подход к эффективной реализации численных алгоритмов // Проблемы информатики. 2025. № 1. С. 29–44. DOI: 10.24412/2073-0667-2025-1-29-44.
4. Алеева В. Н., Кузнецов Е. К. Эффективная реализация алгоритмов сортировки с помощью концепции Q -детерминанта // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 2. С. 5–17. DOI: 10.25205/1818-7900-2025-23-2-5-17.
5. Алеева В. Н., Сапожников А. С. Эффективная реализация алгоритмов обучения нейронных сетей с помощью концепции Q -детерминанта // Проблемы информатики. 2025. № 3. С. 5–16. DOI: 10.24412/2073-0667-2025-3-5-16.
6. Алеева В. Н. Автоматизированное проектирование и исполнение эффективных программ для численных алгоритмов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 31–49. DOI: 10.14529/cmse230303.
7. Manatin P., Aleeva V. Efficient Implementation of Numerical Algorithms Based on a Lexical Analyzer // Parallel Computational Technologies. PCT 2024. Communications in Computer and Information Science. 2024. Vol. 2241. P. 107–121. DOI: 10.1007/978-3-031-73372-7_8.
8. Алеева В. Н. Разработка программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов на основе концепции Q -детерминанта // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 1. С. 5–18. DOI: 10.25205/1818-7900-2025-23-1-5-18.
9. Ершов Ю. Л., Палютин Е. А. Математическая логика. М.: Наука, 1987. 336 с.
10. Алеева В. Н., Зотова П. С., Склезнев Д. С. Расширение возможностей исследования ресурса параллелизма численных алгоритмов с помощью программной Q -системы // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 66–81. DOI: 10.14529/cmse210205.
11. Aleeva V. N. Improving Parallel Computing Efficiency // Proceedings — 2020 Global Smart Industry Conference, GloSIC 2020. IEEE. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.
12. Aleeva V., Bogatyreva E., Skleznev A., et al. Software Q -system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. 2019. Vol. 1129. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52.
13. Соколов М. П. Разработка программного обеспечения для формирования Q -детерминантов численных алгоритмов с Q -термами большой длины // Вып. квалиф. работа бакалавра по направлению «Фундаментальная информатика и информационные технологии»: 02.03.02 / Южно-Уральский государственный университет. Челябинск. 2020. 32 л. [Электрон. рес.]: http://omega.sp.susu.ac.ru/publications/bachelorthesis/2020_402_sokolovmp.pdf (Дата обращения: 15 декабря 2025 года).
14. Суперкомпьютер «Торнадо ЮУрГУ». [Электрон. рес.]: <http://supercomputer.susu.ru/computers/tornado/> (Дата обращения: 15 декабря 2025 года).
15. Соколов М. П. Формирование представлений алгоритмов в форме Q -детерминантов на кластерных вычислительных системах // Вып. квалиф. работа магистра по направлению «Фундаментальная информатика и информационные технологии»: 02.04.02 / Южно-Уральский го-

сударственный университет. Челябинск. 2022. 40 л. [Электрон. рес.]: http://omega.sp.susu.ru/publications/masterthesis/2022_220_sokolovmp.pdf (Дата обращения: 15 декабря 2025 года).

16. Oseledets I., Tyrtyshnikov E. TT-cross approximation for multidimensional arrays // *Linear Algebra and its Applications*, 2010. Vol. 432. N 1, P. 70–88. DOI: 10.1016/j.laa.2009.07.024.

17. Oseledets I. V. Tensor-Train Decomposition // *SIAM Journal on Scientific Computing (SISC)*, 2011. Vol. 33. N 5, P. 2295–2317. DOI: 10.1137/090752286.



Алеева Валентина Николаевна — e-mail: aleevavn@susu.ru; тел.: +7-351-267-90-89. В 1972 году окончила Новосибирский государственный университет с присвоением квалификации «Математика, прикладная математика».

В 1986 году решением Совета при Вычислительном центре СО АН СССР ей присуждена ученая степень кандидата физико-математических наук. В настоящее время она является доцентом кафедры системного программирования Южно-Уральского государственного университета (Национальный исследовательский университет).

Имеет 2 изобретения по архитектуре параллельных вычислительных систем и более 40 публикаций по распараллеливанию численных алгоритмов. Научные интересы: архитектура вычислительных систем, ресурсы параллелизма алгоритмов, проектирование эффективных программ, эффективность параллельных вычислений, в том числе в автоматизированном режиме.

Valentina Nikolaevna Aleeva — e-mail: aleevavn@susu.ru; tel.: +7-351-267-90-89. Graduated from Novosibirsk State University with the Master of Mathematics and Applied Mathematics degree in 1972 and defended his candidate dissertation in physical and mathematical sciences in Computing Centre of SB RAS in 1986. Currently, she is an Associate Professor of the Department of System Programming at South Ural State University (National Research University).

She has 2 inventions on the architecture of parallel computing systems and more than 40 publications on parallelization of numerical algorithms. The scientific interests include the architecture of computing systems, parallelism resources of algorithms, the design of effective programs, the efficiency of parallel computing, particularly in automated mode.



Соколов Михаил Павлович в 2022 году окончил Южно-Уральский государственный университет (Национальный исследовательский университет) и получил степень магистра. В настоящее время является преподавателем и аспирантом кафедры системного программирования Южно-Уральского государственного университета. E-mail: sokolovmp@susu.ru; тел.: +7-351-267-90-89.

Имеет 4 публикации по распараллеливанию алгоритмов. Научные интересы: параллельные вычисления, проектирование параллельных программ, эффективность параллельных вычислений.

Имеет 4 публикации по распараллеливанию алгоритмов. Научные интересы: параллельные вычисления, проектирование параллельных программ, эффективность параллельных вычислений.

Sokolov Mikhail Pavlovich graduated from South Ural State University (National Research University) with the Master degree in 2022. Currently, he is a teacher and postgraduate of the Department of System Programming at South Ural State University. E-mail: sokolovmp@susu.ru; tel.: +7-351-267-90-89.

He has 4 publications on parallelization of algorithms. The scientific interests include parallel computing, the design of parallel programs, the efficiency of parallel computing.

Дата поступления — 26.01.2026

DEBUGGING SYSTEM OF SIMPLE COMPUTATIONAL MODELS IN GRAPH REPRESENTATION

A. Vlasenko, M. Gorodnichev, M. Kurbatov*

Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia

Novosibirsk State Technical University,
630073, Novosibirsk, Russia

*Novosibirsk State University,
630090, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2026-1-24-39

EDN: NPOQMK

Computational models (CM) is a formalism for representing knowledge about computing in problem domains. A CM defines a set of domain quantities (variables) and a set of operations that functionally connect the quantities. The choice of subsets V (known variables) and W (desired variables) determines the formulation of the problem on the CM, and the partial order of operations that allows one to “compute” W by V is called a (V, W) -plan. With appropriate computer implementation of variables and operations, automated problem solving based on CMs becomes possible. Building a CM — as a theory of the problem domain — is a complex task that requires significant skills and experience.

It is convenient to represent a CM as a bipartite graph with vertices of two types: variables and operations. Some operations produce variable values as results that will be input for other operations.

There are several software systems for designing and using CMs (LuNA, HPC Community Cloud, CoCoViLa, Orlando Tools, etc.) These systems have different interfaces and the user experience varies significantly depending on the system. For example, in the HPC Community Cloud, a user can place elements corresponding to values and operations on a canvas in the web interface and link them with arrows. In the LuNA system, the user writes a program in the language of the same name, similar to functional languages.

When using such systems, the user can make a number of mistakes, both when developing a CM and when setting problems. As a result, the solutions to the problem may either not exist at all, or they may differ significantly depending on the choice of the (V, W) -plan.

The following are several types of situations that should be taken into account.

1) The CM-graph has several connectivity components. If the graph is large enough (several dozen or more vertices with many arcs), then this situation may be difficult to detect visually. The existence of several connectivity components may indicate that the modeled subject area should be divided into several independent areas and investigated separately.

2) The inability to calculate any variables from others. In many fields of physics, mathematics, and other sciences, all values are interconnected by formulas, and it is always possible to calculate any value through others. In this case, the presence of a variable that is not the result of any operation is an error in CM-development. Although there may be subject areas in which some variables are always only input and are not calculated through others.

This work was carried out under state contract with ICMMG SB RAS FWNM-2025-0005.

3) A number of possible (V, W) -plans for solving the problem. This situation may be normal, since often the same problem can be solved using different methods, but it may also indicate unnecessary connections that were mistakenly entered into the CM.

4) The computational model design system chooses an inefficient (V, W) -plan according to some criterion. For example, it was not taken into account that the computing system on which the task will be run has graphics accelerators; an inefficient method for solving a problem was chosen (for example, the Kramer method was chosen instead of an iterative method for solving a system of linear equations), etc.

5) When designing the CM, the limitations of the problem domain were not taken into account (for example, that a certain value cannot be negative, that the values are dependent on each other, etc.)

Therefore, it is necessary to develop a CM debugging system that performs a number of checks on the CM in an automated mode and is capable of analyzing the (V, W) -problems. This paper is devoted to the description of such a system.

The system under development receives a CM in the form of a JSON file with a description in a specific format (this format is described in the paper). At the moment, the user is downloading it from the file system. In the future, imports will be implemented directly from CM design systems. To do this, it is necessary to develop a parser for the internal CM representation of each of these systems. After import, the CM graph is displayed in the system's web interface. The user has the opportunity to set a (V, W) -problem by selecting the vertices of the sets V and W . In this case, all existing (V, W) -plans for this problem will be displayed on the screen. If there are no solutions to the problem in this CM, then this will be notified to the user. The system also detects the presence of several connectivity components of the CM graph.

Key words: computational model, high performance computing systems, bipartite graph, automated debugging, visualization.

References

1. Valkovskii V. A., Malyshkin V. E. Sintez paralelnih program i sistem na vichislitelnykh modeliah. Novosibirsk: Nauka, 1988. 129 s.
2. Malyshkin V. E., Korneev V. D. Parallelnoe programmirovaniye multicomputerov: uchebnik. Novosibirsk: Izd-vo NGTU, 2006. 296 s.
3. Tiugu E. H. Konceptualnoye programmirovaniye. M.: Nauka, 1984. 256 s.
4. Gorodnichev M., Lebedev, D. Semantic tools for development of high-level interactive applications for supercomputers // J. Supercomput. 2021. P. 11866–11880. DOI: <https://doi.org/10.1007/s11227-021-03731-6>.
5. Malyshkin V., Akhmed-Zaki D., Perepelkin V. Parallel programs execution optimization using behavior control in LuNA system // J. Supercomput. 2021. P. 9771–9779. DOI: 10.1007/s11227-021-03654-2.
6. Kotkas V., Ojamaa A., Grigorenko P., Maigre R., Harf M., Tyugu E. CoCoViLa as a multifunctional simulation platform // SIMUTOOLS 2011. 4th International ICST Conference on Simulation Tools and Techniques: March 21–25, - Barcelona, Spain: Brussels: ICST, 2011. P. 1–8.
7. Feoktistov A. G., Kostromin R. O., Voskoboynikov M. L., Li-De D. I. Organizatsiya vichislitelnoy sredi razrabotki i primeneniya nauchnykh rabotnykh processov na osnove konteynirizatsii // Vichislitelnye tehnologii. 2023. T. 28. N 6. S. 151–164.
8. Kurbatov M. A. Poisk oshibok vo fragmentirovannykh programmakh s pomosh'yu abstractnogo sintaksicheskogo dereva // Materiali 62-I Mezhdunarodnoy studencheskoy konferentsii. Novosibirsk, 2024. S. 158–159.
9. Malyshkin V., Vlasenko A., Michurov M. Automated Debugging of Fragmented Programs in LuNA System // D. Balandin et al. (Eds.): 22nd International Conference, MMST 2022, Nizhny

Novgorod, Russia, November 14–17, 2022. CCIS 1750, P. 266–280. DOI: 10.1007/978-3-031-24145-1_22.

10. Vlasenko A., Michurov M., Tsarev V., Kurbatov M. Postroenie kompleksa avtomatizirovannoy otladki fragmentirovannih programm // Vestnik NGU. Seriya: Informacionnie tehnologii. 2024. T. 22, N 1. S. 5–20. DOI: 10.25205/1818-7900-2024-22-1-5-20.

11. Gorodnichev M.A., Nalepova E.D., Merkulova E.A., Rudych P.D., Savostyanov A.N. Automation of EEG Data Processing with HPC Community Cloud // IEEE 24th International Conference of Young Professionals in Electron Devices and Materials (EDM), Novosibirsk, Russian Federation, 2023. P. 1320–1323. DOI: 10.1109/EDM58354.2023.10225226.

12. Fielding R.T. REST: Architectural Styles and the Design of Network-Based Software Architectures. Doctoral Dissertation, University of California, 2000.

13. Aho A., Hopcroft D., Ulman D. Structuri danih i algoritmi. M.: Vilyams, 2003. S. 189–197.

14. Knut D. Iskusstvo programmirovaniya. T. 1. 3-e izd. M.: Vilyams, 2006. S. 174–175.

СИСТЕМА ОТЛАДКИ ПРОСТЫХ ВЫЧИСЛИТЕЛЬНЫХ МОДЕЛЕЙ В ГРАФОВОМ ПРЕДСТАВЛЕНИИ

А. Ю. Власенко, М. А. Городничев, М. А. Курбатов*

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

Новосибирский государственный технический университет,
630073, Новосибирск, Россия

*Новосибирский национальный исследовательский государственный университет,
630090, Новосибирск, Россия

УДК 004.4

DOI: 10.24412/2073-0667-2026-1-24-39

EDN: NPOQMK

Вычислительные модели (ВМ) — формализм для представления знаний о вычислениях в предметных областях. ВМ определяет множество переменных, соответствующих величинам предметной области, и множество операций, выражающих функциональные связи между величинами. Выбор подмножеств V — известных величин — и W — искомых величин — определяет постановку задачи на ВМ, а подмножество операций и переменных, связанных отношением частичного порядка, задающее схему вычисления W по V , называется (V, W) -планом. При соответствующей компьютерной реализации величин и операций обеспечивается возможность автоматизированного решения задач на основе ВМ. Построение ВМ — сложная задача, требующая значительных навыков и опыта.

Возможно допустить ряд ошибок, как при разработке ВМ, так и при постановке задач, вследствие чего решений у задачи может либо не быть вообще, либо они могут существенно отличаться в зависимости от выбора (V, W) -плана. Следовательно, необходима разработка системы отладки ВМ, проводящей ряд проверок над ВМ в автоматизированном режиме и способной выполнять анализ поставленных (V, W) -задач. Настоящая статья посвящена описанию такой системы.

Ключевые слова: вычислительная модель, высокопроизводительные вычислительные системы, двудольный граф, автоматизированная отладка, визуализация.

Введение. В рамках этой работы будут рассматриваться простые вычислительные модели (ПВМ). Согласно определению, данному в [1], ПВМ — это пара $M = (X, F)$, где: $X = \{x, y, z, \dots\}$ — конечное множество переменных для представления вычисляемых и измеряемых величин; $F = \{a, b, c, \dots\}$ — конечное множество функциональных символов операций арности $m \times n$, $m \geq 0$; m и n могут быть различны для разных операционных символов. Эти символы служат для выражения функциональных зависимостей между величинами предметной области; с каждым символом операции a арности $m \times n$ связан набор $in(a) = (x_1, \dots, x_m)$ входных и набор $out(a) = (y_1, \dots, y_n)$ выходных переменных, при этом $i \neq j \implies y_i \neq y_j, x_i \neq x_j$.

Исследования выполнены в рамках государственного задания ИВМиМГ СО РАН FWNM-2025-0005.

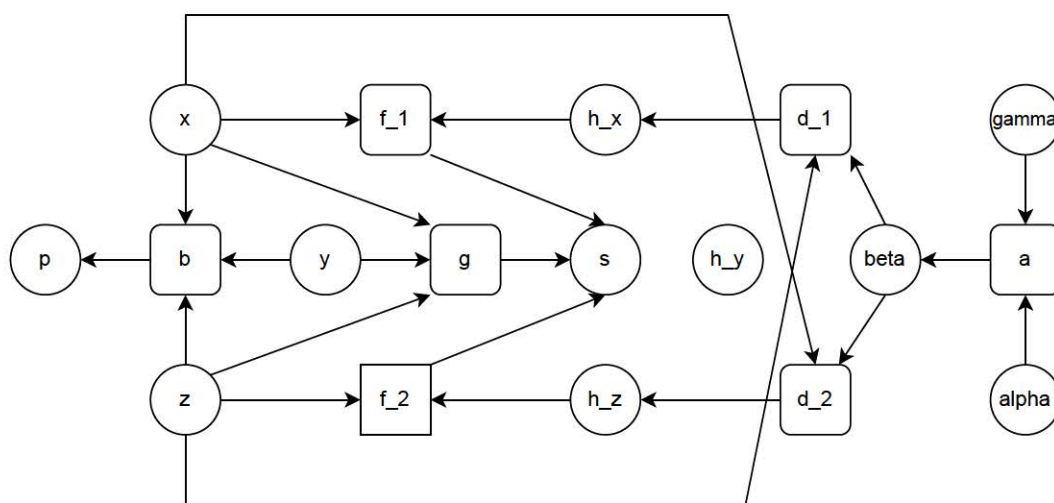


Рис. 1. ПВМ «треугольник»

ПВМ удобно представлять в виде двудольного графа переменных и операций. Пример такого графа для предметной области «треугольник» изображен на рис. 1.

На рис. 1 кругами обозначены символы переменных, а квадратами — символы операций. x, y, z — стороны треугольника; p — периметр; s — площадь; $alpha, beta, gamma$ — углы; h_x, h_y, h_z — высоты, опущенные на соответствующие стороны. Операции d_1 и d_2 вычисляют h_x и h_z соответственно. f_1, f_2 и g вычисляют s . a и b вычисляют $beta$ и p соответственно. Данная модель, естественно, охватывает не всю предметную область и приведена для демонстрации. Так, в модели, например, отсутствуют медианы и биссектрисы треугольника.

Необходимо также ввести понятие термина, которое тесно связано с вычислительными моделями. Терм — математическая абстракция, определяемая по индукции [2]:

- 1) Термами длины 1 называются односимвольные слова из множества переменных.
- 2) Далее, пусть для некоторого $k > 1$ термы длины меньше k уже определены. Тогда слово t длины k называется термом, если оно имеет вид $f(t_1, t_2, \dots, t_n)$, где $t_1, t_2, \dots, t_n$ — термы длины меньшей чем k , f — n -арный функциональный символ (n входных переменных).

Применительно к приведенной выше ПВМ термами являются, например, $g(x, y, z)$; $f_1(d_1(beta))$; $f_2(d_2(a(alpha, gamma)))$. Можно заметить, что терм определяет некоторый алгоритм — задает частичный порядок применения операций.

Для практического использования ПВМ требуется реализовать представление ее в компьютере [3]. Операции и величины должны иметь программную реализацию. Например, операции могут быть реализованы программными модулями, величины представлены в переменных или файлах. Должна быть обеспечена возможность организовать вызов программной реализации операции, подав на вход операции значения входных величин, чтобы вычислить значения выходных.

На ПВМ можно ставить задачи (так называемые (V, W) -задачи) в следующей форме: дано подмножество вершин V , значения каждой из которых известны. Необходимо найти значения всех вершин другого подмножества W . Используя операции из F , входными

параметрами для которых являются переменные из V , можно вычислить другие переменные, после чего применить следующую группу операций и еще расширить множество вычисленных переменных. Этот процесс можно продолжать до тех пор, пока среди вычисленных переменных не окажутся все элементы подмножества W . Однако может так оказаться, что далеко не все операции действительно было необходимо применять и не все полученные результаты нужны для решения задачи. В то же время вычислительные ресурсы на получение данных результатов были затрачены.

В связи с этим при решении подобных задач требуется составление так называемого (V, W) -плана вычислений, представляющего собой частично-упорядоченное подмножество операций и вычисляемых переменных. Применение операций в заданном порядке обеспечивает вычисление значения всех переменных из W при заданных значениях V . При этом в (V, W) -план не должно включаться выполнение «лишних» операций, не требующихся для нахождения значений элементов множества W . Данный план по сути своей задает алгоритм решения задачи в терминах операций, и таких планов для одной и той же задачи на ПВМ может быть множество. Следует отметить, что (V, W) -план удобно задавать термом, определяющим порядок операций, где термами длины 1 являются входные переменные. Однако это справедливо только для тех ВМ, где каждая операция имеет единственную выходную величину, либо множество выходных величин представимо в виде вектора, который поступает на вход следующей операции целиком.

Существуют системы описания вычислительных моделей, которые могут выглядеть совершенно различным образом. Например, HPC Community Cloud (HPC2C) [4] — система управления вычислениями на распределенных высокопроизводительных вычислительных ресурсах — предоставляет возможность в web-интерфейсе размещать на холсте символы операций и переменных, соединяя их связями, а система LuNA [5] содержит текстовый язык описания вычислительных моделей, схожий с функциональными языками. Система CoCoViLa [6] предлагает богатые визуальные и текстовые средства для описания сложных схем и детального описания структурированных компонентов. Программный комплекс Orlando Tools использует язык BPEL для представления ВМ с целью создания распределенных научных приложений в гетерогенной вычислительной среде [7].

Существенной проблемой при описании сложных предметных областей с помощью ПВМ и последующем их использовании для решения задач является возможность допустить множество семантических ошибок или получить неэффективный (относительно интересующих пользователя нефункциональных требований) (V, W) -план. Примерами могут служить включение в модель некорректных модулей в качестве реализации операций (вычисляющих не ту функцию, которая предполагалась разработчиком модели), установка некорректных связей (ошибка в составлении наборов *in* и *out* для операций), постановка неразрешимой на данной ПВМ (V, W) -задачи, построение неоптимального по некоторому критерию (V, W) -плана и др. Если ПВМ насчитывает десятки или сотни элементов (переменных и операций), то для обнаружения таких ошибок «вручную» может потребоваться очень длительное время. Соответственно, актуальной задачей является разработка алгоритмов обнаружения ошибок и программной системы, реализующей эти алгоритмы.

В связи с этим была поставлена задача разработки системы отладки с возможностью импорта ПВМ и (V, W) -задач из различных систем и последующего их анализа с целью выявления ошибок.

Стоит отметить, что некоторые системы конструирования ВМ обладают собственными средствами отладки. Так, например, для системы LuNA велись разработки инструментов

статического анализа [8], анализа по трассе [9], а в настоящее время поддерживается программный комплекс ADAPT [10], комбинирующий несколько подходов. Однако подобные инструменты способны определять ошибки, свойственные конкретной системе или технологии и не предназначены для обнаружения ошибок, характерных вычислительным моделям в общем.

1. Система графового представления простых вычислительных моделей. В рамках разрабатываемой системы необходимо было решить вопрос хранения ПБМ в некотором формате. ПБМ представима в виде двудольного графа. Таким образом, для его компьютерного представления можно использовать такие структурные языки, как JSON или XML. За основу взят формат ПБМ в HPC2C [11]. На базе данного формата был создан упрощенный унифицированный формат, который имеет ту же структуру и упрощенный набор полей: идентификатор (id), название (title) и тип (node_type) — переменная (variable) или операция (operation) для списка узлов (nodes) и идентификаторы начала (source) и конца (target) для дуг графа (edges). Таким образом, система отладки хранит граф как список вершин и список ребер.

Листинг 1. Пример графа в унифицированном формате.

```
{
  "nodes": [
    {
      "id": 0,
      "node_type": "VARIABLE",
      "title": "VARIABLE 2",
    },
    {
      "id": 1,
      "node_type": "OPERATION",
      "title": "OPERATION 0",
    },
    {
      "id": 2,
      "node_type": "VARIABLE",
      "title": "VARIABLE 2",
    }
  ],
  "edges": [
    { "source": 0, "target": 1 },
    { "source": 1, "target": 2 }
  ]
}
```

Унифицированный формат удобен для хранения ПБМ и построения визуализаций. Также подходит для решения задачи импорта ПБМ из различных систем (подробнее об этом далее). Но для задачи поиска необходимой информации в графе в целях анализа модели данный формат оказывается неподходящим. Например, для поиска всех входных переменных, нужных для работы определенной операции, в JSON-файле необходимо пе-

Таблица 1

Сложности операций для графовых структур данных

Структура данных	Поиск вершин, которые зависят от текущей	Перечисление всех дуг	Поиск вершин, от которых зависит текущая	Проверка существования дуги между u и v	Затраты по памяти на хранение структуры
Перечисление множеств	$\Theta(E)$	$\Theta(E)$	$\Theta(E)$	$\Theta(E)$	$\Theta(E + V)$
Матрица смежности	$\Theta(V)$	$\Theta(V ^2)$	$\Theta(V)$	$\Theta(1)$	$\Theta(V ^2)$
Матрица инцидентности	$\Theta(E)$	$\Theta(V \times E)$	$\Theta(E)$	$\Theta(E)$	$\Theta(V \times E)$
Перечень дуг	$\Theta(E)$	$\Theta(E)$	$\Theta(E)$	$\Theta(E)$	$\Theta(E)$
Вектор/ массив/ список смежности	$\Theta(\deg(v))$	$\Theta(E)$	$\Theta(E)$	$\Theta(\min(\deg(u), \deg(v)))$, если список вершин не отсортирован $\Theta(\min(\deg(u), \deg(v)))$, если список вершин отсортирован	$\Theta(V + E)$
Структура с оглавлением	$\Theta(\deg(v))$	$\Theta(E)$	$\Theta(E)$	$\Theta(\min(\deg(u), \deg(v)))$, если список вершин отсортирован	$\Theta(V + E)$
Список вершин и список дуг	$\Theta(E)$	$\Theta(E)$	$\Theta(E)$	$\Theta(E)$	$\Theta(V + E)$

ребирать все дуги (edges), после чего по идентификаторам искать нужную информацию в узлах (nodes).

В целях обеспечения возможности экономичного поиска и анализа ПВМ был проведен обзор структур данных, служащих для представления графов (таблица 1). В таблице приведены сложности операций, требующихся для анализа ПВМ.

В таблице 1 употреблены следующие обозначения:

- 1) Θ — оператор, означающий, что количество операций пропорционально аргументу;
- 2) $|V|$ — количество вершин;
- 3) $|E|$ — количество дуг;
- 4) $\deg(u)$ — полная степень вершины u .

При помощи «Массива смежности» (рис. 2) можно достаточно быстро находить пути между вершинами, и она не занимает много места в памяти, в связи с чем было принято решение использовать данную структуру в целях дальнейшей отладки ПВМ. В терминах ПВМ эта структура выглядит следующим образом. Все вершины операций объединены в массив, и каждая имеет ссылку на список вершин переменных — множество входящих в эту операцию дуг. Дополнительно, каждая операция имеет ссылку на вершину переменной (или ссылки на несколько вершин), которая является результатом данной операции.

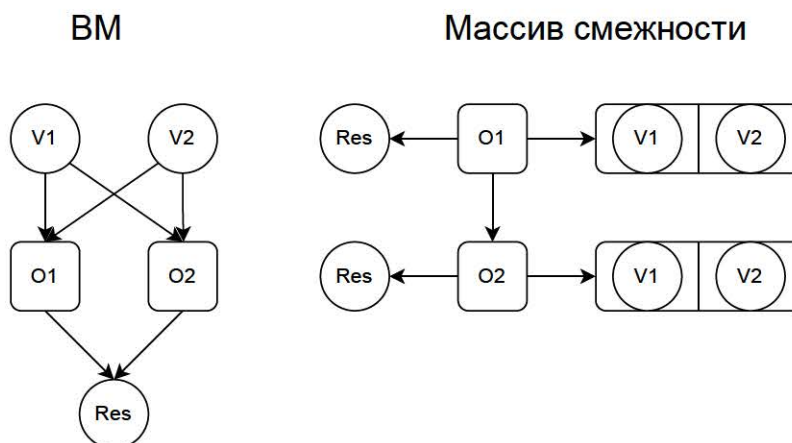


Рис. 2. Массив смежности

Разрабатываемая система отладки не хранит массив смежности в файловой системе. Он создается после импорта ПВМ перед проведением проверок, информация о которых изложена далее.

2. Обнаружение ошибок и выполнение проверок. Пользователь сам конструирует вычислительные модели, исходя из знаний о предметной области. При создании ВМ и (V, W) -планов могут возникать разного рода ошибки, которые могут приводить к некорректным вычислениям (результаты не соответствуют ожидаемой функциональной зависимости в моделируемой предметной области) или оптимизационным проблемам. Далее рассмотрены ситуации, потенциально свидетельствующие о наличии таких ошибок.

Во-первых, стоит обратить внимание на количество компонент связности (КС) в графе ВМ. Если их несколько (более одной), то нужно поставить вопрос об адекватности составленной модели предметной области и о разбиении одной модели на несколько независимых. Это дает преимущество в производительности последующего анализа ВМ за счет разбиения большого графа на несколько более мелких. Однако данный недочет не приводит к ошибкам вычислений, а скорее является плохой практикой.

Далее важно проанализировать возможные варианты вычислений переменных множества W по переменным из V . Если при конструировании неправильно проставить связи между вершинами или забыть добавить нужные операции, связать лишние переменные с операциями и т. д., то это вполне может привести к отсутствию возможности вычислить какую-либо переменную из W . На исходной же ПВМ (безотносительно поставленных (V, W) -задач) можно проверять тот факт, что любую переменную можно вычислить хотя бы одним способом. Противное может и не являться ошибкой, если, например, в ПВМ существуют переменные, которые всегда должны быть только входными. Однако во многих областях математики и физики все величины, которые имеет смысл заложить в ПВМ при моделировании, связаны между собой формулами, используя которые, можно выразить любую переменную через другие.

Часто встречаются ситуации, когда одну и ту же переменную из W можно вычислить по переменным из V несколькими разными способами. Например, систему линейных уравнений можно решать методом Гаусса, Крамера или итерационным методом, и все эти

модули решения СЛАУ могут присутствовать в ПВМ одновременно. Соответственно, для одной и той же (V, W) -задачи в таком случае можно построить несколько (V, W) -планов. Эта множественность вычислений допускается при конструировании ПВМ и не является ошибкой. Однако если при реализации операций допущены ошибки, то результат решения задачи будет зависеть от выбранного плана. Это может объясняться, например, ошибками округления и с прикладной точки зрения быть приемлемым, а может быть следствием принципиальных ошибок. Такие ситуации также необходимо выявлять, а для определения прикладной эквивалентности результатов должна быть соответствующая процедура. Очевидно, что такая процедура не может быть реализована только на основе графа ВМ, поскольку требуется непосредственный запуск операций, составляющих (V, W) -планы.

При составлении (V, W) -плана встает вопрос об оптимальности каждого возможного варианта вычислений по каким-либо критериям. Важным может быть учет следующих параметров:

- 1) Время работы наиболее часто вызываемых модулей на конкретном вычислителе.
- 2) Потребление оперативной памяти модулями.
- 3) Максимальная масштабируемость модулей по процессорным ядрам и вычислительным узлам.
- 4) Возможность использовать спец.ускорители (GPU, NPU и др.)
- 5) И др.

Разные ВМ могут быть связаны с разными предметными областями. Например, с биологией, физикой, математикой и т. д. Важно понимать, что каждая предметная область имеет свои ограничения. Например, сумма углов треугольника в евклидовой геометрии всегда равна 180 градусов, при нормальном атмосферном давлении вода кипит при 100 градусах Цельсия и т. д. Данные ограничения можно закладывать в ВМ при ее конструировании. Для этого необходимо выйти за рамки абстракции ПВМ и предоставить специалисту в предметной области возможность формализации подобных ограничений в прикладном инструменте работы с ВМ.

По ходу вычисления переменных можно собирать статистику использования элементов ВМ и результатов работы операций. Если периодически какая-то операция выдает некорректный результат, то выдавать составителю ВМ сообщение об этом и при составлении (V, W) -планов использовать альтернативные маршруты. Если по статистике использования выясняется, что какие-то переменные или операции из вычислительной модели не используются, то стоит поднять вопрос корректности и/или целесообразности хранения данных элементов в ВМ.

3. Архитектура и реализация системы отладки ПВМ.

3.1. *Архитектура системы.* Для возможности выполнения обозначенных проверок над ПВМ и (V, W) -планами в автоматизированном режиме разрабатывается система отладки, представляющая собой веб-приложение, в которое загружается ПВМ в описанном выше JSON-формате. Система позволяет отрисовывать граф ПВМ и проводить ряд проверок, список которых приведен ниже. Она состоит из трех слоев (рис. 3):

1) Слой “Frontend” (интерфейс приложения) предоставляет пользователю возможность отрисовывать граф ВМ, искать по нему нужную информацию и запускать проверки.

2) Слой “Backend” реализует всю логику работы с графами, термами и компонентами связности. На данный момент этот слой представлен монолитным компонентом. В дальнейшем возможен переход к микросервисам: поиск термов и анализ связности могут быть вынесены в отдельные сервисы по мере развития системы.

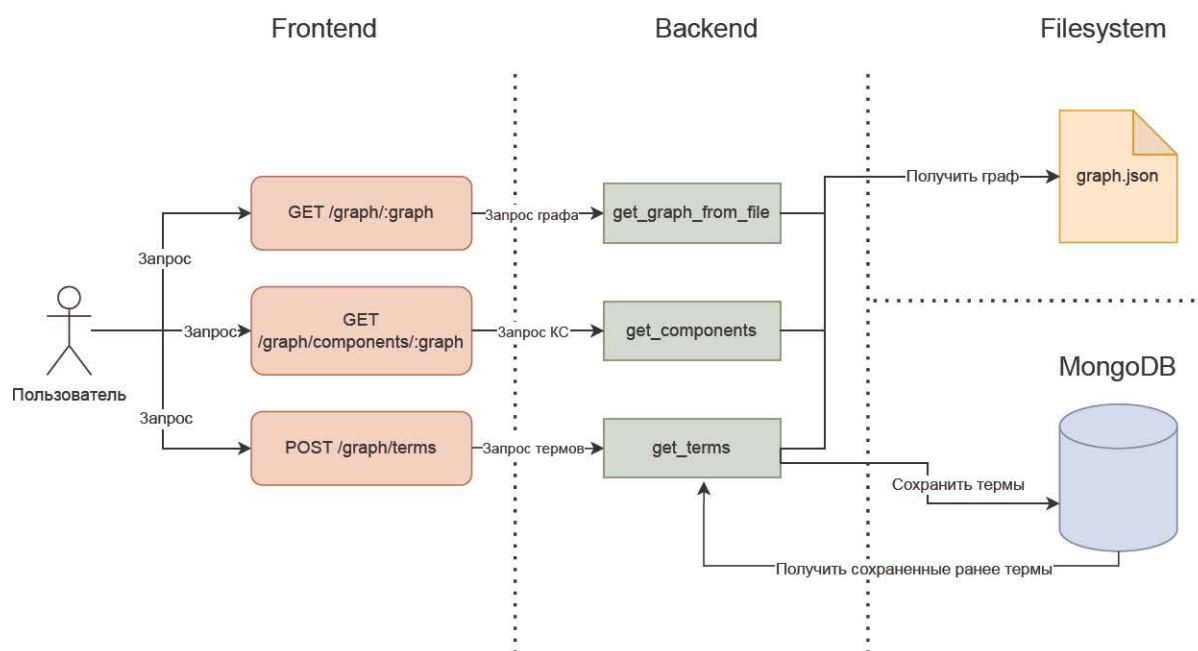


Рис. 3. Архитектура системы

3) Файловая система и БД (база данных). Данный слой хранит всю информацию о графах и результатах их анализа. Исходные графы хранятся в файловой системе (на рис. 3 — “Filesystem”), а найденные термы в базе MongoDB.

3.2. *Технологический стек.* В качестве основного языка программирования используется Rust, веб-приложение строится на основе фреймворк Axum. Для разработки фронтенда используется реактивный фреймворк Vue.js в версии с TypeScript и сборщиком Vite. Управление состоянием реализовано через глобальное хранилище Pinia, а для визуализации графов применяется библиотека Vis.js (vis-network и vis-data).

3.3. *Доступные функции.* На данный момент реализованы следующие функции:

1) «Импортировать граф» — загрузка графа ПБМ из JSON-файла с персонального компьютера пользователя и визуализация графа. В этом файле граф должен быть представлен в унифицированном формате, описанном в разделе 1. Система выполняет укладку графа на плоскость при помощи библиотеки Vis.js, автоматически задавая при этом координаты вершин. Вершины переменных отображаются сплошной границей, а операции пунктирной (рис. 1). В дальнейшем планируется добавить импорт графов из ряда других систем конструирования ВМ.

2) «Получить компоненты связности» — запрос данных о КС графа по его названию. Для удобства восприятия каждая компонента отображается своим цветом.

3) «Выбор V и W» — каждую из вершин переменных можно добавить во множество входных (V) или выходных (W) переменных, создав таким образом (V, W)-задачу. Входные вершины помечаются синим цветом, а выходные — красным.

4) «Поиск термов» — если заданы множества V и W, то можно определить все возможные пути вычисления выходных переменных из входных (см. 4.2. «Алгоритм поиска термов»). Используемые переменные и пути вычисления отображаются зеленым цветом, а целевые переменные — синим. Термы отображаются в правом верхнем углу в форме $z = f(y, x, w)$ (рис. 4).

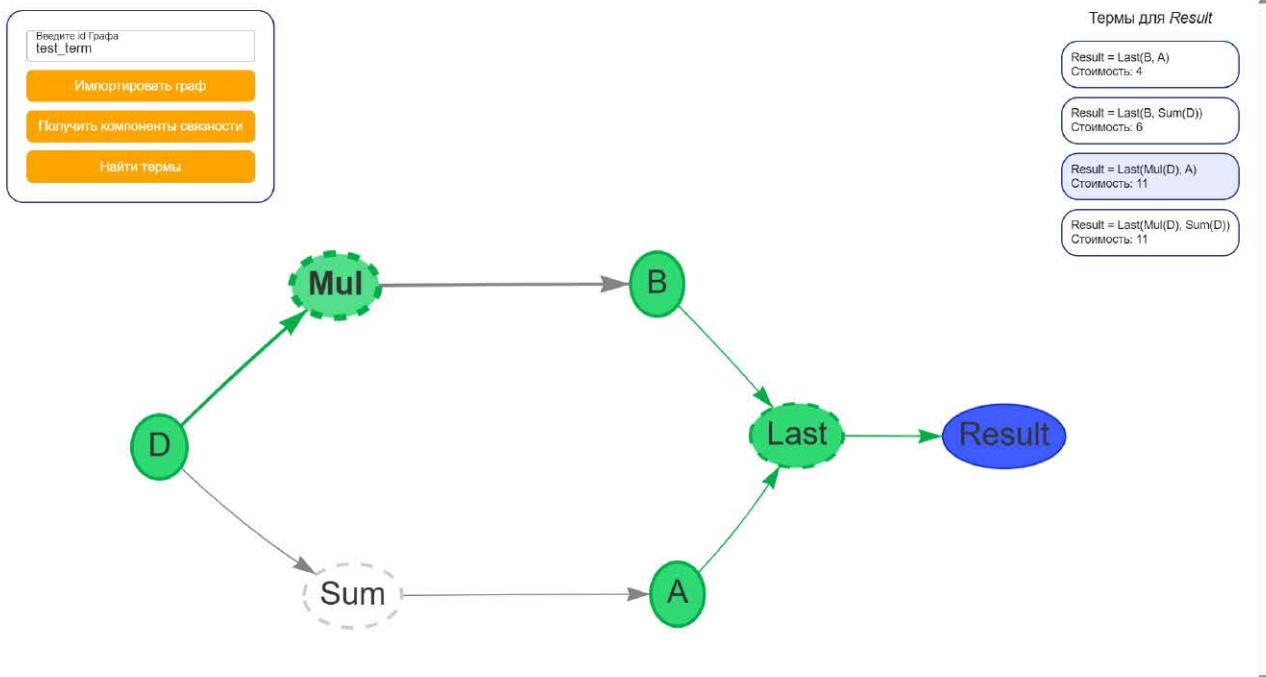


Рис. 4. Найденные термы

3.4. *База данных.* В качестве системы управления базами данных выбрана MongoDB, потому как данные в системе не имеют сложных связей и зависимостей между собой. В базе сохраняются множества термов, которые могут вычислить заданные переменные. Пример записи в базе данных, который показывает термы, вычисляющие переменную с $id = 6$.

Листинг 2. Результат поиска термов.

```
{
  "id": "681f93884fef4f937f95ac29",
  "graph_id": "big",
  "terms": [
    {
      "operations": [4],
      "variables": [5, 3],
      "result_var": 6
    },
    {
      "operations": [1],
      "variables": [0, 2],
      "result_var": 6
    }
  ],
  "result_var": 6,
  "created_at": "2025-05-10T17:57:28.777Z"
}
```

В листинге 2:

- 1) `id` — идентификатор записи, сгенерированный MongoDB.
- 2) `graph_id` — название файла, содержащего ПВМ.
- 3) `terms` — список термов (список `id` операций и переменных и `id` вычисляемой переменной).
- 4) `result_var` — `id` переменной, которую вычисляют термы.
- 5) `created_at` — дата создания записи.

3.5. Импорт ПВМ из других систем. В перспективе в описываемой системе отладки ПВМ предполагается реализовать возможность работать с множеством систем конструирования ВМ благодаря конвертации разных представлений вычислительных моделей в унифицированный формат. Для того чтобы добавить поддержку нового формата ПВМ из какой-либо системы, достаточно создать соответствующий парсер. Например, HPC2C предоставляет RESTful интерфейс [12] для управления вычислительными моделями. Имеется возможность автоматически получить доступ к вычислительным моделям, провести их анализ, модифицировать или внести сведения об ошибках в метаданные вычислительной модели, операций и переменных и сохранить исправления в хранилище моделей HPC2C.

4. Алгоритмы анализа вычислительных моделей. Выполнение проверок. После импорта графа и построения массива смежности в разрабатываемой системе выполняются проверки в соответствии с описанными ниже алгоритмами.

4.1. Алгоритм поиска компонент связности. Ориентированный граф рассматривается как неориентированный, направления дуг игнорируются. Строится система непересекающихся множеств (СНМ) по следующему алгоритму [13]:

- 1) Создается n множеств, где каждая вершина графа образует отдельное множество, т. е. является единственным представителем своего множества.

- 2) Для каждого ребра объединяются множества вершин, которые оно соединяет. Пусть ребро соединяет две вершины: x и y . Если они являются представителями одного и того же множества, то переходим к другому ребру. Иначе объединяем, присоединяя множество с меньшим рангом (мощность множества во внутреннем представлении СНМ) к множеству с большим рангом. Если ранги равны — выбираем любое множество для присоединения.

Сложность данного алгоритма построения СНМ: $O(E \times \alpha(V))$, где E — число ребер, V — число вершин, α — обратная функция Аккермана [14].

4.2. Алгоритм поиска термов. Поиск термов — основополагающая задача при анализе многих ошибок в ПВМ. Например, выполнить анализ вариантов и определить саму возможность вычисления переменной можно только при наличии информации о том, в каких термах она участвует.

Для поиска термов был реализован алгоритм обхода графа в глубину от каждой выходной переменной (листинг 2). Пусть V — множество входных переменных, а W — множество выходных, `visited` — список уже посещенных вершин. Требуется найти все термы, которые бы вычисляли каждую переменную w из W , используя лишь множество V и операции из F . Для каждого w из W выполняется процедура *build_term* (листинг 3).

Листинг 3. Алгоритм поиска термов.

```
fn build_term(graph, w, V, visited) -> Term[]:
0: terms = []
1: Если visited содержит w,
   то вернуть пустой массив термов terms.
2: Добавить w в visited.
3: Если нет операции, вычисляющей w,
   и w не содержится во множестве входных переменных V,
   то вернуть пустой список термов (terms).
   Однако если w содержится в V,
   то вернуть единственный терм - сам w.
4: Для каждой операции, вычисляющей w:
   4.1: Рекурсивно построить термы для аргументов (build_term)
   4.2: Скомбинировать все варианты
        подтермов (что определяет множество
        вариантов вычисления w) и добавить в terms
5: Вернуть terms
```

Также дополнительно результаты анализа сохраняются в MongoDB для последующего использования.

4.3. *Реализованные проверки.* На данный момент система отладки ПВМ способна обнаруживать следующие ситуации:

- 1) Наличие более чем одной компоненты связности в графе ПВМ.
- 2) Отсутствие термов, которые бы могли вычислить выходную переменную по наперед заданному множеству входных (невозможность решить (V, W) -задачу).
- 3) Возможность вычисления выходной переменной различными способами (множество возможных (V, W) -планов).

Заключение. Синтез программ на вычислительных моделях — актуальное направление исследований в области автоматизации конструирования программ. Описав предметную область в виде ВМ, исследователи получают возможность решать на ней различные задачи в автоматизированном режиме, при этом процесс получения решения является отслеживаемым, объяснимым, воспроизводимым. Существуют и развиваются системы для организации решения задач на основе этого формализма.

Существенной проблемой в разработке ВМ является возможность допустить ряд ошибок, как со стороны специалиста в предметной области при конструировании ВМ, так и со стороны разработчиков при создании системы для работы с ВМ, в результате чего (V, W) -планы могут строиться не оптимальным образом по выбранным критериям или даже некорректно. Если ВМ достаточно громоздкая, то обнаружить ошибки «вручную» может быть крайне затруднительно.

Описанная в настоящей статье система отладки предназначена для автоматизации поиска ошибок в ПВМ и (V, W) -планах. Она обладает web-интерфейсом, где пользователь может загрузить ПВМ из JSON-файла определенного формата. В перспективе будет добавлена возможность импорта ПВМ и (V, W) -планов из ряда систем управления вычислительными моделями. В интерфейсе системы отображается граф загруженной ПВМ, система осуществляет поиск ситуаций, свидетельствующих о возможных ошибках в ПВМ, результаты анализа визуализируются в интерфейсе.

В полной функциональности система будет обладать возможностями обнаружения нескольких компонент связности, невозможности вычислить какую-либо переменную из других, множественности решения (V, W) -задачи, а также будет предоставлять специалисту в предметной области возможность формализации ограничений на элементы ПВМ. На данный момент значительная часть из перечисленных функций уже реализована. Отдельной веткой исследования является разработка алгоритмов анализа ПВМ, эффективно (в смысле использования доступных ресурсов) реализуемых на параллельных вычислительных машинах, что актуально для обработки больших графов ПВМ.

Предоставляемые системой функции должны значительно сократить время специалистов, затрачиваемое на отладку ПВМ и (V, W) -планов.

Список литературы

1. Вальковский В. А., Малышкин В. Э. Синтез параллельных программ и систем на вычислительных моделях. Новосибирск: Наука, 1988. 129 с.
2. Малышкин В. Э., Корнеев В. Д. Параллельное программирование мультимикрокомпьютеров: учебник. Новосибирск: Изд-во НГТУ, 2006. 296 с.
3. Тыугу Э. Х. Концептуальное программирование. М.: Наука, 1984. 256 с.
4. Gorodnichev M., Lebedev, D. Semantic tools for development of high-level interactive applications for supercomputers // J. Supercomput. 2021. P. 11866–11880. DOI: <https://doi.org/10.1007/s11227-021-03731-6>.
5. Malyshkin V., Akhmed-Zaki D., Perepelkin V. Parallel programs execution optimization using behavior control in LuNA system // J. Supercomput. 2021. P. 9771–9779. DOI: [10.1007/s11227-021-03654-2](https://doi.org/10.1007/s11227-021-03654-2).
6. Kotkas V., Ojamaa A., Grigorenko P., Maigre R., Harf M., Tyugu E. CoCoViLa as a multifunctional simulation platform // SIMUTOOLS 2011 — 4th International ICST Conference on Simulation Tools and Techniques: March 21–25. Barcelona, Spain: Brussels: ICST, 2011. P. 1–8.
7. Феоктистов А. Г., Костромин Р. О., Воскобойников М. Л., Ли-Дэ Д. И. Организация вычислительной среды разработки и применения научных рабочих процессов на основе контейнеризации // Вычислительные технологии. 2023. Т. 28. № 6. С. 151–164.
8. Курбатов М. А. Поиск ошибок во фрагментированных программах с помощью абстрактного синтаксического дерева // Материалы 62-й Международной научной студенческой конференции. Новосибирск, 2024. С. 158–159.
9. Malyshkin V., Vlasenko A., Michurov M. Automated Debugging of Fragmented Programs in LuNA System // D. Balandin et al. (Eds.): 22nd International Conference, MMST 2022, Nizhny Novgorod, Russia, November 14–17, 2022. CCIS 1750. P. 266–280. DOI: [10.1007/978-3-031-24145-1_22](https://doi.org/10.1007/978-3-031-24145-1_22).
10. Власенко А. Ю., Мичуров М. А., Царев В. Д., Курбатов М. А. Построение комплекса автоматизированной отладки фрагментированных программ // Вестник НГУ. Серия: Информационные технологии. 2024. Т. 22. № 1. С. 5–20. DOI: [10.25205/1818-7900-2024-22-1-5-20](https://doi.org/10.25205/1818-7900-2024-22-1-5-20).
11. Gorodnichev M. A., Nalepova E. D., Merkulova E. A., Rudych P. D., Savostyanov A. N. Automation of EEG Data Processing with HPC Community Cloud // IEEE 24th International Conference of Young Professionals in Electron Devices and Materials (EDM), Novosibirsk, Russian Federation, 2023. P. 1320–1323. DOI: [10.1109/EDM58354.2023.10225226](https://doi.org/10.1109/EDM58354.2023.10225226).
12. Fielding R. T. REST: Architectural Styles and the Design of Network-Based Software Architectures. Doctoral Dissertation, University of California, 2000.
13. Ахо А. В., Хопкрофт Дж. Э., Ульман Дж. Д. Структуры данных и алгоритмы. М.: Вильямс, 2003. С. 189–197.
14. Кнут Д. Э. Искусство программирования. Т. 1. 3-е изд. М.: Вильямс, 2006. С. 174–175.



Власенко Андрей Юрьевич — канд. техн. наук, научный сотрудник лаборатории синтеза параллельных программ ИВМиМГ СО РАН, доцент кафедры параллельных вычислений НГУ, доцент ка-

федры параллельных вычислительных технологий НГТУ. Научные интересы: системы поддержки параллельных вычислений, автоматизированная отладка и оптимизация параллельных программ. Автор более 30 печатных работ. E-mail: a.vlasenko@g.nsu.ru.

Andrey Vlasenko — candidate of Engineering Sciences, associate research at the Parallel Program Synthesis Laboratory of ICMMG SB RAS, associate professor of the Parallel Computing Department of Novosibirsk State University, associate professor of the Parallel Computational Technologies Department of Novosibirsk State Technical University. Scientific interests: parallel computing support systems, automated debugging and optimization of parallel programs. Author of more than 30 publications. E-mail: a.vlasenko@g.nsu.ru.

Городничев Максим Александрович — научный сотрудник Института вычислительной математики и математической геофизики СО РАН. Старший преподаватель в Новосибирском государственном университете и

Новосибирском государственном техническом университете. Научные интересы: автоматизация решения задач на суперкомпьютерах, интеллектуальные системы, разработка научного программного обеспечения. E-mail: maxim@ssd.sccc.ru.



Maksim Gorodnichev — researcher at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS. Senior Lecturer at Novosibirsk State University and Novosibirsk State Technical University. Research interests: automation of solving problems on supercomputers, intelligent systems, scientific software development. E-mail: maxim@ssd.sccc.ru.



Курбатов Максим Андреевич — магистрант 2 года обучения факультета информационных технологий НГУ. Научные интересы: параллельное программирование, системы поддержки параллельных вычислений. E-mail: m.kurbatov@g.nsu.ru.

Maksim Kurbatov — 2nd year undergraduate student of the Information Technology Faculty of NSU. Research interests: parallel programming, parallel computing support systems. E-mail: m.kurbatov@g.nsu.ru.

Дата поступления — 18.02.2026

OPTIMIZATION OF THE k -SHORTEST PATHS ALGORITHM

A. R. Gerb, E. E. Deviatykh, G. A. Omarova

Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia
Novosibirsk State University
630090, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2026-1-40-49
EDN: RGXRWO

This paper examines algorithms for finding k -shortest paths. An optimized algorithm for finding k -shortest simple (loop-free) paths in a directed graph is implemented. The algorithm is based on the ideas of the classical version of Yen's algorithm and the reverse tree construction algorithm. Both implementations have complexity $O(kn(m + n \log n))$.

Key words: graph, path, simple path, tree, graph algorithm, k shortest simple paths, space-time trade-off.

References

1. Egli W., Kraus M. Shortest paths in chemical kinetic applications // Phys. Chem. Chem. Phys. 2003. Vol. 5. P. 3916–3920.
2. Yen Jin Y., Finding the K shortest loopless paths in a network // Manag. Sci. 1971. Vol. 17. P. 712–716.
3. Hoffman R., Pavley R. A method for the solution of the n th best path problem // J. of the Association for Computing Machinery. 1959. Vol. 6, N 4. P. 506–515.
4. Bellman R., Kalaba R. On k th best policies // J. of SIAM. 1960. Vol. 8, N 4. P. 582–588.
6. Bock F., Kanther H., Haynes J. An algorithm (the rh best path algorithm) for finding and ranking paths through a network // Research Report, Armour Research Foundation. Chicago, Illinois, 1957.
7. Clarke S., Krikorian A., Rausan J. Computing the N best loopless paths in a network // J. of SIAM. 1963. Vol. 11, N 4. P. 1096–1102.
9. Dijkstra E. W. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1. P. 269–271.
8. Pollack M. The k th best route through a network // Operations Research. 1961. Vol. 9, N 4. P. 578.
5. Sakarovitch M. The k shortest routes and the k shortest chains in a graph // Operations Research Center Report N 66–32. University of California, Berkeley. 1966.
10. Coudert D., Al Zoobi A., Nisse N. Space and time trade-off for the k shortest simple paths problem space and time trade-off for the k shortest simple paths problem // 18th International Symposium on Experimental Algorithms (SEA 2020). Art. N 18. P. 18:1–18:13.

This work was carried out under state contract with ICMMG SB RAS FWNM-2025-0001.

-
11. Kurz D., Mutzel P. A sidetrack-based algorithm for finding the k shortest simple paths in a directed graph // In Int. Symp. on Algorithms and Computation (ISAAC), Schloss Dagstuhl, 2016. Vol. 64 of LIPIcs. P. 49:1–49:13.
 12. Frigioni D., Marchetti-Spaccamela A., Nanni U. Fully dynamic algorithms for maintaining shortest paths trees // J. of Algorithms. 2000. Vol. 34, N 2. P. 251–281.
 13. [Electrol. Res.]: <https://www.diag.uniroma1.it/challenge9/download.html>.
 14. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-AstroPh.html>.
 15. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-CondMat.html>.
 16. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-GrQc.html>.
 17. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-HepTh.html>.

ОПТИМИЗАЦИИ АЛГОРИТМА ПОИСКА k КРАТЧАЙШИХ ПУТЕЙ

А. Р. Герб, Е. Е. Девярых, Г. А. Омарова

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия
Новосибирский государственный университет
630090, Новосибирск, Россия

УДК 519.17+51-7

DOI: 10.24412/2073-0667-2026-1-40-49

EDN: RGXRWO

В работе исследуются алгоритмы поиска k кратчайших путей. Реализован оптимизированный алгоритм для нахождения k кратчайших простых (без петель) путей в ориентированном графе. Алгоритм основан на идеях классической версии алгоритма Йена и алгоритма построения обратного дерева, обе реализации имеют сложность $O(kn(m + n \log n))$.

Ключевые слова: граф, путь, простой путь, дерево, обход графа.

Введение. Практические задачи вычисления кратчайшего пути слишком многочисленны. Они включают ситуации, в которых фактический путь является желаемым выходом, планирование движения робота, проектирование автомагистралей и линий электропередач, а также маршрутизации сетевых соединений. К ним также относятся задачи планирования, вычисление критического пути в диаграммах. Задачи оптимизации, решаемые методом динамического программирования или более сложными методами поиска матриц, выравнивание последовательностей в молекулярной биологии, построение оптимальных вписанных многоугольников могут быть выражены как задачи поиска кратчайшего пути.

Задача о k кратчайших путях является естественным и давно изученным обобщением задачи о кратчайших путях, в которой ищется не один, а несколько путей в порядке возрастания длины. Для заданного ориентированного графа G с неотрицательными весами ребер, положительного целого числа k и двух вершин s и t требуется найти k кратчайших путей из s в t в порядке возрастания длины. Методы поиска k кратчайших путей используются во многих задачах. Иногда возникает необходимость найти путь, удовлетворяющий определенным ограничениям, помимо небольшой длины, однако эти ограничения могут быть плохо определены или их трудно оптимизировать. Типичное решение — вычислить несколько коротких путей, и затем выбрать один из них с учетом других критериев. Пути могут использоваться для моделирования задач, имеющих известные решения, независимо от формулировки пути. Например, в модели k кратчайших путей автоматического перевода между естественными языками правильный перевод может быть найден экспертом-человеком. Перечисляя пути до появления этого известного решения, можно определить,

Исследования выполнены в рамках государственного задания ИВМиМГ СО РАН FWNM-2025-0001.

насколько хорошо модель соответствует задаче с точки зрения количества неверных путей, обнаруженных до правильного. Информацию можно использовать для настройки модели, а также для определения количества путей, которые необходимо сгенерировать при применении дополнительных ограничений для поиска правильного решения.

Вычисляя более одного кратчайшего пути, можно определить, насколько чувствительно оптимальное решение к изменению параметров задачи. Полезно изучить не только оптимальное решение задачи, но и более широкий класс решений, чтобы лучше понять проблему. Например, состояния сложной системы можно представить как конечный автомат, по сути просто граф с различными вероятностями на каждом ребре перехода состояний. В такой модели, вероятно, потребуется знать не только цепочку событий, которая с наибольшей вероятностью приведет к состоянию отказа, но и все цепочки, вероятность отказа которых превышает некоторый порог. Логарифмирование вероятностей перехода превращает эту задачу в задачу поиска всех путей короче заданной длины.

Применение концепции кратчайшего пути и алгоритма поиска k кратчайших путей в химическом анализе было предложено Эгли в 2003 году [1]. В работе был построен оргграф, узлы которого включали не только вещества, но и реакции. Хотя работа не была направлена на создание метода генерации скелетных механизмов, она предложила новую и полезную идею выделения основных путей реакции.

В данной работе представлена реализация алгоритма ранжирования беспетлевых путей Йена с оптимизацией. Обычная реализация прямо следует классической версии алгоритма Йена [2], оптимизация использует подход построения обратного дерева, поскольку узлы каждого k -го кратчайшего пути без петель анализируются в обратном порядке. Хотя обе реализации имеют сложность $O(kn(m + n \log n))$, в худшем случае анализ новой реализации показал лучшую производительность в вычислительных экспериментах, проведенных для оценки их поведения в среднем случае.

Решение задачи поиска k кратчайших путей важно для ряда задач оптимизации и часто используется для ранжирования путей по неубывающему порядку их стоимости, пока не будут найдены один или несколько путей с небольшой стоимостью, удовлетворяющих некоторым ограничениям.

1. Алгоритм поиска k кратчайших путей. Одно из первых решений задачи поиска k кратчайших путей, или ранжирования k кратчайших путей между парой узлов в сети, было предложено Хоффманом и Павли в 1959 г. [3]. Авторы описали алгоритм, в котором для заданного целого числа $k \geq 1$ последовательно определяются 1-й кратчайший путь, 2-й кратчайший путь и так до k -го кратчайшего пути между заданной парой узлов. Существуют два типа задач поиска k кратчайших путей. Первый тип – это нахождение путей от источника s до стока t , имеющих наименьшую длину, в которых петли разрешены. Алгоритмы для решения этого типа задач предложены в работах [3, 4, 5]. Второй тип задач – поиск k путей от s до t , имеющих наименьшую длину, в которых петли не допускаются [6, 7, 8, 5].

В работе [6] авторы вводят перечисление всех возможных путей из k кратчайших путей в сети. Процедура перечисляет все пути от s до t , а затем выбирает из них k путей с наименьшей длиной. Данный алгоритм имеет два основных недостатка: во-первых, требуется очень большое количество вычислений и памяти; во-вторых, для решения задачи с малым k требуется столько же усилий, сколько и для решения задачи с большим k , и чем больше k , тем затруднительнее использование этого алгоритма.

В работе [8] автор для поиска k -го кратчайшего пути сначала получает $k - 1$ кратчайший путь. Затем расстояние каждой дуги в каждом из 1-го, 2-го, ..., $(k - 1)$ -го кратчайших путей поочередно устремляется к бесконечности. Далее задача поиска кратчайшего пути решается для каждого такого случая. Лучший из этих полученных кратчайших путей является искомым k -м кратчайшим путем. Количество вычислений, требуемых для этой процедуры, экспоненциально возрастает с ростом значения k , следовательно, если k не мало, алгоритм Поллака [8] перегружает вычислительные ресурсы.

В работе [7] авторы предлагают процедуру ветвей и границ для поиска k кратчайших путей в сети: сначала находится кратчайший путь, а затем — k кратчайших путей среди всех путей, ответвляющихся от кратчайшего. Эффективность этого алгоритма зависит от конкретной сети. Если второй кратчайший путь «немедленно ответвляется» от первого, третий кратчайший путь «немедленно ответвляется» от второго кратчайшего пути и т. д., эта процедура может очень быстро определить все k кратчайших путей. Однако в общем случае этот алгоритм, вероятно, потребует огромного количества вычислений и памяти. Поэтому эффективность этой процедуры сложно оценить.

В работе [5] автор описывает следующий алгоритм поиска k кратчайших путей: сначала с помощью процедуры, аналогичной (но менее эффективной), чем алгоритм Хоффмана и Павли, находятся $H > k$ кратчайших путей, которые могут содержать петли [3]. Затем H путей исследуются на k кратчайших путей, не содержащих петель. Эффективность этого алгоритма зависит от конкретной сети. Если H кратчайших путей, полученных алгоритмом Хоффмана, не содержат петель, алгоритм Сакаровича [5] может очень быстро определить k кратчайших путей без петель. Однако для этого алгоритма сложно указать верхнюю вычислительную границу.

1.1. Определения. Пусть $G = (V, E)$ — ориентированный граф с множеством вершин V и множеством ребер E ; $n = |V|$ — количество вершин; $m = |E|$ — количество ребер в графе G .

Подграфом в графе $G(V, E)$ называется граф $H(V_H, E_H)$, такой что $\forall v \in V_H, v \in V$ и $\forall e \in E_H, e \in E$.

Пусть $e \in E$, такое что $e = (u, v)$, тогда обозначим $head(e) = v$ — начало дуги e и $tail(e) = u$ — конец дуги e .

Обозначим $w_G : E \rightarrow R^+$ функцию веса для ребер графа G .

Пусть $u, v \in V$, тогда путем из вершины u в вершину v в графе G называется последовательность вершин $P = P_{uv}^G = (v_0 = u, v_1, \dots, v_r = v)$, такая что $(v_i, v_{i+1}) \in E$ для $\forall i : 0 \leq i < r$. Путь P будем обозначать как $u \rightarrow v$.

Путь называется простым тогда и только тогда, когда $v_i \neq v_j$ для $\forall i, j : 0 \leq i < j \leq r$.

Для возможности вычисления длины пути расширим функцию w_G на множество всех путей следующим образом: $w_G(P) = \sum_{i=0}^{r-1} w_G(v_i v_{i+1})$.

Обозначим путь $v_1 \rightarrow w_p$ как $(v_1, \dots, v_r, Q)$ — путь, полученный сложением двух путей $(v_1, \dots, v_r)$ и $Q = (w_1, \dots, w_p)$, если $(v_r, w_1) \in E$.

Пусть существует путь $u \rightarrow v$, тогда длину кратчайшего пути $u \rightarrow v$ в графе G обозначим $d_G(u, v)$.

Пусть $t \in V$, тогда обратным деревом кратчайших путей T с корнем в t назовем любой подграф G , для которого выполняются условия: 1) $\forall u \in T \setminus \{t\}$ имеет максимум одно исходящее ребро, а t вообще не имеет таких ребер; 2) $\forall u \in T$ длина пути P_{ut}^T равна $d_G(u, t)$, то есть P_{ut}^T является кратчайшим путем.

Пусть $s, t \in V$, тогда S — множество k попарно неравных кратчайших путей, такое что $w_G(P_1) \leq w_G(P_2)$, где $P_1 \in S$ и $P_2 \notin S$.

1.2. *Описание алгоритма Йена поиска k кратчайших путей.* Задача поиска k кратчайших путей заключается в нахождении k путей, имеющих наименьшую длину от начальной до конечной вершины взвешенного графа, при этом каждый путь является простым, то есть не посещает одну и ту же вершину более одного раза [2]. Алгоритм Йена является классическим методом решения задачи в ориентированных графах с неотрицательными весами дуг.

Алгоритм строится на основе индукции по мощности множества кратчайших путей и работает до тех пор, пока не будет найдено заданное количество кратчайших путей.

База индукции $k' = 1$. Вычисляется кратчайший путь $P_0 = (v_0 = s, v_1, \dots, v_r = t)$ от вершины s до вершины t с помощью алгоритма Дейкстры [9] (P_0 существует, иначе решений нет). Второй кратчайший путь является простым кратчайшим ответвлением P_0 .

Алгоритм Йена вычисляет ответвление для каждой вершины v_i пути P_0 следующим образом:

1) Для $\forall i, i = \overline{0, r-1}$ создается подграф $G_i(P_0)$ графа G путем удаления вершин $v_0, \dots, v_{i-1}$, чтобы избежать ответвления, не являющегося простым, и удалением ребра (v_i, v_{i+1}) , чтобы получить путь, отличающийся от P_0 .

2) Для $\forall i, i = \overline{0, r-1}$ вычисляется кратчайший путь Q_i от v_i до t в подграфе $G_i(P_0)$ с использованием алгоритма Дейкстры. Если не существует пути, то данное i не рассматривается на шаге 3.

3) Для $\forall i, i = \overline{0, r-1}$ простое ответвление $(v_0, \dots, v_i, Q_i)$ пути P_0 добавляется в множество кандидатов C . Индекс ответвления i сохраняется явно в паре с ответвлением $(v_0, \dots, v_i, Q_i)$ для последующего использования.

Вторым искомым путем будет самый короткий путь из множества кандидатов C .

Индукционное предположение для $0 < k' < k$. Предположим, что множество k' кратчайших путей S вычислено, множество кандидатов C содержит простые пути $s \rightarrow t$, а также существует кратчайший путь $Q \in C$, такой что $S \cup Q$ является множеством $k' + 1$ кратчайших путей.

Шаг индукции $k' \rightarrow k' + 1$. Пусть $Q = (v_0 = s, \dots, v_r = t)$ и $j (0 \leq j < r)$ — индекс ответвления. Q удаляется из множества кандидатов C и добавляется в результирующее множество S . Затем вычисляются все простые ответвления и добавляются в множество кандидатов C :

1) Для $\forall j, j \leq i < r$ создается подграф $G_i(Q)$ графа G путем удаления вершин $v_0, \dots, v_{i-1}$ и всех дуг вида (v_i, w) , таких что множество кандидатов C уже содержит путь с префиксом $(v_0, \dots, v_i, w)$.

2) Для $\forall i, j \leq i < r$ вычисляется кратчайший путь Q_i от v_i до t в $G_i(Q)$. Если не существует пути, то данное i не рассматривается на шаге 3.

3) Для $\forall j, j \leq i < r$ ответвление $(v_0, \dots, v_i, Q_i)$ пути Q добавляется в множество кандидатов C с его индексом ответвления i .

Следующим $(k' + 2)$ кратчайшим путем будет кратчайший путь из множества кандидатов.

1.3. *Оптимизации алгоритма поиска.* Оптимизация алгоритма проведена на основе алгоритма, предложенного в работе Али Аль Зуби [10], улучшающего метод поиска k кратчайших путей Даниэля Курца и Петра Муцеля [11]. Улучшение направлено на уменьшение потребления оперативной памяти.

Основная идея алгоритма сводится к построению обратных деревьев кратчайших путей. Пусть C_{simple} – приоритетная очередь, построенная на основе двоичной кучи. Данная очередь хранит первые k простых путей кандидатов и их длину в виде $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}), lb)$, где T_i – обратное дерево кратчайших путей; $e_i \in E$ – дуга; lb – длина пути в графе G . Дерево T_{h+1} может храниться в множестве T_{cache} , в противном случае оно вычисляется с помощью алгоритма Дейкстры. При этом $\forall i, i = \overline{0, h}$ T_i гарантированно вычислено и хранится в T_{cache} . Чтобы извлечь из данного представления путь, необходимо пройти от вершины s до $tail(e_0)$ в обратном дереве кратчайших путей T_0 , проследовать по дуге e_0 в T_1 до $tail(e_1)$ и так далее до тех пор, пока не будет достигнута вершина t дерева T_{h+1} .

Вторая очередь $C_{not-simple}$ хранит группы ответвлений-кандидатов с их нижней оценкой длин lb . Данные группы кандидатов требуют явного вызова алгоритма Дейкстры для решения проблемы их простоты.

В начале работы алгоритма вычисляется обратное дерево кратчайших путей T_0 , содержащее кратчайший путь $s \rightarrow t$, если такого пути нет, то нет и решения задачи. Дерево T_0 добавляется в C_{simple} . Цикл выполняется до тех пор, пока не будет найдено k кратчайших путей или не исчерпаны очереди C_{simple} и $C_{not-simple}$. В последнем случае результирующее множество S будет содержать меньше k элементов.

В каждой итерации цикла алгоритм сначала выбирает лучший элемент φ из двух очередей (с приоритетом для C_{simple} , если оценки lb для лучших элементов обеих очередей совпадают). Если φ был извлечен из C_{simple} , то φ имеет вид $((T_0, e_0, T_1, \dots, T_h, e_h = (u_h, v_h), T_{h+1}), lb)$, тогда алгоритм действует следующим образом. Если T_{h+1} не сохранен в T_{cache} , то он вычисляется и сохраняется. Затем извлекается уже описанным способом простой путь P из φ и добавляется в результирующее множество S . Далее алгоритм обрабатывает все ответвления.

1) Для дуги $e = (u, v) \in E, u \in P_{vht}^{T_{h+1}}, v \notin \{s, \dots, v_h, \dots, u\}$ проверяем, является ли простым путь P_e , следующий сначала вдоль P от вершины s до v_h , затем вдоль $P_{vhu}^{T_{h+1}}$, вдоль ребра e и, наконец, вдоль $P_{vt}^{T_{h+1}}$.

2) Если P_e является простым, то добавляем в C_{simple} элемент вида $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, e, T_{h+1}), lb_e)$, где lb_e – длина пути P_e .

3) Если ответвление e не является простым, то оно добавляется в множество $X = \{f_1, \dots, f_r\}$, такое что для всех $i, j, l : 1 \leq i < j < l \leq r$ $tail(f_j)$ находится между (или равен) $tail(f_i)$ и $tail(f_l)$ на пути $P_{vht}^{T_{h+1}}$.

4) Если множество $X \neq \emptyset$, то необходимо добавить X как группу кандидатов в $C_{not-simple}$.

Пусть $f_i = (u_i, v_i)$, тогда обозначим $lb_i = lb_{f_i} = w_G(P_1) + w_G(P_{vhu_i}^{T_{h+1}}) + w_G(f_i) + w_G(P_{vit}^{T_{h+1}})$, в этом случае $lb = \min_{1 \leq i \leq r} lb_i$, где P_1 путь от s до v_h вдоль P .

В случае, если элемент φ извлекается из очереди $C_{not-simple}$, то φ имеет вид $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, X, T_{h+1}), lb)$, $e_h = (u_h, v'_h)$ и $P_1 = (s = x_0, \dots, x_h = v'_h)$ – префикс пути, представленного элементом φ , $P_{v'_ht}^{T_{h+1}} = (v'_h, v'_{h+1}, \dots, v'_p = t)$, и для $\forall j$ $f_j = (v'_{i_j}, v_j)$. $\exists i^* : 1 \leq i^* \leq r$, такой что $lb = lb_{i^*}$, алгоритм обрабатывает одновременно несколько ответвлений $f_r, f_{r-1}, \dots, f_{i^*}$.

1) Строится обратное дерево кратчайших путей T'_r в графе $G_r = G - \{x_0, \dots, x_h = v'_h, \dots, v'_{i_r}\}$ с использованием алгоритма Дейкстры. В очередь C_{simple} добавляется элемент

вида $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, f_r, T'_r), w_G(Q_r))$, где $Q_r = P_1 P_{v'_h v'_{ir}}^{T_{h+1}} P_{v_r t}^{T'_r}$ — простой путь от s до t . Дерево T'_r не сохраняется в T_{cache} , но сохраняется «указатель» на него.

2) Для каждого j от $r - 1$ до i^* (по убыванию) строится дерево T'_j в графе $G_r = G - \{x_0, \dots, x_h = v'_h, \dots, v'_{ir}\}$ на основе T'_{j+1} с использованием алгоритма обновления обратного дерева кратчайших путей [12]. В очередь C_{simple} добавляется элемент вида $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, f_j, T'_j), w_G(Q_j))$, где $Q_j = P_1 P_{v'_h v'_{ij}}^{T_{h+1}} P_{v_j t}^{T'_j}$.

Оставшаяся же часть ответвлений $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, X', T_{h+1}), \min_{1 \leq j < i^*} lb_j)$, где $X' = \{f_1, \dots, f_{i^*-1}\}$ возвращается в $C_{not-simple}$.

Сложность алгоритма в худшем случае аналогична сложности базовой версии — алгоритма Йена, то есть $O(kn(m + n \log n))$.

2. Численные эксперименты.

2.1. *Описание графов.* Для тестирования алгоритмов использованы графы, полученные из различных источников [13–15].

— **NW** (USA-road-d/USA-road-t). Дорожная сеть Северо-Запада США. Граф ориентированный, дуги соответствуют направлениям движения, веса — расстояниям. Размеры: 1 207 945 вершин и 2 840 208 дуг [13].

— **FLA** (USA-road-d/USA-road-t). Дорожная сеть штата Флорида. Граф аналогичен указанному выше. Размеры: 1 070 376 вершин и 2 712 798 дуг [13].

— **Rome99**. Большой фрагмент городской дорожной сети Рима (Италия). Размеры: 3 353 вершин и 8 870 дуг [13].

— **14** (SNAP). Сеть соавторств в категории *Astro Physics* на arXiv: неориентированный граф, ребро между авторами означает совместную публикацию. Размеры: 18 772 вершин и 198 110 ребер [14].

— **15** (SNAP). Сеть соавторств в категории *Condensed Matter* (arXiv), неориентированный граф. Размеры: 23 133 вершин и 93 497 ребер [15].

— **16** (SNAP). Сеть соавторств в категории *General Relativity and Quantum Cosmology* (arXiv), неориентированный граф. Размеры: 5 242 вершин и 14 496 ребер [16].

— **17** (SNAP). Сеть соавторств в категории *High Energy Physics – Theory* (arXiv), неориентированный граф. Размеры: 9 877 вершин и 25 998 ребер [17].

2.2. *Результаты вычислений.* Результаты работы алгоритма были протестированы на графах, описанных выше.

В табл. 1, 2 приведены время выполнения и объем памяти, использованной при выполнении алгоритма поиска k кратчайших путей для 1000 произвольных пар вершин при различных значениях k .

В табл. 1 сделан акцент на ускорение по времени, при этом увеличивается объем использованной памяти. В табл. 2 реализован оптимальный алгоритм, в котором была поставлена цель уменьшить затраты памяти, при этом увеличилось время исполнения. Для графов sa-AstroPh и sa-CondMat в табл. 1 получена и оптимизация по памяти.

Реализованный оптимизированный алгоритм поиска k кратчайших путей показал хороший результат. В случае, когда графы небольшие, время с увеличением k растет незначительно. Для первых двух графов, в которых количество вершин больше миллиона, с увеличением k растет и время.

Заключение. Алгоритм использовался для реализации поиска глобальных путей по графам элементов в задачах химической кинетики. Глобальным путем в заданном графе потока элемента является путь от вещества в исходной смеси до конечного продукта,

Таблица 1

Оптимизация по времени

Граф	k=2		k=3		k=4		k=5		k=10	
	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб
NW	148	45227	208	48119	234	49409	314	53414	556	65846
FLA	127	41065	172	43385	227	45095	304	48684	595	60721
Rome99	0,01	1813	0,03	1836	0,1	1857	0,19	1888	0,77	2019
ca-AstroPh	3,6	2683	3,3	2706	3,6	2726	3,5	2750	3,9	2859
ca-CondMat	2,6	2792	2,6	2792	4	2746	4,6	2759	3,6	2799
ca-GrQc	0,002	1821	0,002	1827	0,001	1831	0,003	1810	0,015	1817
ca-НepTh	0,71	2091	0,67	2098	0,69	2113	0,67	2114	0,7	2127

Таблица 2

Оптимизация по памяти

Граф	k=2		k=3		k=4		k=5		k=10	
	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб
NW	253	32690	365	34536	450	35336	591	37058	1225	46079
FLA	281	30406	374	32204	454	33312	693	35842	1148	44361
Rome99	0,08	1596	0,2	1598	0,39	1607	0,64	1615	1,47	1705
ca-AstroPh	8,7	2905	8,4	2897	8,4	2899	8,6	2908	9,2	2932
ca-CondMat	8,7	3004	8,8	3024	8,7	3015	8,9	3020	8,6	3024
ca-GrQc	0,7	1750	0,7	1751	0,7	1748	0,7	1746	0,7	1741
ca-НepTh	2,3	1991	2,4	1994	2,5	2001	2,4	2005	2,4	2009

проходящий через вещество-концентратор. Глобальный путь позволяет идентифицировать последовательные цепочки превращения соединений от начальных реагентов до конечных продуктов всего реакционного процесса. Учитывая основные глобальные пути, через которые проходят наибольшие потоки элемента, можно идентифицировать наиболее важные цепочки превращения, которые должны быть сохранены в скелетном механизме.

Для каждого вещества-концентратора существует множество глобальных путей, поэтому для идентификации основных глобальных путей используется алгоритм нахождения k кратчайших независимых путей в ориентированном взвешенном графе. Глобальный путь вычисляется как комбинация пути от вещества из исходной смеси до вещества-концентратора с путем от вещества-концентратора до одного из продуктов. На выходе получается множество мощностью k^2 глобальных путей. Все такие пути для каждого вещества-концентратора и каждой пары заданных соединений в каждом графе потока элемента сохраняются и образуют множество веществ, входящих в итоговый скелетный механизм.

В работе рассмотрена реализация алгоритма для дорожных сетей с большим количеством путей, и проведена оптимизация, которая позволила сократить ресурсы по времени и памяти.

В дальнейшем алгоритм будет использоваться в качестве одного из методов исследования и построения фактор-деревьев и древовидных разбиений графа для ускорения времени решения систем линейных уравнений.

Список литературы

1. Egli W., Kraus M. Shortest paths in chemical kinetic applications // Phys. Chem. Chem. Phys. 2003. Vol. 5. P. 3916–3920.
2. Yen Jin Y., Finding the K shortest loopless paths in a network // Manag. Sci. 1971. Vol. 17. P. 712–716.
3. Hoffman R., Pavley R. A method for the solution of the nth best path problem // J. of the Association for Computing Machinery. 1959. Vol. 6, № 4. P. 506–515.
4. Bellman R., Kalaba R. On kth best policies // J. of SIAM. 1960. Vol. 8, № 4. P. 582–588.
6. Bock F., Kanther H., Haynes J. An algorithm (the rh best path algorithm) for finding and ranking paths through a network // Research Report, Armour Research Foundation. Chicago, Illinois, 1957.
7. Clarke S., Krikorian A., Rausan J. Computing the N best loopless paths in a network // J. of SIAM. 1963. Vol. 11, № 4. P. 1096–1102.
9. Dijkstra E. W. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1. P. 269–271.
8. Pollack M. The kth best route through a network // Operations Research. 1961. Vol. 9, № 4. P. 578.
5. Sakarovitch M. The k shortest routes and the k shortest chains in a graph // Operations Research Center Report № 66–32. University of California, Berkeley. 1966.
10. Coudert D., Al Zoobi A., Nisse N. Space and time trade-off for the k shortest simple paths problem space and time trade-off for the k shortest simple paths problem // 18th International Symposium on Experimental Algorithms (SEA 2020). Art. № 18. P. 18:1–18:13.
11. Kurz D., Mutzel P. A sidetrack-based algorithm for finding the k shortest simple paths in a directed graph // In Int. Symp. on Algorithms and Computation (ISAAC), Schloss Dagstuhl, 2016. Vol. 64 of LIPIcs. P. 49:1–49:13.
12. Frigioni D., Marchetti-Spaccamela A., Nanni U. Fully dynamic algorithms for maintaining shortest paths trees // J. of Algorithms. 2000. Vol. 34, № 2. P. 251–281.
13. [Электрон. рес.]: <https://www.diag.uniroma1.it/challenge9/download.html>.
14. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-AstroPh.html>.
15. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-CondMat.html>.
16. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-GrQc.html>.
17. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-HepTh.html>.



Герб Артем Родионович — аспирант Института вычислительной математики и математической геофизики СО РАН. В 2023 году окончил механико-математический факультет НГУ, степень магистра. E-mail: gerb-artem@mail.ru.

ru.

Gerb Artem is a postgraduate student of the Institute of Computational Mathematics and Mathematical Geophysics SB RAS. Graduated from Novosibirsk State University's Math Faculty in 2023.

Девятых Егор Евгеньевич — студент 1 курса магистратуры механико-математического факультета Новосибирско-

го государственного университета. E-mail: klasifate@gmail.com.



Devyatykh Egor is a 1st year master's student of the Faculty of Mechanics and Mathematics of Novosibirsk State University.

Омарова Гульзира Алимовна — канд. физ.-мат. наук, науч. сотр. Института вычислительной математики и математической геофизики СО РАН; e-mail: gulzira@rav.sssc.ru.



Omarova Gulzira is PhD in Physics and Mathematics, Research Fellow at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS.

Дата поступления — 15.12.2025

A HYBRID COMBINATIONAL APPROACH TO SENSITIVITY QUANTIFICATION IN MEMBERSHIP FUNCTION SELECTION DURING FUZZIFICATION OF SERVER PARAMETERS

S. S. Kenjaev, A. R. Akhatov, M. R. Tojiev

Samarkand State University
703004, Samarkand, Uzbekistan

DOI: 10.24412/2073-0667-2026-1-50-64

EDN: YHQJXQ

In the process of efficiently managing request flows within information systems, accounting for the individual characteristics of servers plays a crucial role. Traditional load balancing algorithms—such as Round Robin, Weighted Round Robin, and regional distribution methods—typically allocate requests in a cyclic manner or based solely on predefined weight coefficients. These approaches fail to consider real-time server load levels, response speed, or operational state. This paper proposes a novel fuzzy logic-based model for request flow management, with particular attention given to the selection of appropriate membership functions for the linguistic representation of server parameters. During the fuzzification phase, the sensitivity of membership functions is quantified and optimized through a combinational hybrid approach. Experimental results demonstrate that the proper selection of membership functions enhances the accuracy of decision-making and reduces the degree of uncertainty. Consequently, the proposed approach significantly improves the flexibility, precision, and stability of request flow management in distributed server environments.

Key words: fuzzification, membership functions, sensitivity quantification, combinational hybrid approach, server load balancing, fuzzy logic, system reliability, adaptive control, linguistic modeling, request flow optimization, normalized sensitivity index, system.

References

1. Zadeh L. A. Fuzzy sets // Information and Control, 1965. N 8(3). P. 338–353. DOI: [https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X).
2. Smith J., Doe A. A fuzzy logic analysis method for evaluating human sensitivities // Journal of Fuzzy Systems, 2018. N 45(3). P. 123–136. DOI: <https://doi.org/10.1016/j.jfuzz.2018.01.005>.
3. Gures E., Shayea I., Ergen M., Azmi M.H., El-Saleh A.A. Machine Learning-Based Load Balancing Algorithms in Future Heterogeneous Networks: A Survey // IEEE Access 2022. N 10. P. 37689–37717.
4. Akhatov A.R., Kenjaev S.S., Tojiev M.R. Improved Round Robin Algorithm Based on Fuzzy Logic and Genetic Algorithm for Server Load Balancing // International Russian Automation Conference (RusAutoCon), Sochi, Russian Federation, 2025. P. 296–301. DOI: 10.1109/RusAutoCon65989.2025.11177403.
5. Raximov N., Kuvandikov J., Khasanov D. The importance of loss function in artificial intelligence // International Conference on Information Science and Communications Technologies (ICISCT 2022), DOI: 10.1109/ICISCT55600.2022.10146883.

6. Asghari M.Z., Ozturk M., Hämmäläinen J. Reinforcement Learning Based Mobility Load Balancing with the Cell Individual Offset // In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 2021. P. 25–28.
7. Axatov A., Nurmamatov M., Nazarov F., Sariyev Sh. Genetic algorithm application technology in multi-parameter optimization problems // AIP Conf. Proc., 2024. V. 3244, art. N 030025, DOI: 10.1063/5.0242074.
8. Forrester Research, Global Digital Experience Benchmark Report, 2023.
9. Garcia M., Wong T. Fuzzy Membership Function Evaluation by Non-Linear Regression // International Journal of Fuzzy Systems. N 23(1). P. 34–45. DOI: <https://doi.org/10.1109/IJFS.2021.000567>.
10. Scholz-Reiter B. et al. Fuzzy-based load balancing in production control // CIRP Annals, 2015.
11. Parmanov B.Z., Nazarov F.M., Tojiyev M.R. Development of an Algorithm for Intelligent Analysis of Data From a Microprocessor-Based Portable Spectrophotometer // in Proceedings International Russian Smart Industry Conference, SmartIndustryCon 2025, Institute of Electrical and Electronics Engineers Inc., 2025, P. 664–669. DOI: 10.1109/SmartIndustryCon65166.2025.10985982.
12. Chen W., Huang J. Sensitivity and robustness analysis of adaptive neuro-fuzzy inference systems // IEEE Transactions on Neural Networks and Learning Systems, 2020. N 31(4). P. 945–957. DOI: <https://doi.org/10.1109/TNNLS.2019.2949824>.
13. Ghatak G., De Domenico A., Coupechoux M. Coverage Analysis and Load Balancing in HetNets With Millimeter Wave Multi-RAT Small Cells // IEEE Trans. Wirel. Commun. 2018. N 17. P. 3154–3169.
14. Maruf T. Hazard recognition system based on violation of the integrity of the field and changes in the intensity of illumination on the video image // International Conference on Information Science and Communications Technologies, ICISCT 2022, Institute of Electrical and Electronics Engineers Inc., 2022. DOI: 10.1109/ICISCT55600.2022.10146933.
15. Axatov A., Nurmamatov M., Nazarov F., Sariyev Sh.. Genetic algorithm application technology in multi-parameter optimization problems // AIP Conf. Proc., 2024. V. 3244, art. N 03002524. DOI: 10.1063/5.0242074.
16. Nixon M. S. et al. Real-time fuzzy logic-based resource allocation // Journal of Real-Time Systems, 2021.
17. Parsa S. Adaptive fuzzy load balancing algorithm for heterogeneous distributed systems // Journal of Supercomputing, 2017.
18. Kenjaev S.S., Rashidov A.E. File storage methods and algorithms for optimal management of various types of data // “Descendants of Al-Farghani” electronic scientific journal of Fergana branch of TATU named after Muhammad al-Khorazmi, 2024. V 1. N 3. DOI: 10.5281/zenodo.13954911.
19. Hatipoğlu A., Başaran M., Yazici, M.A., Durak-Ata L. Handover-based Load Balancing Algorithm for 5G and Beyond Heterogeneous Networks // In Proceedings of the 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Brno, Czech Republic, 2020. P. 5–7.

ГИБРИДНЫЙ КОМБИНАЦИОННЫЙ ПОДХОД К КОЛИЧЕСТВЕННОЙ ОЦЕНКЕ ЧУВСТВИТЕЛЬНОСТИ ПРИ ВЫБОРЕ ФУНКЦИИ ПРИНАДЛЕЖНОСТИ В ПРОЦЕССЕ ФАЗЗИФИКАЦИИ ПАРАМЕТРОВ СЕРВЕРА

С. С. Кенжаев, А. Р. Ахатов, М. Р. Тоджиев
Самаркандский государственный университет,
703004, Самарканд, Узбекистан

DOI: 10.24412/2073-0667-2026-1-50-64

EDN: YHQPJXQ

В процессе эффективного управления потоками запросов в информационных системах учет индивидуальных характеристик серверов играет решающую роль. Традиционные алгоритмы балансировки нагрузки, такие как Round Robin, Weighted Round Robin, и методы регионального распределения обычно распределяют запросы циклически или на основе только предопределенных весовых коэффициентов. Эти подходы не учитывают уровни нагрузки серверов в реальном времени, скорость ответа или рабочее состояние. В данной статье предлагается новая модель управления потоками запросов на основе нечеткой логики, с особым вниманием к выбору соответствующих функций принадлежности для лингвистического представления параметров серверов. На этапе фаззификации чувствительность функций принадлежности количественно оценивается и оптимизируется с помощью комбинированного гибридного подхода. Экспериментальные результаты показывают, что правильный выбор функций принадлежности повышает точность принятия решений и снижает степень неопределенности. Следовательно, предложенный подход значительно повышает гибкость, точность и стабильность управления потоками запросов в распределенных серверных средах.

Ключевые слова: фаззификация, функции принадлежности, количественная оценка чувствительности, комбинированный гибридный подход, балансировка нагрузки серверов, нечеткая логика, надежность системы, адаптивное управление, лингвистическое моделирование, оптимизация потока запросов, нормализованный индекс чувствительности, система.

Introduction. In modern information and communication systems, the rapid increase in the number of users, the growing complexity of computational processes, and the operation of services in real-time mode have led to a significant rise in the volume of requests directed to servers. Under the constraints of limited computational resources, this situation has made it increasingly challenging to ensure continuous system operation, maintain response speed, and preserve the quality of service. Consequently, the issue of achieving effective load balancing has become a critical problem in contemporary distributed computing environments [1–3].

According to statistical data, the volume of global Internet traffic is projected to reach 175 zettabytes per year by 2025 [4]. At the same time, Gartner analytics reports indicate that poorly balanced server infrastructures cause companies to experience an average annual economic loss of approximately 2.5 million USD [5–7].

In most cases, information systems deliver services through multiple servers or computing nodes. In such environments, the problem of uneven resource utilization arises, where

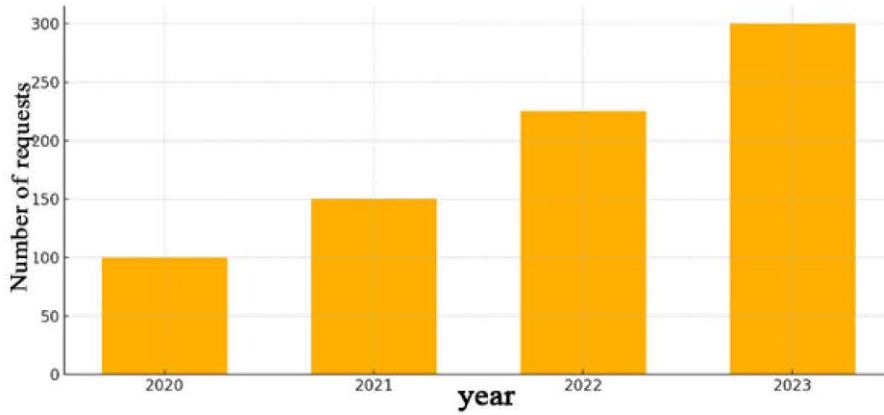


Fig. 1. Growth dynamics of online request volumes during the period 2020–2023

certain servers become overloaded while others remain underutilized. Since servers differ in computational power, memory capacity, and network latency, achieving optimal load distribution becomes a complex combinatorial problem [8]. This challenge becomes even more pronounced in systems based on microservice architectures, where significant variations in request flow direction, intensity, and priority are observed. As a result, dynamic load balancing strategies are required to ensure efficient and stable system performance [9–11].

According to Forrester Research analytics, between 2020 and 2023, the average number of user requests recorded by online stores increased by more than 300 % [12]. This trend underscores the growing necessity of implementing effective load balancing mechanisms that ensure the continuous, stable, and well-balanced operation of not only network infrastructures but also server resources (Fig. 1). Consequently, in the process of optimizing server request flows, researchers have increasingly moved beyond traditional deterministic approaches, adopting artificial intelligence-based methods, particularly fuzzy logic models, which enable the assessment of uncertainty and enhance flexibility in decision-making.

In information systems, the volume of load directed to servers continuously fluctuates over time. This variability is referred to as the dynamic load imbalance problem, which directly affects the overall system performance, including request processing speed, latency level, and the efficiency of resource utilization [13]. One of the key indicators for evaluating system performance is the degree of load distribution among servers. This indicator is typically measured using variance, which reflects the extent to which individual server loads deviate from the average load value.

The variance of load imbalance is expressed by the following formula:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (L_i - \mu)^2, \quad (1)$$

where σ^2 — the variance of load imbalance in the system; L_i — the current load value of the i -th server; μ — the average load across all servers, defined as:

$$\mu = \frac{1}{N} \sum_{i=1}^N L_i, \quad (2)$$

where N — the total number of servers in the network.

Table 1

Priority levels by request types

Request type	Response time requirement	System significance
Emergency medical alert	< 100 ms	Yuqori (hayotiy)
Online payment	< 200 ms	Yuqori (moliyaviy)
Video upload	< 500 ms	O'rta
Statistical report	< 2000 ms	Past

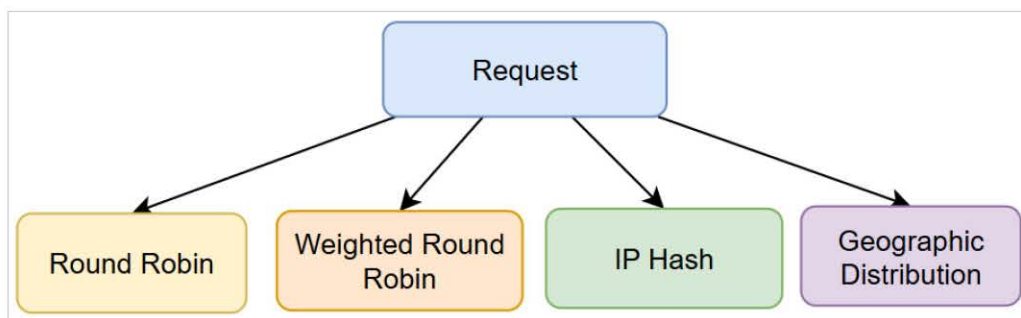


Fig. 2. Static load distribution algorithms

The smaller the variance value, the more balanced the load distribution within the system. Conversely, an increase in variance indicates greater disproportionality in resource utilization and a higher degree of load imbalance.

The requirements for system response time and the semantic significance of user requests vary considerably across different contexts. For instance, real-time processing requests—such as those in medical or financial services—demand high processing speed and minimal latency from the system, whereas informational or analytical requests impose relatively less stringent timing requirements. These distinctions are summarized in Table 1.

The problem of efficiently distributing request flows in information systems is addressed through a variety of algorithmic approaches. These approaches are generally classified into static and dynamic types, depending on their operational principles and responsiveness to system conditions. This classification, on the one hand, defines how an algorithm reacts to changes within the system, and on the other hand, determines the nature of the information it relies on during the decision-making process.

Static request flow distribution algorithms are typically designed based on predefined conditions established before the system operation begins and rely on general information about the servers—such as their computational capacity or number of connections (Fig. 2). These algorithms operate according to fixed rules and do not account for variations in system load; therefore, they yield effective results only under stable load conditions.

One of the most widely used classical approaches to request distribution is the Round Robin algorithm (Fig. 3).

In this method, incoming requests are distributed sequentially and cyclically among the available servers. The RR algorithm was originally developed in the 1960s for multi-user systems and continues to be extensively applied in network load balancers and operating systems today. According to Cisco (2023), more than 75 % of modern load balancing systems are based on RR or its modified versions.

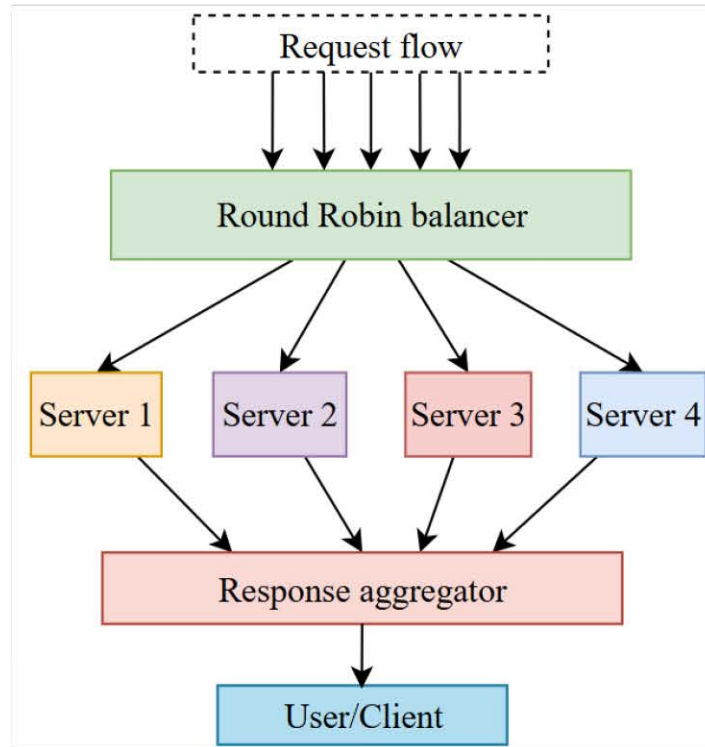


Fig. 3. Mechanism of the Round Robin algorithm

To enhance the efficiency of request flow management and distribution in information systems, it is essential to conduct an in-depth analysis of system parameters, identify their interdependencies, and employ intelligent approaches that account for uncertainty. This paper introduces a fuzzy logic-based mechanism for managing request flows through the linguistic modeling of server parameters. In the proposed model, the principle of sensitivity quantification is applied during the fuzzification stage to select appropriate membership functions corresponding to server characteristics, thereby enabling the optimization of overall system sensitivity. As a result, the system can evaluate its state in an imprecise yet semantically consistent manner, improving the effectiveness of decision-making and request distribution under uncertain conditions.

1. Methodology. Information theory provides a scientific foundation for the use of mathematical modeling in evaluating system reliability, quantitatively analyzing risk levels, and monitoring sensitivity. This approach enables the optimization of system performance by mathematically expressing stability and risk management mechanisms. In particular, the principles of Information Theory proposed by Claude Shannon (1948) and Warren Weaver remains a fundamental framework for analyzing system reliability and efficiency [14]. Their methodology allows for the statistical evaluation of key operational parameters, such as risk coefficients (probabilistic indicators) and the system's response speed to failures (e. g., recovery times), providing a quantitative basis for assessing the robustness and adaptability of complex information systems.

In the process of distributing requests across servers using fuzzy logic based systems, the fuzzification and aggregation of membership functions play a crucial role. For such systems to operate efficiently, it is essential to select appropriate membership functions corresponding

to low, medium, and high states. In fuzzy logic systems, evaluating the sensitivity of input parameters is vital to ensure optimal system performance. During the sensitivity assessment process, even small variations in the parameters of membership functions can significantly affect the system's output results. Thus, addressing this issue remains a key factor in enhancing the effectiveness of fuzzy logic systems.

The sensitivity of membership functions has also been evaluated using the nonlinear regression method [15]. The authors emphasized the high efficiency of this approach in analyzing sensitivity within fuzzy logic systems. The nonlinear regression technique was applied to determine the activation levels of membership functions within the FLS structure and to assess their sensitivity indicators, thereby contributing to an improvement in the accuracy and adaptability of the overall model.

Furthermore, studies conducted on Adaptive Neuro-Fuzzy Inference Systems (ANFIS) have analyzed the sensitivity and stability of such systems [16–17]. Researchers developed methods to ensure optimal system performance by examining the sensitivity levels of input parameters. As a result of this approach, the flexibility of the model increased, and the accuracy of output results was significantly improved.

The process of selecting membership functions has a direct impact on the dynamic characteristics of a fuzzy logic system. This selection serves as a critical factor determining the system's overall efficiency, stability, and decision-making accuracy. If membership functions are chosen incorrectly or do not correspond to the inherent characteristics of the system, it may lead to uncertainty, misclassification, and prediction errors during the decision-making process. Consequently, the system's performance efficiency decreases, and the stability of output results deteriorates. Therefore, it is essential to consider the parametric sensitivity of the system and optimally adapt the membership functions to ensure reliable and precise operation.

In this study, the proposed combinational approach—which involves applying triangular, trapezoidal, and Gaussian membership functions according to the operational state of each server—is aimed at ensuring the optimal sensitivity level of the system. Specifically, under low-load conditions, the triangular membership function is utilized due to its simplicity and rapid responsiveness. For medium-load conditions, the trapezoidal membership function is employed, as it enables smooth transitions between operating states while maintaining the stability of the system. In contrast, under high-load conditions, the Gaussian membership function is adopted because it provides high accuracy and sensitivity, thereby reducing uncertainties within the system. Consequently, this combinational selection strategy enhances the overall adaptability, robustness, and precision of the fuzzy inference process.

Each membership function provides the most appropriate response to the corresponding operational state of the system through its mathematical form and sensitivity parameters. In this way, the combinational approach enables balanced and adaptive management of the request flow by considering such parameters as server load level, response time, resource utilization coefficient, and operational efficiency.

The analytical stages of this combinational approach are illustrated in Fig. 4. In the figure, the sensitivity levels for the triangular, trapezoidal, and Gaussian functions are explicitly defined, demonstrating how each function operates to ensure system efficiency under different conditions. This diagram provides a clear conceptual understanding of how to apply the appropriate method for selecting the optimal sensitivity level corresponding to each system state.

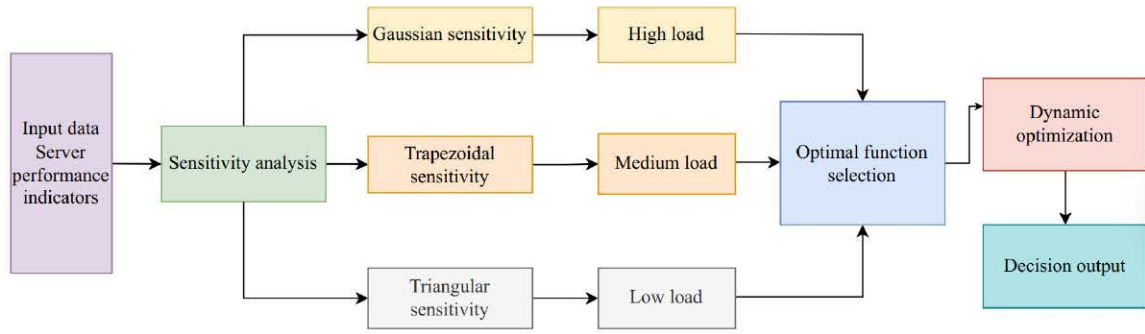


Fig. 4. Stages of combinational sensitivity analysis in the selection of membership functions

During the evaluation process of the combinational approach, various combinations of membership functions are analyzed with the aim of enhancing the overall sensitivity of the system, and the most efficient configuration is subsequently selected. Each function — triangular, trapezoidal, and Gaussian — exhibits a distinct influence on the sensitivity level based on its mathematical characteristics. Therefore, to ensure the system operates with optimal sensitivity, it becomes essential to perform both normalization of sensitivity indicators and an optimization procedure to achieve the most effective balance among these functions.

The normalized sensitivity index is one of the key indicators used to evaluate the overall sensitivity of a system. This index allows for a comparative assessment of sensitivity levels across different system states and determines the range of their variation. For each state, both the maximum and minimum sensitivity values are considered, which together serve as the basis for assessing the system's overall stability and adaptability.

The mathematical representation of the normalized sensitivity index is as follows:

$$S_{norm} = \frac{1}{3} \sum_{k=1}^3 \frac{S_i - S_{min,k}}{S_{max,k} - S_{min,k}}, \quad (3)$$

where S_k — sensitivity level for each state; $S_{max,k}$ — maximum sensitivity for each state; $S_{min,k}$ — minimum sensitivity for each state; S_{norm} — normalized sensitivity index.

This formulation implies that the normalized sensitivity index accounts for the quantitative differences among the sensitivity values corresponding to each operational state of the system and evaluates their relative proportions in a normalized manner. Such an analytical approach enables a comprehensive assessment of the system's overall sensitivity level and provides a foundation for its mathematical optimization.

As a result, the comparative evaluation of maximum and minimum sensitivity values contributes to reducing systemic uncertainty, achieving a balanced sensitivity distribution, and enhancing the overall efficiency of adaptive control mechanisms. Hence, the S_{norm} index serves as a fundamental quantitative criterion for the integrated evaluation of the system's sensitivity state, ensuring the stability, robustness, and adaptability of its functional performance.

Applying the combinational approach in analyzing the sensitivity level of each membership function enables the identification of the most effective function for achieving optimal performance. This approach contributes to the normalization of the system's overall sensitivity, allowing the optimization of sensitivity in each operational state. Furthermore, within the decision-making process, it becomes essential to monitor the system's sensitivity parameters

Table 2

Sensitivity levels and normalized sensitivity index for different approaches

Approach	Low	Medium	High	S_{norm}
Only Triangular	0.92	0.65	0.58	0.717
Only Gaussian	0.61	0.72	0.91	0.747
Combinational	0.95	0.88	0.93	0.920

and select appropriately combined functions that ensure maximal performance and adaptive response.

The evaluation of the combinational approach facilitates smooth system operation, minimizes uncertainty, and enhances overall sensitivity. As a result, the system maintains efficient performance under varying conditions, improves predictive accuracy, and reduces response time, thereby achieving a higher degree of stability and adaptability.

The results of the calculations based on different membership functions' sensitivity levels and the S_{norm} expression used to evaluate the overall system sensitivity are presented in Table 2. This table demonstrates how the S_{norm} value reflects both the system's general sensitivity and its optimal operational performance. A higher S_{norm} index indicates an improvement in the system's sensitivity optimization, enhances the effectiveness of the decision-making process, and contributes to better management of request flows within the system.

Evaluating the reliability of the proposed combinational approach plays a crucial role in ensuring the optimal and stable operation of the system. In information systems, reliability refers to the system's ability to maintain continuous operation under the influence of various internal and external factors. Therefore, reliability is regarded as one of the key indicators for assessing and predicting the performance efficiency of complex systems.

The primary objective of system reliability evaluation is to determine how consistently the system performs under different load conditions, as well as to identify potential risk scenarios and assess the system's ability to manage them. In this process, it becomes essential to mathematically model the interdependence between the system's risk coefficient α_k and the sensitivity–reliability interaction parameter β .

Within this framework, the overall system reliability is expressed through the following general mathematical relationship:

$$R_{system} = \prod_{k=1}^3 (1 - \alpha_k \cdot e^{-\beta S_k}), \quad (4)$$

where α_k — the weighting coefficient of the k -th component; β — the system's criticality parameter; S_k — the load factor affecting the k -th component; $e^{-\beta S_k}$ — represents the conditional probability of component failure under the given load.

The study examined the problem of evaluating the reliability of server systems and optimizing their sensitivity levels. In the proposed model, the interrelationship between external load factors affecting the server, the reliability of its components, and sensitivity indicators was thoroughly analyzed. These parameters play a critical role in ensuring the stable operation of the server system and effectively managing potential risk factors.

Shannon's information theory, which focuses on assessing the interaction between signal and noise within a system, primarily addresses issues related to entropy and information loss during

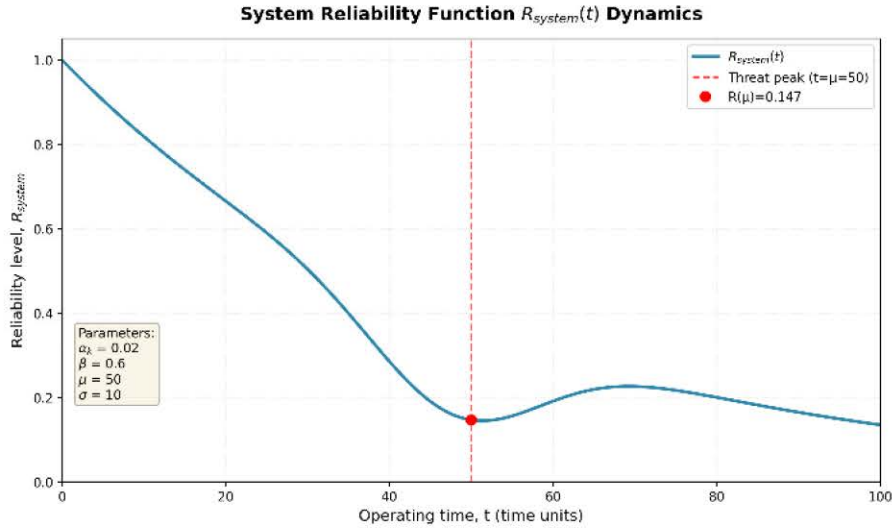


Fig. 5. Reliability function of the system over time

data transmission. However, this approach cannot directly capture the reliability, sensitivity, and stability of server systems operating under risk conditions. The Shannon model does not account for the system's dynamic response to stress factors such as load intensity, resource constraints, or latency. Consequently, it is not fully applicable to complex technical systems where sensitivity optimization, stability maintenance, and risk management are of fundamental importance.

In the present study, a model aimed at evaluating the system's reliability level, sensitivity coefficient, and response to risk impact was employed. As a result, the system's sensitivity level under risk conditions and its reliability parameters are assessed in an optimized manner. The overall reliability level of the system is determined using the following equation:

$$R_{system} = e^{-\alpha_k t} \cdot \left(1 - \beta e^{-\frac{(t-\mu)^2}{2\sigma^2}} \right), \quad (5)$$

where α_k — denotes the failure rate of the k -th component, i. e., the probability of component failure per unit of time; t — represents the operational time or the duration of the system's activity; β — maximum amplitude of risk, indicating the highest intensity of potential risk over time; μ — corresponds to the mean time of risk occurrence, representing the time interval during which the probability of system malfunction is the highest; σ — denotes the dispersion coefficient of the risk parameter, reflecting how widely the risk is distributed over time.

The variation dynamics of the system reliability function R_{system} with respect to the time factor are illustrated in Fig. 5.

Initially, the system operates with a high level of reliability; however, over time, the reliability level gradually decreases due to factors such as component aging or increased load. During the average load period ($t \approx \mu t$), the system's reliability indicator undergoes the most significant change. This situation clearly demonstrates the interrelation between the system's sensitivity parameters and risk coefficients, reflecting how variations in risk exposure directly affect the overall reliability behavior of the system.

2. Results. In the process of managing the query flow based on fuzzy logic, a series of systematic tests were conducted to evaluate system reliability by adapting server parameters

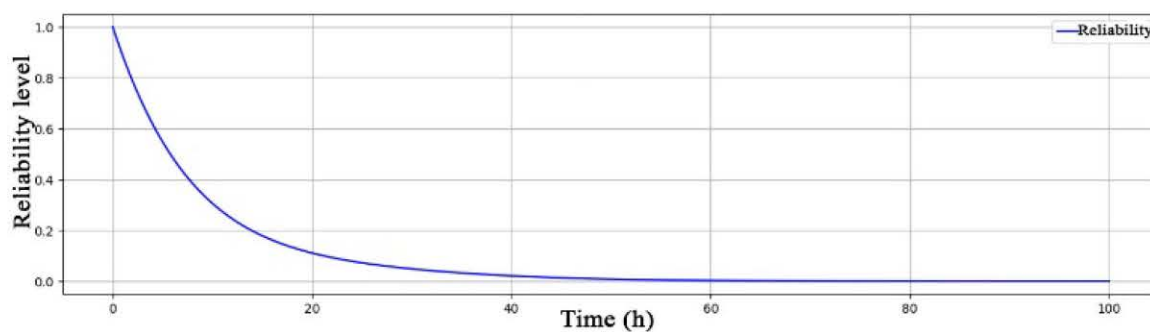


Fig. 6. Reliability dynamics for the triangular membership function

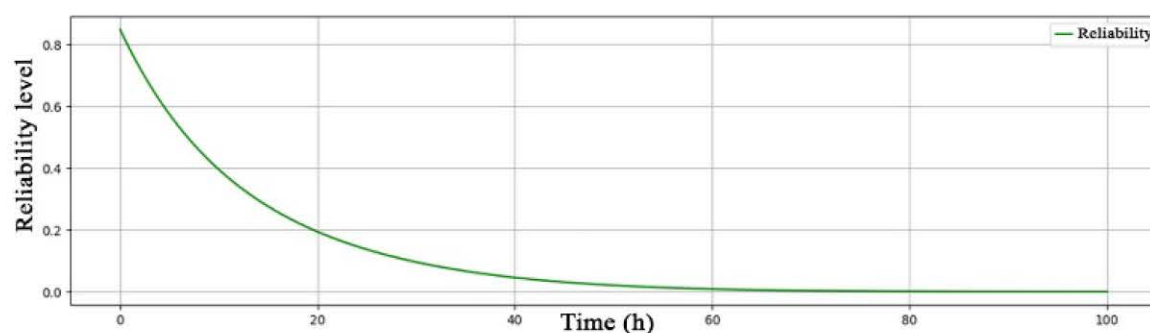


Fig. 7. Reliability dynamics for the trapezoidal membership function

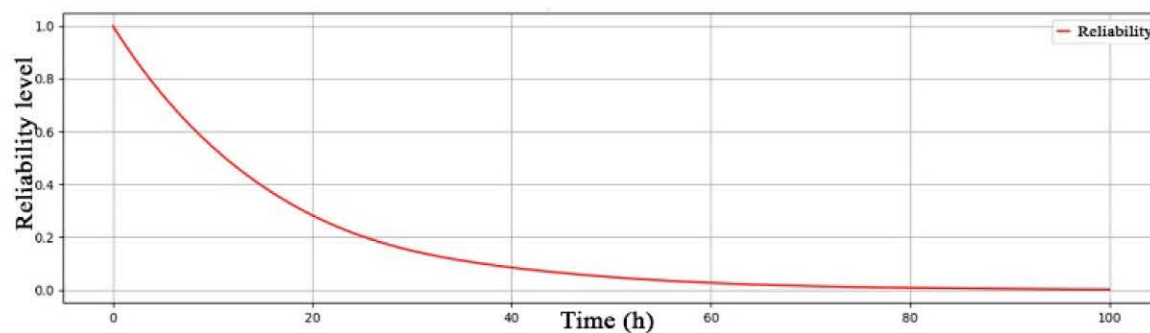


Fig. 8. Reliability dynamics for the Gaussian membership function

to the corresponding linguistic variables. The outcomes of these tests were recorded separately for each membership function. Initially, the system was tested using the triangular membership function, and the corresponding results are illustrated in Fig. 6.

For stable operating conditions, the tests were performed with the trapezoidal membership function, and the obtained results are presented in Fig. 7.

For high-load conditions, the tests were carried out using the Gaussian membership function, and the results are shown in Fig. 8. These findings are particularly important for assessing the system's sensitivity in managing the query flow under conditions of uncertainty and heavy workload.

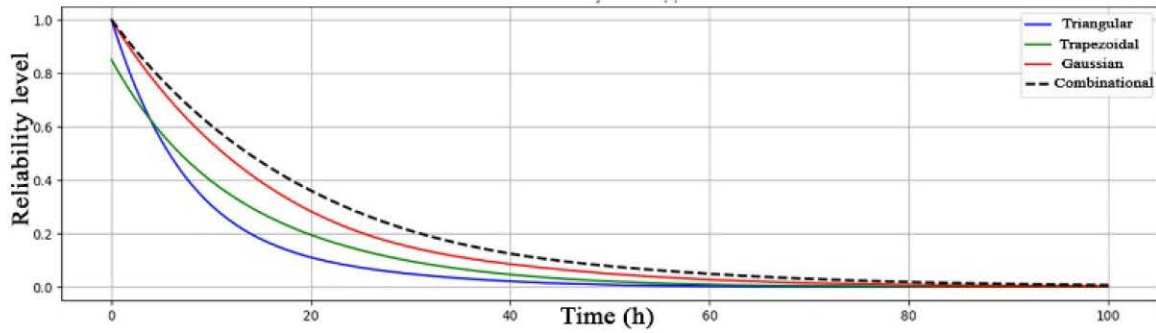


Fig. 9. Reliability dynamics for the Gaussian type membership function

Table 3

Reliability results for different membership functions

Membership function	α_k	β	S_k	R_{system}
Triangular	[0.2, 0.3, 0.5]	0.1	[0.92, 0.88, 0.93]	0.587
Trapezoidal	[0.25, 0.35, 0.45]	0.1	[0.90, 0.85, 0.91]	0.587
Gaussian	[0.15, 0.25, 0.35]	0.1	[0.95, 0.92, 0.94]	0.587
Combinational	[0.18, 0.28, 0.38]	0.1	[0.93, 0.89, 0.92]	0.587

Additionally, under increased load conditions, further evaluations with the Gaussian-type membership function were conducted, the results of which are presented in Fig. 9. These results emphasize the significance of analyzing the system's adaptive response and robustness when operating under unstable or uncertain environments, providing deeper insights into the fuzzy-based reliability control mechanism.

The reliability results calculated for each membership function are presented in Table 3. This table provides the basis for analyzing the system's reliability levels under different operational conditions, as well as for identifying the influence of risk and sensitivity parameters on the system's overall performance. The obtained results serve as a significant scientific foundation for the optimization of the system and ensuring its stable and reliable operation.

In this study, particular attention was devoted to evaluating the sensitivity of membership functions during the fuzzification stage and to optimizing their aggregation states. The appropriate selection of membership functions for low, medium, and high conditions plays a crucial role in ensuring the system's efficient performance. Therefore, the chosen combinational approach allowed achieving optimal sensitivity for each linguistic state of the system, which in turn enhanced both its stability and overall effectiveness.

The sensitivity of membership functions is a fundamental factor in managing uncertainties within the system, as even minor variations in parameters can have a significant impact on the output results. Through the implementation of an optimal combinational strategy, the uncertainties arising in the decision-making process are minimized, thereby improving the system's overall operational efficiency and robustness.

Conclusion. The study demonstrated that applying a single type of membership function to all input parameters reduces the system's sensitivity and limits decision accuracy. Therefore, an approach combining triangular, trapezoidal, and Gaussian functions—selected according to the semantic characteristics of the parameters—was developed. This approach ensured

higher sensitivity and discriminative capability in fuzzy systems, minimized uncertainty in the decision-making process, and enhanced overall system stability. Experimental results confirmed that the combinational approach outperforms traditional methods by reducing response time, balancing server load, and increasing decision accuracy. Consequently, selecting membership functions based on semantic relevance significantly improves system efficiency.

Список литературы

1. Zadeh L. A. Fuzzy sets // *Information and Control*, 1965. N 8(3). P. 338–353. DOI: [https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X).
2. Smith J., Doe A. A fuzzy logic analysis method for evaluating human sensitivities // *Journal of Fuzzy Systems*, 2018. N 45(3). P. 123–136. DOI: <https://doi.org/10.1016/j.jfuzz.2018.01.005>.
3. Gures E., Shayea I., Ergen M., Azmi M.H., El-Saleh A.A. Machine Learning-Based Load Balancing Algorithms in Future Heterogeneous Networks: A Survey // *IEEE Access* 2022. N 10. P. 37689–37717.
4. Akhatov A.R., Kenjaev S.S., Tojiev M.R. Improved Round Robin Algorithm Based on Fuzzy Logic and Genetic Algorithm for Server Load Balancing // *International Russian Automation Conference (RusAutoCon)*, Sochi, Russian Federation, 2025. P. 296–301. DOI: [10.1109/RusAutoCon65989.2025.11177403](https://doi.org/10.1109/RusAutoCon65989.2025.11177403).
5. Raximov N., Kuvandikov J., Khasanov D. The importance of loss function in artificial intelligence // *International Conference on Information Science and Communications Technologies (ICISCT 2022)*, DOI: [10.1109/ICISCT55600.2022.10146883](https://doi.org/10.1109/ICISCT55600.2022.10146883).
6. Asghari M.Z., Ozturk M., Hämmäläinen J. Reinforcement Learning Based Mobility Load Balancing with the Cell Individual Offset // *In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, Helsinki, Finland, 2021. P. 25–28.
7. Axatov A., Nurmamatov M., Nazarov F., Sariyev Sh. Genetic algorithm application technology in multi-parameter optimization problems // *AIP Conf. Proc.*, 2024. V. 3244, art. N 030025, DOI: [10.1063/5.0242074](https://doi.org/10.1063/5.0242074).
8. Forrester Research, *Global Digital Experience Benchmark Report*, 2023.
9. Garcia M., Wong T. Fuzzy Membership Function Evaluation by Non-Linear Regression // *International Journal of Fuzzy Systems*. N 23(1). P. 34–45. DOI: <https://doi.org/10.1109/IJFS.2021.000567>.
10. Scholz-Reiter B. et al. Fuzzy-based load balancing in production control // *CIRP Annals*, 2015.
11. Parmanov B.Z., Nazarov F.M., Tojiyev M.R. Development of an Algorithm for Intelligent Analysis of Data From a Microprocessor-Based Portable Spectrophotometer // *In Proceedings International Russian Smart Industry Conference, SmartIndustryCon 2025*, Institute of Electrical and Electronics Engineers Inc., 2025, P. 664–669. DOI: [10.1109/SmartIndustryCon65166.2025.10985982](https://doi.org/10.1109/SmartIndustryCon65166.2025.10985982).
12. Chen W., Huang J. Sensitivity and robustness analysis of adaptive neuro-fuzzy inference systems // *IEEE Transactions on Neural Networks and Learning Systems*, 2020. N 31(4). P. 945–957. DOI: <https://doi.org/10.1109/TNNLS.2019.2949824>.
13. Ghatak G., De Domenico A., Coupechoux M. Coverage Analysis and Load Balancing in HetNets With Millimeter Wave Multi-RAT Small Cells // *IEEE Trans. Wirel. Commun.* 2018. N 17. P. 3154–3169.
14. Maruf T. Hazard recognition system based on violation of the integrity of the field and changes in the intensity of illumination on the video image // *International Conference on Information Science and Communications Technologies, ICISCT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. DOI: [10.1109/ICISCT55600.2022.10146933](https://doi.org/10.1109/ICISCT55600.2022.10146933).

15. Axatov A., Nurmamatov M., Nazarov F., Sariyev Sh.. Genetic algorithm application technology in multi-parameter optimization problems // AIP Conf. Proc., 2024. V. 3244, art. N 03002524. DOI: 10.1063/5.0242074.

16. Nixon M. S. et al. Real-time fuzzy logic-based resource allocation // Journal of Real-Time Systems, 2021.

17. Parsa S. Adaptive fuzzy load balancing algorithm for heterogeneous distributed systems // Journal of Supercomputing, 2017.

18. Kenjaev S. S., Rashidov A. E. File storage methods and algorithms for optimal management of various types of data // "Descendants of Al-Farghani" electronic scientific journal of Fergana branch of TATU named after Muhammad al-Khorazmi, 2024. V 1. N 3. DOI: 10.5281/zenodo.13954911.

19. Hatipoğlu A., Başaran M., Yazici, M. A., Durak-Ata L. Handover-based Load Balancing Algorithm for 5G and Beyond Heterogeneous Networks // In Proceedings of the 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Brno, Czech Republic, 2020. P. 5–7.



Кенжаев Санжар Соби-рович — аспирант Самаркандского государственного университета имени Шарофа Рашидова. E-mail: s.kenjaye22@gmail.com. В 2008 году получил степень бакалавра в Сам-

ГУ (механика-математика), в 2021 году — степень магистра СамГУ (информационные технологии и мультимедиа). В 2021 году стажировался в Инженерной школе EPL в Труа (Франция), с октября 2024 по март 2025 — в Санкт-Петербургском политехническом университете (Россия).

Sanjar Kenjaev — Sanjar Sobirovich Kenjaev is a PhD at Samarkand State University. E-mail: s.kenjaye22@gmail.com. In 2008, he received a bachelor's degree from SamSU (Maths), and in 2021, he earned a master's degree from SamSU (information technology and multimedia). In 2021, he interned at the EPL Engineering School in Troyes (France), from October 2024 to March 2025 — at St. Petersburg Polytechnic University (Russia).

Ахатов Акмаль Рустамович — e-mail: akmalar@rambler.ru, тел: +998902716418. Д-р техн. наук, профессор, проректор по международному сотрудничеству Самаркандского государственного университета имени Шарофа Рашидова по специальности 05.01.02 — «Системный анализ, управление и обработка информации». Коли-



чество опубликованных научных работ — более 100. Область научных интересов: обработка данных, повышение и обеспечение достоверности этого процесса, автоматизация процессов принятия решений и управления при обработке информации, системы искусственного интеллекта и большие данные.

Akmal Rustamovich Akhatov — e-mail: akmalar@rambler.ru, phone: +998902716418. Doctor of technical sciences, professor, vice-rector for international cooperation of Samarkand State University named after Sharof Rashidov, specialty 05.01.02 — "Systematic analysis, management and information processing". The number of published scientific works is more than 100. Field of scientific interests: data processing, increasing and ensuring the reliability of this process, automation of decision-making and management processes in information processing, artificial intelligence systems and Big Data.

Тоджиев Маруф Рузikuлович — PhD, доцент кафедры «Теория управления и информационная безопасность» Самаркандского государственного университета имени Шарофа Рашидова. E-mail: mrtojiev6886@gmail.com. ORCID: 0000-0002-2899-4495



В 2014 году получил степень бакалавра в Джизакском политехническом институте, в 2017 — степень магистра в Ташкентском университете информационных технологий имени

Мухаммада аль-Хорезми (компьютерный инжиниринг). В 2024 году стал базовым докторантом кафедры компьютерных наук и программирования Джизакского филиала НУУз.

Научные интересы охватывают области компьютерного зрения, компьютерного моделирования, кибербезопасности и квантовых вычислительных алгоритмов. Ведет преподавательскую деятельность по указанным дисциплинам, а также осуществляет научно-исследовательскую работу в данных направлениях.

Tojiyev Maruf Ruzikulovich — PhD, Associate Professor of the Department of Management Theory and Information Security of

the Sharof Rashidov Samarkand State University. E-mail: mrtojiyev6886@gmail.com. ORCID: 0000-0002-2899-4495.

In 2014, he received a bachelor's degree from the Jizzakh Polytechnic Institute, and in 2017, he earned a master's degree from the Muhammad al-Khwarizmi Tashkent University of Information Technologies (Computer Engineering). In 2024, he became a basic doctoral student at the Department of Computer Science and Programming at the Jizzakh branch of NUUz.

His research interests include computer vision, computer modeling, cybersecurity, and quantum computing algorithms. He teaches these subjects and conducts research in these areas.

Дата поступления — 14.11.2025

ON COMBINING ACTIVE KNOWLEDGE CONCEPT AND MACHINE LEARNING

В. А. Перепелкин*, С. Б. Арыков**

Institute of computational mathematics and mathematical geophysics SB RAS,
630090, Novosibirsk, Russia

*Novosibirsk State University,
630090, Novosibirsk, Russia

**Novosibirsk State Technical University,
630073, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2026-1-65-78

EDN: YKXJIB

The problem of automatic program construction remains relevant in the context of continuously progressing computerization of all spheres of society. Despite significant achievements in automation tools such as compilers, programming languages, integrated development environments, and artificial intelligence methods, program construction largely remains a creative task that is difficult to automate. Universal approaches to automatic program construction that would produce sufficiently efficient programs for practical use within acceptable timeframes are algorithmically complex and have only niche applications. Therefore, it is important to explore and develop different approaches to automatic program construction and methodologies for their application and combination. In particular, it is valuable to utilize the experience of manual program development in various subject domains and create programming automation tools based on this experience.

This paper examines the active knowledge concept as one such approach and explores how it relates to and can be combined with machine learning methods. The active knowledge concept is a methodology for automatic program construction in specific subject domains. It is based on the theory of synthesis of parallel programs and systems on computational models [1]. The methodology enables automatic construction of sufficiently efficient programs for solving problems of a certain class within a particular subject domain, where efficiency is understood from the perspective of non-functional properties essential to that domain, such as execution time, memory consumption, network load, etc.

The fundamental principles of the active knowledge concept include non-universality, axiomatic theory as a foundation, partial description of the subject domain, use of computational models, multi-level construction, consideration of static and dynamic aspects, application to well-developed subject domains, inclusion of good solutions, variability, recommendations, alternative approach to program creation, accumulation of knowledge in active form, and instrumental support through systems like LuNA (Language for Numerical Algorithms). The LuNA system, developed and maintained at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS, includes a language for describing active knowledge bases and a system for automatic construction and execution of parallel programs.

Machine learning methods, particularly those based on large language models, offer new opportunities for automating programming tasks. A practically widely used technology is the creation

This research was carried out under the state contract with ICM&MG SB RAS FWNM-2025-0005.

of AI assistants — digital assistants capable of not only executing predefined commands but also understanding context and adapting their behavior accordingly. In software development, AI assistants can help with code generation, documentation, quality checking, education, and auxiliary technical tasks. Popular solutions include Cursor, Codex, GitHub Copilot, Gemini Code Assist, and domestic developments like SourceCraft Code Assistant and GigaCode.

This paper explores the potential synergies between the active knowledge concept and machine learning methods. Several principal possibilities for combining these approaches are considered: (1) An AI assistant for training and methodological support of LuNA system users, helping to overcome the entry barrier; (2) AI assistance in creating active knowledge bases, acting as a methodological consultant guiding users through key stages; (3) Automatic creation of active knowledge bases from human-oriented texts using machine learning; (4) AI translation from informal problem statements to formal specifications; (5) Improvement of active knowledge bases through analysis and optimization using large language models.

The paper presents a prototype implementation of an AI assistant for the LuNA system (LuNA AI), focused initially on documentation-related tasks to help overcome the entry barrier in learning and using the active knowledge concept. Technically, LuNA AI is a service that accepts context from the user (files, questions), combines this information with the available knowledge base (documentation, templates, examples), forms a prompt, calls a large language model, and displays the LLM's response to the user.

The implementation utilizes Yandex AI Assistant API, which combines the YandexGPT large language model with RAG (Retrieval-Augmented Generation) technology for retrieving relevant information from knowledge bases. The system architecture includes a frontend application built with Vue and NuxtUI frameworks and a backend developed in Python with FastAPI. The user's query is vectorized and, together with the most relevant context from the knowledge base, is fed to the LLM. For convenient updating of the knowledge base, the concept of a “project” is implemented at the UI level — a set of files forming a knowledge base, with support for multiple projects to maintain different versions of the LuNA system.

The current demonstration version of the AI assistant solves part of the documentation-related tasks: users can request information about the LuNA system, and the AI assistant builds relevant responses using RAG. Although the quality of responses currently requires further improvement, particularly in enhancing the knowledge base quality, even in this form the assistant can provide significant help to those newly acquainted with the LuNA language.

The research demonstrates that the active knowledge concept and machine learning methods have significant potential for combined use. Active knowledge bases, based on axiomatic theories, can serve as a flexible, rigorous mathematical apparatus allowing practically unlimited accumulation of knowledge that remains unchanged over time with the possibility of automatic application. Machine learning methods can rely on this apparatus, solving optimization problems within given axiomatic theories, as well as serving as a bridge to the informal human context.

Future work includes improving the quality of the knowledge base, integrating the AI assistant into popular development environments like VS Code, and expanding its functionality with additional capabilities such as code generation and optimization suggestions within the LuNA framework.

Key words: active knowledge concept, LuNA system, LLM, deep learning, AI assistant.

References

1. Sintezy paralelnykh programm i sistem na vychislitelnykh modelyakh / V. A. Valkovsky, V. E. Malyshev; Onv. red. V. E. Kotov; AN SSSR, Sib. otd-nie, VC. — Novosibirsk : Nauka. Sib. otd-nie, 1988. 126 s. (In Russian).

2. Malyshkin V. Active Knowledge, LuNA and Literacy for Oncoming Centuries // LNCS, Vol. 9465. 2015. S. 292-303. DOI: 10.1007/978-3-319-25527-9_19.
3. Malyshkin V. E., Perepelkin V. A. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Proceedings of the 11th International Conference on Parallel Computing Technologies (PaCT-2011), LNCS, 2011. T. 6873. S. 53–61. DOI: 10.1007/978-3-642-23178-0_5.
4. Arykov S. Defining Order of Execution in Aspect Programming Language // LNCS. 2017. Vol. 10421. P. 255–261. ISBN 978-3-319-62932-2. DOI 10.1007/978-3-319-62932-2_25.
5. Weisz et al. Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise // Proceedings of the Extended Abstracts of the CHI EA '25 Conference on Human Factors in Computing Systems. Article No. 673. P. 1–13. ISBN: 9798400713958. DOI: 10.1145/3706599.3706670.
6. Jiang et al. CursorCore: Assist Programming through Aligning Anything // CoRR. Vol. abs/2410.07002. 2024. ISSN 2331-8422. DOI 10.48550/arXiv.2410.07002.
7. Mark Chen et al. Evaluating Large Language Models Trained on Code // CoRR. Vol. abs/2107.03374. 2021. ISSN 2331-8422. DOI 10.48550/arXiv.2107.03374.
8. Pandey et al. Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects // CoRR. 2024. Vol. abs/2406.17910. ISSN 2331-8422. DOI 10.48550/arXiv.2406.17910.
9. Gemini Code Assist | AI coding assistant [Electron. Res.]: <https://developers.google.com/gemini-code-assist/docs/overview> (accessed 24.11.2025).
10. Grok Code Fast 1 Model Card [Electron. Res.]: <https://data.x.ai/2025-08-26-grok-code-fast-1-model-card.pdf> (accessed 03.02.2025).
11. SourceCraft. Code Assistant. [Electron. Res.]: <https://sourcecraft.dev/portal/code-assistant/> (accessed 22.11.2025).
12. GigaCode [Electron. Res.]: <https://gigacode.ru/> (accessed 22.11.2025).
13. Tal Kadosh et al. OMPar: Automatic Parallelization with AI-Driven Source-to-Source Compilation // CoRR. Vol. abs-2409-14771. 2024. ISSN 2331-8422. DOI 10.48550/arXiv.2409.14771.
14. Mahmud et al. AutoParLLM: GNN-Guided Automatic Code Parallelization using Large Language Models // CoRR. Vol. abs/2310.04047. 2023. ISSN 2331-8422. DOI 10.48550/arXiv.2310.04047.
15. Nichols et al. HPC-Coder: Modeling Parallel Programs using Large Language Models // ISC High Performance 2024 Research Paper Proceedings (39th International Conference). Hamburg, Germany. 2024. P. 1–12. DOI 10.23919/ISC.2024.10528929.
16. Yandex AI Assistant API. [Electron. Res.]: <https://yandex.cloud/ru/docs/ai-studio/concepts/assistant/> (accessed: 24.11.2025).
17. Oche et al. A Systematic Review of Key Retrieval-Augmented Generation (RAG) Systems: Progress, Gaps, and Future Directions // CoRR. 2025. Vol. abs/2507.18910. ISSN 2331-8422. DOI 10.48550/arXiv.2507.18910.

СОВМЕСТНОЕ ПРИМЕНЕНИЕ КОНЦЕПЦИИ АКТИВНЫХ ЗНАНИЙ И МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

В. А. Перепелкин*, С. Б. Арыков**

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

*Новосибирский национальный исследовательский государственный университет,
630090, Новосибирск, Россия

**Новосибирский государственный технический университет,
630073, Новосибирск, Россия

УДК 004.4'242

DOI: 10.24412/2073-0667-2026-1-65-78

EDN: YKXJIB

В работе рассматриваются возможности совместного использования концепции активных знаний и методов машинного обучения для повышения уровня автоматизации разработки параллельных программ. Концепция активных знаний представляет собой методологию автоматического конструирования программ в конкретных предметных областях. В ее основе лежит идея создания базы активных знаний — специального формализованного описания конкретной предметной области, благодаря чему в этой предметной области существенно снижается сложность автоматического конструирования программ. Методы машинного обучения могут способствовать автоматизации и упрощению создания и использования таких баз активных знаний, снижать порог вхождения в работу с системой автоматического конструирования программ (например, системой LuNA). В работе представлен ИИ-помощник, способный в неформальном общении отвечать на вопросы о системе LuNA и концепции активных знаний. Он реализован на основе большой языковой модели и использует подход RAG для работы с подготовленной для этого текстовой базой знаний. В дальнейшем планируется развивать функциональность ИИ-помощника.

Ключевые слова: концепция активных знаний, система LuNA, машинное обучение, большие языковые модели, ИИ-ассистент.

Введение. На фоне постоянно прогрессирующей компьютеризации практически всех сфер жизни общества не вызывает сомнений актуальность проблемы автоматизации конструирования программ. Она находится под постоянным вниманием ученых и программистов, и ее разработка весьма плодотворна с точки зрения того, какое количество ручного труда удастся переложить на компьютер с приемлемым (а зачастую и превосходящим ручное) качеством этого труда. К таким результатам можно отнести компиляторы, языки и системы программирования, интегрированные среды разработки, генераторы документации, методы искусственного интеллекта и т. п.

Несмотря на колоссальные усилия и результаты в этой области, проблема автоматического конструирования программ на текущий момент остается открытой, и запрос на

Работа выполнена в рамках государственного задания ИВМиМГ СО РАН FWNM-2025-0005.

развитие средств автоматизации программирования только возрастает. Это связано с тем, что в значительной степени конструирование программ остается творческой задачей, плохо поддающейся автоматизации, а прямолинейные и универсальные подходы к автоматическому конструированию за приемлемое время достаточно хороших для практического использования программ являются алгоритмически труднорешаемыми и потому имеют только нишевое практическое применение.

Как следствие, важно исследовать и развивать разные подходы к автоматическому конструированию программ и методологию их применения и сочетания. В частности, важно использовать опыт ручной разработки программ в различных предметных областях и на его основе создавать средства автоматизации программирования. В статье рассматривается один из таких подходов — концепция активных знаний — с точки зрения того, как он соотносится и как может сочетаться с некоторыми методами машинного обучения. Также в статье представлены текущие результаты по реализации одного из таких возможных сочетаний — созданию программного ассистента на основе больших языковых моделей для системы активных знаний LuNA.

Остальная часть статьи структурирована следующим образом. В разделе 1 приводится принципиальное описание концепции активных знаний. В разделе 2 дается характеристика методам искусственного интеллекта, затрагиваемых в статье. В разделе 3 рассматриваются возможности совместного использования концепции активных знаний и методов машинного обучения. В разделе 4 представлены текущие результаты по созданию ИИ-помощника для системы LuNA. В заключении подводятся итоги статьи.

1. О концепции активных знаний. Концепция активных знаний — это методология автоматического конструирования программ в конкретных предметных областях. Она базируется на теории синтеза параллельных программ и систем на вычислительных моделях [1] и была изложена в [2]. Эта методология позволяет обеспечить автоматическое конструирование достаточно эффективных программ решения задач определенного класса в некоторой предметной области. Тут и далее эффективность понимается с точки зрения существенных в той или иной предметной области нефункциональных свойств программы, таких как время выполнения, расход памяти, нагрузка на сеть и т. п. Под функциональными свойствами в работе будем понимать функцию, которую вычисляет программа, т. е. как она отображает свои входные данные в выходные.

Рассмотрим кратко основные принципы концепции активных знаний, дающие представление об этой методологии в целом.

Неуниверсальность. В общей постановке задача автоматического конструирования оптимальных программ по их высокоуровневой спецификации является алгоритмически труднорешаемой задачей, поэтому поиск универсального способа нецелесообразен. Тем не менее, во многих предметных областях существуют различные частные и эвристические подходы и алгоритмы, способные обеспечить за разумное время конструирование достаточно хороших для практического применения программ. Для этого требуется построить базу активных знаний — формализованное частичное описание предметной области, которое содержит информацию о существенных с точки зрения автоматического конструирования программ особенностях этой предметной области и накопленных в ней существующих программных модулях решения отдельных подзадач. Существование достаточно хорошей базы активных знаний позволяет системе конструирования программ (например, системе LuNA [3]) автоматически конструировать в этой предметной обла-

сти программы удовлетворительного качества. Чем уже предметная область, тем проще построить для нее базу активных знаний.

Аксиоматическая теория в основе. В основе базы активных знаний лежит аксиоматическая теория, которая частично описывает предметную область. Аксиоматическая теория представляет собой универсальный математический инструмент, позволяющий строго описывать практически что угодно в предметной области и однозначно интерпретировать это описание, в том числе с помощью компьютера. Наличие такого формального описания позволяет строить описания предметных областей, которые могут быть использованы неограниченно долго в будущем, что позволяет использовать базы активных знаний как способ накопления информации о существующих решениях (алгоритмах, программах, математических моделях, методах и т. п.) для их автоматического использования в решении новых задач.

Частичность описания предметной области. Аксиоматическая теория описывает не всевозможные детали о предметной области, а только частично. В аксиоматическую теорию входит описание ключевых понятий предметной области; хороших, проверенных практикой и отлаженных программных модулей; сложившейся практики применения этих модулей для решения практических задач; существующих эвристик и частных алгоритмов, обеспечивающих конструирование достаточно эффективных программ. Другие детали в аксиоматической теории не описываются, чтобы множество комбинаций не становилось слишком большим, и база активных знаний могла использоваться прагматично.

Использование вычислительных моделей. В качестве представления аксиоматических теорий используются вычислительные модели (см. [1]). Вычислительная модель является математической моделью, позволяющей описывать аксиоматические теории частного вида не с помощью аксиом и правил вывода, а в представлении, подобном двудольному ориентированному графу, заданному параметрически. Вычислительная модель прямо описывает понятия предметной области в виде множества переменных, могущих иметь значения. Также она включает множество операций, описывающих возможность вычислять значения одних понятий из других с помощью вычислительных модулей, называемых фрагментами кода. В качестве фрагментов кода могут выступать готовые программные модули — библиотечные процедуры, программы и т. п., разработанные на традиционных языках программирования.

Использование вычислительных моделей вместо классического представления аксиоматической теории в виде множества аксиом и правил вывода имеет прагматический смысл. Он заключается в том, что даже незначительное изменение аксиоматической теории (добавление, удаление или модификация какого-либо правила вывода или аксиомы) может кардинально и труднопредсказуемо изменить множество ее теорем. Изменение же вычислительной модели локально и гораздо легче понять и оценить последствия такого изменения. Это, как правило, добавление или удаление какого-либо понятия (переменной) или способа вычислять одни понятия из других (операция). Также теория синтеза параллельных программ и систем на вычислительных моделях уже хорошо развита в части автоматического конструирования алгоритмов и программ с учетом их нефункциональных свойств.

Многоуровневое конструирование. Процесс конструирования программы логически (а не хронологически) разделяется на несколько этапов. На первом уровне имеется постановка задачи и база активных знаний. Далее на этой основе строится алгоритм решения задачи (в [1] — планирование, построение VW-плана). На следующем уровне строится

отображение операций и переменных алгоритма на ресурсы параллельного вычислителя и определяется управление, которое задает частичный порядок выполнения операций [4]. Это задает программу решения задачи. Последний уровень — это исполнение программы, на котором обеспечиваются ее динамические свойства исполнительной системой или интерпретатором.

Такое разделение на уровне задает декомпозицию задачи автоматического конструирования программы на подзадачи (вывод алгоритма, генерация программы, обеспечение динамических свойств), сводя решение задачи к решению подзадач, и тем самым снижая сложность процесса автоматического конструирования программ.

Статика и динамика. Процесс конструирования и исполнения программы можно рассматривать как процесс принятия решений. Исходно, когда определена только задача, то на базе активных знаний может быть выведено, вообще говоря, множество различных решений (реализаций). Когда фиксируется алгоритм решения задачи, то множество реализаций сокращается до тех, которые реализуют выбранный алгоритм. И так далее, по мере принятия решений — о распределении ресурсов, управлении и т. п. — множество реализаций сокращается до одноэлементного, соответствующего фактически реализованному.

С этой точки зрения все решения могут приниматься как статически, на этапе конструирования кода программы, так и динамически — во время исполнения. В пределе это дает генерацию полностью статического кода или полностью динамическую интерпретацию. Промежуточный вариант означает исполнение частично определенной программы с помощью исполнительной системы, обеспечивающей динамические свойства (динамическая сборка мусора, динамическая балансировка нагрузки на узлы и т. п.). По возможности все решения следует принимать статически, чтобы минимизировать накладные расходы во время исполнения, а на время исполнения оставлять принятие только тех решений, которые необходимо принимать динамически для обеспечения динамических свойств.

Освоенные предметные области. Концепцию активных знаний следует применять для более или менее освоенных (разработанных) предметных областей — то есть таких, в которых уже имеются накопленные программные модули, которые могут быть переиспользованы в качестве решения подзадач при решении новых задач в этой предметной области, уже сложилась практика применения этих модулей, выработались необходимые эвристики, обеспечивающие хорошую эффективность программ, и т. п. Без этого материала в базу активных знаний будет нечего включать, и ее полезность будет сомнительна. Выгода по сравнению с ручным программированием заключается в том, что эти знания теперь используются в процессе конструирования программы автоматически, без необходимости человека их осваивать. Именно это и понимается под термином «активные знания» в противоположность «пассивным знаниям», примером которых может служить учебник — для применения содержащихся в нем знаний необходим человек, который их освоит.

Включение хороших решений. Так как конструирование новых программ в концепции активных знаний состоит в комбинировании уже имеющихся программных модулей в рамках, определенных аксиоматической теорией, то база активных знаний должна содержать достаточно таких модулей для того, чтобы охватить класс практически значимых задач. Качество результирующей программы существенно зависит от качества модулей, включенных в базу активных знаний. Качество результирующей программы зависит не только от качества модулей, но и от аксиоматической теории, и от алгоритмов конструирования программы.

Вариативность. В базу активных знаний включается множество различных решений одних и тех же задач/подзадач. Решения должны различаться между собой своими нефункциональными свойствами, и эти нефункциональные свойства должны быть формально описаны в базе активных знаний. Благодаря этому система конструирования сможет отдавать предпочтения тем вариантам, которые лучше подходят под ситуацию — особенности и конфигурацию аппаратного обеспечения, свойства обрабатываемых данных, состояние загруженности вычислителя и т. п.

Рекомендации. Программа конструируется автоматически, но возможно вмешательство человека в этот процесс для его настройки. Концепция активных знаний предполагает возможность включения в базу активных знаний рекомендаций — формально описанных высокоуровневых предложений о том, как следует конструировать программу в этой предметной области. Примером может служить формула, отображающая множество операций и переменных алгоритма на узлы мультимонитора, или заданное отношение частичного порядка на множестве операций, определяющая рекомендуемый частичный порядок их исполнения. Конструирование достаточно эффективной программы — сложная задача, часто требующая творческого подхода к своему решению. Рекомендации позволяют включить в базу активных знаний интуитивные представления человека о том, как следует конструировать и исполнять программу, существенно расширив область применения концепции активных знаний и повысив качество конструируемых программ. В перспективе, когда системные алгоритмы будут превосходить по качеству советы человека, рекомендации могут быть безболезненно отброшены из базы активных знаний.

Подход к созданию программ. Концепция активных знаний предлагает альтернативный традиционному подход к созданию программ. Вместо программирования как такового во главу угла становится математическая логика, которая используется для выражения знаний о предметной области и постановке задачи. Если на существующей базе активных знаний нужное решение построить невозможно, или оно оказывается недостаточно эффективным, то это решается включением в базу активных знаний новых элементов (модулей, алгоритмов, понятий и т. п.), а не дописыванием или переписыванием программы.

Накопление знаний в активной форме. Концепция активных знаний дает основу для практически неограниченного накопления знаний в активной форме с целью их последующего автоматического использования в решении новых задач. Это достигается за счет того, что база активных знаний — это отчуждаемое представление знаний, которое может быть проанализировано людьми (или компьютером): модули и аксиоматические теории, проверенные временем и практикой, включаются в базы активных знаний; в случае обнаружения ошибок — исключаются; более эффективные модули включаются в базы активных знаний, со временем вытесняя менее эффективные альтернативы. Этот процесс подобен научному освоению предметных областей, когда со временем полезные и практические понятия, теоремы и методы накапливаются в арсенале научного сообщества, а ошибочные теории и менее эффективные решения постепенно выявляются, опровергаются, исключаются из обихода. Ключевым тут является отчуждаемость и однозначная интерпретация знаний, представленных в виде базы активных знаний, их неизменность во времени. Это отличает концепцию активных знаний от некоторых методов машинного обучения, где затруднительно выявить и подвергнуть анализу знания, заложенные в искусственную нейросеть, а дообучение нейронной сети не гарантирует неизменность предыдущих знаний.

Система LuNA. Концепция активных знаний как методология может быть поддержана инструментально. Примером такого инструмента является система LuNA [3] (от Language for Numerical Algorithms — язык для численных алгоритмов), созданная и развиваемая в ИВМиМГ СО РАН. Эта система включает в себя одноименный язык, позволяющий описывать базы активных знаний, и систему автоматического конструирования и исполнения параллельных программ.

2. Разработка программ с применением методов машинного обучения. Активно применяемой на практике технологией на основе машинного обучения является создание ИИ-ассистентов (или ИИ-помощников). По сути, это цифровой помощник, умеющий не только выполнять набор заранее заданных команд, но понимающий контекст и меняющий свое поведение с его учетом.

Современные ИИ-ассистенты могут видеть, слышать, читать и работать с цифровыми инструментами. Они находят широкое применение как в личном использовании (например, могут разобрать почту или запланировать путешествие), так и в производственных задачах (например, выполняют функции тех. поддержки первой линии). Применительно к разработке программного обеспечения, ИИ-ассистент — это инструмент для помощи в выполнении задач, связанных с кодированием, документированием и тестированием кода [5]. Далее везде под ИИ-ассистентом будем иметь в виду именно помощника по разработке ПО.

Рассмотрим подробнее, как ИИ-ассистент может быть полезен в процессе разработки:

1) Работа с кодом: создать шаблон модуля или класса, предложить автодополнение или оптимизацию на основе контекста (завершение текущей строки или даже всей функции), дать рекомендации по архитектуре, сделать рефакторинг или просто объяснить код.

2) Работа с документацией: дать разъяснения о функционировании проекта, рекомендовать архитектурные решения на основе принятых в проекте стандартов, сгенерировать описание функции.

3) Проверка качества: выявление подозрительных (с точки зрения наличия ошибок) фрагментов кода, генерация юнит-тестов, выполнение UI-тестов, создание тестовых фикстур и моков.

4) Обучение: объяснение сложных концепций языков программирования или нюансов использования библиотек, подбор примеров, сравнительный анализ альтернатив.

5) Вспомогательные технические задачи: генерация конфигурационных файлов, отправка кода в репозиторий и другие.

Представленный перечень не является исчерпывающим, однако позволяет сделать вывод, что спектр помощи, оказываемой ИИ-ассистентом, весьма широк, и потенциально он может значительно повысить продуктивность разработчика.

Существует целый ряд популярных ИИ-ассистентов для промышленной разработки программного обеспечения. К наиболее известным зарубежным решениям можно отнести Cursor [6], Codex [7], GitHub Copilot [8], Gemini Code Assist [9], Grok Code Fast [10], watsonx [5]. Российские инструменты представлены такими решениями как SourceCraft Code Assistant [11] и GigaCode [12].

Ведутся исследовательские проекты по разработке ИИ-ассистентов для специализированных применений. В частности, в области параллельных вычислений можно отметить OMPar [13], AutoParLLM [14], HPC-Coder [15].

Когда дело доходит до высокой производительности (а в параллельных вычислениях это всегда так), специализированные решения для выбранной предметной области име-

ют преимущество перед универсальными системами, так как при их построении можно учесть специфические знания о предметной области и использовать эти знания для оптимизации. Далее рассматриваются некоторые особенности концепции активных знаний, которые могут лечь в основу специализированного ИИ-ассистента.

3. Концепция активных знаний и методы машинного обучения. Рассмотрим некоторые принципиальные возможности сочетания методов машинного обучения и концепции активных знаний.

ИИ-помощник в обучении и методическом сопровождении пользователей системы LuNA. Знакомство с новым языком и системой конструирования программ связано с преодолением порога вхождения, что включает знакомство с терминологией, освоение ключевых принципов и понятий, знакомство с инструментарием и сложившимися практиками. В этой области хорошо могут помочь ИИ-помощники (ассистенты, агенты), которые могут служить связующим звеном между текущим уровнем понимания пользователя и базовым пониманием системы LuNA. ИИ-помощник может помочь в интерпретации сообщений об ошибках, подсказать следующий шаг, ответить на вопрос пользователя, направить его в нужный раздел документации или предложить ознакомиться с публикацией по теме.

Создание базы активных знаний — это сложная задача, имеющая свою специфику в каждой предметной области. Этот процесс имеет свои ключевые этапы и методы выявления и решения задач — определение понятий и концептуализация предметной области, выделение фрагментов кода из имеющегося ПО, определение существенных нефункциональных свойств и способов их представления в базе активных знаний и т. д. ИИ-помощник может существенно помочь в этом процессе, выступив в качестве методического консультанта, проводя пользователя через эти этапы создания базы активных знаний. Отдаленно это можно сравнить с ИИ-консультантом по распараллеливанию последовательных программ, но в данном случае создается не параллельная программа, а база активных знаний, и создается она из существующего программного обеспечения и знаний экспертов в этой предметной области.

Автоматическое создание баз активных знаний. Методы машинного обучения могут использоваться для автоматического создания базы активных знаний из человеко-ориентированных текстов (например, учебников, статей, описаний алгоритмов и т. п.). В простом случае для этого формируется запрос, включающий фрагмент человеко-ориентированного текста, информации о том, что такое база активных знаний и как она представляется на конкретном языке. Так как концепция активных знаний содержит методологию создания баз активных знаний, то этой методологии при определенных ограничениях может следовать и искусственный интеллект. Это направление работ имеет большой потенциал к автоматизированному переводу знаний из пассивной в активную форму. Также в качестве исходного материала могут использоваться не только человеко-ориентированные тексты, но и существующее программное обеспечение.

ИИ-перевод из неформальной постановки задачи в формальную. Решение задач на основе готовой базы активных знаний требует формальной постановки задачи, которую требуется решить. Методы машинного обучения могут быть использованы для перевода постановки задачи из неформального вида в формальный. То же самое касается и интерпретации результатов (решения задачи) — оно может быть переведено языковой моделью в человеко-ориентированную форму, в том числе в ту терминологию, в которой задача была неформально поставлена. Также неформальная постановка задачи может быть

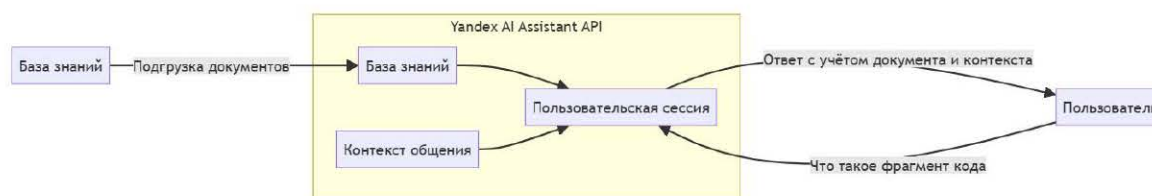


Рис. 1. Архитектура ИИ-ассистента

использована для поиска ИИ-помощником подходящей для этой задачи базы активных знаний.

Улучшение баз активных знаний. Методы машинного обучения, в частности, большие языковые модели, могут использоваться для анализа и улучшения баз активных знаний в том же ключе, как если бы это делал инженер знаний: выявление заведомо избыточных элементов в базе активных знаний, формирование сценариев тестирования работоспособности и эффективности конструируемых программ, определение дублирования и выявление тождественных понятий в различных базах активных знаний.

Также конструирование программ на основе баз активных знаний позволяет строить правильные программы (при условии отсутствия ошибок в самой базе активных знаний), но обеспечение качества конструируемой программы — это сложная оптимизационная проблема, требующая экспертных знаний. Методы машинного обучения могут использоваться для ее решения, имитируя анализ и принятие решений человека при ручной реализации этой же задачи. Построенное таким образом решение может оказаться не самым эффективным или неэффективным вовсе, но, по крайней мере, оно будет работать правильно. Во многих же случаях методы ИИ смогут находить таким образом вполне разумные способы решения задачи.

Подводя итоги разделу, можно сказать, что концепция активных знаний и машинное обучение имеют потенциал к совместному использованию. Базы активных знаний, базируясь на аксиоматических теориях, могут выступать в качестве гибкого строгого математического аппарата, позволяющего практически неограниченно накапливать знания, которые будут сохраняться неизменно во времени с возможностью из автоматического применения. Методы машинного обучения могут опираться на него, решая оптимизационные задачи в рамках заданных аксиоматических теорий и выступая связующим звеном с неформальным контекстом человека.

4. ИИ-ассистент для системы LuNA. Для системы LuNA первостепенной задачей представляется помощь в преодолении «входного барьера» при освоении и использовании концепции активных знаний. Поэтому в первой версии ИИ-ассистента LuNA (далее — LuNA AI) было решено сосредоточиться на задачах, связанных с работой с документацией.

Технически LuNA AI представляет собой сервис, который принимает от пользователя контекст (файлы, вопросы), при необходимости комбинирует эту информацию с имеющейся базой знаний (документацией, шаблонами, примерами), формирует промт, вызывает большую языковую модель (LLM) и отображает пользователю ответ LLM.

Для экономии времени на разработку LuNA AI построен с использованием Yandex AI Assistant API [16], который объединяет большую языковую модель YandexGPT и технологию RAG [17], добавляющую возможности поиска и извлечения релевантной информации из больших баз данных или документов.

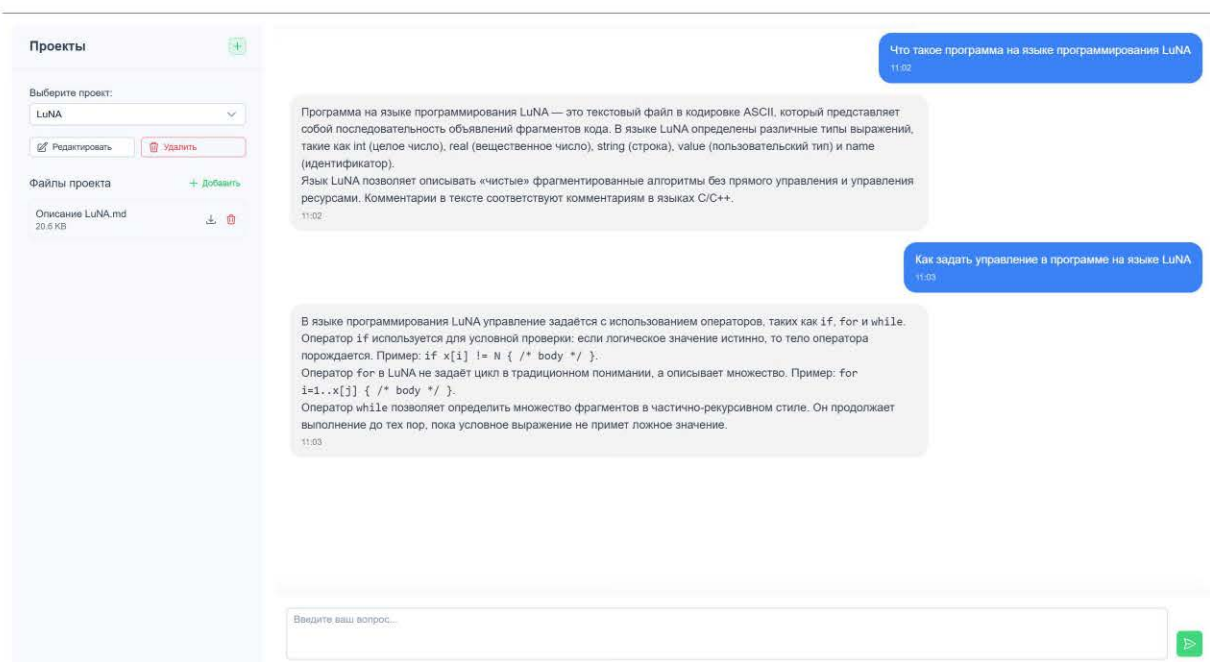


Рис. 2. Пример работы с LuNA AI

Для встраивания ИИ-ассистента в процесс разработки может создаваться отдельная IDE (как в случае с Cursor), ИИ-ассистент может являться плагином (GitHub Copilot или Gemini Code Assist), либо отдельным сервисом, основанным на универсальной (GPT-5.1, Gemini 3 Pro и др.) или специализированной для разработки программного кода (DeepSeek Coder, Qwen3-Coder и др.) модели. В последнем случае требуется дополнительная ручная работа: пользователь самостоятельно формирует запрос к модели, а затем, при необходимости, самостоятельно переносит результат в IDE. С учетом ориентации на задачи по работе с документацией, первая версия LuNA AI реализована в форме отдельного сервиса.

Реализована демонстрационная версия ИИ-ассистента, решающая часть задач по работе с документацией: пользователь может запрашивать информацию о системе LuNA, а ИИ-ассистент с использованием RAG строит релевантный ответ.

Архитектура сервиса представлена на рис. 1. На основе существующей документации по системе LuNA была сформирована база знаний и загружена в созданное в Yandex Cloud пространство через API. Запрос пользователя обрабатывается фронтенд-приложением, построенным с помощью фреймворка Vue и библиотеки NuxtUI, и передается посредством REST-запроса на бэкенд, разработанный на языке Python с применением FastAPI. Далее запрос векторизуется и вместе с наиболее релевантным контекстом из базы знаний подается на вход LLM.

Пример работы с LuNA AI приведен на рис. 2. Пользователь задает вопросы о системе программирования LuNA, а ИИ-ассистент дает ответы.

Для удобного обновления базы знаний на уровне UI реализовано понятие «проекта» — набора файлов, формирующего базу знаний. Проектов может быть несколько. Таким образом можно поддерживать несколько версий системы LuNA, которые значительно отличаются друг от друга.

Заключение. В работе принципиально рассмотрены концепция активных знаний и некоторые ИИ-подходы с точки зрения автоматического конструирования программ. Рас-

смотрено, как они могут сочетаться, дополняя друг друга. Представлена демонстрационная версия ИИ-ассистента, обеспечивающая работу с документацией по системе LuNA.

На текущий момент качество выдаваемых ИИ-ассистентом ответов не является оптимальным и требует дальнейшего улучшения, прежде всего, в части повышения качества базы знаний. Однако даже в таком варианте ИИ-ассистент способен оказывать существенную помощь тем, кто впервые знакомится с языком LuNA.

В настоящий момент ИИ-ассистент реализован как отдельный сервис. В дальнейшем планируется интегрировать его в популярную открытую среду разработки VS Code, а также расширить его функционал дополнительными возможностями.

Список литературы

1. Вальковский В. А., Малышкин В. Э. Синтез параллельных программ и систем на вычислительных моделях / Отв. ред. Котов В. Е.; АН СССР, Сиб. отд-ние, ВЦ. Новосибирск: Наука. Сиб. отд-ние, 1988. 126 с.
2. Malyshkin V. Active Knowledge, LuNA and Literacy for Oncoming Centuries // LNCS, Vol. 9465. 2015. С. 292–303. DOI: 10.1007/978-3-319-25527-9_19.
3. Malyshkin V. E., Perepelkin V. A. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Proceedings of the 11th International Conference on Parallel Computing Technologies (PaCT-2011), LNCS, T. 6873, 2011, С. 53–61. DOI: 10.1007/978-3-642-23178-0_5.
4. Arykov, S. Defining Order of Execution in Aspect Programming Language // LNCS. 2017. Vol. 10421. P. 255–261. ISBN 978-3-319-62932-2. DOI: 10.1007/978-3-319-62932-2_25.
5. Weisz et al. Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise // Proceedings of the Extended Abstracts of the CHI EA '25 Conference on Human Factors in Computing Systems. Article No. 673. P. 1–13. ISBN: 9798400713958. DOI: 10.1145/3706599.3706670.
6. Jiang et al. CursorCore: Assist Programming through Aligning Anything // CoRR. Vol. abs/2410.07002. 2024. ISSN 2331–8422. DOI: 10.48550/arXiv.2410.07002.
7. Mark Chen et al. Evaluating Large Language Models Trained on Code // CoRR. Vol. abs/2107.03374. 2021. ISSN 2331–8422. DOI: 10.48550/arXiv.2107.03374.
8. Pandey et al. Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects // CoRR. Vol. abs/2406.17910. 2024. ISSN 2331–8422. DOI: 10.48550/arXiv.2406.17910.
9. Gemini Code Assist | AI coding assistant. [Электронный ресурс]: <https://developers.google.com/gemini-code-assist/docs/overview> (дата обращения: 24.11.2025).
10. Grok Code Fast 1 Model Card. [Электронный ресурс]: <https://data.x.ai/2025-08-26-grok-code-fast-1-model-card.pdf> (дата обращения: 03.02.2025).
11. SourceCraft. Code Assistant. [Электронный ресурс]. <https://sourcecraft.dev/portal/code-assistant/> (дата обращения: 22.11.2025).
12. GigaCode. [Электронный ресурс]. <https://gigacode.ru/> (дата обращения: 22.11.2025).
13. Tal Kadosh et al. OMPAr: Automatic Parallelization with AI-Driven Source-to-Source Compilation // CoRR. Vol. abs-2409-14771. 2024. ISSN 2331-8422. DOI: 10.48550/arXiv.2409.14771.
14. Mahmud et al. AutoParLLM: GNN-Guided Automatic Code Parallelization using Large Language Models // CoRR. Vol. abs/2310.04047. 2023. ISSN 2331–8422. DOI: 10.48550/arXiv.2310.04047.
15. Nichols et al. HPC-Coder: Modeling Parallel Programs using Large Language Models // ISC High Performance 2024 Research Paper Proceedings (39th International Conference). Hamburg, Germany. 2024. P. 1–12. DOI: 10.23919/ISC.2024.10528929.

16. Yandex AI Assistant API. [Электронный ресурс]: <https://yandex.cloud/ru/docs/ai-studio/concepts/assistant/> (дата обращения: 24.11.2025).

17. Oche et al. A Systematic Review of Key Retrieval-Augmented Generation (RAG) Systems: Progress, Gaps, and Future Directions // CoRR. Vol. abs/2507.18910. 2025. ISSN 2331-8422. DOI: 10.48550/arXiv.2507.18910.



Перепелкин Владислав Александрович — кандидат технических наук, старший научный сотрудник Института вычислительной математики и математической геофизики СО РАН; доцент кафедры параллельных вычислений факультета информационных технологий Новосибирского государственного университета. Тел.: (383) 330-89-94, e-mail: perepelkin@sssd.sccc.ru. В 2008 году окончил Новосибирский государственный университет с присуждением степени магистра техники и технологии по направлению «Информатика и вычислительная техника». В 2023 году защитил кандидатскую диссертацию. Имеет более 20 опубликованных статей по теме автоматизации конструирования параллельных программ в области численного моделирования. Является одним из основных разработчиков экспериментальной системы автоматического конструирования численных параллельных программ для мультикомпьютеров LuNA (от Language for Numerical Algorithms). Область профессиональных интересов включает автоматизацию конструирования параллельных программ, языки и системы параллельного программирования, высокопроизводительные вычисления.

Perepelkin Vladislav Aleksandrovich — PhD in Computer Science. Graduated from Novosibirsk State University in 2008 with the master degree in engineering and technology in computer science. Defended his PhD thesis in 2023. Nowadays has the senior researcher position at the Institute of Computational Mathematics and Mathematical Geophysics of the Siberian Branch of Russian Academy of Sciences, and also has a part time associated professor position in the Novosibirsk State University. He is an author of

more than 20 papers on automation of numerical parallel programs construction. He is one of main developers of system LuNA (Language for Numerical Algorithms) for automatic construction of numerical parallel programs. Professional interests include automation of parallel programs construction, languages and systems of parallel programming, high performance computing.



Арыков Сергей Борисович — кандидат физико-математических наук, младший научный сотрудник Института вычислительной математики и математической геофизики СО РАН; доцент кафедры параллельных вычислительных технологий факультета прикладной математики и информатики Новосибирского государственного технического университета. E-mail: arykov@sscc.ru. Окончил Новосибирский государственный университет. Автор более 20 научных работ. Область профессиональных интересов включает автоматизацию конструирования параллельных программ, языки и системы параллельного программирования, технологии искусственного интеллекта.

Sergey Arykov — PhD in Computer Science, a junior researcher at the Institute of Computational Mathematics and Mathematical Geophysics of the Siberian Branch of the Russian Academy of Sciences. He is also an associated professor at the Department of Parallel Computing Technologies, Faculty of Applied Mathematics and Computer Science, Novosibirsk State Technical University. E-mail: arykov@sscc.ru. Graduated from Novosibirsk State University. The author of more than 20 scientific publications. Professional interests include the automation of parallel program design, parallel programming languages and systems and artificial intelligence technologies.

Дата поступления — 04.02.2026

Правила представления и подготовки рукописей для публикации в журнале „ПРОБЛЕМЫ ИНФОРМАТИКИ“

Общие требования.

Редакция принимает к рассмотрению статьи в электронном виде (*исходный файл в L^AT_EX* и файл PDF, с приложением оригиналов рисунков в формате тех программ, в которых они были сделаны, отдельными файлами).

Файлы, содержащие текст статьи, иллюстрации и дополнительные материалы, можно пересылать на электронный адрес редакции: **problem-info@sscc.ru**.

Принимаются файлы, архивированные архиваторами ZIP/7Z или RAR; применение самораспаковывающихся архивов не допускается.

При повторной отправке материалов, а также при внесении в исходный текст дополнений или исправлений необходимо сообщить об этом в редакцию в тексте электронного письма.

Направляя статью в редакцию журнала, автор (соавторы) на безвозмездной основе передает(ют) издателю на срок действия авторского права по действующему законодательству РФ исключительное право на использование статьи или отдельной ее части (в случае принятия редколлегией Журнала статьи к опубликованию) на территории всех государств, где авторские права в силу международных договоров Российской Федерации являются охраняемыми, в том числе следующие права: на воспроизведение, на распространение, на публичный показ, на доведение до всеобщего сведения, на перевод на иностранные языки и переработку (и исключительное право на использование переведенного и (или) переработанного произведения вышеуказанными способами), на предоставление всех вышеперечисленных прав другим лицам.

Журнал „Проблемы информатики“ является некоммерческим изданием. Плата с авторов за публикацию статей не взимается.

К статье должны быть приложены:

- **разрешение на публикацию** от экспертного совета организации, в которой выполнена работа (для авторов из России);

- **оригинал рецензии;**

портретные фотографии авторов разрешением не менее 300 dpi.

- Блоки информации и на русском, и на английском языках просьба присылать отдельными файлами:

- **Название** статьи;

- **Инициалы и фамилии** авторов;

- **Места работы** авторов: полное наименование организации, почтовый индекс, город, страна;

- **Код(ы) классификации УДК;**

- **Аннотации**, содержащие краткую постановку задачи и описание метода решения: на русском языке объемом не более 1000 знаков, на английском языке расширенную, объемом от 4000 до 8000 знаков, что соответствует требованиям ВАК и Scopus.

- **Ключевые слова;**

- **Списки используемой литературы** в соответствии с ГОСТ Р 7.0.5—2008 (в английской версии необходимо выполнить **транслитерацию** неанглоязычных элементов списка литературы в соответствии с ГОСТ Р 7.0.34-2014) — составляются по ходу упоминания источников в тексте;

- **Краткие биографии (БИО)** авторов с указанием ключевых научных достижений (включая ученую степень, ученое звание — при наличии; основные области научных интересов и формулировку основных результатов, место работы, занимаемую должность, контактные данные — почтовый адрес с индексом, адрес электронной почты, контактный телефон).

Подготовка статьи.

1. Материал статьи должен быть изложен в следующей последовательности:

1.1. название статьи на английском языке;

1.2. инициалы и фамилия автора(ов) на английском языке;

1.3. место работы автора(ов) (на английском языке): полное наименование организации, индекс, город, страна;

1.4. англоязычная аннотация;

1.5. ключевые слова на английском языке;

1.6. references+транслитерация неанглоязычных элементов списка литературы;

1.7. название статьи на русском языке;

1.8. инициалы и фамилии и авторов;

1.9. место работы авторов: полное наименование организации, почтовый индекс, город, страна;

1.10. индекс УДК;

1.11. аннотация на русском языке;

- 1.12. ключевые слова (не более 8);
- 1.13. текст статьи;
- 1.14. список литературы, оформленный в соответствии с требованиями ГОСТ;
- 1.15. краткие биографии авторов на английском и русском языках с указанием ключевых научных достижений (ученую степень, ученое звание — при наличии; место работы, занимаемую должность, контактные данные — почтовый адрес с индексом, адрес электронной почты, контактный телефон, основные области научных интересов и формулировка основных результатов).

2. Требования к формулам:

- Нумерация формул сквозная, выносные формулы центрируются, номер выровнен по правому краю.

3. Требования к рисункам:

- Файлы с рисунками присылаются отдельно в формате программ, в которых они были выполнены: в формате MS Excel (для графиков и диаграмм), eps, pdf, png, tiff, bmp или jpeg (с максимальным качеством).

- Рисунки с подрисуночными подписями завершаются в текст статьи.
- Тексты, являющиеся частью рисунка, выполняются шрифтом TimesNewRoman.
- Фотографии должны иметь разрешение не менее 300 dpi.

4. Дополнительные требования:

- В текст статьи необходимо включать ссылки на рисунки и таблицы, а также подрисуночные подписи и заголовки таблиц. Все буквенные обозначения, приведенные на рисунках, необходимо пояснить в основном тексте или в подрисуночных подписях.

- Сокращения слов не допускаются (кроме общепринятых).

- Векторные переменные обозначаются полужирным шрифтом без курсива.

- Таблицы не должны быть громоздкими. Значения физических величин в таблицах, на графиках и в тексте должны указываться в единицах измерения СИ.

- Графики, если их на рисунке несколько, а также отдельные детали на чертежах, узлы и линии на схемах следует обозначать цифрами, набранными курсивом.

- Нумеровать следует только те формулы и уравнения, на которые имеются ссылки в тексте, нумерация сквозная.

- Ссылки на источники в тексте заключаются в квадратные скобки.

- Иностранные источники приводятся на языке оригинала. Ссылки на неопубликованные работы не допускаются.

Все статьи, опубликованные в журнале «Проблемы информатики», доступны на сайте https://elibrary.ru/title_about.asp?id=30275 и на сайте журнала <http://problem-info.sssc.ru> спустя год после опубликования.

Пример оформления статей можно посмотреть на сайте журнала <http://problem-info.sssc.ru>.