ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

ВЕСТНИК ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

BULLETIN OF THE SOUTH URAL STATE UNIVERSITY

СЕРИЯ

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

2026, том 15, № 2

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2026, volume 15, no. 2

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2026
Т. 15, № 2

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2026
Vol. 15, no. 2

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents

УДК 004.421, 519.688 DOI: 10.14529/cmse260201

ОПИСАНИЕ И АНАЛИЗ СВОЙСТВ, ОСОБЕННОСТЕЙ И ХАРАКТЕРИСТИК АЛГОРИТМОВ В ОТКРЫТОЙ ЭНЦИКЛОПЕДИИ СВОЙСТВ АЛГОРИТМОВ ALGOWIKI*

© 2026 А.С. Антонов

Научно-иссследовательский вычислительный центр
Московского государственного университета имени М.В. Ломоносова
(119234 Москва, ул. Ленинские горы, д. 1, стр. 4)
E-mail: asa@parallel.ru

Поступила в редакцию: 16.03.2026

Исследование свойств алгоритмов является важнейшим этапом в процессе их эффективной реализации на высокопроизводительных вычислительных системах. В данной статье рассматриваются свойства, особенности и характеристики алгоритмов, которые могут быть полезны для проведения такого анализа. Рассмотрены необходимые предварительные шаги, которые нужно выполнить для приведения алгоритмов к виду, в котором их можно анализировать и сравнивать между собой. Выделяются аналитические характеристики алгоритмов, то есть те свойства, которые могут быть описаны в числовом или формульном виде, при этом акцент делается на свойствах, связанных с параллелизмом. Многие из рассматриваемых свойств именно для алгоритмов предлагаются впервые, в то же время являясь аналогичными свойствам, которые принято рассматривать для программных реализаций. Все рассматриваемые свойства иллюстрируются несколькими примерами для хорошо известных алгоритмов, таких как суммирование элементов вектора, скалярное произведение двух векторов, перемножение двух плотных квадратных матриц и метод Гивенса (вращений) QR-разложения квадратной матрицы. В дальнейшем на базе исследования рассмотренных свойств предполагается решать задачи суперкомпьютерного кодизайна по совместному анализу свойств алгоритмов, программных реализаций и суперкомпьютерных систем.

Ключевые слова: алгоритм, параллелизм, суперкомпьютер, кодизайн, вычислительная сложность, параллельная сложность, ресурс параллелизма, ускорение, граф информационных зависимостей, AlgoWiki.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Антонов А.С. Описание и анализ свойств, особенностей и характеристик алгоритмов в Открытой энциклопедии свойств алгоритмов AlgoWiki // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2026. Т. 15, № 2. С. 5–25. DOI: 10.14529/cmse260201.

Введение

Для описания структуры вычислительных алгоритмов могут применяться разные методы. Подробно они были рассмотрены в статье [1]. Наиболее часто на практике используются следующие методы: словесное описание, блок-схемы, описания на традиционных языках программирования или с использованием псевдокода.

Описывать свойства параллельных алгоритмов оказывается еще сложнее [2]. Часто свойства алгоритмов дают только в словесном описании, без четких определений и объяснения их сущности. Такие описания не могут использоваться для автоматического анализа. Для большей строгости описания информационной структуры алгоритма часто используют графовые модели, в частности, граф информационных зависимостей или граф алгоритма [3] (см. раздел 1.6). Полезно бывает визуализировать графы информационных

*Статья рекомендована к публикации Программным комитетом Всероссийской научной конференции с международным участием «Параллельные вычислительные технологии (ПаВТ) 2026».

зависимостей, для чего можно использовать, например, систему AlgoView [4, 5], требующую описания информационной структуры анализируемых алгоритмов на специальном языке Algolang [6].

Наиболее проработанное описание структуры алгоритмов используется в Открытой энциклопедии свойств алгоритмов AlgoWiki [7]. Предложенная в ней структура описания вычислительного алгоритма [8] включает следующие разделы:

1. Свойства и структура алгоритмов
   - 1.1 Общее описание алгоритма
   - 1.2 Математическое описание алгоритма
   - 1.3 Вычислительное ядро алгоритма
   - 1.4 Макроструктура алгоритма
   - 1.5 Схема реализации последовательного алгоритма
   - 1.6 Последовательная сложность алгоритма
   - 1.7 Информационный граф
   - 1.8 Ресурс параллелизма алгоритма
   - 1.9 Входные и выходные данные алгоритма
   - 1.10 Свойства алгоритма
2. Программная реализация алгоритма
   - 2.1 Особенности реализации последовательного алгоритма
   - 2.2 Возможные способы и особенности параллельной реализации алгоритма
   - 2.3 Результаты прогонов
   - 2.4 Выводы для классов архитектур
3. Литература

Описание алгоритмов по этой схеме включает множество полезных свойств и является весьма содержательным. Однако энциклопедия AlgoWiki реализуется в виде wiki-проекта [9] на базе движка MediaWiki [10]. Использование идеологии wiki-проектов дает большие возможности по привлечению авторов к работе по совместному наполнению контента сайта, но приводит к тому, что вся информация хранится в слабо структурированном текстовом виде, практически непригодном для проведения автоматического анализа.

Поэтому, когда встает задача автоматически анализировать свойства описанных в проекте алгоритмов, приходится частично или полностью отходить от идеологии wiki-проектов. Отдельные страницы энциклопедии AlgoWiki, например, основная страница классификации алгоритмов [11], реализующая хорошо известную схему описания предметной области в виде цепочек «задача–метод–алгоритм–реализация» [12, 13], уже реализованы без использования wiki-технологии [9].

Первый шаг по направлению к автоматическому исследованию свойств алгоритмов в рамках энциклопедии AlgoWiki был сделан, когда на страницах описаний алгоритмов стали делать выноску с несколькими базовыми свойствами. Пример выноски с базовыми свойствами метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] приведен на рис. 1.

На таких выносках стали размещать как наиболее важные свойства последовательных алгоритмов (последовательная сложность, объем входных и выходных данных), так и некоторые характеристики, связанные с параллелизмом (высота и ширина ярусно-параллельной формы). Так эти свойства стали более заметными, однако решением вышеописанной проблемы это не является, потому что данные выноски формируются на базе текстовых описаний

Метод Гивенса (вращений) QR-разложения квадратной матрицы	
Последовательный алгоритм	
Последовательная сложность	$2n^3$
Объём входных данных	n^2
Объём выходных данных	n^2
Параллельный алгоритм	
Высота ярусно-параллельной формы	$11n - 16$
Ширина ярусно-параллельной формы	$O(n^2)$

Рис. 1. Пример выноски с базовыми свойствами алгоритма

с использованием механизма шаблонов [15]. При этом сами данные по-прежнему являются плохо структурированными и не подходят для автоматического анализа.

Решением, подходящим для автоматического анализа, является ввод данных для подобных выносок путем заполнения полей специальных веб-форм. Лучше всего, если поля форм в этом случае заполняются путем выбора из заранее подобранных фиксированных вариантов. Чтобы это стало возможно, требуется строго описать те свойства, которые важны для описания характеристик алгоритмов и будут в дальнейшем использоваться для их автоматического анализа, чему и посвящена данная статья. Для вариантов числовых параметров, характеризующих эти свойства, нужно зафиксировать варианты наиболее часто используемых на практике их значений.

В разделе 1 данной статьи рассмотрены подходы к описанию и анализу свойств, особенностей и характеристик алгоритмов, позволяющие выделить алгоритмы, которые можно автоматически оценивать и сравнивать между собой на основе вводимых в разделе 2 характеристик.

1. Подходы к описанию и анализу свойств, особенностей и характеристик алгоритмов

1.1. Оценки в лучшем, в худшем и в среднем случае

Многие численные характеристики алгоритмов существенно зависят от заданных входных данных. Для одних их значений алгоритм может работать значительно дольше, чем для других. Поэтому анализ обычно «заключается в нахождении минимального (для оптимистов), максимального (для пессимистов), среднего (для специалистов по теории вероятностей) значения» [16] (здесь автор писал об оценке времени исполнения алгоритма). Для произвольных характеристик часто приводят оценки в лучшем, худшем и среднем случае.

Про получение оценок в лучшем, худшем и среднем случае пишут многие другие авторы книг по исследованию свойств алгоритмов [17–22]. Оценки в худшем случае позволяют гарантировать, что при отсутствии других важных действующих факторов значения изучаемых характеристик в реализации не окажутся хуже этих оценок. Оценки в лучшем случае показывают, к чему может приблизиться рассматриваемая характеристика на са-

мом удачном наборе входных данных. Но разница между худшим/лучшим случаем и тем, что получается на практике, иногда может оказаться весьма существенной. Оценки в среднем случае несут в себе некоторые предположения о значениях реальных входных данных. Однако получение оценок в среднем случае не всегда осмысленно и не всегда приводит к правильным результатам. Заметим также, что есть разные подходы к тому, что в конкретном случае считать средним.

1.2. Точные оценки и асимптотические оценки

Практически во всех книгах по анализу свойств алгоритмов (например, [16, 18, 20, 22–28]) для оценок ключевых характеристик используют не их абсолютные значения, а оценивают их через функцию от входных данных задачи, зачастую приводя ее асимптотику, чаще всего в формате «О большое».

Строгое определение этого понятия приводится, например, в книге [16]: «Запись $O(f(n))$ всегда обозначает следующее: существуют положительные константы M и n_0, такие, что величина x_n, представленная в виде $O(f(n))$, удовлетворяет условию $|x_n| \leq M|f(n)|$ для всех целых $n \geq n_0$».

«О большое» позволяет оценить порядок рассматриваемой величины, в некоторых источниках называется также «скорость роста (rate of growth)» или «порядок роста (order of growth)» [18, 21].

Использование семантики «О большого» во многих случаях дает возможность заметно упростить оценки характеристик алгоритмов, но оставить возможность их асимптотического сравнения. Например, автор страницы «Метод сдваивания Стоуна для решения двудиагональных СЛАУ» [29] в энциклопедии AlgoWiki приводит оценку последовательной сложности алгоритма как $3(n-1)(\lceil \log_2(n-1) \rceil + 2)$. Такая формула сложна как просто для восприятия, так и для сравнения с другими алгоритмами. Приведение оценки к семантике «О большого» даст оценку $n \log_2 n$, что уже значительно проще и может использоваться в дальнейшем анализе.

1.3. Выделение главного параметра

Вообще говоря, алгоритмы могут иметь несколько входных параметров. При построении формул для их характеристик получаются многопараметрические функции, которые анализировать значительно сложнее; также существенно усложняется сравнение этих характеристик для различных используемых алгоритмов. Во многих источниках анализируют только простые однопараметрические алгоритмы.

В классической книге [30] сказано: «Нам хотелось бы связать с каждой конкретной задачей некоторое число, называемое ее размером, которое выражало бы меру количества входных данных». В книге [27] это описано более детально: «Для простоты входные данные представляются параметром n. Этот параметр пропорционален величине обрабатываемого набора данных и может обозначать:

- размер массива или файла при сортировке или поиске;
- степень полинома;
- количество символов в строке;
- другую абстрактную меру объема рассматриваемой задачи».

Выделение главного параметра рассматривается, в частности, также в работе [22].

Некоторые многопараметрические алгоритмы, не теряя общности с точки зрения оценки их характеристик, можно свести к однопараметрическим вариантам. Например, во многих случаях, когда в алгоритмах используются прямоугольные матрицы размера $m \times n$ для анализа характеристик таких алгоритмов зачастую достаточно рассмотреть частный случай квадратной матрицы размера $n \times n$.

Сведение многопараметрических алгоритмов к однопараметрическим вариантам зачастую может оказаться неприемлемым, но анализ и сравнение многопараметрических функций, описывающих характеристики таких алгоритмов, является сложной задачей, к которой можно будет вернуться в будущем, если возникнет такая необходимость.

1.4. Выделение основного типа данных

Почти в любом алгоритме операции выполняются над данными, которые в конкретных реализациях описываются объектами каких-то типов, используемых в языке программирования. В одной программной реализации для разных объектов могут использоваться данные разных типов, но обычно можно выделить один основной тип данных, над которыми выполняется большинство наиболее важных операций. Далее все основные характеристики алгоритма оцениваются с учетом операций только над данными этого типа.

Так, общепринятой практикой является при решении задачи суммирования элементов вектора вещественных чисел учитывать при оценках характеристик алгоритма только операции над самими вещественными числами, хотя очевидно, что для реализации этого алгоритма на многих распространенных языках программирования потребуются дополнительные операции над вспомогательными переменными (чаще всего целочисленными) для организации цикла, в котором выполняется суммирование элементов вектора. Обычно считается, что подобными операциями можно пренебречь, к тому же этих операций нет в самом алгоритме, они появляются только в его программной реализации.

1.5. Выделение значимых операций

Шаги любого алгоритма могут состоять из самых разных операций (арифметических, логических, операций чтения/записи данных, пересылки данных и т.д.) Для построения простых оценок характеристик алгоритма нужно выделить из этих операций самые важные с точки зрения анализа свойств алгоритма [28]. В [22] сказано: «Первый шаг при анализе алгоритма состоит в определении абстрактных операций, на которых основан алгоритм, чтобы отделить анализ от реализации». Это важное замечание, потому что программные реализации алгоритма могут содержать новые операции, которых непосредственно алгоритм не предусматривает. Некоторые из таких операций (чтения/записи, пересылки данных и т.п.) могут играть решающую роль для эффективности выполнения программы, но не являются операциями самого реализуемого алгоритма, и поэтому не должны входить в оценки его характеристик. Задача совместного анализа алгоритма, программной реализации и целевой вычислительной системы с учетом характеристик всех этих трех сущностей является более сложной и решается в рамках проекта по решению задачи суперкомпьютерного кодизайна [31].

На практике при подсчете числовых характеристик алгоритмов обычно ограничиваются одним основным набором операций. Часто для простоты предполагают, что операции из одного набора примерно равноценны по времени выполнения. Во многих численных алгоритмах таким основным набором является подмножество реализуемых арифметических

операций. Характеристики не численных алгоритмов оцениваются с использованием тех типов операций, на которые они ориентированы. Так, например, в алгоритмах сортировки и в алгоритмах на графах основными элементарными операциями являются операции сравнения, перестановки и т.д.

Если алгоритм описан на основе макроопераций (и его можно назвать в этом случае макроалгоритмом), то подсчитывать его численные характеристики можно либо в числе этих макроопераций (но тогда его, вероятно, будет сложно сравнивать с другими алгоритмами), либо предварительно расшифровывая макрооперации в наборы элементарных операций.

1.6. Граф информационных зависимостей

Одним из основных подходов при анализе свойств алгоритмов, особенно связанных с параллелизмом, является построение и исследование *графа информационных зависимостей* (называемого также *графом алгоритма*) [3]. Это ориентированный ациклический граф, вершины которого соответствуют срабатываниям операций алгоритма, а дуги — информационным зависимостям между ними. Состав операций, соответствующих вершинам графа, может быть разным — от элементарных операций до макроопераций (в этом случае используют понятие *макрографа информационных зависимостей*).

Часто используемым представлением графа информационных зависимостей является его *ярусно-параллельная форма (ЯПФ)* (*parallel form*) — такое представление графа алгоритма, в котором:

- все вершины разбиты на перенумерованные подмножества ярусов;
- начальная вершина каждой дуги расположена на ярусе с номером меньшим, чем номер яруса конечной вершины;
- между вершинами, расположенными на одном ярусе, не может быть дуг.

Под *высотой ЯПФ* понимают число ее ярусов. *Ширина яруса* — это число вершин, расположенных на данном ярусе, а *ширина ЯПФ* — это максимальная ширина ярусов в ЯПФ.

Канонической ярусно-параллельной формой называется ЯПФ, высота которой на единицу больше длины критического пути, а все входные вершины расположены на первом ярусе. Для заданного графа информационных зависимостей его каноническая ЯПФ единственна [3].

Для построения графов информационных зависимостей и получения их канонической ярусно-параллельной формы можно использовать, например, систему AlgoView [4, 5]. На рис. 2 показан макрограф информационных зависимостей для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант), приведенный на странице описания данного алгоритма в энциклопедии AlgoWiki [14]. На графе ярко-оранжевым цветом помечены вершины, относящиеся к текущему ярусу канонической ярусно-параллельной формы, темно-зеленым цветом — вершины, относящиеся к предыдущим ярусам, светло-зеленым цветом — вершины, относящиеся к последующим ярусам.

2. Аналитические характеристики алгоритмов

Разные авторы выделяют разные характеристики алгоритмов. Так, Дональд Кнут (Donald Knuth) считает необходимыми такие их особенности, как конечность, определенность, ввод, вывод и эффективность [16].

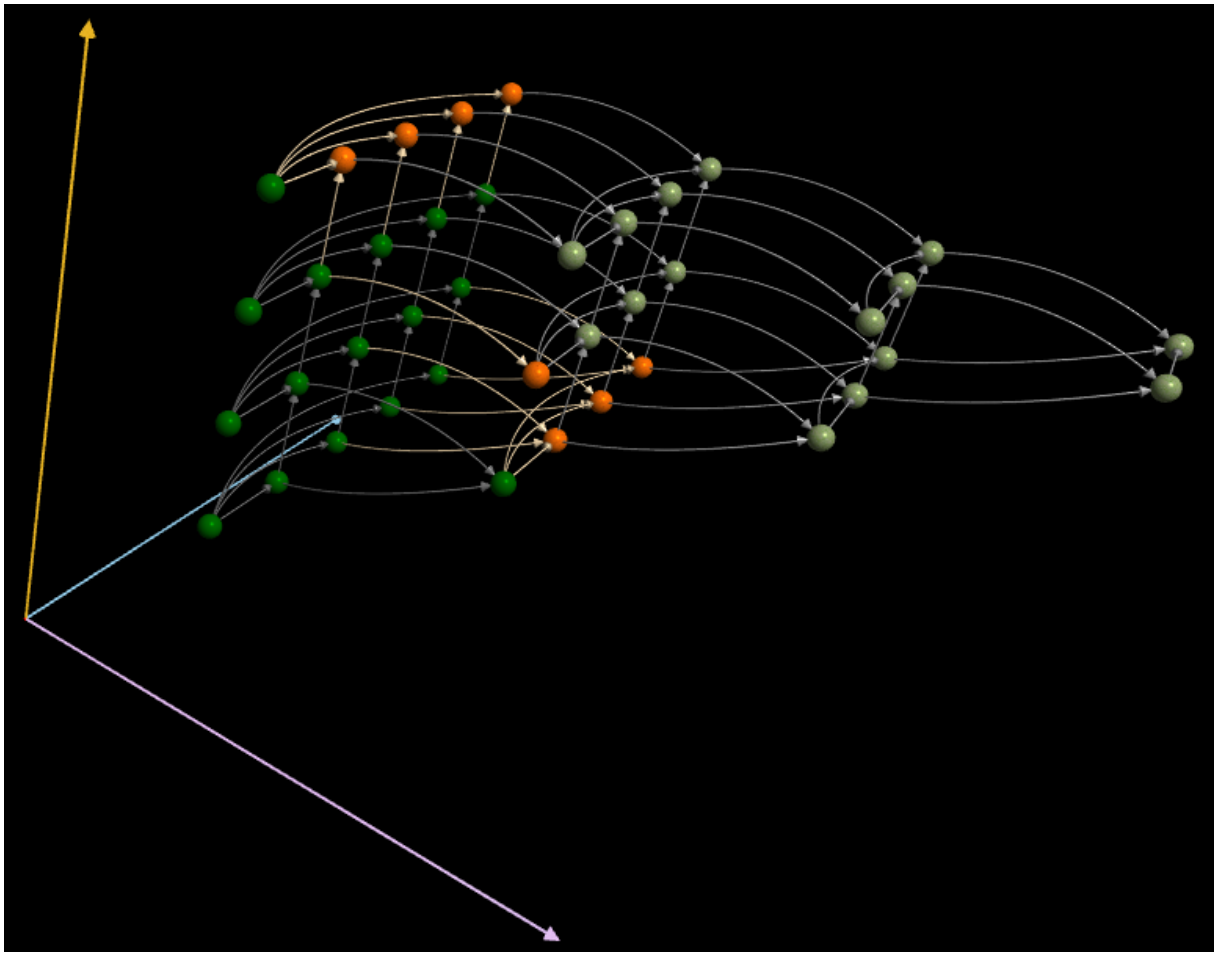


Рис. 2. Пример графа информационных зависимостей с выделением одного яруса канонической ЯПФ

В другом источнике [27] написано: «Алгоритмы должны обладать следующими формальными свойствами:

- понятность;
- дискретность;
- детерминированность;
- результативность;
- массовость».

В книге [18] сформулировано: «Хороший алгоритм должен обладать тремя свойствами: быть корректным, эффективным и легкореализуемым».

Таким образом, каждый автор выделяет свой набор ключевых характеристик алгоритмов, зачастую не давая строгих определений этим характеристикам. И даже в случае, если характеристика имеет четкое определение, она зачастую является описательной и не может быть просто охарактеризована числовым параметром.

Если же ставить цель автоматического или автоматизированного анализа свойств алгоритмов, то эти свойства должны быть четко описаны и измеримы. Основную идею анализа алгоритмов Д. Кнут [16] описывает как «взять конкретный алгоритм и определить его количественные характеристики». Целевой функцией такого анализа должна быть потенциальная возможность предсказания динамических характеристик алгоритма при его

реализации на целевой вычислительной системе: «Анализ алгоритма заключается в том, чтобы предсказать требуемые для его выполнения ресурсы» [21].

2.1. Детерминированность и безусловность

Важным свойством алгоритмов является их детерминированность. *Детерминированный алгоритм* — это алгоритм с постоянной структурой вычислительного процесса, который выдает уникальный и предопределенный результат для заданных входных данных. National Institute of Standards and Technology (NIST) в словаре Dictionary of Algorithms and Data Structures [32] определяет понятие детерминированный алгоритм как «Алгоритм, поведение которого можно полностью предсказать на основе входных данных (An algorithm whose behavior can be completely predicted from the input)». В книге [27] понятие детерминированности поясняется так: «Каждое правило алгоритма должно быть четким, однозначным и выдавать для одних исходных данных всегда один и тот же результат».

В книге [21] проводится четкое разделение между детерминированными и вероятностными (randomized) алгоритмами: «В общем случае алгоритм называется рандомизированным (randomized), если его поведение определяется не только набором входных величин, но и значениями, которые выдает генератор случайных чисел». То же самое разделение проводится и в книге [20]: «Алгоритмы, не использующие рандомизацию, называются детерминированными (Algorithms not using randomization are called deterministic)», а также в книге [18].

Однако не для всех детерминированных алгоритмов можно легко получить рассматриваемые в последующих пунктах характеристики. Так, итерационные алгоритмы (по определению зачастую являющиеся детерминированными) обычно требуют выполнения своих итераций до достижения определенной точности. Для одной такой итерации определение характеристик обычно возможно, но количество итераций заранее неизвестно и существенно зависит от входных данных. Для подобных алгоритмов получение подобных характеристик сильно затруднено и требует отдельного изучения.

В книге [3] дано следующее определение детерминированности: «Если программа не имеет условных операторов, то как ее саму, так и описанный ею алгоритм будем называть детерминированными. В противном случае будем называть их недетерминированными». Напрямую данное определение противоречит перечисленным выше классическим вариантам, но может быть основой для определения класса *безусловных* (*branchless*) алгоритмов. Итерационные алгоритмы не войдут в класс безусловных алгоритмов, поскольку для определения необходимости следующей итерации требуется проверка условия на достижение необходимой точности. Перечисленные в последующих пунктах характеристики рассматриваются для алгоритмов, принадлежащих к этому классу.

2.2. Вычислительная (последовательная) сложность

Под *вычислительной сложностью* (*computational complexity*) задачи W понимают количество основных вычислительных шагов лучшего последовательного алгоритма, необходимых для решения задачи на одном процессе. Поскольку дается характеристика последовательной версии алгоритма, то вычислительную сложность часто называют также *последовательной сложностью*.

Не всегда очевидно, что понимать под основными вычислительными шагами алгоритма. Это могут быть как выделенные значимые операции (см. раздел 1.5), так и более крупные

шаги, состоящие из множества элементарных операций. Набор используемых для оценки операций определяется как особенностями самого рассматриваемого алгоритма, так и теми целями, которые ставятся при его анализе.

Вычислительная сложность условно характеризует время, требуемое для последовательной реализации алгоритма, поэтому многие авторы книг по свойствам алгоритмов [17, 22, 23, 28, 33] говорят непосредственно об оценке времени исполнения. При этом надо понимать, что это условное время выражается не в секундах, а в количестве основных вычислительных шагов. Если для простоты делается допущение, что любой основной вычислительный шаг выполняется за единицу времени, то эти две величины совпадают. В книге [21] явно говорится «время работы алгоритма ... измеряется в количестве элементарных операций». В книгах [27, 30] используется термин временная сложность.

В ряде источников [24, 26, 34] вычислительную сложность соотносят с выполняемой работой (work), Д. Кнут [16] описывает ее как «ожидаемую скорость выполнения алгоритма». Иногда разделяют понятия трудоемкости алгоритма, под которой понимают точную оценку количества выполняемых операций, и сложность алгоритма, дающую асимптотическую (см. раздел 1.2) оценку [19].

Вычислительная сложность, как и многие другие рассматриваемые далее характеристики алгоритмов, обычно является некоторой функцией от размера входных данных рассматриваемого алгоритма, выделенного в качестве главного параметра (см. раздел 1.3).

Эту и последующие характеристики покажем на трех простых примерах: алгоритме суммирования элементов вектора длины n, алгоритме скалярного произведения двух векторов длины n и алгоритме перемножения двух плотных квадратных матриц размера $n \times n$. На рис. 3 показаны графы информационных зависимостей трех рассматриваемых алгоритмов. На графах квадратные (кубические) вершины соответствуют операциям перемножения двух чисел, а шарообразные вершины — операциям суммирования двух чисел.

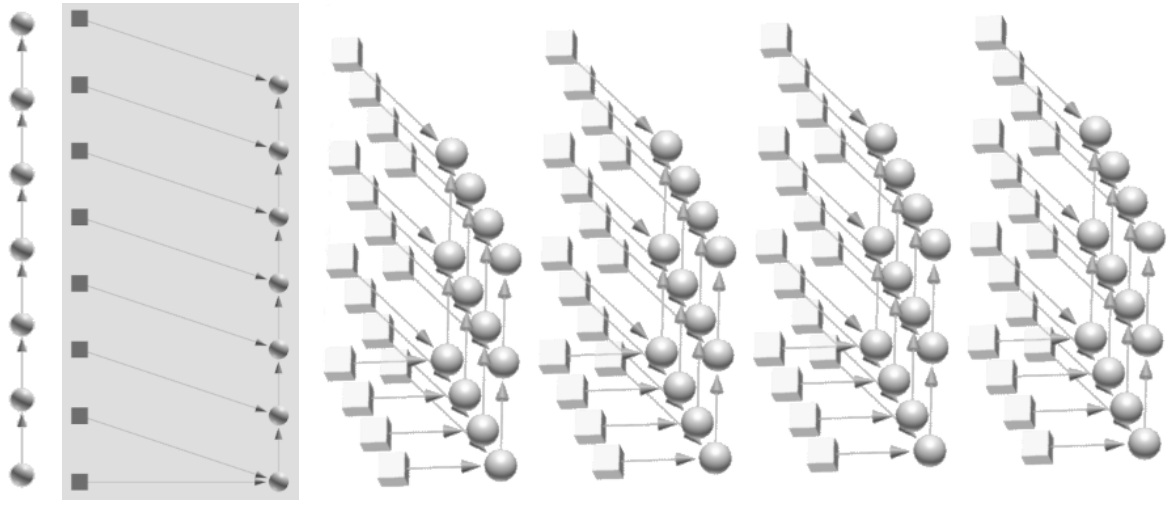


Рис. 3. Графы информационных зависимостей трех алгоритмов:
а) суммирования элементов вектора (для $n = 8$);
б) скалярного произведения двух векторов (для $n = 8$);
в) перемножения двух плотных квадратных матриц (для $n = 4$)

Вычислительная сложность алгоритма суммирования элементов вектора длины n, выраженная в количестве суммирований двух чисел, равна $W = n - 1$, вычислительная сложность алгоритма скалярного произведения двух векторов длины n при условии, что операции суммирования и перемножения двух чисел считаем равнозначными, составляет $W = 2n - 1$, а вычислительная сложность алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равна $W = n^2(2n - 1)$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] автор описания приводит оценку последовательной сложности как $W = 2n^3$, что после приведения к семантике «О большого» (см. раздел 1.2) можно было бы записать в виде $O(n^3)$.

2.3. Параллельная сложность

Под *параллельной сложностью* (*parallel complexity*) алгоритма W_p понимают количество основных вычислительных шагов, за которое можно выполнить данный алгоритм в предположении доступности неограниченного числа необходимых вычислительных ресурсов (процессоров, функциональных устройств, вычислительных узлов, ядер и т.п.). Встречаются различные варианты названия этой характеристики: параллельная сложность (parallel complexity) [24], глубина вычислительной сети (depth of the computational network) [33], глубина (depth) [26], вычислительные шаги (computational steps) [17].

Параллельная сложность алгоритма соответствует высоте канонической ярусно-параллельной формы его графа информационных зависимостей (см. раздел 1.6), а также на единицу больше длины критического пути в данном графе, и характеризует минимально возможное число основных вычислительных шагов при параллельном исполнении рассматриваемого алгоритма.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Параллельная сложность алгоритма суммирования элементов вектора длины n равна последовательной сложности, т. е. $W_p = W = n - 1$ (алгоритм является строго последовательным; схема сдваивания и другие подобные методы изменяют информационную структуру, а значит, являются другими алгоритмами), параллельная сложность алгоритма скалярного произведения двух векторов длины n составляет $W_p = n$ (один шаг на все попарные произведения и $n-1$ шаг на суммирование), а параллельная сложность алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ тоже равна $W_p = n$ (все элементы результирующей матрицы можно вычислять параллельно, для вычисления каждого нужно выполнить скалярное произведение соответствующих строки и столбца исходных матриц).

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] автор описания приводит оценку параллельной сложности как $W_p = 11n - 16$, что после приведения к семантике «О большого» (см. раздел 1.2) можно было бы записать в виде $O(n)$.

2.4. Ресурс параллелизма

Одной из метрик *ресурса параллелизма* алгоритма можно считать ширину канонической ярусно-параллельной формы его графа информационных зависисимостей (см. раздел 1.6) W_r. Она характеризует максимальное число вершин графа информационных зависимостей, попавших на один ярус канонической ЯПФ. Такие вершины соответствуют

операциям, которые можно выполнять параллельно при наличии достаточного количества исполняющих устройств. В книге [17] соответствующее понятие называется числом процессоров (number of processors), при этом имеется в виду необходимое число процессоров для максимально параллельного выполнения алгоритма.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Ресурс параллелизма алгоритма суммирования элементов вектора длины n равен $W_r = 1$, ресурс параллелизма алгоритма скалярного произведения двух векторов длины n при условии, что операции суммирования и перемножения двух чисел считаем равнозначными, составляет $W_r = n$ (для попарных произведений), а ресурс параллелизма алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равен $W_r = n^3$ (все элементы результирующей матрицы можно вычислять параллельно, для вычисления каждого нужно выполнить скалярное произведение соответствующих строки и столбца исходных матриц).

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] автор описания приводит оценку ресурса параллелизма как $W_r = O(n^2)$.

2.5. Алгоритмическое ускорение

Понятие *ускорения* (*speedup*) чаще применяется как характеристика программных реализаций и определяется как $S = T_1/T_p$ — отношение времени работы некоторой программы на одном процессе T_1 ко времени работы распараллеленной программы при использовании p процессов T_p. Похожую характеристику можно рассмотреть и для алгоритмов. Поскольку вычислительная сложность (см. раздел 2.2) является некоторым приближением времени выполнения последовательного алгоритма, а параллельная сложность (см. раздел 2.3) — некоторым приближением времени выполнения параллельного алгоритма, то их отношение [28] $S_{max} = W/W_p$ будет аналогом ускорения для алгоритмов. В данном случае S_{max} определяет максимальное *алгоритмическое ускорение*, которое теоретически можно получить для реализации данного алгоритма в отсутствие влияния других значимых факторов при наличии бесконечного количества исполняющих устройств (в концепции неограниченного параллелизма [3]).

В книге [26] аналогичную характеристику называют просто параллелизм (parallelism), а в книге [17] — ускорение (speedup).

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Алгоритмическое ускорение суммирования элементов вектора длины n равно $S_{max} = (n-1)/(n-1) = 1$, алгоритмическое ускорение скалярного произведения двух векторов длины n составляет $S_{max} = (2n-1)/n$, а алгоритмическое ускорение перемножения двух плотных квадратных матриц размера $n \times n$ равно $S_{max} = n^2(2n-1)/n = 2n^2 - n$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] алгоритмическое ускорение можно оценить как $S_{max} = O(n^3)/O(n) = O(n^2)$.

2.6. Алгоритмическая эффективность распараллеливания

Понятие *эффективность распараллеливания* (*parallel efficiency*) чаще применяется как характеристика программных реализаций и определяется как $E_p = S/p$ (где S — полученное ускорение работы программы, а p — число используемых процессов. Похожую характе-

ристику можно рассмотреть и для алгоритмов. Поскольку алгоритмическое ускорение (см. раздел 2.5) S_{max} является некоторым приближением ускорения для параллельного алгоритма, а ресурс параллелизма (см. раздел 2.4) W_r — некоторым приближением необходимого количества процессов, то их отношение $E_{p_{max}} = S_{max}/W_r$ будет аналогом эффективности распараллеливания для алгоритмов. В данном случае $E_{p_{max}}$ определяет *алгоритмическую эффективность распараллеливания*, которая показывает, насколько хорошо может быть распараллелен данный алгоритм.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Алгоритмическая эффективность распараллеливания суммирования элементов вектора длины n равна $E_{p_{max}} = 1$ (в этом вырожденном случае высокая алгоритмическая эффективность распараллеливания не означает наличия параллельной реализации), алгоритмическая эффективность распараллеливания скалярного произведения двух векторов длины n составляет $E_{p_{max}} = \frac{2n-1}{n^2}$, а алгоритмическая эффективность распараллеливания алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равна $E_{p_{max}} = \frac{2n^2-n}{n^3} = \frac{2n-1}{n^2}$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] алгоритмическую эффективность распараллеливания можно оценить как $E_{p_{max}} = O(n^2)/O(n^2) = O(1)$.

2.7. Алгоритмическая стоимость

Понятие *стоимости* (*cost*) чаще применяется как характеристика программных реализаций и определяется как $C = pT_p$ — произведение количества процессов p на время работы распараллеленной программы T_p. Похожую характеристику можно рассмотреть и для алгоритмов. Поскольку параллельная сложность (см. раздел 2.3) W_p является некоторым приближением времени выполнения параллельного алгоритма, а ресурс параллелизма (см. раздел 2.4) W_r — некоторым приближением необходимого количества процессов, то их произведение $C_{max} = W_r * W_p$ будет аналогом стоимости для алгоритмов. В данном случае C_{max} задает *алгоритмическую стоимость*, которая определяется как суммарное количество операций, которые выполнились бы в параллельной реализации алгоритма при условии максимальной загрузки всех процессов. Оценки алгоритмической стоимости зачастую могут оказаться сильно завышенными, поскольку могут включать учет операций, которых нет в изначальном алгоритме.

Похожая характеристика стоимости (cost) рассматривается в ряде источников [17, 24, 26, 28, 34], а в книге [33] она называется «объем работы, выполняемой параллельным алгоритмом (the amount of work performed by a parallel algorithm)».

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Алгоритмическая стоимость суммирования элементов вектора длины n равна $C_{max} = n - 1$, алгоритмическая стоимость скалярного произведения двух векторов длины n составляет $C_{max} = n^2$, а алгоритмическая стоимость алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равна $C_{max} = n^4$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] алгоритмическую стоимость можно оценить как $C_{max} = O(n^3)$.

2.8. Алгоритмические накладные расходы

Понятие *накладных расходов* (*total overhead*) чаще применяется как характеристика программных реализаций и определяется как $T_0 = C - T_1$ — разница стоимости и времени работы программы на одном процессе T_1. Похожую характеристику можно рассмотреть и для алгоритмов. Поскольку алгоритмическая стоимость (см. раздел 2.7) C_{max} является некоторым аналогом стоимости для алгоритмов, а вычислительная сложность (см. раздел 2.2) W является некоторым приближением времени выполнения последовательного алгоритма, то их разность $T_{0max} = C_{max} - W$ будет аналогом накладных расходов для алгоритмов. В данном случае T_{0max} определяет *алгоритмические накладные расходы*, то есть оценку того, насколько суммарное количество операций параллельного алгоритма при условии максимальной загрузки всех процессов больше количества операций последовательного алгоритма.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Алгоритмические накладные расходы алгоритма суммирования элементов вектора длины n равны $T_{0max} = 0$ (что не удивительно для последовательного алгоритма), алгоритмические накладные расходы алгоритма скалярного произведения двух векторов длины n составляют $T_{0max} = n^2 - (2n - 1)$, а алгоритмические накладные расходы алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равны $T_{0max} = n^4 - n^2(2n - 1)$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] алгоритмические накладные расходы можно оценить как $T_{0max} = O(n^3) - O(n^3) = O(n^3)$.

2.9. Алгоритмическая избыточность

Иногда бывает более показательно сравнивать алгоритмическую стоимость (см. раздел 2.7) и вычислительную сложность (см. раздел 2.2) рассмотрением не разности, а отношения данных величин. При этом получается *алгоритмическая избыточность* (*redundancy*) $R = C_{max}/W$.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Алгоритмическая избыточность суммирования элементов вектора длины n равна $R = 1$, алгоритмическая избыточность скалярного произведения двух векторов длины n составляют $R = \frac{n^2}{2n-1} = O(n)$, а алгоритмическая избыточность перемножения двух плотных квадратных матриц размера $n \times n$ равна $R = \frac{n^4}{n^2(2n-1)} = O(n)$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] алгоритмическую избыточность можно оценить как $R = O(n^3)/O(n^3) = O(1)$.

2.10. Оценка необходимого объема памяти

Важным при анализе алгоритмов является не только оценка выполняемых операций, но и оценка *необходимого объема памяти*. Поскольку объем данных в единицах измерения информации (битах, байтах) определяется реализацией, то для алгоритмов необходимый объем памяти оценивают в количестве элементов основного типа данных (см. раздел 1.4). Эту характеристику называют также емкостная сложность [30], сложность по памяти [28] или пространственная сложность [27].

Оценка необходимого объема памяти состоит из двух частей: оценки объема входных данных [21] алгоритма V_{in} и оценки объема его выходных данных V_{out}.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Объем входных данных алгоритма суммирования элементов вектора длины n равен $V_{in} = n$, а объем выходных данных этого алгоритма $V_{out} = 1$, объем входных данных алгоритма скалярного произведения двух векторов длины n составляет $V_{in} = 2n$, а объем выходных данных этого алгоритма $V_{out} = 1$, объем входных данных алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равен $V_{in} = 2n^2$, а объем выходных данных этого алгоритма $V_{out} = n^2$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] объем входных и выходных данных можно оценить как $V_{in} = V_{out} = n^2$.

2.11. Вычислительная мощность

Для программных реализаций хорошо известна такая характеристика, как арифметическая интенсивность (arithmetic intensity [35]). Она соотносит интенсивность вычислений в программе с интенсивностью работы с оперативной памятью и определяется как отношение количества операций, выполняемых в программе, к количеству обращений к памяти. Для алгоритмов оценку количества выполняемых операций дает вычислительная сложность 2.2, а количество обращений к памяти в общем случае оценить достаточно сложно. Поэтому как характеристику алгоритмов рассматривают *вычислительную мощность* (*computational power*) — отношение вычислительной сложности (см. раздел 2.2) алгоритма к суммарному объему его входных и выходных данных (см. раздел 2.10) $P = \frac{W}{V_{in}+V_{out}}$.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Вычислительная мощность алгоритма суммирования элементов вектора длины n равна $P = \frac{n-1}{n+1}$, вычислительная мощность алгоритма скалярного произведения двух векторов длины n составляет $P = \frac{2n-1}{2n+1}$, а вычислительная мощность алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равна $P = \frac{n^2(2n-1)}{3n^2} = \frac{2n-1}{3}$.

Для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] вычислительную мощность можно оценить как $P = \frac{2n^3}{2n^2} = n$.

2.12. Свойства графа информационных зависимостей

Важные свойства алгоритмов можно получать, анализируя свойства соответствующих графов информационных зависимостей (см. раздел 1.6). Часть таких свойств однозначно соответствует описанным в предыдущих пунктах характеристикам:

- *Количество вершин* графа информационных зависимостей соответствует вычислительной сложности (см. раздел 2.2) алгоритма;
- *Высота канонической ярусно-параллельной формы* графа информационных зависимостей соответствует параллельной сложности (см. раздел 2.3) алгоритма;
- *Ширина канонической ярусно-параллельной формы* графа информационных зависимостей характеризует ресурс параллелизма (см. раздел 2.4) алгоритма.

Другие свойства графов информационных зависимостей дают дополнительные полезные характеристики соответствующих алгоритмов.

2.12.1. Количество дуг в графе информационных зависимостей

Количество дуг E в графе информационных зависимостей является одной из базовых характеристик графа, а применительно к анализу алгоритмов характеризует общее количество информационных зависимостей, что в конечном счете означает число обращений в память и в какой-то степени необходимых пересылок данных при параллельной реализации.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Количество дуг в графе информационных зависимостей алгоритма суммирования элементов вектора длины n равно $E = n-2$, количество дуг в графе информационных зависимостей алгоритма скалярного произведения двух векторов длины n составляет $E = 2(n-1) = 2n-2$, а количество дуг в графе информационных зависимостей алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равно $E = n^2(2n-2)$.

2.12.2. Плотность графа информационных зависимостей

Плотностью графа (*graph density*) принято называть соотношение количества дуг и вершин $D = E/W$. Иногда эту величину называют средней степенью вершины. Применительно к графам информационных зависимостей плотность характеризует интенсивность работы с данными в соответствующем алгоритме.

Рассмотрим примеры алгоритмов с графами информационных зависимостей, приведенными на рис. 3. Плотность графа информационных зависимостей алгоритма суммирования элементов вектора длины n равна $D = \frac{n-2}{n-1}$, плотность графа информационных зависимостей алгоритма скалярного произведения двух векторов длины n составляет $D = \frac{2n-2}{2n-1}$, а плотность графа информационных зависимостей алгоритма перемножения двух плотных квадратных матриц размера $n \times n$ равна $D = \frac{n^2(2n-2)}{n^2(2n-1)} = \frac{2n-2}{2n-1}$.

2.12.3. Количество типов дуг в графе информационных зависимостей

Типы дуг в графе информационных зависимостей задают разные образцы обращения к данным в алгоритме: считаем, что тип дуг определяется единым направляющим вектором и равной длиной при фиксированных значениях параметров. Поскольку направляющие векторы и длины дуг существенно зависят от расположения вершин, то определять разные типы дуг можно при некотором фиксированном (стандартном) способе их расположения.

На рис. 3 видно, что граф информационных зависимостей алгоритма суммирования элементов вектора содержит дуги только одного типа, а графы информационных зависимостей алгоритма скалярного произведения двух векторов и алгоритма перемножения двух плотных квадратных матриц содержат дуги трех типов.

2.12.4. Наличие в графе информационных зависимостей вершин со степенью исхода, зависящей от внешних параметров

Степень исхода (*degree of a vertex*) вершины графа информационных зависимостей — это количество исходящих из данной вершины информационных дуг. Если в графе информационных зависимостей есть вершины, степень исхода которых зависит от внешних параметров, то в программной реализации это во многих случаях будет соответствовать использованию массовых рассылок с помощью, например, коллективных операций в техно-

логии MPI. Такая информация будет полезной при написании программы, но может быть зафиксирована уже в момент анализа структуры алгоритма.

Графы информационных зависимостей алгоритмов на рис. 3 не содержат таких вершин, а в макрографе информационных зависимостей для метода Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) [14] на рис. 2 такие вершины присутствуют.

Заключение

В данной статье предложен набор свойств, особенностей и характеристик алгоритмов, предназначенных для обеспечения возможности их автоматического анализа. Следующим шагом планируется внесение этих свойств в описания алгоритмов на страницах Открытой энциклопедии свойств алгоритмов AlgoiWiki [7]. Для дальнейшего их автоматического анализа данные свойства будут вноситься авторами статей не в wiki-формате, а путем заполнения полей специальной веб-формы с занесением в базу данных проекта. В более отдаленной перспективе планируется разработка технологий автоматического получения таких свойств по некоторому описанию информационной структуры алгоритма (например, на языке Algolang [6]) или по тексту его программной реализации на традиционном языке программирования. И далее полученные свойства будут использоваться для решения задач суперкомпьютерного кодизайна [31] по совместному анализу свойств алгоритмов, программных реализаций и суперкомпьютерных систем.

Исследование выполнено за счет гранта Российского научного фонда № 25–11–00181, `https://rscf.ru/project/25-11-00181/`. *Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М. В. Ломоносова [36].*

Литература

1. Антонов А. Обзор методов описания структуры алгоритмов // Вычислительные методы и программирование. 2025. Т. 26, № 4. С. 548–568. DOI: 10.26089/NumMet.v26r436.
2. Voevodin V., Antonov A., Dongarra J. Why is it hard to describe properties of algorithms? // Procedia Computer Science. 2016. Vol. 101. P. 4–7. DOI: 10.1016/j.procs.2016.11.002.
3. Воеводин В.В., Воеводин В.В. Параллельные вычисления. СПб: БХВ-Петербург, 2002. 608 с.
4. Antonov A., Volkov N. Information Graph Visualization Using AlgoView Software Tool // Lobachevskii J. Math. 2020. Vol. 41, no. 6. P. 1427–1434. DOI: 10.1134/S199508022008003X.
5. Skryabin G., Gadieva T., Antonov A. A New Version of the AlgoView System for 3D Visualization and Interactive Analysis of Information Graphs of Algorithms // Parallel Computational Technologies. Vol. 2241 / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer, 2024. P. 19–33. Communications in Computer and Information Science. DOI: 10.1007/978-3-031-73372-7_2.
6. Antonov A., Volkov N. Study of the Algorithms Information Structure as the Basis of a Training Workshop // Supercomputing. Vol. 1510 / ed. by V. Voevodin, S. Sobolev.

Cham: Springer, 2021. P. 404–414. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-92864-3_31.

7. Открытая энциклопедия свойств алгоритмов. URL: https://algowiki-project.org (дата обращения: 23.01.2026).
8. Структура описания свойств алгоритмов — Алговики. URL: https://algowiki-project.org/en/Description_of_algorithm_properties_and_structure (дата обращения: 23.01.2026).
9. Antonov A. Wiki Representation and Analysis of Knowledge About Algorithms // Supercomputing. Vol. 13708 / ed. by V. Voevodin, S. Sobolev, M. Yakobovskiy, R. Shagaliev. Cham: Springer, 2022. P. 604–616. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-22941-1_44.
10. MediaWiki. URL: https://www.mediawiki.org (дата обращения: 23.01.2026).
11. Классификация алгоритмов — Алговики. URL: https://algowiki-project.org/en/Algorithm_classification (дата обращения: 23.01.2026).
12. Antonov A., Frolov A., Konshin I., Voevodin V. Hierarchical Domain Representation in the AlgoWiki Encyclopedia: From Problems to Implementations // Parallel Computational Technologies. Vol. 910 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2018. P. 3–15. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_1.
13. Popov A., Nikitenko D., Antonov A., Voevodin V. Formal Model of Problems, Methods, Algorithms and Implementations in the Advancing AlgoWiki Open Encyclopedia // CEUR Workshop Proc. Vol. 2281. 2018. P. 1–11. URL: http://ceur-ws.org/Vol-2281/#paper-01.
14. Метод Гивенса (вращений) QR-разложения квадратной матрицы (вещественный точечный вариант) — Алговики. URL: https://algowiki-project.org/en/Givens_method (дата обращения: 23.01.2026).
15. Справка: Шаблоны — MediaWiki. URL: https://www.mediawiki.org/wiki/Help:Templates/ru (дата обращения: 23.01.2026).
16. Кнут Д. Искусство программирования. Том 1. Основные алгоритмы. 3-е издание. М.: ООО «И.Д. Вильямс», 2001. 720 с.
17. Akl S. The Design and Analysis of Parallel Algorithms. Prentice Hall, 1989. 412 p.
18. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд.: Пер. с англ. СПб: БХВ-Петербург, 2011. 720 с.
19. Лобанова В., Воронина О., Лобанова Н. Теория алгоритмов: учебное пособие. Орёл: ОГУ имени И.С. Тургенева, 2017. 95 с.
20. Mehlhorn K., Sanders P. Algorithms and Data Structures: The Basic Toolbox. Springer Science & Business Media, 2008. 305 p. DOI: 10.1007/978-3-540-77978-0.
21. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ, 2-е издание. Пер. с англ. М.: Издательский дом «Вильямс», 2011. 1296 с.
22. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск. К.: Издательство «ДиаСофт», 2001. 688 с.

23. Reingold E. Basic Techniques for Design and Analysis of Algorithms. Chapman, Hall/CRC, 2004. 3–1 p. DOI: 10.1201/9780429171529-4.

24. JaJa J. An Introduction to Parallel Algorithms. Addison Wesley Longman Publishing Co., Inc., United States, 1992. 566 p.

25. Alsuwaiyel M. Parallel algorithms. World Scientific Publishers, 2022. 400 p. DOI: 10.1142/12744.

26. Blelloch G., Maggs B. Parallel Algorithms. Computer science handbook / editor-in-chief, Allen B. Tucker. 2nd ed. 2004. 10-10–41 p.

27. Селиванова И., Блинов В. Фундаментальные алгоритмы на C++. Построение и анализ алгоритмов обработки данных: учебно-методическое пособие. Екатеринбург: Издательство Уральского университета, 2015. 108 с.

28. Макконнелл Д. Основы современных алгоритмов. 2-е дополненное издание. М.: Техносфера, 2004. 368 с.

29. Метод сдваивания Стоуна для решения двудиагональных СЛАУ — Алговики. URL: https://algowiki-project.org/en/Stone_doubling_algorithm_for_solving_bidiagonal_SLAEs (дата обращения: 23.01.2026).

30. Ахо А., Хопкрофт Д., Ульман Д. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.

31. Antonov A., Maier R., Nikitenko D., Voevodin V. An Approach to Solving the Problem of Supercomputer Co-design // Lobachevskii J Math. 2024. Vol. 45, no. 7. P. 2965–2973. DOI: 10.1134/S1995080224603680.

32. Deterministic algorithm. URL: https://xlinux.nist.gov/dads/HTML/deterministicAlgorithm.html (дата обращения: 23.01.2026).

33. Smith J. The Design and Analysis of Parallel Algorithms. Oxford University Press, 1993. 470 p.

34. Casanova H., Legrand A., Robert Y. Parallel Algorithms. CRC PRESS, 2020. 335 p.

35. Паттерсон Д., Хеннеси Д. Архитектура компьютеров и проектирование компьютерных систем. 4-е. СПб: Питер, 2012. 919 с. «Классика computer science».

36. Voevodin V., Antonov A., Nikitenko D., *et al.* Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community // Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

Антонов Александр Сергеевич, к.ф.-м.н., ведущий научный сотрудник, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

DOI: 10.14529/cmse260201

DESCRIBING AND ANALYZING PROPERTIES, FEATURES AND CHARACTERISTICS OF ALGORITHMS IN THE ALGOWIKI OPEN ENCYCLOPEDIA OF PARALLEL ALGORITHMIC FEATURES*

© 2026 A.S. Antonov

Research Computing Center, Lomonosov Moscow State University
(GSP-1, Leninskie Gory 1, building 4, Moscow, 119234 Russia)
E-mail: asa@parallel.ru

Received: 16.03.2026

Investigating the properties of algorithms is a crucial step in their effective implementation on high-performance computing systems. This paper examines the properties, features, and characteristics of algorithms that can be useful for such analysis. It also discusses the necessary preliminary steps to transform algorithms into a form suitable for analysis and comparison. Analytical characteristics of algorithms – that is, those properties that can be described numerically or formulaically – are highlighted, with an emphasis on properties related to parallelism. Many of the properties discussed are proposed for algorithms for the first time, while at the same time being similar to those commonly considered for software implementations. All of the properties discussed are illustrated with several examples for well-known algorithms, such as summation of vector elements, the scalar product of two vectors, the multiplication of two dense square matrices, and the Givens (rotation) method of the QR factorization of a square matrix. In the future, based on the study of the considered properties, it is proposed to solve problems of supercomputer co-design for the joint analysis of the properties of algorithms, software implementations and supercomputer systems.

Keywords: algorithm, parallelism, supercomputer, co-design, computational complexity, parallel complexity, parallelism resource, speedup, information dependency graph, AlgoWiki.

FOR CITATION

Antonov A.S. Describing and Analyzing Properties, Features and Characteristics of Algorithms in the AlgoWiki Open Encyclopedia of Parallel Algorithmic Features. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2026. Vol. 15, no. 2. P. 5–25. (in Russian) DOI: 10.14529/cmse260201.

References

1. Antonov A. Review of methods for describing the structure of algorithms. Numerical Methods and Programming. 2025. Vol. 26, no. 4. P. 548–568. (in Russian) DOI: 10.26089/NumMet.v26r436.
2. Voevodin V., Antonov A., Dongarra J. Why is it hard to describe properties of algorithms?. Procedia Computer Science. 2016. Vol. 101. P. 4–7. DOI: 10.1016/j.procs.2016.11.002.
3. Voevodin V., Voevodin V. Parallel computing. BHV-Peterburg, 2002. 608 p. (in Russian).

*The article was recommended for publication by the program committee of the international scientific conference "Parallel Computing Technologies (PaVT) 2026".

4. Antonov A., Volkov N. Information Graph Visualization Using AlgoView Software Tool. Lobachevskii J. Math. 2020. Vol. 41, no. 6. P. 1427–1434. DOI: 10.1134/S199508022008003X.

5. Skryabin G., Gadieva T., Antonov A. A New Version of the AlgoView System for 3D Visualization and Interactive Analysis of Information Graphs of Algorithms. Parallel Computational Technologies. Vol. 2241 / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer, 2024. P. 19–33. Communications in Computer and Information Science. DOI: 10.1007/978-3-031-73372-7_2.

6. Antonov A., Volkov N. Study of the Algorithms Information Structure as the Basis of a Training Workshop. Supercomputing. Vol. 1510 / ed. by V. Voevodin, S. Sobolev. Cham: Springer, 2021. P. 404–414. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-92864-3_31.

7. Open Encyclopedia of Parallel Algorithmic Features – Algowiki. URL: https://algowiki-project.org/en (accessed: 23.01.2026).

8. Description of algorithm properties and structure – Algowiki. URL: https://algowiki-project.org/en/Description_of_algorithm_properties_and_structure (accessed: 23.01.2026).

9. Antonov A. Wiki Representation and Analysis of Knowledge About Algorithms. Supercomputing. Vol. 13708 / ed. by V. Voevodin, S. Sobolev, M. Yakobovskiy, R. Shagaliev. Cham: Springer, 2022. P. 604–616. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-22941-1_44.

10. MediaWiki. URL: https://www.mediawiki.org (accessed: 23.01.2026).

11. Algorithm classification – Algowiki. URL: https://algowiki-project.org/en/Algorithm_classification (accessed: 23.01.2026).

12. Antonov A., Frolov A., Konshin I., Voevodin V. Hierarchical Domain Representation in the AlgoWiki Encyclopedia: From Problems to Implementations. Parallel Computational Technologies. Vol. 910 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2018. P. 3–15. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_1.

13. Popov A., Nikitenko D., Antonov A., Voevodin V. Formal Model of Problems, Methods, Algorithms and Implementations in the Advancing AlgoWiki Open Encyclopedia. CEUR Workshop Proc. Vol. 2281. 2018. P. 1–11. URL: http://ceur-ws.org/Vol-2281/#paper-01.

14. Givens method – Algowiki. URL: https://algowiki-project.org/en/Givens_method (accessed: 23.01.2026).

15. Help:Templates – MediaWiki. URL: https://www.mediawiki.org/wiki/Help:Templates/en (accessed: 23.01.2026).

16. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition. 2001. 672 p.

17. Akl S. The Design and Analysis of Parallel Algorithms. Prentice Hall, 1989. 412 p.

18. Skiena S. The Algorithm Design Manual Second Edition. Springer, 2011. 739 p.

19. Lobanova V., Voronina O., Lobanova N. Theory of Algorithms: A Tutorial. Orel: Orel State University named after I.S. Turgenev, 2017. 95 p. (in Russian).

20. Mehlhorn K., Sanders P. Algorithms and Data Structures: The Basic Toolbox. Springer Science & Business Media, 2008. 305 p. DOI: 10.1007/978-3-540-77978-0.

21. Cormen T., Leiserson C.E., Rivest R.L., Stein C. Introduction To Algorithms, 3rd Edition. Milton, Qld., Jacaranda Wiley, 2009. 1292 p.

22. Sedgewick R. Algorithms in C, Parts 1–4: Fundamentals, Data Structures, Sorting, Searching. Addison-Wesley Professional, 1997. 702 p.

23. Reingold E. Basic Techniques for Design and Analysis of Algorithms. Chapman, Hall/CRC, 2004. 3–1 p. DOI: 10.1201/9780429171529-4.

24. JaJa J. An Introduction to Parallel Algorithms. Addison Wesley Longman Publishing Co., Inc., United States, 1992. 566 p.

25. Alsuwaiyel M. Parallel algorithms. World Scientific Publishers, 2022. 400 p. DOI: 10.1142/12744.

26. Blelloch G., Maggs B. Parallel Algorithms. Computer science handbook / editor-in-chief, Allen B. Tucker–2nd ed. 2004. 10-10–41 p.

27. Selivanova I., Blinov V. Fundamental Algorithms in C++. Construction and Analysis of Data Processing Algorithms: A Tutorial. Ekaterinburg: Ural University Publishing House, 2015. 108 p. (in Russian).

28. McConnell J. Analysis of Algorithms. Jones & Bartlett Learning, 2008. 451 p.

29. Stone doubling algorithm for solving bidiagonal SLAEs. URL: https://algowiki-project.org/en/Stone_doubling_algorithm_for_solving_bidiagonal_SLAEs (accessed: 23.01.2026) (in Russian).

30. Aho A., Hopcroft J., Ullman J. The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc., 1974. 480 p.

31. Antonov A., Maier R., Nikitenko D., Voevodin V. An Approach to Solving the Problem of Supercomputer Co-design. Lobachevskii J Math. 2024. Vol. 45, no. 7. P. 2965–2973. DOI: 10.1134/S1995080224603680.

32. Deterministic algorithm. URL: https://xlinux.nist.gov/dads/HTML/deterministicAlgorithm.html (accessed: 23.01.2026).

33. Smith J. The Design and Analysis of Parallel Algorithms. Oxford University Press, 1993. 470 p.

34. Casanova H., Legrand A., Robert Y. Parallel Algorithms. CRC PRESS, 2020. 335 p.

35. Patterson D., Hennessy J. Computer Organization and Design. 4th Edition. Elsevier, 2012. 919 p.

36. Voevodin V., Antonov A., Nikitenko D., *et al.* Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

УДК 519.853.4 DOI: 10.14529/cmse260202

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ И ЕГО ИСПОЛЬЗОВАНИЕ ДЛЯ РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ ХИМИЧЕСКОЙ КИНЕТИКИ*

© 2026 К.А. Баркалов[1,2], Е.Н. Варсеева[1], И.Г. Лебедев[1,2], А.Л. Хашпер[3], А.С. Урлуков[4,5], И.М. Губайдуллин[6], И.Ф. Хисаметдинов[7]

[1]*Нижегородский государственный университет им. Н.И. Лобачевского (603022 Н. Новгород, пр. Гагарина, д. 23),*

[2]*Научно-образовательный математический центр «Математика технологий будущего» (603022 Н. Новгород, пр. Гагарина, д. 23, корп. 2),*

[3]*ООО «СИБИНТЕК-СОФТ» (141401 Московская обл., Химки, Коммунальный проезд, стр. 30, помещ. 191),*

[4]*Институт катализа СО РАН (630090 Новосибирск, пр. Академика Лаврентьева, д. 5),*

[5]*Новосибирский государственный университет (630090 Новосибирск, ул. Пирогова, д. 1),*

[6]*Институт нефтехимии и катализа УФИЦ РАН (450075 Уфа, пр. Октября, д. 141),*

[7]*Уфимский государственный нефтяной технический университет (450064 Уфа, ул. Космонавтов, д. 1)*

E-mail: konstantin.barkalov@itmm.unn.ru, varseeva@unn.ru, ilya.lebedev@itmm.unn.ru, anna.khashper@gmail.com, aurlukov@mail.ru, irekmars@mail.ru, iskanderkhisametdinov1@gmail.com

Поступила в редакцию: 20.03.2026

В статье рассматривается использование параллельных вычислений для выбора параметров математической модели процесса низкотемпературной паровой конверсии углеводородов, входящих в состав попутного нефтяного газа. Для указанного химического процесса необходимо сформировать его кинетическую модель, то есть определить соответствующие кинетические параметры реакций. Для этого решается обратная задача, в которой ищутся значения кинетических параметров на основе экспериментальных данных. С формальной точки зрения обратная задача химической кинетики является задачей глобальной оптимизации. Для ее решения был использован параллельный информационно-статистический алгоритм глобального поиска в сочетании с локальным уточнением лучшего решения. Алгоритм является детерминированным и основан на предположении липшицевости целевой функции, что является типичным для многих других подходов к построению методов глобальной оптимизации. При этом решение многомерных задач сводится к решению эквивалентных им одномерных задач. Соответствующая редукция основана на использовании кривой Пеано, отображающей единичный отрезок вещественной оси на гиперкуб. Распараллеливание алгоритма организовано по схеме «мастер-рабочие» с использованием общей памяти. Оптимальные параметры модели, найденные с помощью параллельного алгоритма оптимизации, позволили адекватно смоделировать процесс низкотемпературной паровой конверсии легких углеводородов с использованием катализатора.

Ключевые слова: глобальная оптимизация, многоэкстремальные функции, параллельные вычисления, химическая кинетика, обратные задачи.

*Статья рекомендована к публикации программным комитетом всероссийской научной конференции «Параллельные вычислительные технологии (ПаВТ) 2026».

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Баркалов К.А., Варсеева Е.Н., Лебедев И.Г., Хашпер А.Л., Урлуков А.С., Губайдуллин И.М., Хисаметдинов И.Ф. Параллельный алгоритм глобальной оптимизации и его использование для решения обратных задач химической кинетики // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2026. Т. 15, № 2. С. 26–46. DOI: 10.14529/cmse260202.

Введение

Сжигание попутного нефтяного газа (ПНГ) на текущий момент остается актуальной проблемой российской нефтедобывающей промышленности. ПНГ представляет собой смесь предельных углеводородов, растворенных в нефти и выделяющихся при ее добыче. Одним из наиболее перспективных способов эффективной утилизации попутного газа является его использование непосредственно на месте добычи, для выработки электроэнергии и тепла на малых электростанциях, которые обеспечивают собственные нужды месторождения [1]. Однако исходный ПНГ не может быть использован в качестве топлива из-за высокого содержания в нем $C_2{}^+$-углеводородов, которые вызывают поломки двигателей внутреннего сгорания. Для решения проблемы утилизации ПНГ предлагается применять низкотемпературную паровую конверсию (НТПК) $C_2{}^+$-углеводородов, входящих в состав ПНГ, с преимущественным образованием метан-водородных смесей [2]. Полученные таким образом метан-водородные смеси могут быть использованы в качестве топлива.

Для проведения НТПК $C_2{}^+$-углеводородов были широко исследованы катализаторы на основе никеля [1, 3–5]. Однако основной проблемой никелевых катализаторов является их невысокая стабильность, связанная с зауглероживанием или спеканием никелевых частиц во время эксплуатации катализатора.

Сотрудники Института катализа СО РАН в своих работах показали, что родиевые катализаторы, нанесенные на смешанные оксиды церия-циркония, проявляют высокую активность и стабильность в процессе НТПК пропана [6–10], а катализатор 1 вес. % $Rh/Ce_{0.4}Zr_{0.5}Y_{0.05}La_{0.05}O_2$ является оптимальным вариантом среди родий-содержащих катализаторов [7]. Однако для более детального понимания кинетики процесса НТПК углеводородов на Rh катализаторах и оптимизации условий его проведения нужно построить и проанализировать математическую модель указанного процесса (так называемую кинетическую модель).

При построении кинетической модели многостадийной химической реакции решается обратная задача химической кинетики: по экспериментальным данным о динамике процесса восстанавливаются численные параметры модели — константы скоростей одностадийных реакций. Формально все сводится к поиску глобального минимума многоэкстремальной целевой функции, поэтому очень важно использовать адекватный метод оптимизации.

Метод решения таких оптимизационных задач должен учитывать их специфику. Во-первых, целевая функция фактически представляет собой «черный ящик»: она задается не явной формулой, а алгоритмом, вычисляющим ее значения. Во-вторых, даже одно вычисление функции требует ресурсов, поскольку подразумевает проведение численного моделирования. Из-за этого мультистартовые методы, основанные на многократном применении градиентного спуска, неприменимы: вычисление градиента слишком дорого, и даже многократный запуск градиентных процедур может не обеспечить сходимость к решению многоэкстремальной задачи.

Перечисленные особенности обратных задач учитываются в разработанном в Нижегородском университете параллельном алгоритме глобальной оптимизации. Теоретической основой данного алгоритма служит информационно-статистический подход, подробно изложенный в [11]. Суть подхода заключается в редукции исходной многомерной задачи к эквивалентной одномерной задаче, которую можно решить с помощью эффективного алгоритма оптимизации функций одной переменной. Вопросы распараллеливания предложенного алгоритма на различных архитектурах рассматриваются в [12].

В настоящей статье представлены результаты применения параллельного алгоритма глобального поиска к решению обратной задачи химической кинетики. Статья имеет следующую структуру. В разделе 1 приведено описание математической модели изучаемой химической реакции, сформулирована обратная задача. В разделе 2 излагается стандартная постановка задачи глобальной оптимизации, приведена вычислительная схема параллельного алгоритма ее решения, сформулированы его теоретические свойства, кратко рассмотрены вопросы, связанные с техническими аспектами реализации параллельного алгоритма. Содержательная интерпретация результатов численных расчетов с использованием разработанного параллельного алгоритма представлена в разделе 3. В заключении содержатся выводы и планы дальнейших исследований.

1. Постановка задачи

Для описания низкотемпературной паровой конверсии пропана достаточно учесть следующие реакции:

$$\begin{aligned} &\mathrm{C_3H_8} + 6\,\mathrm{H_2O} \longrightarrow 3\,\mathrm{CO_2} + 10\,\mathrm{H_2},\\ &\mathrm{C_3H_8} + \mathrm{H_2} \longrightarrow \mathrm{C_2H_6} + \mathrm{CH_4},\\ &\mathrm{C_2H_6} + \mathrm{H_2} \longrightarrow 2\,\mathrm{CH_4},\\ &\mathrm{CO_2} + 4\,\mathrm{H_2} \rightleftharpoons \mathrm{CH_4} + 2\,\mathrm{H_2O}. \end{aligned} \tag{1}$$

Было проведено математическое моделирование НТПК этана и пропана на 1 вес. % $\mathrm{Rh/Ce_{0.4}Zr_{0.5}Y_{0.05}La_{0.05}O_2}$ катализаторе с использованием реакций (1) и механизма Ленгмюра–Хиншельвуда для задания скоростей некоторых реакций [13, 14]:

$$\begin{cases} w_1 = k_1 \cdot \dfrac{P_{\mathrm{C_3H_8}} \cdot P_{\mathrm{H_2O}}}{(P_{\mathrm{H_2O}} + \alpha \cdot \sqrt{P_{\mathrm{H_2}}})^2},\\ w_2 = k_2 \cdot P_{\mathrm{C_3H_8}} \cdot P_{\mathrm{H_2}},\\ w_3 = k_4 \cdot \dfrac{\sqrt{P_{\mathrm{H_2}}} \cdot P_{\mathrm{C_2H_6}}}{(P_{\mathrm{H_2O}} + \alpha \cdot \sqrt{P_{\mathrm{H_2}}})^2},\\ w_4 = k_4 \cdot P_{\mathrm{CO_2}} \cdot P_{\mathrm{H_2}} - k_{-4} \cdot P_{\mathrm{H_2O}} \cdot P_{\mathrm{CH_4}}, \end{cases} \tag{2}$$

где P_X — парциальное давление вещества X, атм; w_i — скорость i-ой реакции, атм/(л·с); k_i — константа скорости i-ой реакции; t — время, с; α — отношение констант адсорбции $\mathrm{H_2}$ и $\mathrm{H_2O}$ на выбранном катализаторе.

Система дифференциальных уравнений для описания кинетики НТПК пропана записывается в виде

$$\begin{cases} u \cdot \dfrac{dP_{\mathrm{C_3H_8}}}{dl} = -w_1 - w_2, \\ u \cdot \dfrac{dP_{\mathrm{H_2O}}}{dl} = -6w_1 + 2w_4, \\ u \cdot \dfrac{dP_{\mathrm{CO_2}}}{dl} = 3w_1 - w_4, \\ u \cdot \dfrac{dP_{\mathrm{H_2}}}{dl} = 10w_1 - w_2 - w_3 - 4w_4, \\ u \cdot \dfrac{dP_{\mathrm{CH_4}}}{dl} = w_2 + 2w_3 + w_4, \\ u \cdot \dfrac{dP_{\mathrm{C_2H_6}}}{dl} = w_2 - w_3, \end{cases} \tag{3}$$

где u — линейная скорость потока реакционной смеси, м/с; l — высота слоя катализатора, м.

Прямая задача химической кинетики предполагает расчет состава многокомпонентной реагирующей смеси на основе кинетической модели с известными параметрами [15]. С математической точки зрения прямая задача представляет собой задачу Коши для системы из n обыкновенных дифференциальных уравнений:

$$x' = f(x,t), x(t_0) = x_0, t \in [t_0, t_{cont}], \tag{4}$$

где $x(t)$ — вещественная n-мерная функция, представляющая собой парциальные давления веществ; x_0 — вектор начальных условий; t_0 — начальный момент времени; t_{cont} — время контакта с катализатором; $t \in [t_0, t_{cont}]$ — время (независимая переменная). Начальное и конечное значения t_0 и t_{cont}, а также исходные парциальные давления веществ x_0 считаются заданными. Решением прямой задачи является набор (t_i, x_i), где t_i — момент времени, а x_i — вектор парциальных давлений всех веществ системы в i-й момент времени [15].

Для решения прямой задачи при различных температурах реакции был реализован модифицированный метод Рунге—Кутты 4 порядка точности в соответствии со следующими формулами:

$$\begin{aligned} &F_1 = f(x_i) \cdot h, \\ &F_2 = f(x_i + F_1/3) \cdot h, \\ &F_3 = f(x_i + (F_1 + F_2)/6) \cdot h, \\ &F_4 = f(x_i + (F_1 + 3 \cdot F_3)/8) \cdot h, \\ &x_{s+1} = x_i + (F_1 - 3 \cdot F_3 + 4 \cdot F_4)/2, \\ &F_5 = f(x_{i+1}) \cdot h, \\ &x^*_{i+1} = x_i + (F_1 + 4 \cdot F_4 + F_5)/6. \end{aligned} \tag{5}$$

Построение математических моделей химических реакций предполагает наличие в них неизвестных кинетических параметров (предэкспоненциальные множители констант скоростей, энергии активации, наблюдаемые порядки по компонентам), которые можно найти, решая задачу минимизации отклонения между расчетными и экспериментальными данными. Таким образом, возникает задача идентификации математической модели (обратная задача химической кинетики) [16–18].

Решение обратной задачи химической кинетики сводится к поиску минимума целевой функции [15]. В данной работе в качестве критерия минимизации была использована целе-

вая функция вида

$$E = \sum_{i=1}^{n} \left| \overline{P_i} - \widetilde{P_i} \right| \cdot v_i, \tag{6}$$

где $\overline{P_i}$, $\widetilde{P_i}$ — экспериментальные и расчетные значения парциальных давлений веществ, атм; v_i — весовые коэффициенты; n — количество измеряемых веществ. Требуется найти набор констант скоростей k, при котором значение функции (6) минимально.

2. Параллельный алгоритм для решения задач глобальной оптимизации

2.1. Задача глобальной оптимизации и редукция размерности

Рассмотрим задачу глобальной оптимизации в общем виде:

$$\varphi(y^*) = \min\{\varphi(y) : y \in D\}, \tag{7}$$

$$D = \left\{ y \in \mathbb{R}^N : a_i \leqslant y_i \leqslant b_i,\ a_i, b_i \in \mathbb{R},\ 1 \leqslant i \leqslant N \right\}, \tag{8}$$

где целевая функция является «черным ящиком» и предполагается удовлетворяющей условию Липшица

$$|\varphi(y_1) - \varphi(y_2)| \leqslant L \|y_1 - y_2\|,\ y_1, y_2 \in D,$$

с константой L, $L < \infty$, неизвестной априори.

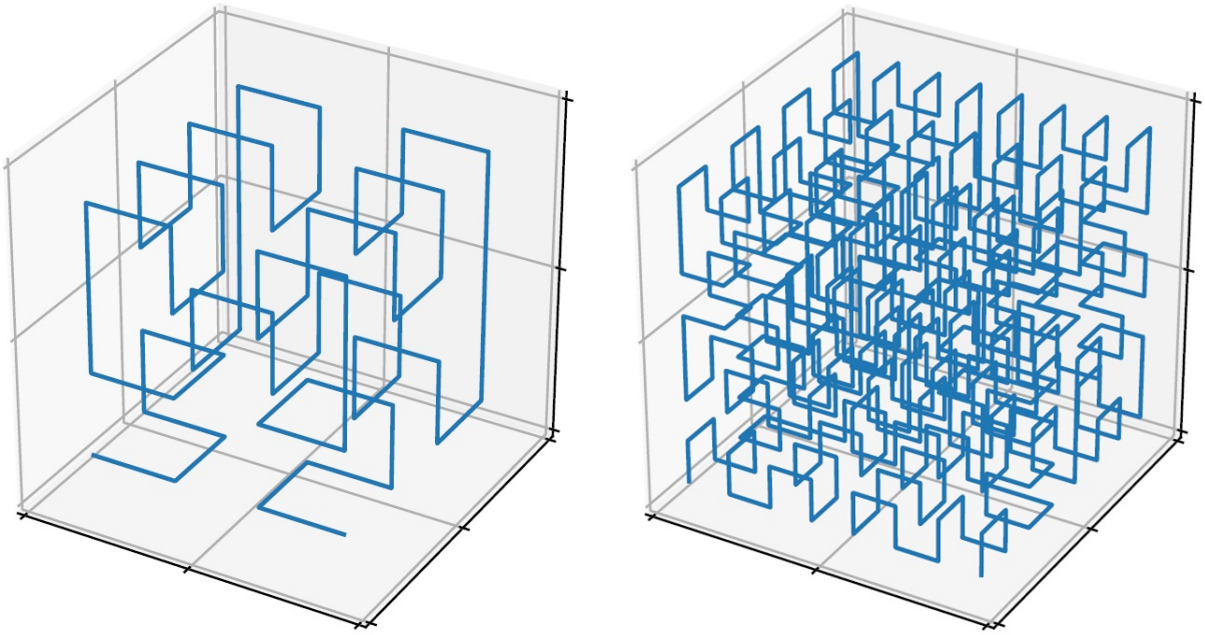
Предположение о выполнении условия Липшица для целевой функции используется многими исследователями при разработке детерминированных алгоритмов глобальной оптимизации. Первые методы такого класса были предложены еще в начале 1970-х годов [19, 20]. Активная работа в этой области продолжается и сейчас [21–24].

Отметим, что в настоящее время распространенным инструментом для решения задач с функциями вида «черный ящик» являются также и метаэвристические алгоритмы, основанные на имитации природных процессов и использующие элементы стохастики (см. [25–27]). Их практическая привлекательность объясняется простотой использования, однако по эффективности и надежности поиска оптимума они, как правило, уступают детерминированным методам [28, 29].

Эффективный алгоритм решения задач многоэкстремальной оптимизации — *информационно-статистический алгоритм глобального поиска* — был разработан в ННГУ им. Н.И. Лобачевского [11]. Данный алгоритм является детерминированным и в исходном виде был ориентирован на решение задач безусловной оптимизации, однако позднее были разработаны его модификации для решения задач с функциональными ограничениями, а также — с несколькими критериями. Также для алгоритма глобального поиска был предложен ряд способов его распараллеливания [12, 30, 31].

Многие современные алгоритмы глобальной оптимизации используют идею редукции размерности при решении многомерных задач; см., например, методы на основе диагональных [32] или симплексных [33] разбиений. Параллельный алгоритм глобального поиска, описываемый в данном разделе, использует кривую Пеано [11, 34], отображающую единичный отрезок $[0, 1]$ на N-мерный куб D из (8). С помощью такого отображения исходная многомерная задача (7) сводится к одномерной задаче

$$\varphi(y^*) = \varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0, 1]\},$$



а) Развертка при $m = 3$ б) Развертка при $m = 4$

Рис. 1. Развертки в трехмерном пространстве

где функция $\varphi(y(x))$ удовлетворяет равномерному условию Гёльдера

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \leqslant H\,|x_1 - x_2|^{1/N}$$

с константой Гёльдера H, связанной с константой Липшица L соотношением $H = 2L\sqrt{N+3}$, а $y(x)$ — кривая Пеано, отображающая $[0,1]$ на D. Поэтому, не ограничивая общности, можно говорить о минимизации одномерной функции

$$f(x) = \varphi(y(x)), x \in [0,1],$$

удовлетворяющей условию Гёльдера.

Известно, что точная кривая Пеано $y(x)$ задается как предельный объект и вычислить точное значение $y(x)$ не представляется возможным. Поэтому на практике используются аппроксимации истинной кривой, называемые *развертками*. Численные методы построения разверток изложены в работах [11, 34]. Точность развертки определяется параметром ее построения m и имеет порядок 2^{-m} по каждой координате. Примеры разверток для размерности $N = 3$ и различных значений m показаны на рис. 1.

2.2. Параллельный алгоритм глобального поиска

Будем называть процесс вычисления значения функции $f(x) = \varphi(y(x))$ *испытанием*, а пару $\{x, z = f(x) = \varphi(y(x))\}$ — результатом испытания. При описании параллельного алгоритма будем использовать термин *итерация* для обозначения одновременного (параллельного) выполнения нескольких испытаний (каждое испытание выполняется в отдельном потоке или процессе). Число испытаний на n-й итерации обозначим p, а общее число испытаний, выполненных за все n итераций, обозначим $s(n)$. Иными словами, предполагается, что при выполнении n-й итерации имеется в распоряжении p потоков или процессов.

Вычислительная схема параллельного алгоритма глобального поиска состоит в следующем. Первые p испытаний выполняются в граничных точках $x^0 = 0$, $x^1 = 1$, а также в произвольных внутренних точках $x^2, ..., x^{p-1}$ интервала $(0, 1)$. Предположим, что завершено $n \geqslant 1$ итераций метода, в ходе которых проведены испытания в $s = s(n)$ точках x^i, $0 \leqslant i \leqslant s$. Тогда точки $x^{s+1}, ..., x^{s+p}$ для проведения испытаний на следующей $(n + 1)$-й итерации определяются согласно следующим правилам.

1. Перенумеровать прообразы всех точек, в которых уже проведены испытания,

$$y^0 = y(x^0), y^1 = y(x^1), ..., y^s = y(x^s) \tag{9}$$

по возрастанию их координат, то есть

$$0 = x_0 < x_1 < \cdots < x_s = 1, \tag{10}$$

и сопоставить им вычисленные значения $z_i = \varphi(y(x_i)), 0 \leqslant i \leqslant s$.

2. Вычислить максимальное абсолютное значение разделенных разностей

$$\mu = \max_{1 \leqslant i \leqslant s} \frac{|z_i - z_{i-1}|}{\Delta_i}, \tag{11}$$

где $\Delta_i = (x_i - x_{i-1})^{1/N}$. Если $\mu = 0$, полагаем $\mu = 1$.

3. Для каждого интервала (x_{i-1}, x_i), $1 \leqslant i \leqslant s$, вычислить величину

$$R(i) = r\mu\Delta_i + \frac{(z_i - z_{i-1})^2}{r\mu\Delta_i} - 2(z_i + z_{i-1}) \tag{12}$$

называемую *характеристикой* интервала; вещественное число $r > 1$ является параметром алгоритма.

4. Упорядочить характеристики $R(i), 1 \leqslant i \leqslant s$, по убыванию

$$R(t_1) \geqslant R(t_2) \geqslant \cdots \geqslant R(t_s) \tag{13}$$

и выбрать p интервалов с номерами $t_j, 1 \leqslant j \leqslant p$, которым соответствуют наибольшие характеристики.

5. Провести новые испытания параллельно, в точках $x^{s+j} \in (x_{t_j-1}, x_{t_j}), 1 \leqslant j \leqslant p$, вычисляемых по формуле

$$x^{s+j} = \frac{x_{t_j} + x_{t_j-1}}{2} - \operatorname{sign}(z_{t_j} - z_{t_j-1}) \frac{1}{2r} \left[\frac{|z_{t_j} - z_{t_j-1}|}{\mu} \right]^N.$$

Алгоритм завершает свою работу, если условие $\Delta_{t_j} < \epsilon$ выполняется хотя бы для одного номера интервала $t_j, 1 \leqslant j \leqslant p$; здесь $\epsilon > 0$ — заданная точность. Алгоритм также останавливается, если проведено максимально допустимое число итераций S_{max}, установленное до начала вычислений.

Введение описанного выше параллелизма не приводит к возникновению новых предельных точек, отличных от предельных точек последовательного алгоритма, т.е. когда $p = 1$. Условия сходимости для параллельного АГП могут быть сформулированы в виде следующей теоремы, доказательство которой приведено в работе [11].

Теорема. *Пусть $\overline{x}$ есть предельная точка последовательности испытаний $\{x^s\}$, порождаемой параллельным АГП при минимизации функции $f(x), x \in [0, 1]$, удовлетворяющей*

условию Гёльдера с константой $H, 0 < H < \infty$; число задействованных вычислительных элементов $p, 1 < p < \infty$, является фиксированным, а в условии остановки используется значение $\epsilon = 0$. Тогда:

1. *сходимость к точке $\overline{x} \in (0, 1)$ является двусторонней;*
2. *точка $\overline{x}$ является локально оптимальной, если функция $f(x)$ имеет конечное число локальных экстремумов;*
3. *если наряду с точкой $\overline{x}$ существует другая предельная точка $\widehat{x}$, то $f(\overline{x}) = f(\widehat{x})$;*
4. *при любом $s > 1$ выполняется неравенство $f(x_s) \geqslant f(\overline{x})$;*
5. *если на некотором шаге для величины $r\mu$ из (12) выполняется условие $r\mu > 2^{2-1/N}H$;*

то множество предельных точек последовательности $\{x^s\}$ совпадает с множеством точек глобального минимума $f(x)$.

Рассматриваемый вариант распараллеливания является *синхронным*, когда переход к следующей итерации осуществляется только после полного завершения текущей, то есть после окончания последнего испытания текущей итерации. Конкретная реализация этой схемы распараллеливания будет зависеть от архитектуры компьютерной системы, а также от требований к программному обеспечению, необходимому для вычисления значений целевой функции. В рассматриваемой задаче химической кинетики для проведения одного испытания требовалось решить систему ОДУ (для решения этой задачи использовался модифицированный метод Рунге—Кутты 4-го порядка). Распараллеливание было организовано с использованием общей памяти и схемы мастер-рабочие. Алгоритм глобального поиска был запущен в потоке-мастере, остальные (подчиненные) потоки проводили параллельные испытания в рамках шага 5 алгоритма. При этом время проведения одного испытания составляло порядка 0.1 секунды, таким образом, накладные расходы на синхронизацию между потоками не оказывали существенного влияния на эффективность распараллеливания.

После определения с помощью параллельного АГП точки, являющейся грубой оценкой глобального минимума, из нее запускается локальный метод. Мы использовали метод Хука—Дживса, который относится к классу методов нулевого порядка. Его вычислительные правила основаны на чередовании исследующего поиска (для выбора направления) и поиска в выбранном направлении (поиска по образцу) [35].

Исследующий поиск можно рассматривать как оператор $y^s = F(w^s)$, отображающий текущую точку w^s в следующую базовую точку y^s:

- Вычисляется величина смещения из текущей точки w^s (она является различной для каждого координатного направления и может изменяться в процессе работы метода).
- Шаг поиска является успешным, если значение целевой функции в новой точке не превышает значение целевой функции в предыдущей точке.
- Если шаг является неуспешным, то происходит возврат к исходной точке и выполняется смещение в обратном направлении.
- После исследования всех N координатных направлений данный этап завершается; полученная точка y^s определяется как базовая для поиска по направлению.

Поиск по направлению определяется смещением от полученной новой базовой точки y^s вдоль прямой, соединяющей y^s и y^{s-1}, т.е.

$$w^{s+1} = y^s + \alpha(y^s - y^{s-1}),$$

где величина смещения задается параметром метода α.

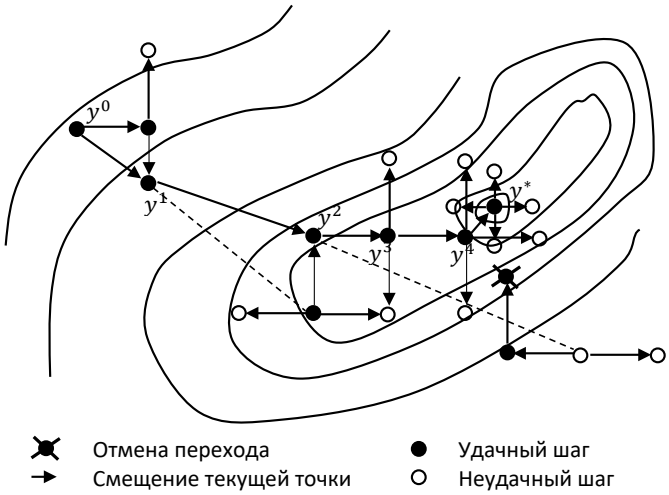


Рис. 2. Иллюстрация работы метода Хука—Дживса

Рисунок 2 иллюстрирует работу локального алгоритма. Показаны линии уровня целевой функции; темные точки соответствуют успешным, а светлые — неуспешным шагам, проведенным в ходе исследовательского поиска.

3. Результаты численных экспериментов

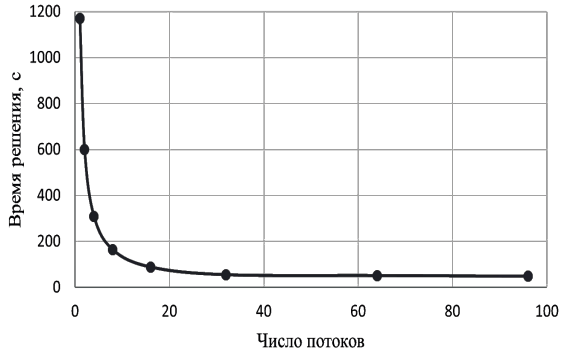
Вычислительные эксперименты проводились на узле суперкомпьютера «Лобачевский» в следующей конфигурации: операционная система Linux CentOS 8.2, 2 процессора Intel Xeon Platinum 8558 2.1/4GHz (48 ядер каждый), оперативная память 2048 Gb. Все рассмотренные алгоритмы реализованы на C++, использовался компилятор Intel C++ Compiler 2023 с поддержкой OpenMP.

Выбор оптимальных кинетических параметров осуществлялся следующим образом. Были получены экспериментальные данные для 18 различных температур проведения реакции. Далее, для каждой из этих температур решалась обратная задача и выбирались константы скоростей реакций, минимизирующие функцию (6) отклонения расчетных концентраций от экспериментальных для каждой температуры. Далее предэкспоненциальный множитель и энергии активации, входящие в уравнения линейно, искались с помощью метода наименьших квадратов.

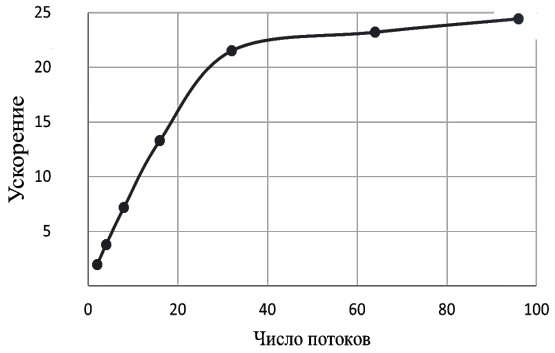
Таким образом, с помощью параллельного АГП было решено 18 обратных задач, каждая для своей температуры. Параллельный алгоритм запускался со следующими параметрами: плотность построения развертки $m = 10$, параметр надежности $r = 1.5$, точность глобального поиска $\epsilon = 10^{-2}$, локального уточнения $\delta = 10^{-4}$; максимальное число итераций глобального поиска $S_{glob} = 64000$, локального поиска $S_{loc} = 500$. Число потоков варьировалось от 1 до 96. В табл. 1 и на рис. 3 приведены усредненные (по всем 18 задачам) оценки времени работы и ускорения параллельного алгоритма. Результаты показывают, что увеличение числа задействованных потоков (более 32) не приводит к значительному ускорению; этот факт обсуждается в заключении. Подробный анализ и сопоставление теоретического (по закону Амдала) и практического ускорения параллельного алгоритма глобального поиска с использованием локального уточнения проведен в работе [36].

Таблица 1. Зависимость времени решения и ускорения от числа потоков

P	Время решения, с	Ускорение
1	1170.00	—
2	600.27	1.95
4	308.34	3.79
8	162.82	7.19
16	87.95	13.30
32	54.43	21.50
64	50.80	23.20
96	47.90	24.42



а) зависимость времени от числа потоков



б) зависимость ускорения от числа потоков

Рис. 3. Графики времени работы и ускорения параллельного алгоритма

С помощью алгоритма глобального поиска для каждой температуры были получены наборы констант скоростей, минимизирующие функцию отклонения расчетных концентраций от экспериментальных (6) для каждой температуры. В качестве весовых коэффициентов v_i в формуле (6) были использованы значения $v_i = 1.0, 1 \leqslant i \leqslant 5, v_6 = 4.4$, что соответствует большему влиянию последнего вещества. В ходе численных экспериментов было установлено, что равномерное взвешивание всех компонентов ($v_i = 1$) приводит к систематическому завышению расчетных значений для C_2H_6. Увеличение его весового коэффициента до 4.4 позволило скорректировать эту погрешность и добиться более равномерного распределения ошибки по всем компонентам.

После того, как с помощью параллельного АГП были получены наборы констант скоростей для каждой температуры, осуществлялся поиск предэкспоненциального множителя k_0 и энергии активации E_a (Дж/моль). Для этого метод наименьших квадратов применялся к соотношению, полученному логарифмированием уравнения Аррениуса:

$$\ln(k) = \ln(k_0) - \frac{E_a}{RT}, \tag{14}$$

где R — универсальная газовая постоянная, Дж/(моль · К); T — температура, K. Здесь в качестве k подставлялись полученные параллельным алгоритмом глобального поиска значения констант скоростей, соответствующие различным температурам T.

Далее методом наименьших квадратов были рассчитаны величины k_0 и E_a для всех стадий; полученные значения приведены в табл. 2.

Таблица 2. Рассчитанные кинетические параметры реакций (1)

№ реакции	$ln(k_0)$	E_a, **кДж/моль**
1	15.09	73.28
2	4.65	7.51
3	3.19	8.88
4	4.28	5.87
4’	10.6	57.31

Полученные значения энергии активации были сопоставлены со значениями, представленными в других работах по исследованию процесса паровой конверсии углеводородов на родиевых катализаторах. В работе [37] был изучен паровой риформинг метана, этана, пропана и бутана на монолитном сотовом катализаторе на основе родия. Эффективные энергии активации паровой конверсии пропана и этана в процессе парового риформинга природного газа составляют 51 и 50 кДж/моль соответственно. В работе [38] был представлен механизм поверхностной реакции частичного окисления и паровой конверсии пропана на катализаторе Rh/Al_2O_3. Процессы разрыва C-C и C-H связей в пропане характеризуются невысокими энергиями активации в пределах 20–70 кДж/моль.

Экспериментальные исследования реакции метанирования CO_2 на родиевых катализаторах [39] показали, что энергия аткивации этого процесса варьируется от 16.2 до 24.0 ккал/моль (67.8–100.4 кДж/моль). Однако эти значения были получены для катализатора Rh/Al_2O_3, в случае же применения оксида Ce0.75Zr0.25O2 в качестве носителя следует ожидать более низкие значения, вследствие наличия в этом оксиде кислородных вакансий, которые сильнее активируют диоксид углерода в процессе метанирования CO_2. Для процессов паровой конверсии метана при использовании родиевых катализаторов наблюдаемая энергия активации выше, чем в паровом риформинге более тяжелых углеводородов. Это связано с более прочной связью C-H в молекуле метана и ее низкой поляризации.

В работах [37, 40, 41] было показано, что для родиевых катализаторов энергия активации паровой конверсии метана сильно зависит от условий проведения реакции и состава используемого катализатора и варьируется от 40 до 90 кДж/моль.

Таким образом, для родиевых катализаторов наблюдаемые энергии активации в процессах конверсии пропана и этана обычно составляют 20–70 кДж/моль, конверсия метана характеризуется более высокими значениями энергии активации (40–90 кДж/моль). Для процесса метанирования диоксида углерода — 67.8–100.4 кДж/моль. Энергии активации, полученные в нашем исследовании, соответствуют литературным данным, что указывает на корректность используемого приближения.

Расчетные парциальные давления были пересчитаны с учетом k_0 и E_a, полученных методом наименьших квадратов. Было рассчитано среднее (по всем температурам) абсолютное отклонение расчетных парциальных давлений от экспериментальных для каждого вещества по формуле

$$\varepsilon_i = \frac{\sum_{j=1}^{T_{count}} \left| \overline{P_{ij}} - \widetilde{P_{ij}} \right|}{T_{count}}, \quad i \in [1, n], \tag{15}$$

где $\overline{P_{ij}}$, $\widetilde{P_{ij}}$ — экспериментальное и расчетное значения парциальных давлений вещества i для температуры j, атм; T_{count} — количество температур; n — количество измеряемых веществ. В табл. 3 представлены средние значения абсолютных отклонений расчетных пар-

циальных давлений от экспериментальных, а также максимальные значения экспериментальных парциальных давлений соответствующих веществ.

Таблица 3. Средние абсолютные отклонения расчетных парциальных давлений от экспериментальных и максимальные экспериментальные парциальные давления

Вещество	ε_i, **атм**	$\overline{P_{max}}$, **атм**
C_3H_8	0.009	0.060
H_2O	0.013	0.370
CO_2	0.010	0.049
H_2	0.015	0.079
CH_4	0.062	0.626
C_2H_6	0.006	0.030

Статистические характеристики можно считать хорошими для H_2O и CH_4, но недостаточными для веществ C_3H_8, CO_2, H_2, C_2H_6. На рис. 4 представлены графики зависимости экспериментальных и расчетных парциальных давлений от температуры для C_3H_8, H_2O, CO_2, H_2, CH_4, C_2H_6. Экспериментальные данные обозначены синим цветом, а расчетные — оранжевым. Полученные результаты подтверждают, что модель соответствует эксперименту и может использоваться для прогнозирования состава продуктов паровой конверсии углеводородов.

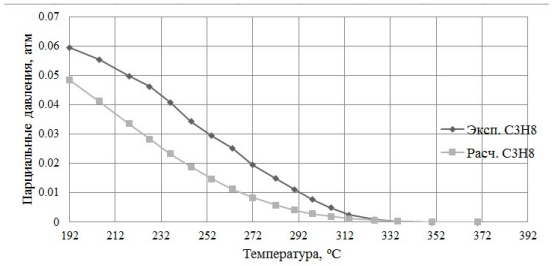
Качественное соответствие модели экспериментальным данным является приемлемым. Однако количественная аппроксимация в настоящий момент не достаточна для веществ C_3H_8, CO_2, H_2, C_2H_6. В продолжение работы предполагается включить в модель дополнительные параметры для более точного описания зависимостей.

Заключение

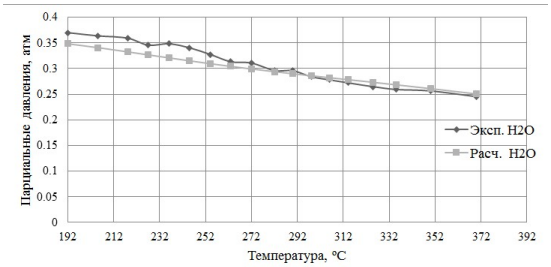
Результатом проведенного исследования является решение задачи выбора кинетических параметров для процесса низкотемпературной паровой конверсии пропана на 1 вес.% $Rh/Ce_{0.4}Zr_{0.5}Y_{0.05}La_{0.05}O_2$ катализаторе. Найденные кинетические параметры будут использоваться для получения газовых смесей с необходимыми характеристиками. Также в дальнейших исследованиях будет проведено усложнение полученной модели для учета реакций паровой конверсии более тяжелых углеводородов (бутана, пентана), входящих в состав ПНГ. Это позволит увеличить диапазон условий применимости модели и заранее оценивать возможность применения процесса НТПК для эффективной утилизации ПНГ на различных месторождениях.

В планы дальнейших работ входит использование других схем распараллеливания, способных повысить эффективность использования параллельных вычислительных систем для решения задач глобальной оптимизации. Сложности с достижением значительных показателей ускорения объясняются следующими двумя обстоятельствами.

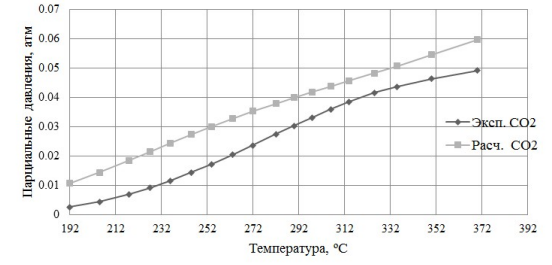
Во-первых, при реализации параллельного алгоритма была применена синхронная схема распараллеливания. Управляющий поток-мастер (в котором работал метод оптимизации) при выполнении шага 5 алгоритма ожидал окончания проведения всех испытаний, проводимых рабочими потоками. В решаемой задаче время проведения одного испытания зависит от времени решения системы ОДУ и может варьироваться в зависимости от конкретных значений параметров. Возможный путь к повышению эффективности распарал-



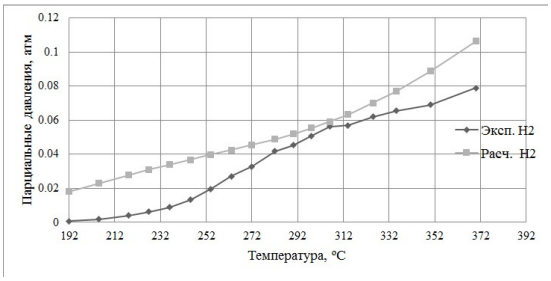
а) C_3H_8



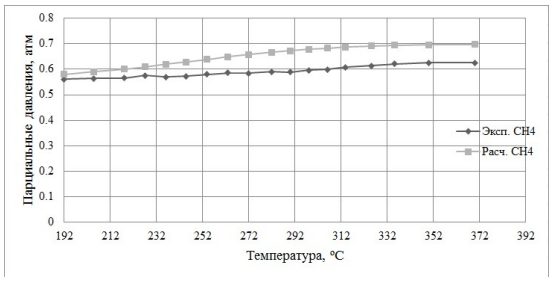
б) H_2O



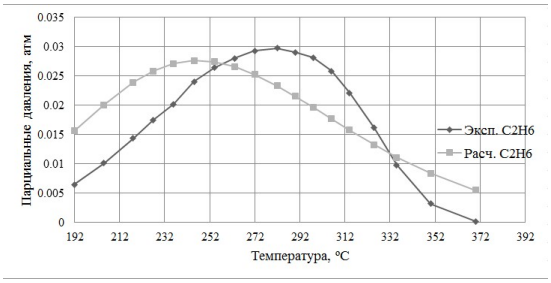
в) CO_2



г) H_2



д) CH_4



е) C_2H_6

Рис. 4. Сопоставление экспериментальных и расчетных парциальных давлений

леливания — реализация асинхронной схемы проведения испытаний на шаге 5 алгоритма. В этом случае результаты испытания, полученные одним из вычислителей, могут быть отправлены для обработки мастеру немедленно, без ожидания окончания других испытаний. После чего вычислитель получает от мастера новую точку, в которой начнет проведение очередного испытания. Асинхронная схема распараллеливания, теоретически, способна обеспечить полную загрузку задействованных вычислительных ресурсов даже в том случае, когда время выполнения испытаний различно в разных точках области поиска.

Во-вторых, при масштабном распараллеливании «узким местом» становится локальный поиск, который выполняется для уточнения решения после завершения фазы глобального поиска. Локальный поиск требует незначительного (по сравнению с глобальным) числа поисковых испытаний, но эти испытания выполняются последовательно. И при параллельном запуске это незначительное число последовательных испытаний становится сопоставимым с числом параллельных итераций глобального поиска. В дальнейшем планируется использовать параллельную реализацию метода локального поиска на завершающем этапе работы, что способно повысить эффективность распараллеливания в целом.

Наконец, планируется развитие математических основ используемых алгоритмов параллельной глобальной оптимизации. В задаче поиска параметров модели, обеспечивающих минимум невязки между расчетными и экспериментальными данными, теоретическое зна-

чение целевой функции в точке глобального минимума является известным — оно равно нулю. При этом точка глобального минимума (соответствующая лучшим значениям параметров модели) является неизвестной и подлежит определению. На практике нулевое значение невязки может и не достигаться, если исследуемая модель плохо соответствует эксперименту, но в таком случае целевая функция будет положительной. Информацию о нижней границе значений целевой функции можно использовать для разработки нового алгоритма оптимизации, который позволит как гарантировать сходимость к глобальному минимуму, так и ускорить процесс его отыскания.

Работа выполнена при поддержке Министерства образования и науки Российской Федерации, проекты № FSWR-2026-0007 (разработка параллельного алгоритма и проведение вычислительных экспериментов, К.А. Баркалов, И.Г. Лебедев) и № FMRS-2025-0031 (разработка кинетической модели, И.М. Губайдуллин).

Литература

1. Uskov S.I., Potemkin D.I., Shigarov A.B., *et al.* Low-temperature steam conversion of flare gases for various applications // Chemical Engineering Journal. 2019. Vol. 368. P. 533–540. DOI: 10.1016/j.cej.2019.02.189.
2. Zyryanova M.M., Snytnikov P.V., Shigarov A.B., *et al.* Kinetic features of methane steam reforming on nickel catalysts // Fuel. 2014. Vol. 135. P. 76–82. DOI: 10.1016/j.fuel.2014.06.032.
3. Enikeeva L.V., Potemkin D.I., Uskov S.I., *et al.* Gravitational search algorithm for determining the optimal kinetic parameters of propane pre-reforming reaction // Reaction Kinetics, Mechanisms and Catalysis. 2021. Vol. 132. P. 111–122. DOI: 10.1007/s11144-021-01927-8.
4. Potemkin D.I., Uskov S.I., Gorlova A.M., *et al.* Low-temperature steam conversion of natural gas to methane–hydrogen mixtures // Catalysis in Industry. 2020. Vol. 12. P. 244–249. DOI: 10.1134/S2070050420030101.
5. Shigarov A.B., Uskov S.I., Potemkin D.I., Snytnikov P.V. Experimental verification of kinetics and internal diffusion impact on low temperature steam reforming of a propane–methane mixture over Ni-based catalyst // Chemical Engineering Journal. 2022. Vol. 429. P. 132205. DOI: 10.1016/j.cej.2021.132205.
6. Урлуков А.С., Усков С.И., Потемкин Д.И., Снытников П.В. Каталитическая конверсия факельного газа на Rh-катализаторах с последующей прямой монетизацией // Катализ в промышленности. 2022. Т. 22, № 4. С. 51—57. DOI: 10.1134/S2070050422040110.
7. Урлуков А.С., Усков С.И., Потемкин Д.И. и др. Оптимизация состава Rh-содержащих катализаторов для применения в низкотемпературной паровой конверсии легких углеводородов // Экология и промышленность России. 2023. Т. 27, № 6. С. 17—23.
8. Garkul I.A., Zadesenets A.V., Filatov E.Y., *et al.* Double oxalates of Rh(III) with Cu(II) and Zn(II)—Effective precursors of nanoalloys for hydrogen production by steam reforming of propane // International Journal of Hydrogen Energy. 2024. Vol. 82. P. 611–623. DOI: 10.1016/j.ijhydene.2024.07.446.

9. Urlukov A.S., Uskov S.I., Sobyanin V.A., *et al.* Ethane formation via catalytic low-temperature steam reforming of C_{3+}-alkanes // Chemical Engineering Journal. 2022. Vol. 446. P. 136993. DOI: 10.1016/j.cej.2022.136993.

10. Zadesenets A.V., Garkul I.A., Filatov E.Y., *et al.* Double oxalates of Rh(III) with Ni(II) and Co(II)—Effective precursors of nanoalloys for hydrocarbons steam reforming // International Journal of Hydrogen Energy. 2023. Vol. 48, no. 59. P. 22428–22438. DOI: 10.1016/j.ijhydene.2023.01.365.

11. Strongin R.G., Sergeyev Y.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000. 416 p. DOI: 10.1007/978-1-4615-4677-1.

12. Barkalov K., Lebedev I. Solving multidimensional global optimization problems using graphics accelerators // Supercomputing. Vol. 687. Cham: Springer, 2016. P. 224–235. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-55669-7_18.

13. Arcotumapathy V., Alenazey F.S., Al-Otaibi R.L., *et al.* Mechanistic investigation of methane steam reforming over Ce-promoted Ni/SBA-15 catalyst // Applied Petrochemical Research. 2015. Vol. 5, no. 4. P. 393–404. DOI: 10.1007/s13203-015-0121-2.

14. Batebi D., Abedini R., Mosayebi A. Kinetic modeling of combined steam and CO_2 reforming of methane over the Ni–Pd/Al_2O_3 catalyst using Langmuir–Hinshelwood and Langmuir–Freundlich isotherms // Industrial & Engineering Chemistry Research. 2021. Vol. 60, no. 2. P. 851–863. DOI: 10.1021/acs.iecr.0c04566.

15. Мазитов А.А., Осипова А.Г., Ахметов И.В., Губайдуллин И.М. Решение обратной задачи химической кинетики на примере реакции синтеза бензилиденбензиламина // Журнал Средневолжского математического общества. 2016. Т. 18, № 3. С. 145—152.

16. Губайдуллин И.М., Рябов В.В., Тихонова М.В. Применение индексного метода глобальной оптимизации при решении обратных задач химической кинетики // Вычислительные методы и программирование. 2011. Т. 12, № 1. С. 137—145.

17. Morlanes N., Lezcano G., Yerrayya A., *et al.* Improving robustness of kinetic models for steam reforming based on artificial neural networks and ab initio calculations // Chemical Engineering Journal. 2022. Vol. 433. P. 133201. DOI: 10.1016/j.cej.2021.133201.

18. Woo Y., Park J.M., Bae J.W., Park M.J. Kinetic modeling of the steam reforming of light hydrocarbon mixture from waste resources: Effects of gas composition on hydrogen production // International Journal of Hydrogen Energy. 2023. Vol. 48, no. 41. P. 15383–15391. DOI: 10.1016/j.ijhydene.2023.01.050.

19. Пиявский С.А. Один алгоритм отыскания абсолютного экстремума функции // Журнал вычислительной математики и математической физики. 1972. Т. 12, № 4. С. 888—896. DOI: 10.1016/0041-5553(72)90115-2.

20. Shubert B. A sequential method seeking the global maximum of a function // SIAM Journal on Numerical Analysis. 1972. Vol. 9, no. 3. P. 379–388. DOI: 10.1137/0709036.

21. Evtushenko Y.G., Posypkin M.A. A deterministic approach to global box-constrained optimization // Optimization Letters. 2013. Vol. 7, no. 4. P. 819–829. DOI: 10.1007/s11590-012-0452-1.

22. Žilinskas A., Žilinskas J. Global optimization based on a statistical model and simplicial partitioning // Computers & Operations Research. 2010. Vol. 37, no. 10. P. 1759–1767.

23. Pintér J.D. Global optimization in action: Continuous and Lipschitz optimization: Algorithms, implementations and applications. Dordrecht: Kluwer Academic Publishers, 1996. 480 p.

24. Jones D.R. The DIRECT global optimization algorithm // Encyclopedia of Optimization. Boston: Springer, 2009. P. 725–735. DOI: 10.1007/978-0-387-74759-0_128.

25. Yang X.-S. Engineering optimization: An introduction with metaheuristic applications. Hoboken: John Wiley & Sons, 2013. 384 p. DOI: 10.1002/9780470640425.

26. Handbook of metaheuristics / ed. by M. Gendreau, J.-Y. Potvin. 2nd ed. New York: Springer, 2010. 640 p. DOI: 10.1007/978-1-4419-1665-5.

27. Eiben A.E., Smith J.E. Introduction to evolutionary computing. 2nd ed. Berlin: Springer, 2015. 294 p. DOI: 10.1007/978-3-662-44874-8.

28. Kvasov D.E., Mukhametzhanov M.S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case // Applied Mathematics and Computation. 2018. Vol. 318. P. 245–259. DOI: 10.1016/j.amc.2017.05.014.

29. Sergeyev Y.D., Kvasov D.E., Mukhametzhanov M.S. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget // Scientific Reports. 2018. Vol. 8. P. 453. DOI: 10.1038/s41598-017-18940-4.

30. Gergel V., Barkalov K., Sysoyev A. A novel supercomputer software system for solving time-consuming global optimization problems // Numerical Algebra, Control & Optimization. 2018. Vol. 8, no. 1. P. 47–62. DOI: 10.3934/naco.2018003.

31. Strongin R., Gergel V., Barkalov K., Sysoyev A. Generalized parallel computational schemes for time-consuming global optimization // Lobachevskii Journal of Mathematics. 2018. Vol. 39, no. 4. P. 576–586. DOI: 10.1134/S1995080218040133.

32. Sergeyev Y.D., Kvasov D.E. Global search based on efficient diagonal partitions and a set of Lipschitz constants // SIAM Journal on Optimization. 2006. Vol. 16, no. 3. P. 910–937. DOI: 10.1137/040621132.

33. Žilinskas A. Branch and bound algorithm for multidimensional scaling with city-block metric // Journal of Global Optimization. 2008. Vol. 43, no. 2–3. P. 357–372. DOI: 10.1007/s10898-008-9306-x.

34. Sergeyev Y.D., Strongin R.G., Lera D. Introduction to global optimization exploiting space-filling curves. New York: Springer, 2013. 148 p. DOI: 10.1007/978-1-4614-8042-6.

35. Химмельблау Д. Прикладное нелинейное программирование. М.: Мир, 1975. 535 с.

36. Barkalov K., Lebedev I., Karchkov D. Analysis and elimination of bottlenecks in parallel algorithm for solving global optimization problems // Supercomputing. Vol. 13708. Cham: Springer, 2022. P. 18–32. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-22941-1_2.

37. Schädel B.T., Duisberg M., Deutschmann O. Steam reforming of methane, ethane, propane, butane, and natural gas over a rhodium-based catalyst // Catalysis Today. 2009. Vol. 142, no. 1–2. P. 42–51. DOI: 10.1016/j.cattod.2009.01.008.

38. Karakaya C., Karadeniz H., Maier L., Deutschmann O. Surface reaction kinetics of the oxidation and reforming of propane over Rh/Al_2O_3 catalysts // ChemCatChem. 2017. Vol. 9, no. 4. P. 685–695. DOI: 10.1002/cctc.201601237.

39. Solymosi F., Erdöhelyi A., Bánsági T. Methanation of CO_2 on supported rhodium catalysts // Studies in Surface Science and Catalysis. 1981. Vol. 7. P. 1448–1449.

40. Zeppieri M., Villa P.L., Verdone N., *et al.* Kinetics of methane steam reforming reaction over nickel- and rhodium-based catalysts // Applied Catalysis A: General. 2010. Vol. 387, no. 1–2. P. 147–154. DOI: 10.1016/J.APCATA.2010.08.017.

41. Jakobsen J.G., Jakobsen M., Chorkendorff I., Sehested J. Methane steam reforming kinetics for a rhodium-based catalyst // Catalysis Letters. 2010. Vol. 140, no. 3. P. 90–97. DOI: 10.1007/s10562-010-0436-7.

Баркалов Константин Александрович, д.т.н., доцент, заведующий кафедрой математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (Н. Новгород, Российская Федерация)

Варсеева Екатерина Николаевна, ассистент кафедры математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (Н. Новгород, Российская Федерация)

Лебедев Илья Генадьевич, к.т.н., доцент кафедры математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (Н. Новгород, Российская Федерация)

Хашпер Анна Леонидовна, главный специалист отдела бизнес-систем, ООО «СИБИНТЕК-СОФТ» (Уфа, Российская Федерация)

Урлуков Артем Сергеевич, аспирант, младший научный сотрудник отдела гетерогенного катализа, Институт катализа им. Г.К. Борескова СО РАН (Новосибирск, Российская Федерация)

Губайдуллин Ирек Марсович, профессор, д.ф.-м.н., заведующий лабораторией математической химии ИНК УФИЦ РАН, профессор кафедры технологии нефти и газа УГНТУ (Уфа, Российская Федерация)

Хисаметдинов Искандер Фанзилевич, бакалавр Евразийской политехнической школы УГНТУ (Уфа, Российская Федерация)

DOI: 10.14529/cmse260202

PARALLEL ALGORITHM OF GLOBAL OPTIMIZATION AND ITS USE FOR SOLVING INVERSE PROBLEMS OF CHEMICAL KINETICS

© 2026 K.A. Barkalov[1,2], E.N. Varseeva[1], I.G. Lebedev[1,2], A.L. Khashper[3], A.S. Urlukov[4,5], I.M. Gubaydullin[6], I.F. Khisametdinov[7]

[1] *Lobachevsky State University of Nizhny Novgorod (pr. Gagarina 23, N. Novgorod, 603022 Russia),*

[2] *Scientific and Educational Mathematical Center "Mathematics of Future Technologies" (pr. Gagarina 23, bldg. 2, N. Novgorod, 603022 Russia),*

[3] *LLC "SIBINTEK-SOFT" (Kommunalnyy Proezd 30, Khimki, 141401 Russia),*

[4] *Catalysis Institute, Siberian Branch of the Russian Academy of Sciences (pr. Academician Lavrentiev 5, Novosibirsk, 630090 Russia),*

[5] *Novosibirsk State University (st. Pyrogova 1, Novosibirsk, 630090 Russia),*

[6] *Institute of Petrochemistry and Catalysis (pr. Oktyabrsky 141, Ufa, 450075 Russia),*

[7] *Ufa State Petroleum Technological University (st. Kosmonavtov 1, Ufa, 450064 Russia)*

E-mail: konstantin.barkalov@itmm.unn.ru, varseeva@unn.ru, ilya.lebedev@itmm.unn.ru, anna.khashper@gmail.com, aurlukov@mail.ru, irekmars@mail.ru, iskanderkhisametdinov1@gmail.com

Received: 20.03.2026

The article discusses the use of parallel computing for selecting parameters of a mathematical model of low-temperature steam conversion process of hydrocarbons contained in associated petroleum gas. For this chemical process, it is necessary to develop its kinetic model, i.e., determine the corresponding kinetic reaction parameters. To achieve this, an inverse problem is solved where values of kinetic parameters are sought based on experimental data. Mathematically, the inverse problem of chemical kinetics corresponds to a global optimization problem. A parallel information-statistical algorithm for global search combined with local refinement of the best solution was used to solve this problem. The algorithm is deterministic and relies on the assumption of Lipschitz continuity of the objective function, which is typical for many other approaches to constructing global optimization methods. In this case, solving multidimensional problems reduces to solving equivalent one-dimensional problems. The corresponding reduction is based on using a Peano curve that maps the unit interval of the real axis onto a hypercube. Parallelization of the algorithm is organized according to the «master-workers» scheme with shared memory usage. The found optimal model parameters allowed adequately simulating the process of low-temperature steam conversion of light hydrocarbons using a catalyst.

Keywords: global optimization, multi-extremal functions, parallel computing, chemical kinetics, inverse problems.

FOR CITATION

Barkalov K.A., Varseeva E.N., Lebedev I.G. Parallel Algorithm of Global Optimization and Its Use for Solving Inverse Problems of Chemical Kinetics. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2026. Vol. 15, no. 2. P. 26–46. (in Russian) DOI: 10.14529/cmse260202.

References

1. Uskov S.I., Potemkin D.I., Shigarov A.B., *et al.* Low-temperature steam conversion of flare gases for various applications. Chemical Engineering Journal. 2019. Vol. 368. P. 533–540. DOI: 10.1016/j.cej.2019.02.189.
2. Zyryanova M.M., Snytnikov P.V., Shigarov A.B., *et al.* Kinetic features of methane steam reforming on nickel catalysts. Fuel. 2014. Vol. 135. P. 76–82. DOI: 10.1016/j.fuel.2014.06.032.
3. Enikeeva L.V., Potemkin D.I., Uskov S.I., *et al.* Gravitational search algorithm for determining the optimal kinetic parameters of propane pre-reforming reaction. Reaction Kinetics, Mechanisms and Catalysis. 2021. Vol. 132. P. 111–122. DOI: 10.1007/s11144-021-01927-8.
4. Potemkin D.I., Uskov S.I., Gorlova A.M., *et al.* Low-temperature steam conversion of natural gas to methane–hydrogen mixtures. Catalysis in Industry. 2020. Vol. 12. P. 244–249. DOI: 10.1134/S2070050420030101.
5. Shigarov A.B., Uskov S.I., Potemkin D.I., Snytnikov P.V. Experimental verification of kinetics and internal diffusion impact on low temperature steam reforming of a propane–methane mixture over Ni-based catalyst. Chemical Engineering Journal. 2022. Vol. 429. P. 132205. DOI: 10.1016/j.cej.2021.132205.
6. Urlukov A.S., Uskov S.I., Potemkin D.I., Snytnikov P.V. Catalytic conversion of flare gas on Rh-based catalysts with subsequent direct monetization. Kataliz v promyshlennosti. 2022. Vol. 22, no. 4. P. 51–57. (in Russian) DOI: 10.1134/S2070050422040110.
7. Urlukov A.S., Uskov S.I., Potemkin D.I., *et al.* Optimization of Rh-containing catalysts composition for application in low-temperature steam conversion of light hydrocarbons. Ecology and Industry of Russia. 2023. Vol. 27, no. 6. P. 17–23. (in Russian).
8. Garkul I.A., Zadesenets A.V., Filatov E.Y., *et al.* Double oxalates of Rh(III) with Cu(II) and Zn(II)—Effective precursors of nanoalloys for hydrogen production by steam reforming of propane. International Journal of Hydrogen Energy. 2024. Vol. 82. P. 611–623. DOI: 10.1016/j.ijhydene.2024.07.446.
9. Urlukov A.S., Uskov S.I., Sobyanin V.A., *et al.* Ethane formation via catalytic low-temperature steam reforming of C_{3+}-alkanes. Chemical Engineering Journal. 2022. Vol. 446. P. 136993. DOI: 10.1016/j.cej.2022.136993.
10. Zadesenets A.V., Garkul I.A., Filatov E.Y., *et al.* Double oxalates of Rh(III) with Ni(II) and Co(II)—Effective precursors of nanoalloys for hydrocarbons steam reforming. International Journal of Hydrogen Energy. 2023. Vol. 48, no. 59. P. 22428–22438. DOI: 10.1016/j.ijhydene.2023.01.365.
11. Strongin R.G., Sergeyev Y.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000. 416 p. DOI: 10.1007/978-1-4615-4677-1.
12. Barkalov K., Lebedev I. Solving multidimensional global optimization problems using graphics accelerators. Supercomputing. Vol. 687. Cham: Springer, 2016. P. 224–235. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-55669-7_18.

13. Arcotumapathy V., Alenazey F.S., Al-Otaibi R.L., *et al.* Mechanistic investigation of methane steam reforming over Ce-promoted Ni/SBA-15 catalyst. Applied Petrochemical Research. 2015. Vol. 5, no. 4. P. 393–404. DOI: 10.1007/s13203-015-0121-2.

14. Batebi D., Abedini R., Mosayebi A. Kinetic modeling of combined steam and CO_2 reforming of methane over the Ni–Pd/Al_2O_3 catalyst using Langmuir–Hinshelwood and Langmuir–Freundlich isotherms. Industrial & Engineering Chemistry Research. 2021. Vol. 60, no. 2. P. 851–863. DOI: 10.1021/acs.iecr.0c04566.

15. Mazitov A.A., Osipova A.G., Akhmetov I.V., Gubaydullin I.M. Solution of the inverse problem of chemical kinetics on the example of benzylidenebenzylamine synthesis reaction. Journal of Middle Volga Mathematical Society. 2016. Vol. 18, no. 3. P. 145–152. (in Russian).

16. Gubaidullin I.M., Ryabov V.V., Tikhonova M.V. Application of the index method of global optimization for solving inverse problems of chemical kinetics. Computational Methods and Programming. 2011. Vol. 12, no. 1. P. 137–145. (in Russian).

17. Morlanes N., Lezcano G., Yerrayya A., *et al.* Improving robustness of kinetic models for steam reforming based on artificial neural networks and ab initio calculations. Chemical Engineering Journal. 2022. Vol. 433. P. 133201. DOI: 10.1016/j.cej.2021.133201.

18. Woo Y., Park J.M., Bae J.W., Park M.J. Kinetic modeling of the steam reforming of light hydrocarbon mixture from waste resources: Effects of gas composition on hydrogen production. International Journal of Hydrogen Energy. 2023. Vol. 48, no. 41. P. 15383–15391. DOI: 10.1016/j.ijhydene.2023.01.050.

19. Piyavskii S.A. An algorithm for finding the absolute extremum of a function. USSR Computational Mathematics and Mathematical Physics. 1972. Vol. 12, no. 4. P. 57–67. (in Russian) DOI: 10.1016/0041-5553(72)90115-2.

20. Shubert B. A sequential method seeking the global maximum of a function. SIAM Journal on Numerical Analysis. 1972. Vol. 9, no. 3. P. 379–388. DOI: 10.1137/0709036.

21. Evtushenko Y.G., Posypkin M.A. A deterministic approach to global box-constrained optimization. Optimization Letters. 2013. Vol. 7, no. 4. P. 819–829. DOI: 10.1007/s11590-012-0452-1.

22. Žilinskas A., Žilinskas J. Global optimization based on a statistical model and simplicial partitioning. Computers & Operations Research. 2010. Vol. 37, no. 10. P. 1759–1767.

23. Pintér J.D. Global optimization in action: Continuous and Lipschitz optimization: Algorithms, implementations and applications. Dordrecht: Kluwer Academic Publishers, 1996. 480 p.

24. Jones D.R. The DIRECT global optimization algorithm. Encyclopedia of Optimization. Boston: Springer, 2009. P. 725–735. DOI: 10.1007/978-0-387-74759-0_128.

25. Yang X.-S. Engineering optimization: An introduction with metaheuristic applications. Hoboken: John Wiley & Sons, 2013. 384 p. DOI: 10.1002/9780470640425.

26. Handbook of metaheuristics / ed. by M. Gendreau, J.-Y. Potvin. 2nd ed. New York: Springer, 2010. 640 p. DOI: 10.1007/978-1-4419-1665-5.

27. Eiben A.E., Smith J.E. Introduction to evolutionary computing. 2nd ed. Berlin: Springer, 2015. 294 p. DOI: 10.1007/978-3-662-44874-8.

28. Kvasov D.E., Mukhametzhanov M.S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. Applied Mathematics and Computation. 2018. Vol. 318. P. 245–259. DOI: 10.1016/j.amc.2017.05.014.

29. Sergeyev Y.D., Kvasov D.E., Mukhametzhanov M.S. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Scientific Reports. 2018. Vol. 8. P. 453. DOI: 10.1038/s41598-017-18940-4.

30. Gergel V., Barkalov K., Sysoyev A. A novel supercomputer software system for solving time-consuming global optimization problems. Numerical Algebra, Control & Optimization. 2018. Vol. 8, no. 1. P. 47–62. DOI: 10.3934/naco.2018003.

31. Strongin R., Gergel V., Barkalov K., Sysoyev A. Generalized parallel computational schemes for time-consuming global optimization. Lobachevskii Journal of Mathematics. 2018. Vol. 39, no. 4. P. 576–586. DOI: 10.1134/S1995080218040133.

32. Sergeyev Y.D., Kvasov D.E. Global search based on efficient diagonal partitions and a set of Lipschitz constants. SIAM Journal on Optimization. 2006. Vol. 16, no. 3. P. 910–937. DOI: 10.1137/040621132.

33. Žilinskas A. Branch and bound algorithm for multidimensional scaling with city-block metric. Journal of Global Optimization. 2008. Vol. 43, no. 2–3. P. 357–372. DOI: 10.1007/s10898-008-9306-x.

34. Sergeyev Y.D., Strongin R.G., Lera D. Introduction to global optimization exploiting space-filling curves. New York: Springer, 2013. 148 p. DOI: 10.1007/978-1-4614-8042-6.

35. Himmelblau D.M. Applied nonlinear programming. Moscow: Mir, 1975. 535 p. (in Russian).

36. Barkalov K., Lebedev I., Karchkov D. Analysis and elimination of bottlenecks in parallel algorithm for solving global optimization problems. Supercomputing. Vol. 13708. Cham: Springer, 2022. P. 18–32. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-22941-1_2.

37. Schädel B.T., Duisberg M., Deutschmann O. Steam reforming of methane, ethane, propane, butane, and natural gas over a rhodium-based catalyst. Catalysis Today. 2009. Vol. 142, no. 1–2. P. 42–51. DOI: 10.1016/j.cattod.2009.01.008.

38. Karakaya C., Karadeniz H., Maier L., Deutschmann O. Surface reaction kinetics of the oxidation and reforming of propane over Rh/Al_2O_3 catalysts. ChemCatChem. 2017. Vol. 9, no. 4. P. 685–695. DOI: 10.1002/cctc.201601237.

39. Solymosi F., Erdöhelyi A., Bánsági T. Methanation of CO_2 on supported rhodium catalysts. Studies in Surface Science and Catalysis. 1981. Vol. 7. P. 1448–1449.

40. Zeppieri M., Villa P.L., Verdone N., *et al.* Kinetics of methane steam reforming reaction over nickel- and rhodium-based catalysts. Applied Catalysis A: General. 2010. Vol. 387, no. 1–2. P. 147–154. DOI: 10.1016/J.APCATA.2010.08.017.

41. Jakobsen J.G., Jakobsen M., Chorkendorff I., Sehested J. Methane steam reforming kinetics for a rhodium-based catalyst. Catalysis Letters. 2010. Vol. 140, no. 3. P. 90–97. DOI: 10.1007/s10562-010-0436-7.

УДК 519.612 DOI: 10.14529/cmse260203

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ РЕШЕНИЙ СПЕЦИАЛЬНОГО ВИДА В ЗАДАЧЕ МОДЕЛИРОВАНИЯ РАЗВИТИЯ ТРЕЩИНЫ ГИДРАВЛИЧЕСКОГО РАЗРЫВА ПЛАСТА*

© 2026 О.С. Борщук[1], Р.К. Газизов[1,2], Н.С. Белевцов[1,2]

[1] *ООО «РН-Технологии»*

(119607 Москва, Раменский б-р, д. 1),

[2] *Уфимский университет науки и технологий*

(450076 Уфа, ул. Заки Валиди, д. 32)

E-mail: os_ borshchuk@rosneft.ru, rk_ gazizov2@rosneft.ru, nsbelevtsov@rn-t.ru

Поступила в редакцию: 20.04.2026

Рассматриваются методы численного решения систем линейных алгебраических уравнений (СЛАУ) специального вида, возникающих при численном моделировании гидродинамических процессов в пористых средах и развития трещины гидравлического разрыва пласта (ГРП). Обсуждаемые методы используются в корпоративной линейке программного обеспечения, развиваемой в ПАО «НК «Роснефть». Особое внимание уделяется параллельным методам решения разреженных систем, применяемым в корпоративных симуляторах. Приводится математическая постановка задачи Академического турнира ПАО «НК «Роснефть» 2025 года, связанная с моделированием процесса ГРП. Особенностью задачи является совместное моделирование упругого поведения породы и течения жидкости в трещине, что приводит к необходимости многократного решения систем с матрицей блочного вида, содержащей как разреженные блоки, так и заполненный, отвечающий за уравнения геомеханики. Наличие заполненного блока существенно отличает рассматриваемые системы от классических разреженных систем гидродинамического моделирования и требует разработки специализированных алгоритмов. В рамках турнира участникам предлагалось решить последовательность из 551 СЛАУ специального вида. Проводится сравнительный анализ решений команд-финалистов с точки зрения эффективности используемых параллельных методов решения линейных систем.

Ключевые слова: системы линейных алгебраических уравнений, гидравлический разрыв пласта, предобуславливание, итерационные методы, разреженные матрицы, параллельные вычисления, академический турнир.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Борщук О.С., Газизов Р.К., Белевцов Н.С. Сравнительный анализ методов решения систем линейных алгебраических решений специального вида в задаче моделирования развития трещины гидравлического разрыва пласта // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2026. Т. 15, № 2. С. 47–64. DOI: 10.14529/cmse260203.

Введение

Во многих задачах моделирования залежей углеводородов наиболее трудоемким является решение систем линейных алгебраических уравнений (СЛАУ), имеющих специальный вид и большую размерность, которое занимает до 90 % общего времени моделирования. Поэтому использование параллельных вычислений является необходимым условием эффек-

*Статья рекомендована к публикации Программным комитетом Всероссийской научной конференции с международным участием «Параллельные вычислительные технологии (ПаВТ) 2026».

тивности решения задач моделирования процессов нефтедобычи, которые часто требуют значительных вычислительных ресурсов.

НК «Роснефть», занимающаяся созданием собственной линейки симуляторов для моделирования физических процессов, происходящих в нефтяных резервуарах, начиная с 2000 года активно использует технологии параллельного программирования. Развитие этих технологий в компании за последние 25 лет тесно связано с развитием параллельных вычислений: от использования многопроцессорных кластерных систем с интерфейсом MPI и технологий многопоточного вычисления OpenMP до современных технологий распараллеливания на видеокартах с использованием технологии CUDA.

Разработка высокопроизводительных алгоритмов решения СЛАУ приобретает особую актуальность в условиях постоянного усложнения геологических, геофизических и гидродинамических моделей. Например, гидродинамическое моделирование месторождений порождает разреженные СЛАУ огромной размерности, число неизвестных в которых может достигать сотен миллионов. Не менее значимую роль решение СЛАУ играет при численном моделировании гидравлического разрыва пласта (ГРП), где дискретизация связанных уравнений механики деформируемого твердого тела, фильтрации жидкости разрыва и распространения трещины приводит к необходимости многократного решения систем с изменяющейся на каждом шаге структурой матрицы. С ростом объемов данных сейсморазведки, геофизического исследования скважин (ГИС) и данных мониторинга эксплуатации месторождений размерность систем также достигает миллионов уравнений. При этом матрицы СЛАУ, возникающие в рассматриваемых задачах, зачастую обладают специфическими свойствами — высокой обусловленностью, определенным характером разреженности, блочной структурой и др. Все вышеперечисленное обуславливает высокую важность разработки и программной реализации эффективных параллельных алгоритмов решения СЛАУ и построения предобуславливателей для цифрового моделирования нефтегазовых месторождений.

С целью привлечения внимания специалистов в области прикладной математики из вузов и академической среды к решению задач нефтегазодобывающей отрасли ПАО «НК «Роснефть» с 2023 года проводит Академические турниры. На турнирах формулируются задачи, представляющие научный и практический интерес для НК «Роснефть». Участникам турнира 2025 года было предложено разработать эффективные алгоритмы численного решения систем линейных алгебраических уравнений специального вида, возникающих в задаче моделирования развития трещины гидравлического разрыва пласта.

Анализ результатов участников турнира показал, что применявшиеся алгоритмы и технологии во многом схожи с теми, которые регулярно применяются при решении СЛАУ с разреженными матрицами. В частности, многие методы использовались нами ранее в задачах моделирования фильтрационных течений многофазных жидкостей в пористых средах, решаемых в корпоративном симуляторе «РН-КИМ». Опыт практического применения таких методов для решения соответствующих СЛАУ обсуждается в разделе 1. Оставшаяся часть статьи посвящена задаче Академического турнира 2025 года: в разделах 2 и 3 приводится математическая модель гидравлического разрыва пласта и формулируется постановка задачи Академического турнира; в разделе 4 обсуждаются подходы к ее решению, предложенные участниками соревнования. Основные результаты работы сформулированы в разделе «Заключение».

1. Решение систем линейных уравнений в задачах гидродинамического моделирования

Основным процессом, рассматриваемым в задачах моделирования нефтедобычи, является фильтрация флюида в пористой среде. Решение таких задач связано с колоссальным объемом вычислений. Поэтому развитие параллельных технологий в ПАО «НК «Роснефть» тесно связано с их использованием в задачах фильтрации. Важно отметить, что параллельные методы решения систем линейных алгебраических уравнений, используемые в корпоративном гидродинамическом симуляторе, также могут быть использованы для решения соответствующих задач в других областях нефтегазового инжиниринга.

Известны несколько подходов к численной реализации математической модели многофазных фильтрационных течений углеводородов в пористой среде (см., например, [1]). В настоящее время наиболее эффективной признана полностью неявная схема (Fully Implicit), реализация которой приводит к необходимости решения СЛАУ с не более чем семью блочными элементами в строке, где размерность блока определяется моделью фильтрации. Так как матрицы соответствующих СЛАУ на практике являются сильно разреженными и плохо обусловленными, для решения принято использовать предобусловленные итерационные методы подпространства Крылова [2], например, метод бисопряженных градиентов со стабилизацией (BiCGStab) или метод обобщенных минимальных невязок (GMRES).

Одним из эффективных предобуславливателей, применяемых в задаче численного моделирования многофазной фильтрации углеводородов, является неполное LU-разложение без заполнения (с нулевым заполнением) — ILU(0), также используются и другие предобуславливатели класса ILU. Более сложный, но в то же время и более эффективный подход заключается в применении специализированного двухступенчатого предобуславливателя CPR (Constrained Pressure Residual) и его модификаций. В рамках CPR выделяется локальный предобуславливатель, в качестве которого часто используется алгебраический многосеточный метод (Algebraic Multigrid Method, AMG), а также глобальный предобуславливатель, например, упомянутый выше ILU(0). Такой подход позволяет эффективно распараллеливать основную часть вычислений. Так, например, в корпоративном гидродинамическом симуляторе «РН-КИМ» для MPI-версии полностью неявной схемы для модели с 1900000 активных ячеек достигнуто ускорение в 12.66 раз на 16 процессорах.

В работе [3] предложена модификация двухступенчатого предобуславливателя CPR (Constrained Pressure Residual) для решения разреженных СЛАУ, возникающих при численном моделировании многофазной фильтрации вязкой сжимаемой жидкости в пористой среде. Предложенная модификация ACPR (Adaptive CPR) основана на декомпозиции расчетной области на устойчивую и неустойчивую части по критерию CFL, что позволяет строить неполное LU-разложение не для всей матрицы, а лишь для подмножества неустойчивых ячеек. Тестирование на реальных двух- и трехфазных моделях размерностью до 1122000 ячеек показало сокращение времени решения линейных систем в среднем на 20% и снижение требуемого объема оперативной памяти на 35–40% по сравнению со стандартным CPR.

Новый этап в решении СЛАУ, возникающих в задаче фильтрации, начался с распространения гибридных вычислительных систем, в которые наряду с центральными процессорами (CPU) привычной архитектуры x86 устанавливались массивно-параллельные ускорители (графические процессоры (GPU) производства компаний NVIDIA, AMD или Intel). В работах [4, 5], посвященных применению графических процессоров для решения разрежен-

ных СЛАУ в рамках той же задачи гидродинамического моделирования, было показано, что решение рассматриваемых систем итерационным методом BiCGStab с предобуславливателем BILU(0) (блочное неполное LU-разложение) позволяет существенно сократить время решения таких систем (в 4.3–7.7 раза).

Кроме того, в [5] было предложено на первом шаге двухступенчатого предобуславливателя CPR вместо AMG использовать метод AIPS (Approximate Inverse Power Series), аппроксимирующий обратную матрицу на основе степенного разложения в ряд Неймана. Проведенное сравнение предобуславливателей BILU(0), CPR-AMG и CPR-AIPS показало, что CPR-AIPS имеет преимущество на некоторых СЛАУ. В частности, в [6] показано преимущество использования такого предобуславливателя за счет применения ориентированного на архитектуру GPU параллельного алгоритма решения линейных систем с трехдиагональной матрицей, состоящей из независимых блоков различного размера.

Другой способ применения метода AIPS приведен в работе [7]. А именно, предложено применять предобуславливатель AIPS на графических процессорах в качестве альтернативы неполному LU-разложению без заполнения ILU(0) при решении СЛАУ. Предпосылкой работы является низкий ресурс параллелизма метода ILU(0), быстродействие которого на GPU зачастую уступает реализации на CPU. Проведена адаптация алгоритмов построения и применения предобуславливателя AIPS с учетом блочной структуры матриц, при этом в параллельном алгоритме задействован метод блочной прогонки для пакетного решения систем с трехдиагональными матрицами. На наборе СЛАУ с размерностью от 2 до 18 миллионов на GPU NVIDIA A100 применение AIPS обеспечило снижение времени решения до 3.6 раза относительно ILU(0) в режиме одноступенчатого предобуславливателя и до 1.5 раза в качестве замены на второй ступени двухступенчатого предобуславливателя CPR.

Рассмотренные подходы реализуются в корпоративном гидродинамическом симуляторе «РН-КИМ». Для рабочих станций реализована многопоточная версия симулятора, которая эффективно задействует все ядра процессора. На практике это дает сокращение времени расчета до 9.5 раз при использовании 16 ядер — решение, оптимальное для повседневной работы с моделями малого и среднего масштаба. Для задач высокой детализации — модели с сотнями миллионов ячеек и десятками тысяч скважин — симулятор поддерживает распределенные вычисления на кластерных и суперкомпьютерных системах. При запуске на 32 узлах кластера достигнуто ускорение до 24 раз, что делает возможным расчет полномасштабных моделей в приемлемые сроки. Также использование кластерной версии позволяет снять ограничение по оперативной памяти для расчета гигантских моделей. Отдельное внимание уделено поддержке графических процессоров. Использование GPU NVIDIA позволяет ускорить расчеты в 1.5–3.4 раза по сравнению с классическим CPU-режимом. Для максимальной производительности доступна гибридная схема на основе технологий MPI и CUDA, обеспечивающая одновременную работу нескольких GPU в рамках одной задачи.

Следует отметить, что во всех перечисленных работах рассматриваются эффективные параллельные реализации итерационных методов и предобуславливателей для решения разреженных СЛАУ, возникающих при дискретизации уравнений фильтрации. В отличие от этого, СЛАУ, получаемые в процессе моделирования ГРП, обладают иной структурой: помимо разреженных блоков, связанных с уравнениями гидродинамики, система содержит плотный блок, отвечающий за уравнения геомеханики, что существенно усложняет задачу построения эффективных решателей.

2. Моделирование процесса развития трещин гидравлического разрыва пласта

Даже в своей самой простой форме гидравлический разрыв пласта является сложным процессом для моделирования, поскольку включает в себя взаимодействие по меньшей мере трех процессов: (1) механическую деформацию, вызванную давлением жидкости на поверхность трещины; (2) течение жидкости внутри трещины; (3) рост самой трещины. Деформация твердого тела (горной породы) моделируется с использованием теории линейной упругости, которая представлена интегральным уравнением, определяющим нелокальную зависимость между шириной трещины и давлением жидкости. Поток жидкости моделируется с использованием теории смазки, представленной нелинейным дифференциальным уравнением в частных производных, которое связывает скорость потока жидкости, ширину трещины и градиент давления. С другой стороны, критерий распространения трещины обычно определяется традиционным подходом к скорости выделения энергии в рамках теории линейной механики упругого разрушения (Linear Elastic Fracture Mechanics, LEFM), т.е. трещина растет, если коэффициент интенсивности напряжений на кончике трещины равен прочности породы.

Таким образом, рассматривается задача совместного моделирования упругого поведения породы и течения жидкости в трещине ГРП [8].

2.1. Уравнение упругости

Уравнения упругости используются для расчета ширины трещины в зависимости от «чистого» давления (локальное давление жидкости минус локальное минимальное смыкающее напряжение) в каждой точке поверхности трещины. Уравнения упругости в 3D для одной плоской трещины могут быть записаны в виде интегрального уравнения вида

$$Fw = \int_{\Omega(t)} C(x,y,\xi,\eta)w(\xi,\eta,t)\,d\xi\,d\eta = p(x,y,t) - \sigma_c(x,y), \tag{1}$$

где $p(x,y,t)$ — давление жидкости внутри трещины, σ_c — локальное минимальное смыкающее напряжение, $w(x,y,t)$ — раскрытие трещины (толщина). Ядро C содержит всю информацию об упругих свойствах среды. Предполагается, что $\Omega(t)$ — площадь, занимаемая трещиной в момент времени t.

2.2. Уравнение гидродинамики

В рассматриваемой задаче считается, что трещина является «плоской», т.е. радиус кривизны трещины много больше ее раскрытия.

Для плоской трещины, которая растет в трехмерной упругой среде, используют двумерные уравнения Рейнольдса движения жидкости

$$\frac{\partial w}{\partial t} = \nabla \cdot [D(w,p)(\nabla p - \rho \mathbf{g})] + Q, \tag{2}$$

где ρ — плотность жидкости; $\mathbf{g} = (0,g)^T$, g — ускорение свободного падения; $Q = Q(x,y,t)$ — функция, связанная с источниками и утечками.

Для неньютоновской (степенной) жидкости:

$$D(w,p) = N'\, w^{\frac{2n'+1}{n'}}\ |\nabla p - \rho\mathbf{g}|^{\frac{1-n'}{n'}}\,, \tag{3}$$

$$N' = \left(\frac{n'}{2n'+1}\right)\left(\frac{1}{2n'+1\,K'}\right)^{\frac{1}{n'}}, \tag{4}$$

$$|\nabla p - \rho\mathbf{g}| = \sqrt{\left(\frac{\partial p}{\partial x} - \rho g_x\right)^2 + \left(\frac{\partial p}{\partial y} - \rho g_y\right)^2}, \tag{5}$$

где n' — степенной показатель жидкости, обычно изменяется в пределах $0.1 < n' < 2$; K' — индекс консистенции; g_i — компонента вектора $\mathbf{g}$.

Для ньютоновских жидкостей $n' = 1$ и $K' = \mu$, где μ — вязкость жидкости.

2.3. Начальные и граничные условия

Вдоль границы трещины рассматривается граничное условие нулевого потока жидкости, заданное выражением

$$D(w,p)\,\mathbf{n}\cdot(\nabla p - \rho\mathbf{g}) = 0, \tag{6}$$

где $\mathbf{n}$ — внешняя единичная нормаль к границе трещины.

Начальные условия:

$$w(x,y,0) = 0, \tag{7}$$

$$p(x,y,0) = \sigma_c(x,y). \tag{8}$$

2.4. Дискретизация по пространству

Для дискретизации уравнения (1) используется метод граничных элементов. В предположении, что трещина плоская, модель 3D упругости можно свести к плоской системе элементов, покрывающих плоскость трещины [9, 10]. Пример плоской сетки представлен на рис. 1: сетка содержит n_x узлов вдоль горизонтальной оси Ox и n_y узлов вдоль вертикальной оси Oy, через Δx и Δy обозначены шаги сетки по осям Ox и Oy соответственно. Отметим, что без внешнего воздействия трещины ГРП на вертикальных скважинах обладают симметрией относительно ствола скважины, поэтому можно рассматривать только половину трещины.

Для дискретизации по пространству уравнения гидродинамики (2) используется 2D метод конечных элементов. Узлы дискретизации для уравнений упругости и гидродинамики совпадают.

После дискретизации по пространству система принимает вид:

$$Fw = p - \sigma_c, \tag{9}$$

$$\frac{\Delta w}{\Delta t} = H(w,p) + Q, \tag{10}$$

где w и p — неизвестные поля давления и раскрытия, Q — содержит все, что относится к источникам и стокам, F — полностью заполненная матрица упругости, $H(w,p)$ — нелинейный оператор, полученный при дискретизации оператора $\nabla\cdot[D(w,p)\nabla p]$ из уравнения (2).

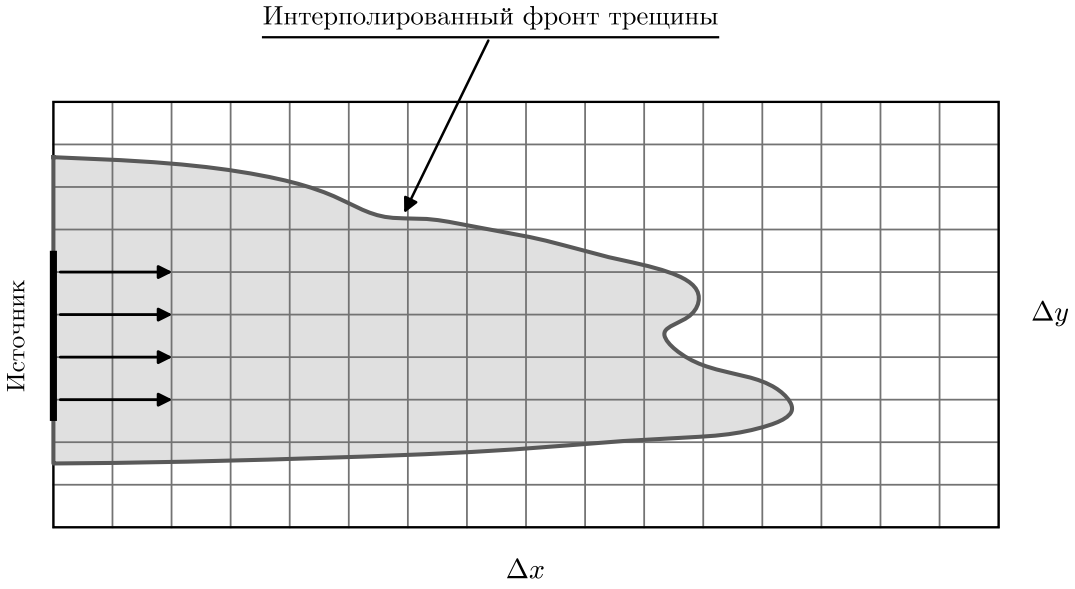


Рис. 1. Расчетная сетка в плоскости трещины

2.5. Дискретизация по времени

Для дискретизации по времени используется полностью неявная схема [8, 11]. Система с учетом начальных условий принимает вид

$$Fw^{k+1} - p^{k+1} + \sigma_c = 0, \tag{11}$$

$$w^{k+1} - w^k - H(w^{k+1}, p^{k+1})\Delta t - Q\Delta t = 0, \tag{12}$$

$$w^0 = 0, \tag{13}$$

$$p^0 = \sigma_c, \tag{14}$$

где w^k — значение поля раскрытия на k-м шаге по времени, w^{k+1} — неизвестное значение на $(k+1)$-м шаге. Аналогично для давления p.

Система (11)–(14) представляет собой систему нелинейных алгебраических уравнений, которую надо решать на каждом шаге по времени.

Стандартом при решении нелинейных алгебраических уравнений $g(x) = 0$ является использование метода Ньютона—Рафсона. Метод базируется на разложении нелинейной функции $g(x)$ в ряд Тейлора в окрестности начального приближения x_0:

$$g(x) = g(x_0) + g'(x_0)(x - x_0) + o(x - x_0). \tag{15}$$

Отбрасывая члены порядка малости $o(x - x_0)$ и используя $g(x) = 0$, получим

$$g'(x_0)\delta x = -g(x_0), \tag{16}$$

где $\delta x = x - x_0$.

Для системы нелинейных алгебраических уравнений $G(\vec{x}) = 0$, итерационный процесс (16) примет вид алгоритма 1, где $J(\vec{x}^k)$ — матрица Якоби для $G(\vec{x})$, элемент в строке i и столбце j равен $J_{ij}(\vec{x}^k) = \frac{\partial G_i}{\partial x_j}(\vec{x}^k)$, $G_i(\vec{x})$ — i-е уравнение в системе нелинейных алгебраических уравнений G.

Algorithm 1 Алгоритм Ньютона—Рафсона для решения систем нелинейных алгебраических уравнений.

$l \leftarrow 0$;
$\vec{x}^l \leftarrow \tilde{x}$ /* *Начальное приближение* */
$\varepsilon \leftarrow \tilde{\varepsilon}$ /* *Требуемая точность* */
while $|G(\vec{x}^l)|_2 \geqslant \varepsilon$ **do**
  Решить СЛАУ: $J(\vec{x}^l)\delta\vec{x}^{l+1} = -G(\vec{x}^l)$;
  $\vec{x}^{l+1} \leftarrow \vec{x}^l + \delta\vec{x}^{l+1}$;
  $l \leftarrow l + 1$;
end while

Далее, для сокращения записи, будем опускать знаки вектора над переменными, например, будем использовать x вместо $\vec{x}$.

В алгоритме 1 основной операцией является решение системы линейных алгебраических уравнений (СЛАУ)

$$Ju = v. \tag{17}$$

Таким образом, на каждой итерации метода Ньютона—Рафсона для системы (11)–(14) необходимо решить систему линейных алгебраических уравнений

$$\begin{bmatrix} F & E \\ C & B \end{bmatrix} \begin{bmatrix} \delta w \\ \delta p \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \tag{18}$$

где E — единичная матрица, F — заполненная матрица, связанная с уравнениями геомеханики, C и B — разреженные матрицы с одинаковым шаблоном разреженности, т.е. ненулевые элементы находятся в одних и тех же местах. C и B отвечают за уравнения гидродинамики. Блоки в системе (18) обладают следующими свойствами:

- F — строится один раз и не меняется в процессе эволюционного расчета, но для оптимизации из расчета исключены узлы с нулевым раскрытием w, из-за чего почти все системы в тесте имеют различающиеся размерности,
- C и B — меняются на каждом шаге по времени и на каждой итерации нелинейного решателя.

Отметим, что поскольку матрица F является заполненной, то с ростом размерности, сложность операции умножения на вектор для матрицы F растет как $O(n^2)$, где n — число строк матрицы F. Для матриц B и C сложность растет как $O(n)$. Очевидно, что при росте размерности матриц, умножение блока F быстро становится доминирующим. Итерационный метод Ньютона—Рафсона допускает использование приближенной матрицы $\tilde{J}$, что ведет к увеличению количества требуемых итераций для достижения сходимости, но в рассматриваемом случае позволяет значительно сократить трудоемкость решения системы (18).

В итоге решается система

$$\begin{bmatrix} \tilde{F} & E \\ C & B \end{bmatrix} \begin{bmatrix} \delta w \\ \delta p \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \tag{19}$$

где $\tilde{F}$ — разреженная матрица, полученная из F занулением элементов f_{ij}, $i \neq j$, таких, что $|f_{ij}| < \varepsilon\sqrt{\sum_{j=0}^{n} f_{ij}^2}$, $\varepsilon \in (0,1)$ — параметр, выбираемый из условия минимизации времени расчета.

В задании для Академического турнира 2025 матрица $\tilde{F}$ для упрощения обозначена F.

3. Задача академического турнира 2025

Перед участниками стояла задача разработки алгоритма и его реализации в виде программного модуля, предназначенного для решения последовательности из $M = 551$ системы линейных алгебраических уравнений (СЛАУ) с матрицами $N \times N$ размером от $N \approx 100$ до $N \approx 22\,000$. Все СЛАУ были получены в рамках решения одной эволюционной задачи развития трещины гидравлического разрыва пласта, математическая постановка которой приведена в предыдущем разделе. Рост размерности систем, как было сказано ранее, связан с ростом трещины ГРП, т.е. увеличением площади ее боковой поверхности. Для сокращения вычислительных затрат в расчет включаются только узлы, влияющие на трещину (на рис. 2 выделены красным); обозначения параметров сетки совпадают с введенными выше.

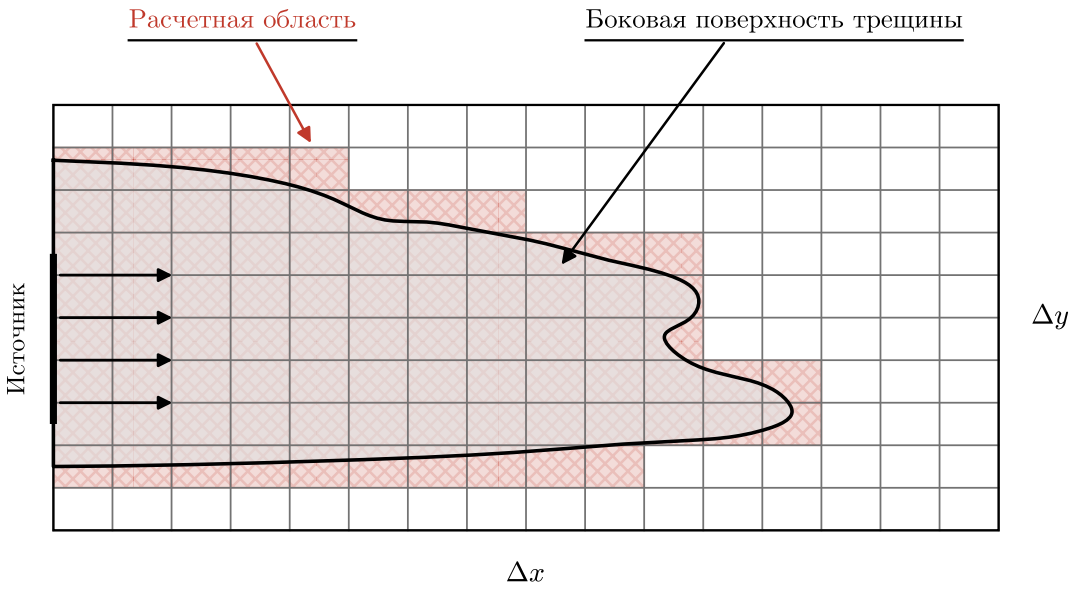


Рис. 2. Расчетная сетка с выделенными активными узлами

Каждая из решаемых систем имеет специальный блочный вид

$$\begin{bmatrix} F & E \\ C & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \tag{20}$$

в котором блоки F, E, C, B являются квадратными матрицами размерности $N_r \times N_r$, а полная система имеет размерность $N \times N = 2N_r \times 2N_r$, F — разреженная матрица с большим количеством ненулевых элементов в строке, B и C — разреженные матрицы с максимум девятью элементами в строке и совпадающими шаблонами разреженности (ненулевые элементы находятся в одних и тех же местах), E — единичная матрица. Матрицы B и C изначально построены на 9-точечном шаблоне, но в результате исключения строк и столбцов, относящихся к неактивным узлам, не имеют выраженной 9-диагональной структуры.

Систему (20) необходимо было решить с точностью τ относительно $l2$-нормы правой части:

$$\|v - A\tilde{u}\|_2 \leqslant \tau\|v\|_2, \tag{21}$$

где

$$A = \begin{bmatrix} F & E \\ C & B \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ b \end{bmatrix},$$

а $\tilde{u}$ — полученное приближенное решение системы.

В качестве исходных данных участникам турнира был предоставлен архив, содержащий $4 \times M$ файлов в формате NPZ библиотеки NumPy, где каждый файл именуется по схеме `prefix_xxxxxx.npz`: префикс `f_`, `b_`, `c_` или `r_` обозначает соответственно F-блок, B-блок, C-блок системы или вектор правой части b, а `xxxxxx` — шестизначный порядковый номер матрицы. Все матричные блоки (F, B, C) хранятся в компактном разреженном формате CSR (Compressed Sparse Row), оптимальном для матриц с высоким уровнем разреженности.

Разрабатываемый участниками программный модуль должен был представлять собой исполняемый файл для операционной системы Linux и обеспечивать решение системы (20) с точностью $\tau = 10^{-8}$. Итоговая оценочная метрика рассчитывалась как

$$T = \sum_{i=1}^{M} t_i Q(N_i), \tag{22}$$

где t_i — полное время работы модуля (загрузка, решение и вывод вектора) для i-й системы размерности N_i, а весовая функция задана выражением

$$Q(N_i) = \frac{1000.0}{\sqrt{N_i}}. \tag{23}$$

Для автоматического запуска тестирования и вычисления итоговой оценочной метрики разработанных программных модулей на тестовой вычислительной машине был реализован скрипт `run_bench.py`, который последовательно обрабатывает системы из тестового набора, измеряет время выполнения каждого шага, проверяет точность решения и суммирует взвешенную метрику T, отображая в выводе детальную статистику.

Отметим, что тестовый набор СЛАУ и исходные коды программных модулей, разработанных участниками турнира, не могут быть размещены в открытом доступе. Тестовые данные сформированы с помощью корпоративного программного обеспечения ПАО «НК «Роснефть» и относятся к информации ограниченного доступа, а программные реализации участников являются результатами интеллектуальной деятельности, права на которые определены регламентом Академического турнира. По этой причине ниже приводится содержательное описание использованных алгоритмов и результатов их единообразного тестирования, достаточное для самостоятельного воспроизведения рассмотренных подходов.

4. Анализ решений финалистов турнира

В 2025 году в Академическом турнире приняло участие 159 человек, представляющих 66 научных организаций и 44 вуза из 25 городов Российской Федерации. В Финал мероприятия были отобраны 10 команд. Далее в статье обсуждаются результаты 6 лучших команд-финалистов.

4.1. Решения призеров турнира

Команда, занявшая третье место, как и все оставшиеся финалисты Академического турнира, выполняла редукцию исходной блочной системы размерности $2N_r$ к системе с матрицей $S = C - BF$ размерности N_r. Участники использовали правое предобуславливание редуцированной системы $Sx = b$, решая при этом эквивалентную систему $SM^{-1}u = b$ с последующим восстановлением $x = M^{-1}u$. Для построения предобуславливателя $M = C - BF'$ было использовано трехдиагональное приближение F' плотной матрицы F. Построенная таким образом матрица оставалась разреженной — число ее ненулевых элементов не более чем втрое превышало число ненулевых элементов в C. Для приближенного вычисления $M^{-1}x$ было применено ILUT-разложение из библиотеки Eigen. Важной особенностью реализации стал отказ от явного формирования плотной матрицы S: матрично-векторное произведение Sx на каждой итерации вычислялось как действие оператора $Sx = Cx - B(Fx)$ тремя последовательными разреженными умножениями. Членами команды было показано, что при числе итераций, значительно меньшем размерности задачи, такой подход существенно выгоднее предварительного вычисления $S = C - BF$, требующего порядка N разреженных умножений матрицы на вектор. В качестве итерационного метода был выбран GMRES с перезапуском после определенного числа шагов и полученным правым предобуславливанием. Для ускорения процедуры ортогонализации в методе использовался классический алгоритм Грама—Шмидта, обеспечивающий эффективное распараллеливание средствами стандарта OpenMP.

Командой, занявшей второе место, было предложено использование блочно-трехдиагонального предобуславливателя, основанного на расширении ленточной матрицы в смысле максимальной энтропии. Предобуславливатель строился для матрицы $Z = C - BE$, где E — разреженное приближение матрицы F, полученное отсечением элементов по порогу ε. Обратная матрица к Z в смысле максимальной энтропии также имеет блочно-трехдиагональную структуру и представляется в виде суммы трех блочно-диагональных матриц, факторизации блоков которых могут выполняться полностью параллельно с использованием библиотеки SuperLU. В качестве итерационного метода был выбран GMRES, при этом матрично-векторные операции умножения вычислялись без явного формирования матрицы S. Для ускорения наиболее ресурсоемкой операции умножения плотной матрицы F на вектор была реализована параллельная схема с распределением строк между потоками стандарта OpenMP и организацией вычислений блоками по 32 элемента для эффективного использования векторных инструкций процессора. Кроме того, при построении предобуславливателя использовалась арифметика одинарной точности, что позволило улучшить использование кэша без ухудшения сходимости итерационного метода.

Победителем турнира был предложен подход, основанный на предобуславливании алгебраическим многосеточным методом (AMG) в форме V-цикла. Предобуславливатель строился не для точного дополнения Шура $S = C - BF$, а для его диагонального приближения $Q = C - B\mathrm{diag}(F)$, что обеспечило лучшую сходимость итерационного метода. В качестве сглаживателя в AMG использовалась комбинация метода Чебышева четвертого типа и ℓ_1-Якоби — беспараметрический подход, в котором верхняя оценка спектра получается из теоремы Гершгорина при масштабировании на ℓ_1-норму строк. Для построения оператора пролонгации применялся расширенный шаблон интерполяции, повышающий устойчивость метода. Редуцированная система $(C - BF)x = b$ решалась методом BiCGStab.

4.2. Подходы остальных финалистов

Помимо призеров соревнования, нельзя не упомянуть еще три команды, предложившие различные подходы к решению поставленной задачи.

Одна из команд-финалистов построила решение на базе библиотеки hypre, используя классический алгебраический многосеточный метод в качестве предобуславливателя для методов BiCGStab и Flexible GMRES. Отличительной чертой данного подхода стал автоматизированный подбор параметров многосеточного метода с помощью разработанного авторами ранее гибридного эволюционного алгоритма оптимизации.

Другая команда разработала оригинальный предобуславливатель на основе ряда Неймана. Идея состояла в аппроксимации обратной матрицы частичными суммами ряда Неймана с использованием масштабированного предобуславливателя Якоби. Для обеспечения сходимости ряда вводился масштабирующий параметр, а матрица упругости заменялась своим диагональным приближением. Авторами были получены теоретические оценки качества аппроксимации и экспериментально подобраны оптимальные параметры с адаптацией для систем различной размерности. Итерационным методом служил BiCGStab, реализация выполнена на языке C++ с использованием стандарта OpenMP и библиотеки OpenBLAS.

Наконец, третья команда применила нестандартный подход к поиску оптимальной стратегии решения, задействовав около двадцати параллельных ИИ-агентов, каждый из которых итеративно подбирал параметры решателя на основе обратной связи от оценочной метрики турнира. В результате такого автоматизированного перебора агенты независимо сошлись к одному и тому же подходу: ILU-предобуславливание с методом BiCGStab. В итоговом решении ILU-разложение вычислялось в арифметике одинарной точности, а решение методом BiCGStab проводилось в два этапа — сначала в арифметике одинарной точности (float32) для быстрого получения начального приближения, затем в арифметике двойной точности (float64) для достижения требуемой точности. Дополнительное ускорение достигалось за счет JIT-компиляции критичных функций средствами компилятора Numba и оптимизации загрузки данных.

4.3. Тестирование результатов

Для сравнительного анализа решений финалистов турнира было проведено единообразное тестирование всех программных модулей на одной и той же вычислительной машине с использованием полного набора из 551 СЛАУ. Помимо измерения общего времени работы модуля и вычисления оценочной метрики (22), каждый программный модуль выводил дополнительную статистическую информацию: число итераций итерационного метода, время построения предобуславливателя, время решения системы и достигнутую нормированную невязку. Это позволило детально проанализировать вклад различных этапов решения в общее время работы.

На рис. 3 представлены графики зависимости числа итераций, времени построения предобуславливателя и общего времени решения от номера системы для шести команд, подходы которых описаны в разделах 4.1 и 4.2. Номер системы монотонно связан с ее размерностью: ранние системы имеют размерность порядка сотен, поздние — порядка 10000. Из графиков числа итераций видно, что подходы на основе AMG-предобуславливания (команды 1 и 4) обеспечивают наименьшее и наиболее стабильное число итераций, тогда как решения на основе ILU-предобуславливания (команды 2 и 3) демонстрируют заметный рост числа итераций с увеличением размерности системы. Решение с предобуславливателем на

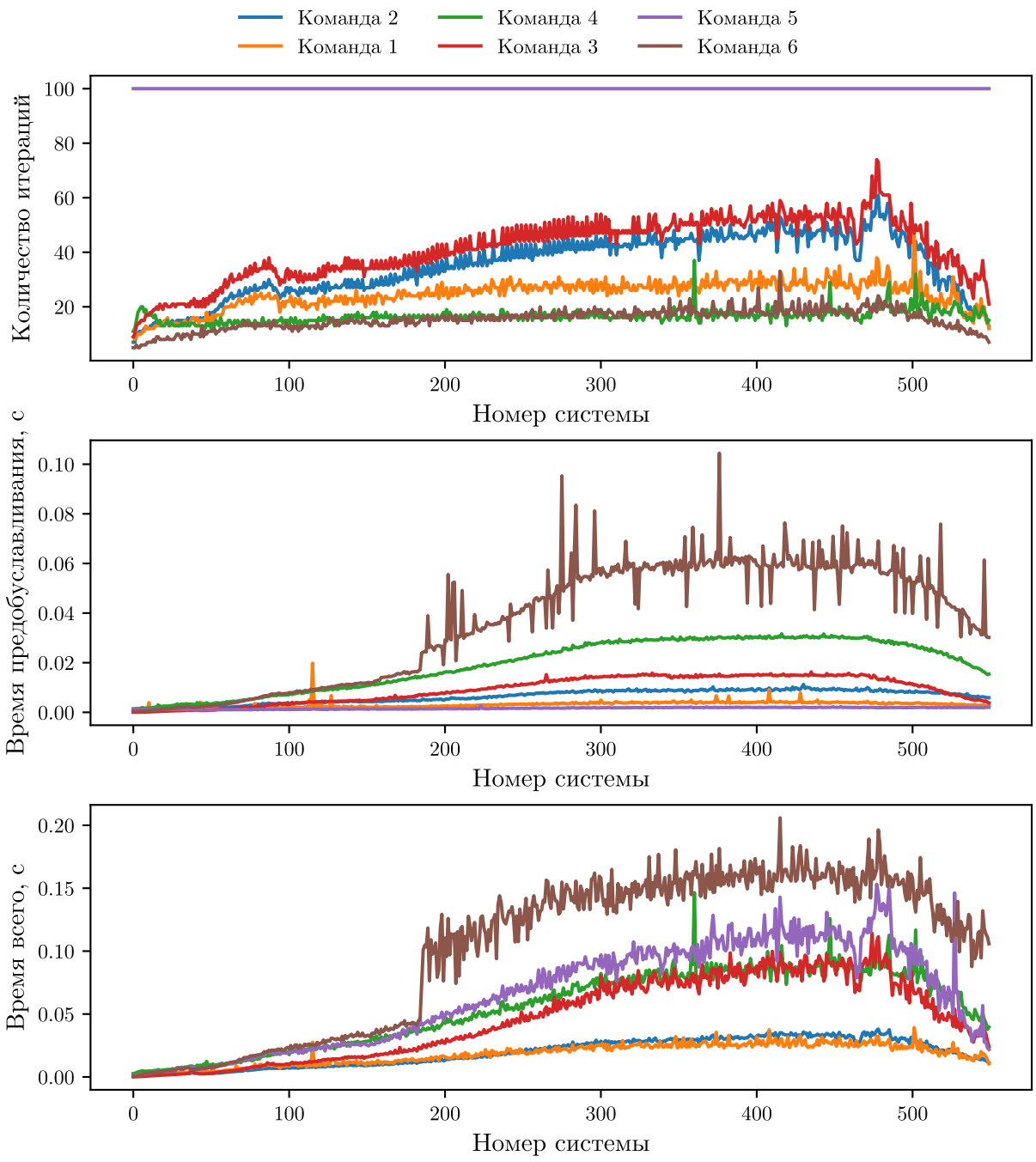


Рис. 3. Результаты тестирования решений шести финалистов турнира

основе разреженного приближения обратной матрицы с фиксированным числом итераций (SPAI) на уровне 100 формально сходилось (команда 5), однако при значительно большем числе итераций по сравнению с остальными подходами.

По времени построения предобуславливателя наиболее затратным оказался подход с ILU-предобуславливанием на основе BiCGStab (команда 6), реализованный на языке Python с использованием компилятора Numba. Наименее затратными оказались подходы, использующие SPAI-предобуславливание (команда 5) и AMG в форме V-цикла (команда 1).

На нижнем графике рис. 3 приведены общие времена решения всех систем для каждой команды. Данный параметр являлся основным критерием при определении победителей

турнира. Отчетливо видно расслоение подходов на несколько групп по быстродействию, при этом разрыв между лидирующей группой и остальными участниками увеличивается с ростом размерности систем.

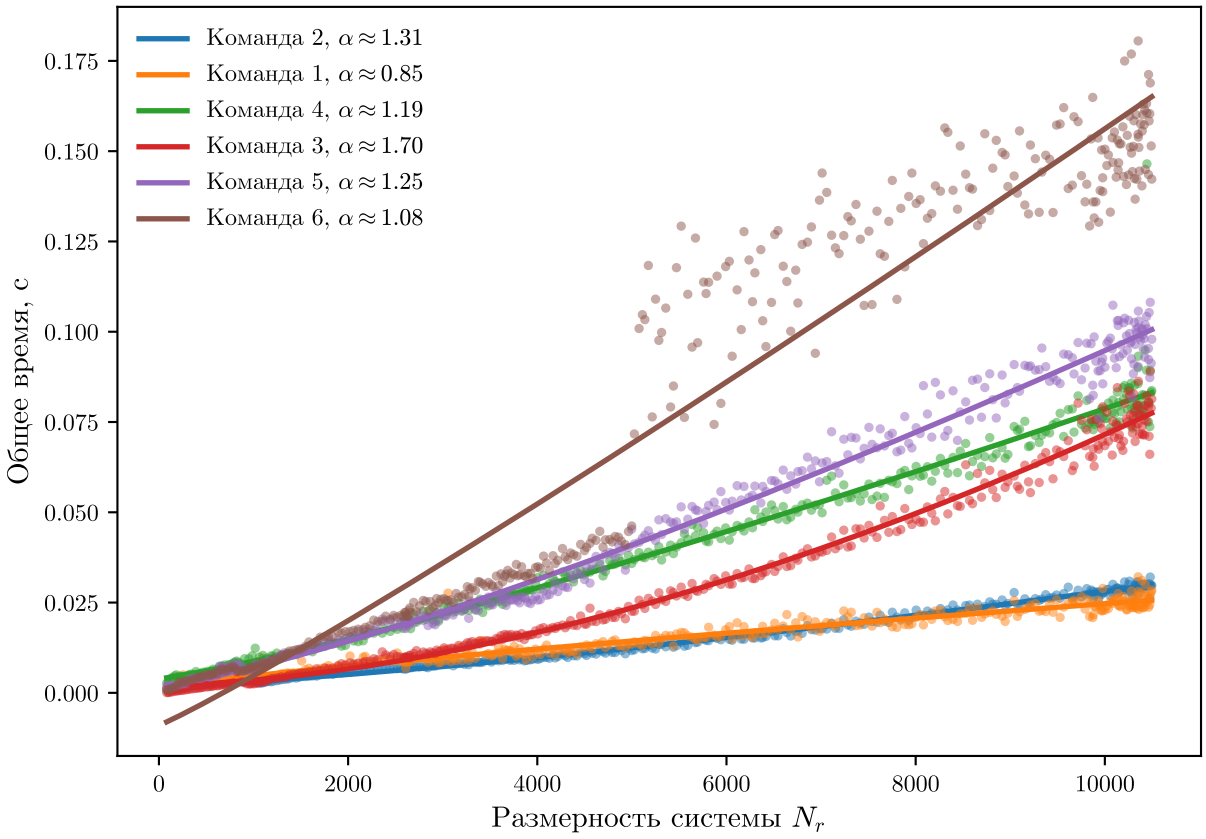


Рис. 4. Зависимость общего времени решения от размерности системы N_r

На рис. 4 представлена зависимость общего времени решения от размерности системы N_r с аппроксимацией степенной функцией вида $t = aN_r^{\alpha} + b$ (сплошные линии) для каждого из шести финалистов турнира. Показатель степени α характеризует асимптотическую сложность алгоритма. Наилучшую асимптотику демонстрирует подход (команда 1) на основе AMG-предобуславливания с методом BiCGStab ($\alpha \approx 0.85$), что согласуется с близкой к линейной сложностью алгебраического многосеточного метода. Решение на основе метода GMRES с блочно-трехдиагональным предобуславливателем (команда 2) показало $\alpha \approx 1.31$.

Из рис. 3 и рис. 4 видно, что два лучших решения оказались чрезвычайно близки по общему времени. Несмотря на принципиально различные подходы — AMG-предобуславливание с BiCGStab (команда 1) против блочно-трехдиагонального предобуславливателя с методом GMRES (команда 2) — оба решения демонстрируют сопоставимое общее время на всем диапазоне размерностей. Кривые общего времени пересекаются в районе размерности $N_r \approx 6000$ (рис. 4): для систем меньшего размера быстрее оказывается решение команды 2, для больших СЛАУ — решение команды 1. Различие в показателях асимптотики ($\alpha = 0.85$ против $\alpha = 1.31$) свидетельствует о том, что при дальнейшем росте размерности преимущество AMG-подхода будет увеличиваться.

Заключение

Рассмотрены системы линейных алгебраических уравнений специального блочного вида, возникающие при численном моделировании развития трещины гидравлического разрыва пласта. Отличительной особенностью этих систем является наличие как заполненного блока, обусловленного уравнениями геомеханики, так и разреженных блоков, отвечающих за гидродинамическую составляющую, что делает плохо применимыми стандартные подходы, ориентированные исключительно на разреженные системы.

Проведен обзор методов решения разреженных СЛАУ, развиваемых в рамках гидродинамического симулятора ПАО «НК «Роснефть», включая двухступенчатые предобуславливатели CPR и AIPS с реализацией на графических процессорах. Описана математическая модель процесса гидравлического разрыва пласта и показано, каким образом дискретизация связанных уравнений упругости и гидродинамики приводит к системам рассматриваемого блочного вида.

На основе результатов Академического турнира 2025 года проведен сравнительный анализ нескольких подходов к решению поставленной задачи. Результаты тестирования продемонстрировали, что наилучшую асимптотическую сложность обеспечивает подход на основе алгебраического многосеточного предобуславливания с методом BiCGStab: зависимость времени решения от размерности системы N_r аппроксимируется степенной функцией с показателем $\alpha \approx 0.85$, что свидетельствует о близкой к линейной сложности алгоритма. Вместе с тем подход на основе блочно-трехдиагонального предобуславливателя с методом GMRES показал сопоставимое быстродействие на системах умеренного размера, что свидетельствует о практической значимости обоих подходов.

Полученные результаты открывают несколько направлений дальнейших исследований. Во-первых, представляет интерес адаптация рассмотренных алгоритмов для выполнения на графических процессорах, что позволит использовать опыт GPU-реализаций, накопленный при решении задач гидродинамического моделирования. Во-вторых, актуальной задачей является исследование эффективности предложенных подходов на системах существенно большей размерности.

Литература

1. Байков В.А., Борщук О.С., Газизов Р.К. и др. Моделирование фильтрационных течений углеводородов в пористой среде на многопроцессорных вычислительных системах // Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ: Труды Всероссийской научной конференции. 2007. С. 187–189.

2. Saad Y. Iterative Methods for Sparse Linear Systems. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003.

3. Борщук О.С. О модификации двухступенчатого метода предобуславливания при численном решении задачи многофазной фильтрации вязкой сжимаемой жидкости в пористой среде // Вестник УГАТУ. Серия: Системный анализ, управление и обработка информации. 2009. Т. 12, № 1(30). С. 146–150.

4. Губайдуллин Р.Р., Репин Н.В., Юлдашев А.В. Опыт применения графических процессоров для решения разреженных систем линейных алгебраических уравнений в рамках задачи гидродинамического моделирования нефтегазовых месторождений // Вестник Уфимского государственного авиационного технического университета. 2015. Т. 19,

№ 4(70). С. 118–123.

5. Газизов Р.К., Губайдуллин Р.Р., Репин Н.В., Юлдашев А.В. Исследование эффективности различных предобуславливателей при параллельном решении разреженных систем линейных алгебраических уравнений на графических процессорах // Вестник Уфимского государственного авиационного технического университета. 2018. Т. 22, № 2(80). С. 105–112.
6. Юлдашев А.В., Репин Н.В., Спеле В.В. Параллельный предобусловливатель на основе степенного разложения обратной матрицы для решения разреженных линейных систем на графических процессорах // Вычислительные методы и программирование. 2019. Т. 20, № 4. С. 444–456. DOI: 10.26089/NumMet.v20r439.
7. Добровольцев А.С., Сохатский М.А., Юлдашев А.В. О возможности применения на GPU предобусловливателя AIPS в качестве альтернативы ILU(0) в задаче гидродинамического моделирования нефтегазовых месторождений // Параллельные вычислительные технологии (ПаВТ'2025): Труды международной научной конференции. 2025. С. 309.
8. Adachi J., Siebrits E., Peirce A.P., Desroches J. Computer simulation of hydraulic fractures // International Journal of Rock Mechanics and Mining Sciences. 2007. Vol. 44, no. 5. P. 739–757. DOI: 10.1016/j.ijrmms.2006.11.006.
9. Peirce A.P., Siebrits E. Uniform asymptotic approximations for accurate modeling of cracks in layered elastic media // International Journal of Fracture. 2002. Vol. 110. P. 205–239. DOI: 10.1023/A:1010861821959.
10. Peirce A.P., Siebrits E. The scaled flexibility matrix method for the efficient solution of boundary value problems in 2D and 3D layered elastic media // Computer Methods in Applied Mechanics and Engineering. 2001. Vol. 190, no. 45. P. 5935–5956. DOI: 10.1016/S0045-7825(01)00206-7.
11. Devloo P., Fernandes P.D., Gomes S.M., *et al.* A finite element model for three dimensional hydraulic fracturing // Mathematics and Computers in Simulation. 2006. Vol. 73, no. 1–4. P. 142–155. DOI: 10.1016/j.matcom.2006.06.020.

Борщук Олег Сергеевич, ООО «РН-Технологии» (Москва, Российская Федерация)

Газизов Рафаил Кавыевич, д.ф.-м.н., профессор, кафедра высокопроизводительных вычислений и дифференциальных уравнений, Уфимский университет науки и технологий (Уфа, Российская Федерация), ООО «РН-Технологии» (Москва, Российская Федерация)

Белевцов Никита Сергеевич, к.ф.-м.н., кафедра высокопроизводительных вычислений и дифференциальных уравнений, Уфимский университет науки и технологий (Уфа, Российская Федерация), ООО «РН-Технологии» (Москва, Российская Федерация)

DOI: 10.14529/cmse260203

COMPARATIVE ANALYSIS OF METHODS FOR SOLVING SYSTEMS OF LINEAR ALGEBRAIC EQUATIONS OF A SPECIAL FORM IN THE HYDRAULIC FRACTURE PROPAGATION MODELING PROBLEM

© 2026 O.S. Borshchuk[1], R.K. Gazizov[1,2], N.S. Belevtsov[1,2]

[1] *RN-Technologies LLC (blvd Ramenskiy 1, Moscow, 119607, Russia),*

[2] *Ufa University of Science and Technology (st. Zaki Validi 32, Ufa, 450076 Russia)*

E-mail: os_ borshchuk@rosneft.ru, rk_ gazizov2@rosneft.ru, nsbelevtsov@rn-t.ru

Received: 20.04.2026

Methods for the numerical solution of systems of linear algebraic equations (SLAE) of a special form arising in the simulation of fluid flow in porous media and of hydraulic fracture propagation are considered. The methods discussed are used in the corporate software product line developed at PJSC Rosneft Oil Company. Particular attention is paid to parallel methods for solving sparse systems that are employed in corporate simulators. The mathematical formulation of the problem of the 2025 Rosneft Academic Tournament, related to the simulation of hydraulic fracturing, is presented. A distinctive feature of the problem is the coupled modeling of the elastic behavior of the rock and of fluid flow within the fracture, which requires the repeated solution of systems with a block matrix containing both sparse blocks and a dense block corresponding to the geomechanics equations. The presence of the dense block substantially distinguishes these systems from the classical sparse systems of reservoir simulation and calls for specialized algorithms. As part of the tournament, participants were asked to solve a sequence of 551 SLAEs of this special form. A comparative analysis of the finalist teams' solutions is carried out in terms of the efficiency of the parallel methods used to solve the linear systems.

Keywords: systems of linear algebraic equations, hydraulic fracturing, preconditioning, iterative methods, sparse matrices, parallel computing, academic tournament.

FOR CITATION

Borshchuk O.S., Gazizov R.K., Belevtsov N.S. Comparative Analysis of Methods for Solving Systems of Linear Algebraic Equations of a Special Form in the Hydraulic Fracture Propagation Modeling Problem. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2026. Vol. 15, no. 2. P. 47–64. (in Russian) DOI: 10.14529/cmse260203.

References

1. Baykov V.A., Borshchuk O.S., Gazizov R.K., *et al.* Simulation of hydrocarbon filtration flows in porous media on multiprocessor computing systems. Scientific Service on the Internet: Multi-Core Computer World. 15 Years of RFBR: Proceedings of the All-Russian Scientific Conference. 2007. P. 187–189. (in Russian)

2. Saad Y. Iterative Methods for Sparse Linear Systems. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003.

3. Borshchuk O.S. On the modification of the two-stage preconditioning method for the numerical solution of the multiphase filtration problem of viscous compressible fluid in a porous medium.

Bulletin of Ufa State Aviation Technical University. Series: System Analysis, Control and Information Processing. 2009. Vol. 12, no. 1(30). P. 146–150. (in Russian)

4. Gubaydullin R.R., Repin N.V., Yuldashev A.V. Experience of using graphics processors for solving sparse systems of linear algebraic equations in the context of hydrodynamic simulation of oil and gas fields. Bulletin of Ufa State Aviation Technical University. 2015. Vol. 19, no. 4(70). P. 118–123. (in Russian)
5. Gazizov R.K., Gubaydullin R.R., Repin N.V., Yuldashev A.V. Efficiency study of various preconditioners for parallel solution of sparse systems of linear algebraic equations on graphics processors. Bulletin of Ufa State Aviation Technical University. 2018. Vol. 22, no. 2 (80). P. 105–112. (in Russian)
6. Yuldashev A.V., Repin N.V., Spele V.V. Parallel preconditioner based on power series expansion of the inverse matrix for solving sparse linear systems on graphics processors. Numerical Methods and Programming. 2019. Vol. 20, no. 4. P. 444–456. (in Russian) DOI: 10.26089/NumMet.v20r439.
7. Dobrovoltsev A.S., Sokhatsky M.A., Yuldashev A.V. On the possibility of applying the AIPS preconditioner on GPU as an alternative to ILU(0) in the problem of hydrodynamic simulation of oil and gas fields. Parallel Computational Technologies (PCT'2025): Proceedings of the International Scientific Conference. 2025. P. 309. (in Russian)
8. Adachi J., Siebrits E., Peirce A.P., Desroches J. Computer simulation of hydraulic fractures. International Journal of Rock Mechanics and Mining Sciences. 2007. Vol. 44, no. 5. P. 739–757. DOI: 10.1016/j.ijrmms.2006.11.006.
9. Peirce A.P., Siebrits E. Uniform asymptotic approximations for accurate modeling of cracks in layered elastic media. International Journal of Fracture. 2002. Vol. 110. P. 205–239. DOI: 10.1023/A:1010861821959.
10. Peirce A.P., Siebrits E. The scaled flexibility matrix method for the efficient solution of boundary value problems in 2D and 3D layered elastic media. Computer Methods in Applied Mechanics and Engineering. 2001. Vol. 190, no. 45. P. 5935–5956. DOI: 10.1016/S0045-7825(01)00206-7.
11. Devloo P., Fernandes P.D., Gomes S.M., *et al.* A finite element model for three dimensional hydraulic fracturing. Mathematics and Computers in Simulation. 2006. Vol. 73, no. 1–4. P. 142–155. DOI: 10.1016/j.matcom.2006.06.020.

УДК 544.18, 544.16, 544.332.2.03, 544.174.3 DOI: 10.14529/cmse260204

КВАНТОВО-ХИМИЧЕСКОЕ ИССЛЕДОВАНИЕ НЕКОТОРЫХ ВЫСОКОЭНЕРГЕТИЧЕСКИХ АЗОЛО-БИС(ТЕТРАЗОЛО)АЗИНОВ*

© 2026 В.М. Волохов[1], Е.С. Амосова[1], В.В. Парахин[2], Д.Б. Лемперт[1], О.А. Гончарова[1], В.В. Воеводин[3]

[1] *Федеральный исследовательский центр проблем химической физики и медицинской химии РАН (142432 Черноголовка, пр. Ак. Семенова, д. 1),*

[2] *Институт органической химии имени Н.Д. Зелинского РАН (119991 Москва, Ленинский пр., д. 47),*

[3] *Научно-исследовательский вычислительный центр Московского государственного университета имени М.В. Ломоносова (119234 Москва, ул. Ленинские горы, д. 1, стр. 4)*

E-mail: vvm@icp.ac.ru, aes@icp.ac.ru, parakhin@ioc.ac.ru, lempert@icp.ac.ru, goa@icp.ac.ru, voevodin@parallel.ru

Поступила в редакцию: 17.04.2026

Высокоэнергетические соединения широко применяются в различных областях промышленности и техники, что обусловливает актуальность разработки новых веществ этого типа и исследования их свойств. В работе представлено квантово-химическое исследование шести высокоэнергетических тетрациклов на основе бис(тетразоло)азинов, аннелированных незамещенными азолами. Цель исследования — определение ключевой энергетической характеристики этих соединений, энтальпии образования в газообразном состоянии, и анализ ее зависимости от молекулярной структуры. Расчеты выполнены в программном комплексе Gaussian 09 методами теории функционала плотности B3LYP/6-311+G(2d,p) и B3LYP/cc-pVTZ, комбинированным методом G4MP2, а также с использованием реализации G4MP2 в программном комплексе NWChem. Энтальпии образования определяли на основе реакций атомизации и изодесмических реакций. Показано, что энтальпия образования в целом возрастает при увеличении содержания азота и числа эндотермических связей в аннелированном азольном фрагменте; максимальное значение получено для трис(тетразоло)-s-триазина. Результаты реализации G4MP2 в NWChem хорошо согласуются с расчетами Gaussian 09, а использование нескольких вычислительных узлов позволяет уменьшить время выполнения задач. Рассчитаны и проанализированы ИК-спектры исследуемых соединений, установлено соответствие характеристических полос поглощения структурным фрагментам молекул. Квантово-химические расчеты выполнены с использованием суперкомпьютера «Ломоносов-2».

Ключевые слова: квантово-химические расчеты, высокоэнергетические материалы, энтальпия образования, тетрациклические соединения, трис(азоло)азины, трис(тетразоло)[1,3,5]триазины.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Волохов В.М., Амосова Е.С., Парахин В.В., Лемперт Д.Б., Гончарова О.А., Воеводин В.В. Квантово-химическое исследование некоторых высокоэнергетических азоло-бис(тетразоло)азинов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2026. Т. 15, № 2. С. 65–81. DOI: 10.14529/cmse260204.

Введение

Высокоэнергоемкие соединения (ВЭС) являются уникальными веществами, обладающими большим запасом химической энергии, которая может быть контролируемо высво-

*Статья рекомендована к публикации Программным комитетом Всероссийской научной конференции с международным участием «Параллельные вычислительные технологии (ПаВТ) 2026».

бождена в процессе горения или детонации. Одной из ключевых характеристик, определяющих энергетический потенциал ВЭС, является энтальпия образования (ΔH_f) [1–6]. Высокую величину ΔH_f ВЭС обеспечивает объединение нескольких азотистых гетероароматических ядер в конденсированную гетероциклическую систему [7–10]. Ранее наша научная группа с помощью квантово-химических вычислений выполнила исследования взаимосвязи структуры и ΔH_f ряда тетрациклических соединений, в которых бензол или 1,3,5-триазин конденсированы с тремя пиррольными, имидазольными, пиразольными, 1,2,3- или 1,2,4-триазольными циклами [11–17]. Но наибольшей величины ΔH_f в этой серии соединений позволяет достичь включение в тетрациклическую структуру тетразольных колец [18–20]. Настоящая работа посвящена изучению параметров структуры и энтальпии образования в газовой фазе ($\Delta H^0_{f(\text{г})}$) ряда незамещенных конденсированных тетрациклов **1–6** (рис. 1), в которых центральное пиразиновое, 1,2,4- или 1,3,5-триазиновое ядро аннелировано с двумя тетразольными кольцами, а также с дополнительным пиррольным, имидазольным, пиразольным, 1,2,3-триазольным, 1,2,4-триазольным или тетразольным циклом:

- пирроло[1,2-b]бис(тетразоло)[1,5-d:5',1'-f][1,2,4]триазин (1),
- имидазо[1,5-b]бис(тетразоло)[1,5-d:5',1'-f][1,2,4]триазин (2),
- пиразоло[1,5-b]бис(тетразоло)[1,5-d:5',1'-f][1,2,4]триазин (3),
- [1,2,4]триазоло[1,5-b]бис(тетразоло)[1,5-d:5',1'-f][1,2,4]триазин (4),
- 8H-[1,2,3]триазоло[4,5-e]бис(тетразоло)[1,5-a:5',1'-c]пиразин (5),
- трис(тетразоло)[1,5-a:1',5'-c:1",5"-e][1,3,5]триазин (6),

являющиеся пока гипотетическими структурами, из которых ранее как ВЭС теоретически рассматривался только тетрацикл **6** [21–24]. Указанные характеристики были установлены в результате высокопроизводительных квантово-химических расчетов. При этом следует отметить, что в представленных структурах отсутствуют атомы-окислители, поэтому коэффициент насыщения кислородом ($C_xH_yN_wO_z$ $\alpha = 2z/(4x+y)$) для всех этих соединений равен нулю $\alpha = 0$.

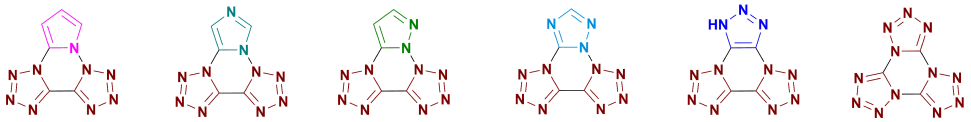


Рис. 1. Объекты исследования — ряд незамещенных азоло-ди(тетразоло)азинов **1–6**

Статья организована следующим образом. В разделе 1 представлена методика исследования. Раздел 2 посвящен обсуждению полученных результатов. В разделе 3 рассматриваются особенности проведения расчетов. В заключении изложены краткий обзор основных результатов исследования и выводы.

1. Методика исследования

Для расчетов использовали квантово-химические комплексы Gaussian 09 [25] и NWChem [26]. Геометрия изучаемых молекул была определена путем полной оптимизации всех геометрических параметров с использованием гибридного функционала плотности B3LYP [27, 28] с базисом 6-311+G(2d,p). Последующий расчет колебательных частот, выполненный с использованием аналитических первых и вторых производных без учета ангармонических поправок (отсутствие мнимых частот), подтвердил стабильность получен-

ных конфигураций. Энтальпия образования в газовой фазе ($\Delta H^0_{f(\mathrm{r})}$) исследуемых веществ рассчитана с использованием метода гибридного функционала плотности B3LYP [27, 28] с базисным набором cc-pVTZ [29] и 6-311+G(2d,p) и с использованием комбинированного метода G4MP2 [30, 31]. Для расчетов $\Delta H^0_{f(\mathrm{r})}$ в программе NWChem был написан модуль, воспроизводящий последовательность вычислений комбинированного метода G4MP2 по соответствующим формулам, опубликованным ранее в работах [31, 32],

$$E_0[G4(MP2)] = CCSD(FC,T)/6-31G(d) + \Delta E_{MP2} + \Delta E_{HF} + \Delta E(SO) + E(HLC) + E(ZPE),$$

где CCSD(FC,T)/6-31G(d) — это расчет энергии на уровне теории связанных кластеров с учетом тройных возбуждений CCSD(T) с базисным набором 6-31G(d) при использовании приближения замороженного остова; $\Delta \mathrm{E_{MP2}}$ и $\Delta \mathrm{E_{HF}}$ — энергетические поправки, рассчитанные соответственно методами MP2 и HF; ΔE(SO) — спин-орбитальная поправка, E(HLC) — поправка высшего уровня, а E(ZPE) — энергия нулевых колебаний.

Расчет ИК-спектров поглощения проводили с использованием метода гибридного функционала плотности B3LYP с базисом cc-pVTZ с введением масштабирующего коэффициента 0.967 [33].

Расчет ΔH^0_f исследуемых веществ проводили с помощью двух подходов:

- на основе реакции атомизации,
- на основе схем изодесмических реакций образования вещества.

Расчет методом, основанным на реакции атомизации для молекулы с формулой $C_xH_yN_z$, включал следующие этапы:

1. Определение энергии атомизации по формуле:

$$\sum D_0 = xE_0(C) + yE_0(H) + zE_0(N) - E_0(C_xH_yN_z),$$

   где $E_0(C)$, $E_0(H)$, $E_0(N)$ и $E_0(C_xH_yN_z)$ — рассчитанные полные энергии атомов и молекул.
2. Определение ΔH^0_f при температуре 0 K:

$$\Delta H^\circ_{\mathrm{f}}(C_xH_yN_z, K0) = x\Delta H^\circ_{\mathrm{f}}(C, K0) + y\Delta H^\circ_{\mathrm{f}}(H, K0) + z\Delta H^\circ_{\mathrm{f}}(N, K0) - \sum D_0,$$

   где первые три слагаемых — это энтальпии образования атомов в газовой фазе, взятые из баз данных NIST-JANAF [34].
3. Определение ΔH^0_f при температуре 298.15 K:

$$\begin{aligned}\Delta H^\circ_{\mathrm{f}}(C_xH_yN_z, K298) = {} & \Delta H^\circ_{\mathrm{f}}(C_xH_yN_z, K0) + \\ & + \big(H^0(C_xH_yN_z, K298) - H^0(C_xH_yN_z, K0)\big) - \\ & - x\big(H^0(\mathrm{C}, K298) - H^0(\mathrm{C}, K0)\big) - \\ & - y\big(H^0(\mathrm{H}, K298) - H^0(\mathrm{H}, K0)\big) - \\ & - z\big(H^0(\mathrm{N}, K298) - H^0(\mathrm{N}, K0)\big),\end{aligned}$$

   где второе слагаемое получают с помощью квантово-химического расчета, а с третьего по шестое слагаемые известны из эксперимента (или рассчитаны из экспериментальных молекулярных констант).

Для расчета ΔH_f^0 использовали следующие схемы изодесмических реакций, представленные на рис. 2. В расчете использовали полные энергии реагентов и продуктов реакций, рассчитанные с использованием гибридного функционала плотности B3LYP [27, 28] с базисом 6-311+G(2d,p) и энтальпии атомизации, рассчитанные с использованием комбинированного метода G4MP2 [30, 31].

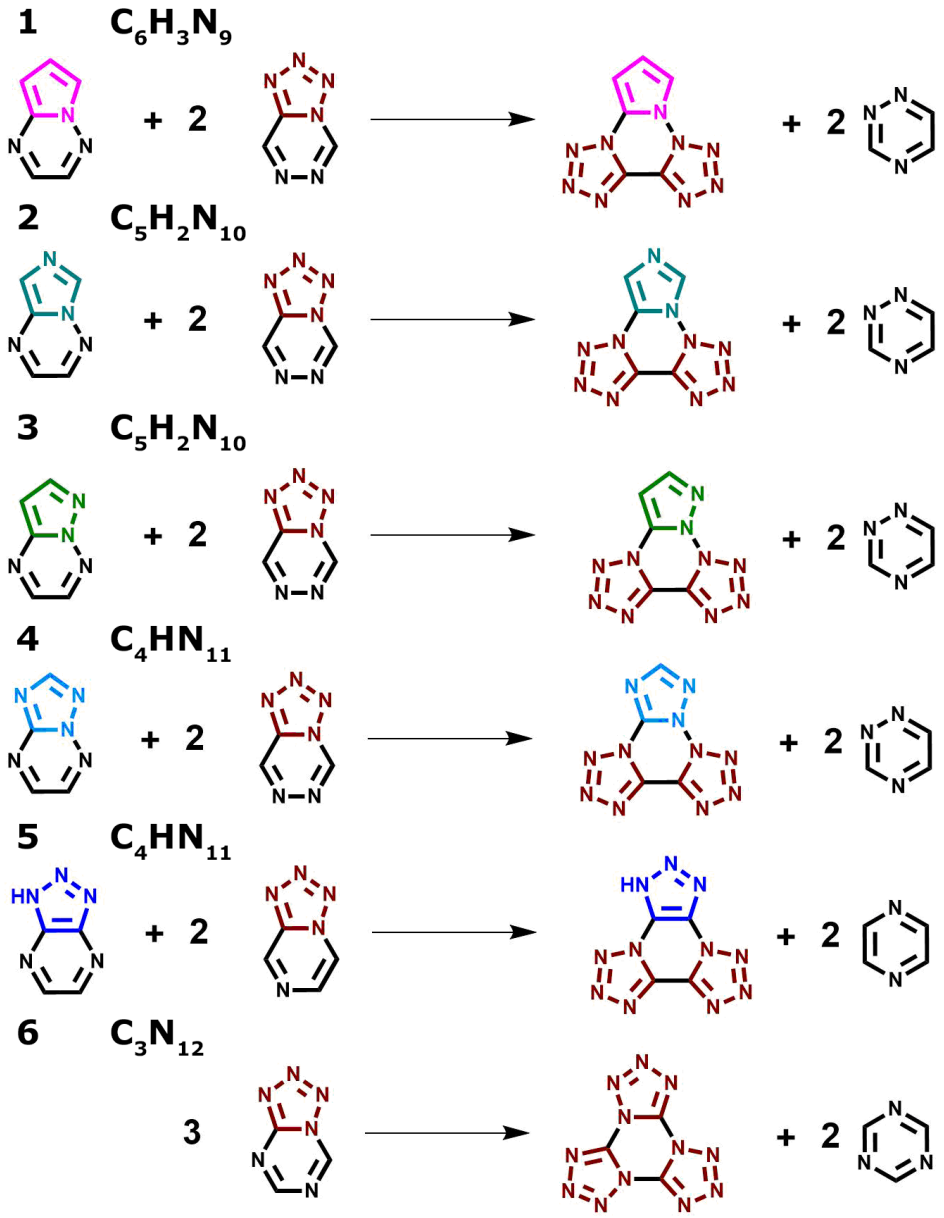


Рис. 2. Схемы изодесмических реакций, использованные для расчета энтальпии образования

2. Обсуждение результатов

2.1. Энтальпии образования

Оптимизированная геометрия представлена на рис. 3. Результаты расчетов ΔH_f^0 разными методами собраны в табл. 1.

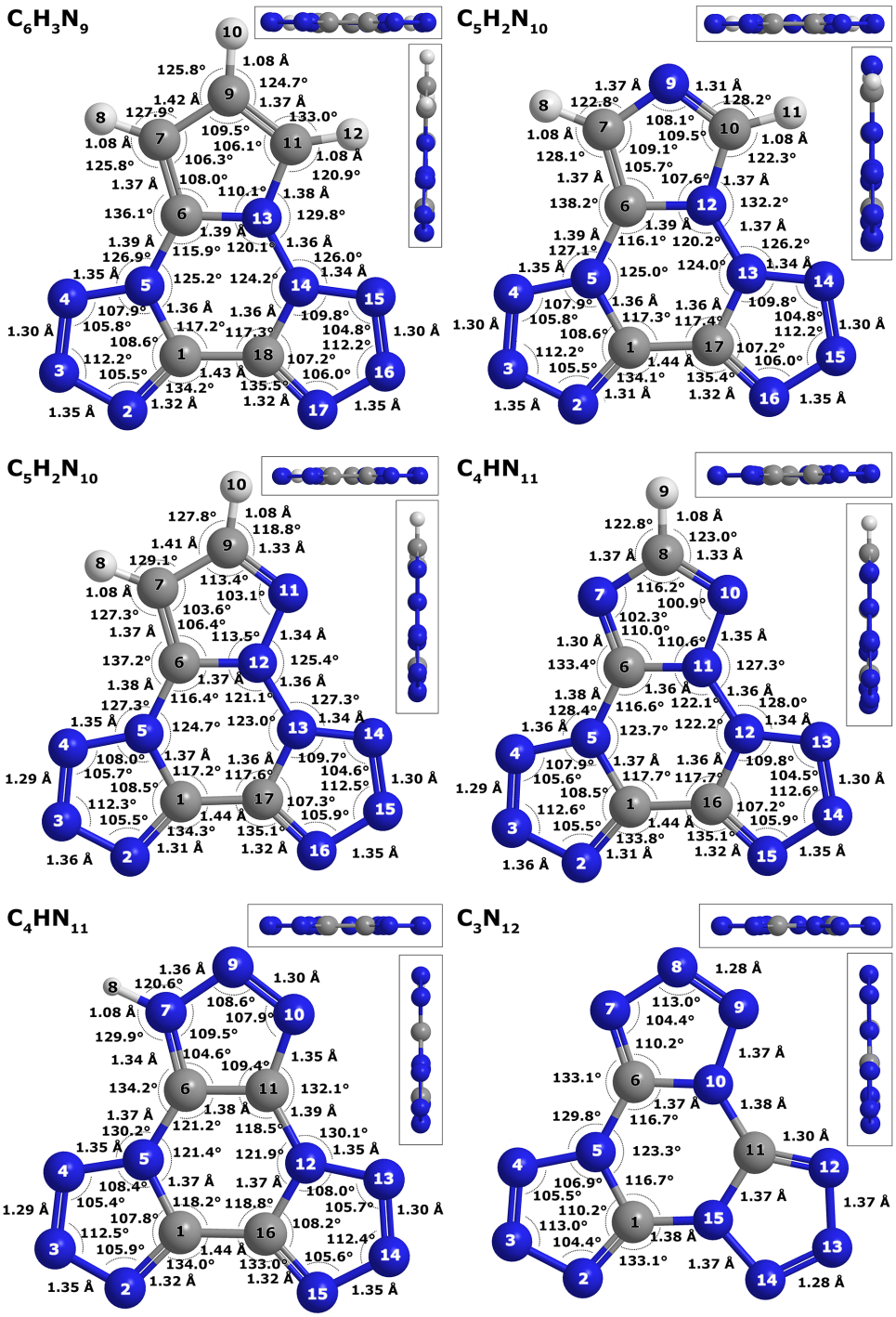


Рис. 3. Структуры и геометрические параметры (Å и °) тетрациклов **1–6** (рассчитанные на уровне B3LYP/6-311+G(2d,p))

Разница между наибольшими и наименьшими значениями ΔH_f^0, полученными с использованием различных методов и подходов к расчету, составляет 64–68 кДж/моль. Самые высокие значения ΔH_f^0 получаются при использовании теории функционала плотности B3LYP с базисным набором 6-311+G(2d,p) (1108–1272 кДж/моль, среднеквадратичное отклонение (СКО) от результатов G4MP2 — 47 кДж/моль). Наименьшие значения получаются при использовании теории функционала плотности B3LYP с базисным набором с

Таблица 1. Энтальпия образования в газовой фазе исследуемых соединений **1–6** (кДж/моль / *кДж/кг*)

No	Формула	N%*	$\Delta H^0_{f(\text{г}}$				
			B3LYP/ 6-311+G(2d,p)	G4MP2	G4MP2 (NWChem)	Reaction	B3LYP/ cc-pVTZ
1	$C_6H_3N_9$	62.7	1107.9 *5510.6*	**1040.5** ***5172.5***	1041.4 *5179.8*	1043.6 *5188.2*	1042.0 *5182.9*
2	$C_5H_2N_{10}$	69.3	1141.0 *5647.2*	**1089.6** ***5390.3***	1089.6 *5390.3*	1091.6 *5402.9*	1075.4 *5322.8*
3	$C_5H_2N_{10}$	69.3	1183.2 *5856.3*	**1132.5** ***5602.6***	1133.4 *5609.8*	1133.8 *5611.5*	1117.7 *5532.0*
4	C_4HN_{11}	75.9	1205.3 *5936.2*	**1169.1** ***5755.2***	1169.3 *5758.9*	1170.1 *5763.0*	1139.5 *5612.3*
5	C_4HN_{11}	75.9	1144.2 *5635.5*	**1109.2** ***5462.7***	1109.2 *5463.0*	1111.3 *5473.2*	1079.9 *5318.8*
6	C_3N_{12}	82.4	1272.3 *6235.7*	**1245.9** ***6106.5***	1245.8 *6105.7*	1247.0 *6141.6*	1203.8 *5899.9*

* N% — массовая доля азота

тройной дзета для всех соединений, кроме **1** (1042–1204 кДж/моль, СКО — 26 кДж/моль). Наименьшее отклонение от результатов, полученных с помощью комбинированного метода G4MP2, наблюдается при использовании этого же метода, реализованного с помощью модуля в NWChem, — СКО составляет менее 1 кДж/моль. Использование изодесмических реакций в расчетах также позволяет получить результаты близкие к G4MP2, СКО — 2 кДж/моль.

Анализ полученных результатов расчета показывает, что энтальпия образования в газовой фазе $\Delta H^0_{f(\text{г})}$ (и мольная, и удельная) в целом возрастает в ряду тетрациклов **1–6** (см. табл. 1). Изучаемые соединения различаются структурой одного из азольных циклов, аннелированных с центральным ядром, чем и обусловлена разница в величине их $\Delta H^0_{f(\text{г})}$. Рост энтальпии образования в ряду **1–6** сопровождается ростом N% от 62.7% до 82.4% и соответствует последовательности изменения аннелированных азолов: пиррол → имидазол → пиразол → триазол → тетразол, и объясняется увеличением в этом ряду количества эндотермических связей C–N, C=N, N–N, N=N, повышающих $\Delta H^0_{f(\text{г})}$. При этом одна из изучаемых структур — тетрацикл **5**, содержащий 1,2,3-триазольный цикл — выпадает из указанной последовательности, и связано это с тем, что в отличие ото всех остальных объектов ряда соединений, построенных вокруг 1,2,4- или 1,2,3-триазинового ядра, в соединении **5** центральным кольцом является пиразин, в котором меньше эндотермических связей с атомами азота по сравнению с триазиновыми циклами и, наоборот, имеет место дополнительная связь углерод–углерод, снижающая $\Delta H^0_{f(\text{г})}$. Следует отметить, что наибольшей энтальпией образования в рассматриваемой серии соединений обладает трис(тетразоло)-s-триазин (**6**), в структуре которого 1,3,5-триазиновое ядро аннелировано сразу с тремя высокоэнергоемкими тетразольными кольцами. Тетрацикл **6** характеризуется очень высоким содержанием азота 82.4% и его $\Delta H^0_{f(\text{г})}$ достигает рекордного значения 1246 кДж/моль (6107 кДж/кг).

Среди исследуемых в настоящей работе соединений лишь тетрацикл **6** ранее рассматривался в литературе. В работе [24] для данного соединения приведены значения полной электронной энергии и энергии нулевых колебаний (ZPE), рассчитанные на уровне B3LYP/aug-cc-pVDZ; при этом указанные величины не использовались для оценки ΔH_f^0 и не представлены в виде термохимических характеристик. В работе [23] также приводится значение энтальпии образования 15560 кДж/кг, рассчитанное для твердой фазы в рамках DFT-подхода на уровне PBE. Указанная величина используется авторами для оценки энергетической плотности материала. В связи с этим прямое количественное сравнение данного значения с результатами настоящей работы, полученными для газовой фазы, не представляется корректным.

2.2. ИК-спектры

Рассчитанные ИК-спектры поглощения исследуемых соединений **1**–**6** представлены на рис. 4.

Для спектров большинства рассматриваемых соединений характерны пики интенсивности в области 1458–1444 см$^{-1}$, связанные с валентными колебаниями связей C=N в двух тетразольных кольцах, и в области 1595–1567 см$^{-1}$ для соединений **1**–**4**, связанные с валентными колебаниями связей C–N и C–C в триазиновых кольцах. Для соединения **5** характерны валентные колебания связей C–N и C–C в пиразиновом кольце, объединенные в один пик в области 1613 см$^{-1}$. Для соединения **6** валентные колебания связей C=N в тетразольных кольцах объединяются с валентными колебаниями связей C=N в триазиновом кольце и отражаются на спектре в виде ярко выраженного пика в области 1581 см$^{-1}$.

Для спектров соединений **1**–**3** характерны пики в области 3186–3148 см$^{-1}$, связанные с валентными колебаниями связей C–H в азольных кольцах, на спектре соединения **4** валентные колебания связи C–H в триазольном кольце отражаются в виде едва заметного пика в области 3160 см$^{-1}$. На спектре соединения **5** можно видеть пик интенсивности в области 3513 см$^{-1}$, связанный с валентными колебаниями связей N–H в триазольном кольце.

Внеплоскостные деформационные колебания водородных связей отражаются на спектрах в виде пиков интенсивности в области ∼770 и ∼680 см$^{-1}$. Полосы интенсивного поглощения в области 1368–1356 см$^{-1}$ и 1265–1243 см$^{-1}$ на спектрах рассматриваемых соединений связаны с комбинацией различных деформационных колебаний водородных связей в азольных кольцах. Полосы интенсивного поглощения в области 1078–1043 см$^{-1}$ связаны с деформационными колебаниями связей N–N в тетразольных кольцах. Все зарегистрированные полосы соответствуют колебаниям, типичным для соответствующих функциональных фрагментов.

3. Особенности проведения расчетов

Квантово-химические расчеты проводились с использованием оборудования Суперкомпьютерного комплекса МГУ (проект 2312) [35, 36] и вычислительных ресурсов ФИЦ ПХФ и МХ РАН. Большая часть расчетов проводилась в разделах суперкомпьютера «Ломоносов-2» *volta1* на процессорах Intel Xeon Gold 6142 (16 ядер, 2.60 GHz — 1331.2 GFlop/s) и *volta2* на процессорах Intel Xeon Gold 6240 (18 ядер, 2.60 GHz — 1497.6 GFlop/s) с графическими ускорителями Nvidia Tesla V100 (900-2G500-0010-000, 1246 MHz, 7 TFlop/s). Расчеты с использованием метода теории функционала плотности B3LYP с базисными наборами 6-311+G(2d,p) и cc-pVTZ для молекул данного уровня слож-

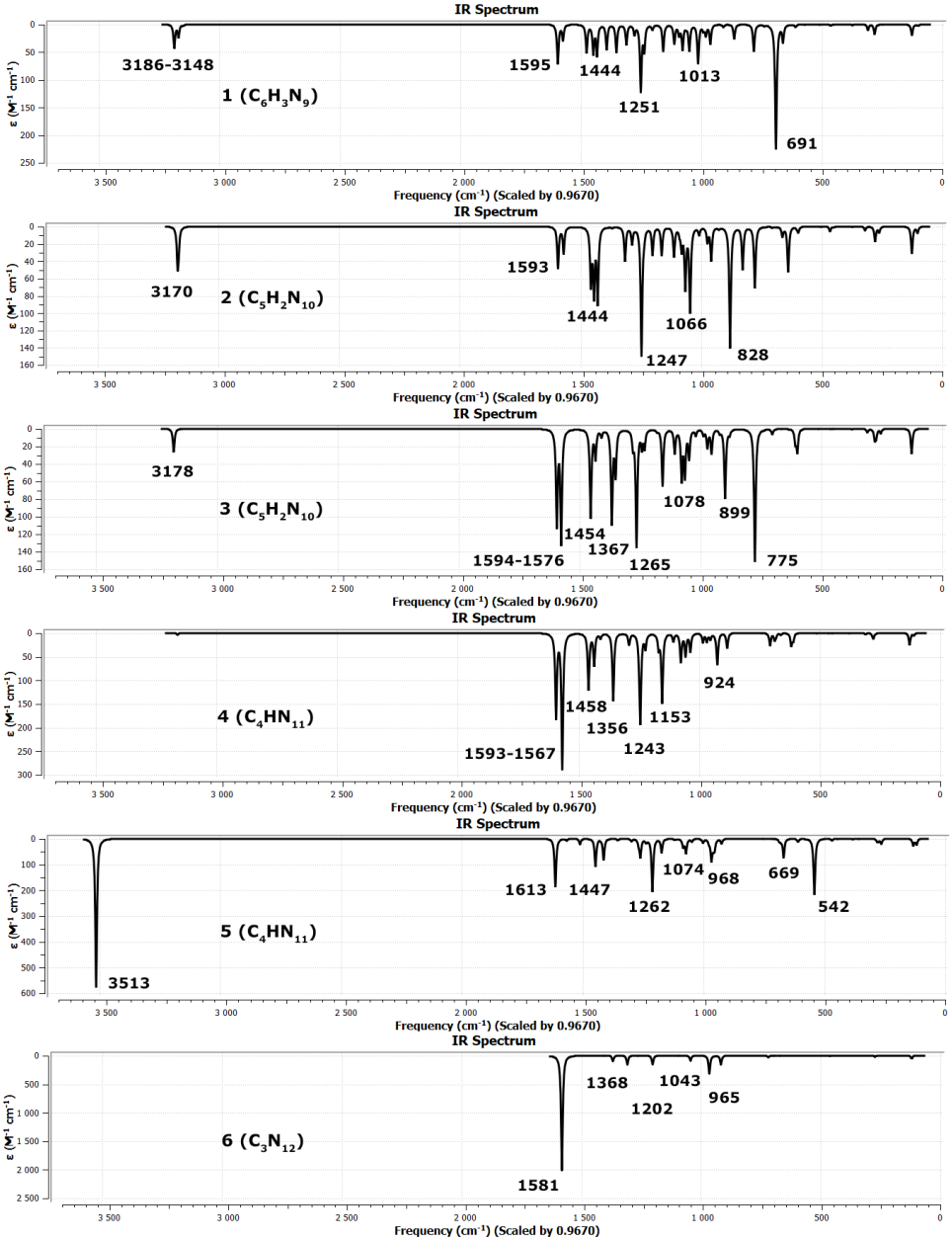


Рис. 4. ИК спектры поглощения исследуемых тетрациклов **1–6** (рассчитанные на уровне B3LYP/cc-pVTZ)

ности заняли от 13 до 24 мин в разделе *volta1* и от 11 до 18 мин в разделе *volta2*. При использовании комбинированного метода G4MP2 разница во времени расчета в разных разделах суперкомпьютера составила от 17 до 43 мин. Время расчета данным методом составило от 1 час. 23 мин для соединения **4** в разделе *volta2* до 2 час. 16 мин для соединения **1** в разделе *volta1*. Gaussian для распараллеливания использует свой функционал Linda, который не допускает распределение задач между разными узлами, но распределяет задачи между процессорами одного узла.

При расчетах с использованием программного модуля для NWChem сравнили время выполнения задач при использовании нескольких узлов раздела. Задачи запускали в разделах *volta1* и *volta*2, используя от 2 до 8 узлов на задачу (табл. 2). Для соединений **1–4** провести расчеты в разделе *volta2* на 6 и 8 узлах не представилось возможным в связи

Таблица 2. Время выполнения расчетов в NWChem в зависимости от использованных вычислительных ресурсов (час.:мин)

Соединение	Раздел суперкомпьютера	Число узлов			
		2	4	6	8
1	Volta1	1:40	0:54	**0:39**	**0:31**
	Volta2	**1:04**	**0:44**	—	—
2	Volta1	1:34	0:51	**0:37**	**0:29**
	Volta2	**0:59**	**0:41**	—	—
3	Volta1	1:35	0:51	**0:37**	**0:31**
	Volta2	**1:00**	**0:42**	—	—
4	Volta1	1:36	0:49	**0:36**	**0:29**
	Volta2	**0:56**	**0:41**	—	—
5	Volta1	1:26	0:47	**0:35**	**0:28**
	Volta2	**0:55**	**0:40**	**0:35**	0:33
6	Volta1	1:21	0:44	**0:32**	**0:26**
	Volta2	**0:52**	**0:38**	0:34	0:34

с большой загруженностью раздела и малым количеством доступных для использования узлов. Из табл. 2 видно, что при использовании 2 и 4 узлов на задачу время расчета ниже в разделе *volta2*, но при увеличении числа используемых узлов время выполнения задач данного уровня сильнее сокращается в разделе *volta1*.

При использовании изодесмических реакций для соединений **1**–**5**, где в реакции участвуют 4 соединения (два реагента и два продукта), для получения конечного результата нужно провести семь промежуточных расчетов: четыре — с использованием теории функционала плотности B3LYP с базисным набором 6-311+G(2d,p) и три — с использованием комбинированного метода G4MP2. Эти промежуточные расчеты проводились как в разделе *volta2* суперкомпьютера, так и на отдельном вычислительном ресурсе с характеристиками Intel(R) Xeon(R) Gold 6140 CPU @ 2.30 GHz, RAM 259 Gb, 20Tb disk space. Суммарное время, потраченное на все промежуточные расчеты на разных ресурсах, составило 68 мин для соединения **6** (меньшее число участников реакции, и следовательно меньшее число промежуточных расчетов) и около 2 час. для соединений **1**–**5**, что сравнимо со временем расчета с использованием комбинированного метода G4MP2 в Gaussian 09. Однако поскольку во всех использованных изодесмических реакциях в числе реагентов и продуктов присутствуют одни и те же соединения (рис. 2), расчеты для которых выполняются лишь один раз и затем используются для всех рассматриваемых систем, суммарным временем, затраченным на эти общие расчеты (59 мин), можно пренебречь. В этом случае общее время расчетов по изодесмическим реакциям для соединений **1**–**5** составляет около 1 час., что существенно меньше, чем при прямом использовании комбинированного метода G4MP2.

Заключение

С использованием квантово-химического программного комплекса Gaussian 09 выполнены расчеты оптимизированной структуры, ИК-спектров и энтальпии образования в газовой фазе ряда тетрациклов на основе бис(тетразоло)азинов, аннелированных незамещенными азолами. Была проанализирована закономерность изменения энтальпии образования

в зависимости от структуры молекул исследуемых соединений (количества и видов связей, структуры гетероциклов в составе тетрацикла). Среди изученных соединений самое высокое значение энтальпии образования показал трис(тетразоло)-s-триазин, в структуре которого 1,3,5-триазиновое ядро аннелировано тремя тетразольными кольцами. Для большинства изученных тетрациклических структур энтальпия образования вычислена впервые. Для всех изученных тетрациклов проанализированы расчетные ИК-спектры и выявлено соответствие характеристических частот поглощения структурным фрагментам их молекул. Показано, что вычислительные затраты при квантово-химических расчетах энтальпии образования существенно зависят как от выбранного метода, так и от конфигурации вычислительного ресурса: прямые расчеты методом G4MP2 характеризуются высокой вычислительной стоимостью, тогда как применение изодесмических реакций и собственного модуля, реализующего метод G4MP2 в программном комплексе NWChem, обеспечивают сопоставимое качество результатов при существенно меньших затратах вычислительного времени.

Расчеты с использованием ресурсов суперкомпьютерного комплекса МГУ и анализ результатов расчетов и вычислительных затрат выполнялись при поддержке Российского научного фонда (грант 23-71-00005). Е.С. Амосова принимала участие в проведении квантово-химических расчетов, анализе их результатов и вычислительных затрат, написании и редактировании статьи. О.А. Гончарова принимала участие в анализе результатов квантово-химических расчетов и исследовании ИК-спектров. В.М. Волохов выполнял работу в соответствии с темой государственного задания, № госрегистрации 124013100856-9. Д.Б. Лемперт выполнял работу в соответствии с темой государственного задания, № госрегистрации No. 124020100045-5. В.В. Парахин участвовал в постановке научной задачи, обзоре литературы, анализе результатов, написании и редактировании статьи. В.В. Воеводин принимал участие в проведении квантово-химических расчетов и анализе вычислительных затрат.

Литература

1. Rice B.M., Pai S.V., Hare J. Predicting heats of formation of energetic materials using quantum mechanical calculations // Combust. Flame. 1999. Vol. 118. P. 445–458. DOI: 10.1016/S0010-2180(99)00008-5.
2. Byrd E.F., Rice B.M. Improved prediction of heats of formation of energetic materials using quantum mechanical calculations // J. Phys. Chem. A. 2006. Vol. 110. P. 1005–1013. DOI: 10.1021/jp0536192.
3. Muthurajan H., Sivabalan R., Anniyappan M., *et al.* Prediction of heat of formation and related parameters of high energy materials // J. Hazard. Mater. 2006. Vol. 133. P. 30–45. DOI: 10.1016/j.jhazmat.2005.10.009.
4. Zhong L., Liu D., Hu M., *et al.* First-principles calculations of solid-phase enthalpy of formation of energetic materials // Commun. Chem. 2025. Vol. 8. P. 1–7. DOI: 10.1038/s42004-025-01544-9.
5. Larin A.A., Muravyev N.V., Pivkina A.N., *et al.* Assembly of tetrazolylfuroxan organic salts: Multipurpose green energetic materials with high enthalpies of formation and excellent detonation performance // Chem. Eur. J. 2019. Vol. 25. P. 4225–4233. DOI: 10.1002/chem.201806378.

6. Luk'yanov O.A., Parakhin V.V., Shlykova N.I., *et al.* Energetic N-azidomethyl derivatives of polynitro hexaazaisowurtzitanes series: CL-20 analogues having the highest enthalpy // New J. Chem. 2020. Vol. 44. P. 8357–8365. DOI: 10.1039/D0NJ01453B.

7. Gao H., Zhang Q., Shreeve J.M. Fused heterocycle-based energetic materials (2012–2019) // J. Mater. Chem. A. 2020. Vol. 8. P. 4193–4216. DOI: 10.1039/c9ta12704f.

8. Wu J.T., Xu J., Li W., Li H.B. Coplanar fused heterocycle-based energetic materials // Propellants Explos. Pyrotech. 2020. Vol. 45, no. 4. P. 536–545. DOI: 10.1002/prep.201900333.

9. Wang S., Huang Y., Ma Q., Chen F.X. Theoretical research on tricyclic-based as high-energy performance energetic materials // J. Mol. Model. 2025. Vol. 31. P. 183. DOI: 10.1007/s00894-025-06401-z.

10. Kuznetsova A.N., Leonov N.E., Anikin O.V., *et al.* Parent 1,4-dihydro-[1,2,3]triazolo[4,5-d][1,2,3]triazole and its derivatives as precursors for the design of promising high energy density materials // New J. Chem. 2025. Vol. 49. P. 311–320. DOI: 10.1039/D4NJ04427D.

11. Volokhov V., Amosova E., Parakhin V., *et al.* Quantum-Chemical Study of Some Tris(pyrrolo)benzenes and Tris(pyrrolo)-1,3,5-triazines // Supercomputing. RuSCDays 2023. Vol. 14388 / ed. by V. Voevodin, S. Sobolev, M. Yakobovskiy, R. Shagaliev. Cham: Springer, 2023. P. 177–189. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-49432-1_14.

12. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Quantum-Chemical Study of Some Trispyrazolobenzenes and Trispyrazolo-1,3,5-triazines // Supercomput. Front. Innov. 2024. Vol. 11, no. 3. P. 64–73. DOI: 10.14529/jsfi240304.

13. Volokhov V., Amosova E., Parakhin V., *et al.* Quantum-Chemical Simulation of Some Triimidazolobenzenes and Triimidazolo-1,3,5-Triazines // Parallel Computational Technologies. PCT 2024. Vol. 2241 / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer, 2024. P. 292–303. Communications in Computer and Information Science. DOI: 10.1007/978-3-031-73372-7_21.

14. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Comparison of Quantum-Chemical Programs and Methods for the Calculation of Enthalpies of Formation of High-Energy Tetracyclic Compounds // Supercomput. Front. Innov. 2025. Vol. 12, no. 2. P. 60–73. DOI: 10.14529/jsfi250205.

15. Volokhov V., Amosova E., Parakhin V., *et al.* Quantum-chemical study of unsubstituted tris(triazolo)benzenes and 1,3,5-triazines // Parallel Computational Technologies. PCT 2025. Vol. 2891 / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer, 2026. P. 386–397. Communications in Computer and Information Science. DOI: 10.1007/978-3-032-22051-6_21.

16. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Quantum-chemical study of high energy derivatives of tris(triazolo)benzenes // Lobachevskii Journal of Mathematics. 2025. Vol. 46, no. 8. P. 3883–3894. DOI: 10.1134/S1995080225610100.

17. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Quantum-chemical study of high energy derivatives of tris(1,2,4-triazolo)-1,3,5-triazines // Lecture Notes in Computer Science. Vol. 16196. Cham: Springer, 2026. P. 159–171. DOI: 10.1007/978-3-032-13127-0_12.

18. Chen B., Lu H., Chen J., *et al.* revealing recent progress on nitrogen-rich energetic materials based on tetrazole skeleton // Top. Curr. Chem. 2023. Vol. 381. A. 25. DOI: 10.1007/s41061-023-00435-8.

19. Manzoor S., Yin X., Zhang J.G. Nitro-tetrazole based high performing explosives: Recent overview of synthesis and energetic properties // Defence Technology. 2021. Vol. 17. P. 1995–2010. DOI: 10.1016/j.dt.2021.02.002.

20. Volokhov V., Parakhin V., Amosova E., *et al.* Quantum-chemical calculations of the enthalpy of formation of isomeric 5/6/5 tricyclic tetrazolotetrazine derivatives annelated with nitroazoles // Supercomputing. RuSCDays 2024. Vol. 15406 / ed. by V. Voevodin, A. Antonov, D. Nikitenko. Cham: Springer, 2024. P. 187–201. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-78459-0_14.

21. Hammerl A., Klapötke T.M., Rocha R. Azide–tetrazole ring–chain isomerism in polyazido-1,3,5-triazines, triazido-*s*-heptazine, and diazidotetrazines // Eur. J. Inorg. Chem. 2006. P. 2210–2228. DOI: 10.1002/ejic.200600100.

22. Hu A., Zhang F. Ab initio molecular dynamics simulations on sensitivity of nitrogen-rich poly-tetrazolo energetic compounds // AIP Conference Proceedings. Vol. 1195, no. 1. American Institute of Physics, 2009. P. 809–812. DOI: 10.1063/1.3295265.

23. Hu A., Zhang F. A nitrogen-rich C_3N_{12} solid transformed from cyanuric triazide under high pressure and temperature // J. Phys.: Condens. Matter. 2010. Vol. 22. A. 505402. DOI: 10.1088/0953-8984/22/50/505402.

24. Liang X.Q., Zhou J.J., Zheng Y., Ma F. Theoretical studies on the mechanism of the azido–tetrazole of azido-*s*-triazine // Natur. Prod. Commun. 2015. Vol. 10. P. 269–272. DOI: 10.1177/1934578x1501000213.

25. Frisch M.J., Trucks G.W., Schlegel H.B., *et al.* Gaussian 09, Revision B.01. Gaussian, Inc., Wallingford CT, 2010.

26. Valiev M., Bylaska E.J., Govind N., *et al.* NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations // Comput. Phys. Commun. 2010. Vol. 181. P. 1477. DOI: 10.1016/j.cpc.2010.04.018.

27. Becke A.D. Density-functional thermochemistry. III. The role of exact exchange // J. Chem. Phys. 1993. Vol. 98, no. 7. P. 5648–5652. DOI: 10.1063/1.464913.

28. Johnson B.J., Gill P.M.W., Pople J.A. The performance of a family of density functional methods // J. Chem. Phys. 1993. Vol. 98, no. 7. P. 5612–5626. DOI: 10.1063/1.464906.

29. Dunning T.H., Jr. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen // J. Chem. Phys. 1989. Vol. 90, no. 2. P. 1007–1023. DOI: 10.1063/1.456153.

30. Curtiss L.A., Redfern P.C., Raghavachari K. Gaussian-4 theory using reduced order perturbation theory // J. Chem. Phys. 2007. Vol. 127, no. 12. A. 124105. DOI: 10.1063/1.2770701.

31. Curtiss L.A., Redfern P.C., Raghavachari K. G*n* theory // Comput. Mol. Sci. 2011. Vol. 1, no. 5. P. 810–825. DOI: 10.1002/wcms.59.

32. Curtiss L.A., Redfern P.C., Raghavachari K. Gaussian-4 theory // J. Chem. Phys. 2007. Vol. 126, no. 8. P. 084108. DOI: 10.1063/1.2436888.

33. CCCBDB Vibrational Frequency Scaling Factors. URL: `https://cccbdb.nist.gov/vsfx.asp` (дата обращения: 25.12.2025).

34. NIST-JANAF Thermochemical Tables. URL: `https://janaf.nist.gov/` (дата обращения: 25.12.2025).

35. Voevodin V.V., Antonov A.S., Nikitenko D.A., *et al.* Supercomputer Lomonosov-2: large-scale, deep monitoring and online analytics for the user community // Supercomput. Front. Innov. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

36. Nikitenko D.A., Voevodin V.V., Zhumatiy S.A. Deep analysis of job state statistics on Lomonosov-2 supercomputer // Supercomput. Front. Innov. 2019. Vol. 5, no. 2. P. 4–10. DOI: 10.14529/jsfi180201.

Волохов Вадим Маркович, д.ф.-м.н., профессор, главный научный сотрудник, отдел вычислительных ресурсов, Федеральный исследовательский центр проблем химической физики и медицинской химии РАН (Черноголовка, Российская Федерация)

Амосова Елена Сергеевна, научный сотрудник, отдел вычислительных ресурсов, Федеральный исследовательский центр проблем химической физики и медицинской химии РАН (Черноголовка, Российская Федерация)

Парахин Владимир Валерьевич, к.х.н., старший научный сотрудник, лаборатория органического синтеза, Институт органической химии имени Н. Д. Зелинского РАН (Москва, Российская Федерация)

Лемперт Давид Борисович, к.х.н., главный научный сотрудник, отдел горения и взрыва, Федеральный исследовательский центр проблем химической физики и медицинской химии РАН (Черноголовка, Российская Федерация)

Гончарова Ольга Андреевна, младший научный сотрудник, отдел функциональных материалов для химических источников энергии, Федеральный исследовательский центр проблем химической физики и медицинской химии РАН (Черноголовка, Российская Федерация)

Воеводин Владимир Валентинович, чл.-корр. РАН, д.ф.-м.н., директор, научно-исследовательский вычислительный центр, Московский государственный университет им. М. В. Ломоносова (Москва, Российская Федерация)

DOI: 10.14529/cmse260204

QUANTUM-CHEMICAL STUDY OF SOME HIGH-ENERGY AZOLO-BIS(TETRAZOLO)AZINES

© 2026 V.M. Volokhov[1], E.S. Amosova[1], V.V. Parakhin[2], D.B. Lempert[1], O.A. Goncharova[1], V.V. Voevodin[3]

[1] *Federal Research Center of Problems of Chemical Physics and Medicinal Chemistry RAS (pr. Ac. Semenova 1, Chernogolovka, 142432 Russia),*

[2] *N.D. Zelinskiy Institute of Organic Chemistry RAS (pr. Leninskiy 47, Mocsow, 119991 Russia),*

[3] *Research Computing Center of Lomonosov Moscow State University (bldg. 4, Leninskie Gory 1, 119234 Russia)*

E-mail: vvm@icp.ac.ru, aes@icp.ac.ru, parakhin@ioc.ac.ru, lempert@icp.ac.ru, goa@icp.ac.ru, voevodin@parallel.ru

Received: 17.04.2026

High-energy compounds are widely used in various branches of industry and technology, which makes the development of new substances of this type and the study of their properties a relevant task. This work presents a quantum-chemical investigation of six high-energy tetracyclic compounds based on bis(tetrazolo)azines annelated with unsubstituted azoles. The aim of the study is to determine their key energetic characteristic, the gas-phase enthalpy of formation, and to analyze its dependence on molecular structure. Calculations were performed with Gaussian 09 using the B3LYP/6-311+G(2d,p) and B3LYP/cc-pVTZ density functional theory methods and the composite G4MP2 method, as well as with a G4MP2 implementation in the NWChem software package. Enthalpies of formation were determined using atomization and isodesmic reactions. The enthalpy of formation generally increases with the nitrogen content and the number of endothermic bonds in the annelated azole fragment; the highest value was obtained for tris(tetrazolo)-s-triazine. The NWChem G4MP2 implementation gives results in good agreement with Gaussian 09 calculations, while the use of several compute nodes reduces task execution time. Infrared spectra of the studied compounds were calculated and analyzed, and characteristic absorption bands were assigned to molecular structural fragments. Quantum-chemical calculations were performed using the «Lomonosov-2» supercomputer.

Ключевые слова: quantum-chemical calculations, high-energy-density materials, enthalpy of formation, tetracyclic compounds, tris(azolo)azines, tris(tetrazolo)[1,3,5]triazine.

FOR CITATION

Volokhov V.M., Amosova E.S., Parakhin V.V., Lempert D.B., Goncharova O.A., Voevodin V.V. Quantum-chemical Study of Some High-energy Azolo-bis(tetrazolo)azines. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2026. Vol. 15, no. 2. P. 65–81. (in Russian) DOI: 10.14529/cmse260204.

References

1. Rice B.M., Pai S.V., Hare J. Predicting heats of formation of energetic materials using quantum mechanical calculations. Combust. Flame. 1999. Vol. 118. P. 445–458. DOI: 10.1016/S0010-2180(99)00008-5.

2. Byrd E.F., Rice B.M. Improved prediction of heats of formation of energetic materials

using quantum mechanical calculations. J. Phys. Chem. A. 2006. Vol. 110. P. 1005–1013. DOI: 10.1021/jp0536192.

3. Muthurajan H., Sivabalan R., Talawar M.B., *et al.* Prediction of heat of formation and related parameters of high energy materials. J. Hazard. Mater. 2006. Vol. 133. P. 30–45. DOI: 10.1016/j.jhazmat.2005.10.009.

4. Zhong L., Liu D., Hu M., *et al.* First-principles calculations of solid-phase enthalpy of formation of energetic materials. Commun. Chem. 2025. Vol. 8. P. 1–7. DOI: 10.1038/s42004-025-01544-9.

5. Larin A.A., Muravyev N.V., Pivkina A.N., *et al.* Assembly of tetrazolylfuroxan organic salts: Multipurpose green energetic materials with high enthalpies of formation and excellent detonation performance. Chem. Eur. J. 2019. Vol. 25. P. 4225–4233. DOI: 10.1002/chem.201806378.

6. Luk'yanov O.A., Parakhin V.V., Shlykova N.I., *et al.* Energetic N-azidomethyl derivatives of polynitro hexaazaisowurtzitanes series: CL-20 analogues having the highest enthalpy. New J. Chem. 2020. Vol. 44. P. 8357–8365. DOI: 10.1039/D0NJ01453B.

7. Gao H., Zhang Q., Shreeve J.M. Fused heterocycle-based energetic materials (2012–2019). J. Mater. Chem. A. 2020. Vol. 8. P. 4193–4216. DOI: 10.1039/c9ta12704f.

8. Wu J.T., Xu J., Li W., Li H.B. Coplanar fused heterocycle-based energetic materials. Propellants Explos. Pyrotech. 2020. Vol. 45, no. 4. P. 536–545. DOI: 10.1002/prep.201900333.

9. Wang S., Huang Y., Ma Q., Chen F.X. Theoretical research on tricyclic-based as high-energy performance energetic materials. J. Mol. Model. 2025. Vol. 31. P. 183. DOI: 10.1007/s00894-025-06401-z.

10. Kuznetsova A.N., Leonov N.E., Anikin O.V., *et al.* Parent 1,4-dihydro-[1,2,3]triazolo[4,5-d][1,2,3]triazole and its derivatives as precursors for the design of promising high energy density materials. New J. Chem. 2025. Vol. 49. P. 311–320. DOI: 10.1039/D4NJ04427D.

11. Volokhov V., Amosova E., Parakhin V., *et al.* Quantum-Chemical Study of Some Tris(pyrrolo)benzenes and Tris(pyrrolo)-1,3,5-triazines. Supercomputing. RuSCDays 2023. Vol. 14388 / ed. by V. Voevodin, S. Sobolev, M. Yakobovskiy, R. Shagaliev. Cham: Springer, 2023. P. 177–189. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-49432-1_14.

12. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Quantum-Chemical Study of Some Trispyrazolobenzenes and Trispyrazolo-1,3,5-triazines. Supercomput. Front. Innov. 2024. Vol. 11, no. 3. P. 64–73. DOI: 10.14529/jsfi240304.

13. Volokhov V., Amosova E., Parakhin V., *et al.* Quantum-Chemical Simulation of Some Triimidazolobenzenes and Triimidazolo-1,3,5-Triazines. Parallel Computational Technologies. PCT 2024. Vol. 2241 / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer, 2024. P. 292–303. Communications in Computer and Information Science. DOI: 10.1007/978-3-031-73372-7_21.

14. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Comparison of Quantum-Chemical Programs and Methods for the Calculation of Enthalpies of Formation of High-Energy Tetracyclic Compounds. Supercomput. Front. Innov. 2025. Vol. 12, no. 2. P. 60–73. DOI: 10.14529/jsfi250205.

15. Volokhov V., Amosova E., Parakhin V., *et al.* Quantum-chemical study of unsubstituted tris(triazolo)benzenes and 1,3,5-triazines. Parallel Computational Technologies. PCT 2025. Vol. 2891 / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer, 2026. P. 386–397. Communications in Computer and Information Science. DOI: 10.1007/978-3-032-22051-6_21.

16. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Quantum-chemical study of high energy derivatives of tris(triazolo)benzenes. Lobachevskii Journal of Mathematics. 2025. Vol. 46, no. 8. P. 3883–3894. DOI: 10.1134/S1995080225610100.

17. Volokhov V.M., Parakhin V.V., Amosova E.S., *et al.* Quantum-chemical study of high energy derivatives of tris(1,2,4-triazolo)-1,3,5-triazines. Lecture Notes in Computer Science. Vol. 16196. Cham: Springer, 2026. P. 159–171. DOI: 10.1007/978-3-032-13127-0_12.

18. Chen B., Lu H., Chen J., *et al.* revealing recent progress on nitrogen-rich energetic materials based on tetrazole skeleton. Top. Curr. Chem. 2023. Vol. 381. A. 25. DOI: 10.1007/s41061-023-00435-8.

19. Manzoor S., Yin X., Zhang J.G. Nitro-tetrazole based high performing explosives: Recent overview of synthesis and energetic properties. Defence Technology. 2021. Vol. 17. P. 1995–2010. DOI: 10.1016/j.dt.2021.02.002.

20. Volokhov V., Parakhin V., Amosova E., *et al.* Quantum-chemical calculations of the enthalpy of formation of isomeric 5/6/5 tricyclic tetrazolotetrazine derivatives annelated with nitroazoles. Supercomputing. RuSCDays 2024. Vol. 15406 / ed. by V. Voevodin, A. Antonov, D. Nikitenko. Cham: Springer, 2024. P. 187–201. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-78459-0_14.

21. Hammerl A., Klapötke T.M., Rocha R. Azide–tetrazole ring–chain isomerism in polyazido-1,3,5-triazines, triazido-*s*-heptazine, and diazidotetrazines. Eur. J. Inorg. Chem. 2006. P. 2210–2228. DOI: 10.1002/ejic.200600100.

22. Hu A., Zhang F. Ab initio molecular dynamics simulations on sensitivity of nitrogen-rich poly-tetrazolo energetic compounds. AIP Conference Proceedings. Vol. 1195, no. 1. American Institute of Physics, 2009. P. 809–812. DOI: 10.1063/1.3295265.

23. Hu A., Zhang F. A nitrogen-rich C_3N_{12} solid transformed from cyanuric triazide under high pressure and temperature. J. Phys.: Condens. Matter. 2010. Vol. 22. A. 505402. DOI: 10.1088/0953-8984/22/50/505402.

24. Liang X.Q., Zhou J.J., Zheng Y., Ma F. Theoretical studies on the mechanism of the azido–tetrazole of azido-*s*-triazine. Natur. Prod. Commun. 2015. Vol. 10. P. 269–272. DOI: 10.1177/1934578x1501000213.

25. Frisch M.J., Trucks G.W., Schlegel H.B., *et al.* Gaussian 09, Revision B.01. Gaussian, Inc., Wallingford CT, 2010.

26. Valiev M., Bylaska E.J., Govind N., *et al.* NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. Comput. Phys. Commun. 2010. Vol. 181. P. 1477. DOI: 10.1016/j.cpc.2010.04.018.

27. Becke A.D. Density-functional thermochemistry. III. The role of exact exchange. J. Chem. Phys. 1993. Vol. 98, no. 7. P. 5648–5652. DOI: 10.1063/1.464913.

28. Johnson B.J., Gill P.M.W., Pople J.A. The performance of a family of density functional methods. J. Chem. Phys. 1993. Vol. 98, no. 7. P. 5612–5626. DOI: 10.1063/1.464906.

29. Dunning T.H., Jr. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. J. Chem. Phys. 1989. Vol. 90, no. 2. P. 1007–1023. DOI: 10.1063/1.456153.

30. Curtiss L.A., Redfern P.C., Raghavachari K. Gaussian-4 theory using reduced order perturbation theory. J. Chem. Phys. 2007. Vol. 127, no. 12. A. 124105. DOI: 10.1063/1.2770701.

31. Curtiss L.A., Redfern P.C., Raghavachari K. Gn theory. Comput. Mol. Sci. 2011. Vol. 1, no. 5. P. 810–825. DOI: 10.1002/wcms.59.

32. Curtiss L.A., Redfern P.C., Raghavachari K. Gaussian-4 theory. J. Chem. Phys. 2007. Vol. 126, no. 8. P. 084108. DOI: 10.1063/1.2436888.

33. CCCBDB Vibrational Frequency Scaling Factors. URL: `https://cccbdb.nist.gov/vsfx.asp` (accessed: 25.12.2025).

34. NIST-JANAF Thermochemical Tables. URL: `https://janaf.nist.gov/` (accessed: 25.12.2025).

35. Voevodin V.V., Antonov A.S., Nikitenko D.A., *et al.* Supercomputer Lomonosov-2: Large-scale, deep monitoring and online analytics for the user community. Supercomput. Front. Innov. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

36. Nikitenko D.A., Voevodin V.V., Zhumatiy S.A. Deep analysis of job state statistics on Lomonosov-2 supercomputer. Supercomput. Front. Innov. 2019. Vol. 5, no. 2. P. 4–10. DOI: 10.14529/jsfi180201.

УДК 004.021 **DOI: 10.14529/cmse260205**

СИНХРОНИЗАЦИЯ В КОНКУРЕНТНЫХ СТРУКТУРАХ ДАННЫХ НА ОСНОВЕ АНАЛИЗА ЗАВИСИМОСТЕЙ МЕЖДУ ОПЕРАЦИЯМИ*

© 2026 Д.С. Коротченко, В.Е. Аксенов

Университет ИТМО

(197101 Санкт-Петербург, Кронверкский пр., д. 49, лит. А)

E-mail: korotchenkods@gmail.com, aksenov.vitaly@gmail.com

Поступила в редакцию: 28.07.2026

В работе рассматривается оптимизация синхронизаций операций в потокобезопасных структурах данных. В различных структурах данных операции могут по-разному конфликтовать друг с другом с точки зрения одновременного исполнения. Например, стандартная блокировка чтения—записи учитывает, что одновременное чтение из разных потоков часто допустимо, в то время как одновременная запись из разных потоков или параллельная запись и чтение требуют синхронизации. Такая зависимость может быть и более сложной: например, несколько одновременных операций увеличения всех элементов в контейнере на заданную величину допустимы с синхронизацией на уровне доступа к элементу, в то время как параллельное присвоение всем элементам контейнера одного числа из одного потока и другого числа из другого потока требует более сложной синхронизации. В представленной работе рассматривается новый подход к обеспечению синхронизации при выполнении операций с потокобезопасной структурой данных, основанный на анализе графа зависимостей между операциями, с целью обеспечения более эффективной работы такой структуры. Также приводятся результаты экспериментов по сравнению различных подходов, которые показывают эффективность предлагаемого решения.

Ключевые слова: потокобезопасность, конкурентные структуры данных, граф зависимостей, семантическая блокировка, линеаризуемость.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Коротченко Д.С., Аксенов В.Е. Синхронизация в конкурентных структурах данных на основе анализа зависимостей между операциями // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2026. Т. 15, № 2. С. 82–103. DOI: 10.14529/cmse260205.

Введение

При разработке многопоточных приложений часто возникает потребность в использовании потокобезопасных структур данных. Такие структуры данных инкапсулируют логику синхронизации потоков внутри себя, предоставляя пользователю удобный интерфейс, позволяющий не задумываться о корректности. Нередко потокобезопасные структуры не просто гарантируют корректное поведение при работе из нескольких потоков, но также пытаются сделать структуру *конкурентной*, то есть обрабатывать вызовы операций над этой структурой данных, совершенные разными потоками, параллельно, когда это возможно. Поддерживающая конкурентность структура данных позволяет улучшить метрику *пропускной способности* — число запросов к данной структуре, обрабатываемых в секунду на заданном профиле нагрузки (числе потоков, с ней работающих, и распределении типов операций между этими потоками).

Стандартные библиотеки большинства языков программирования содержат различные потокобезопасные конкурентные реализации наиболее часто используемых структур, на-

*Статья рекомендована к публикации Программным комитетом Всероссийской научной конференции с международным участием «Параллельные вычислительные технологии (ПаВТ) 2026».

пример словарей, списков, множеств и др. Однако нередко возникает необходимость реализации дополнительных операций на таких структурах (например, преобразование всех элементов структуры или суммирование ее элементов), которые также требуют синхронизации с остальными операциями для сохранения корректности. Также у пользователей может возникнуть потребность в требуемых для работы приложения собственных реализациях структур данных, отличных от представленных в языке по умолчанию.

Конкурентные структуры данных можно разделить на два типа по способу синхронизации — так называемые *lock-free* структуры, использующие специальные алгоритмы для поддержания целостности и корректности без использования блокировок, и так называемые *lock-based* или основанные на блокировках структуры. В этой работе мы сосредоточились на структурах, основанных на блокировках. Несмотря на появление в стандартных и сторонних библиотеках реализаций, основанных на идее *lock-free*, структуры, работающие на блокировках, по-прежнему востребованы и популярны. При этом расширить существующую *lock-free* структуру дополнительной операцией с сохранением свойства *lock-free* чаще всего крайне сложно: для этого требуется большое количество сложного кода, в том числе модификация уже существующих операций. Расширить *lock-based* структуру гораздо проще. Отметим, что при желании *lock-free* структуру можно расширить другими операциями с использованием нашего подхода, потеряв при этом свойство *lock-free*.

Ключевое и наиболее часто встречающееся требование при работе с потокобезопасными структурами данных, подразумеваемое под корректностью — *линеаризуемость*. Интуитивно линеаризуемая структура данных — это структура, на которой любая последовательность вызовов операций из нескольких потоков может быть промоделирована на одном потоке [1]. В разделе 2 мы детальнее опишем, что именно подразумевается под линеаризуемостью и конкурентностью структур данных.

Простейший способ сохранить линеаризуемость при добавлении новых операций — дополнительно синхронизировать все операции с помощью отдельной блокировки. Использование дополнительной синхронизации позволит гарантировать, что никакие две операции не выполняются одновременно в процессе работы со структурой, из чего автоматически следует линеаризуемость.

Однако такой подход является неэффективным — структура фактически перестает быть конкурентной: никакие операции не выполняются параллельно. Обзор различных подходов, позволяющих разрешить исполнение некоторых операций параллельно с целью увеличения производительности, приведен в разделе 1.

При решении проблемы синхронизации операций зачастую возможно более эффективное поведение, чем взятие блокировки на каждую операцию. Так, если в структуре есть операции чтения, которые не модифицируют ее внутреннее состояние в процессе выполнения, а лишь используют информацию о данных, хранящихся в структуре, то выполнение такой операции параллельно с другой операцией чтения не повлияет на результаты их исполнения. В таком случае можно было бы использовать блокировки чтения—записи, то есть такие блокировки, которые допускают параллельное исполнение нескольких операций чтения, но не допускают параллельного исполнения каких-либо операций параллельно с операцией записи. Также блокировку чтения—записи можно применить в ситуации расширения конкурентной структуры данных дополнительными операциями. В этом случае изначально представленные в структуре данных операции уже гарантируют линеаризуемость, используя, возможно, дополнительные оптимизации, чтобы даже те операции, которые мо-

дифицируют структуру данных (операции записи), могли бы исполняться параллельно в отдельных случаях. Такие оптимизации обычно зависят от специфики структуры данных и могут быть различными в различных реализациях. В таком случае с помощью блокировки чтения—записи можно расширить структуру данных новыми операциями, не ухудшая производительность исходных операций (с учетом заложенной в структуру конкурентности) путем взятия для выполнения таких операций блокировки чтения, но при этом не допускать ни параллелизации таких операций с новыми, добавляемыми в структуру данных, операциями, ни параллелизации новых операций между собой за счет взятия для всех новых операций блокировки записи.

Легко заметить, что и такое решение не всегда оптимально, например, если структура расширяется операциями чтения, то их можно было бы исполнить параллельно между собой (или с исходно имеющимися операциями чтения), но для этого необходимо использовать более сложную блокировку. Далее в работе мы приведем и другой пример структуры, в которой зависимость между операциями устроена более сложным образом.

В этой работе мы представляем новый подход, предлагающий использование нового алгоритма блокировки, которая реализует соответствующую логику на основе информации о зависимостях между различными операциями. Далее мы будем называть такую блокировку *семантической*.

Предлагаемый подход актуален в различных практических задачах: системах управления базами данных, выполняющими диапазонные запросы параллельно с точечными обновлениями; кэшах в оперативной памяти с операциями массового обновления или чтения; системах аналитики реального времени, где операции агрегации могут выполняться параллельно операциям обновления. В серверных приложениях типичное число потоков составляет от 4 до 32, что соответствует диапазону, исследованному в экспериментах в рамках этой работы.

Основными вкладами работы являются:

- Формализация зависимостей между операциями в структуре данных в виде графа зависимостей, обобщающая классическую модель блокировок чтения—записи на произвольные зависимости.
- Новый алгоритм семантической блокировки с двумя подходами к реализации логики проверки конфликтов: на основе дополнительных блокировок и на основе атомарных счетчиков.
- Экспериментальная оценка предложенного подхода с использованием бенчмарка Synchrobench, демонстрирующая преимущество семантической блокировки в различных сценариях, особенно с доминированием параллельно-совместимых операций.

Статья организована следующим образом. В разделе 1 приводится обзор связанных работ. В разделе 2 формализуются требования к структурам данных, а также целевая метрика. В разделе 3 детально описаны различные подходы к синхронизации операций. В разделе 4 подробно описаны результаты сравнения производительности тестовой структуры данных с различными способами синхронизации операций, показывающие эффективность предлагаемого решения. В заключении приводится краткая сводка результатов, полученных в работе, и указаны направления дальнейших исследований.

1. Обзор литературы

Наиболее близкой к рассматриваемой постановке классической абстракцией является задача группового взаимного исключения (Group Mutual Exclusion, GME) [2]. В отличие от простого взаимоисключения, которое подразумевает эксклюзивный доступ к ресурсу, в алгоритмах группового взаимного исключения разрешается доступ к ресурсу нескольких параллельных процессов, относящихся к одной группе, но запрещен одновременный доступ процессов из разных групп. Произвольный граф зависимостей операций в структуре данных может выглядеть сложнее — операция A может не конфликтовать с B, но при этом конфликтовать сама с собой, что уже не позволит отнести эти две операции к одному классу. Эффективные алгоритмы решения задачи группового взаимного исключения широко исследуются в различных вариациях [3, 4], однако чаще всего направлены на решение проблемы в распределенных системах, что также отличает их от рассматриваемой в работе задачи.

Большое число работ посвящено исследованию различных подходов для ускорения выполнения долгих операций. Например, исполнение нескольких операций одним батчем под одной блокировкой [5]; подходы на основе идеи коллаборативности — переиспользование заблокированных потоков для быстрейшего выполнения операции, ставшей причиной блокировки [6]; добавление частичной персистентности, то есть хранения нескольких версий структуры для ускорения выполнения или параллелизации отдельных структур данных [7].

В то же время проблема синхронизации операций в структурах данных перекликается с проблемой выполнения транзакций и откатов в базах данных (и транзакционной памяти в общем случае). Для корректной поддержки транзакционности в базах данных широко исследуются различные типы блокировок, например блокировки отрезков [8], предикатные блокировки, позволяющие избавиться от части фантомных конфликтов [9], анализируются операции для выявления более слабых свойств по сравнению с коммутативностью и их применения для корректной обработки ситуаций отката транзакций [10]. Переход от режимов доступа к семантике абстрактного типа данных был формализован в управлении параллелизмом на основе коммутативности: две операции могут выполняться параллельно, если перестановка их порядка не меняет допустимые состояния и наблюдаемые результаты [11].

Обнаружение конфликтов по адресам в транзакционной памяти [12] приводит к ложным конфликтам. Transactional Boosting позволяет обнаружить конфликты на уровне методов: некоммутирующие вызовы захватывают конфликтующие блокировки, а откат выполняется с помощью журнала и обратных операций [13]. Однако эффективное использование транзакций и транзакционной памяти требует возможности выполнения дешевых обратных операций, что затруднено для операций, затрагивающих большую часть элементов структуры данных. Кроме того, расширение структур, реализованных для обычной памяти, потребует в этом случае их полного изменения для работы с транзакционной памятью.

2. Синхронизация операций в потокобезопасных структурах данных

Потокобезопасными структурами данных называются такие структуры данных, которые гарантируют корректное поведение при вызове их операций из нескольких потоков.

Под корректным поведением в этой работе мы будем подразумевать *линеаризуемость* результатов выполнения операций над структурой данных [1].

Определение 1. Результат выполнения вызванных из нескольких потоков операций над структурой данных, на которые получен ответ, *линеаризуем*, если можно последовательно выполнить тот же набор операций в одном потоке так, что результаты всех операций совпадут с исходными. При этом операции, которые изначально исполнялись на одном потоке, должны быть исполнены в том же взаимном порядке.

Линеаризуемость является ключевым и наиболее часто требуемым на практике свойством. Линеаризуемость может быть проверена различными инструментами: так, для языков Java и Kotlin общепринятым инструментом проверки свойства *линеаризуемости* является Lincheck [14]. С помощью Lincheck были проверены все реализации на языке Java описанных далее подходов, а также структуры данных, использованные в экспериментах в рамках этой работы.

Помимо *линеаризуемости* введем также понятие *справедливости* (fair) структуры. Справедливость определяет, как структура обрабатывает ситуацию, в которой поток, ожидающий выполнения операции из-за конфликта с уже исполняемыми операциями, может быть вытеснен новыми операциями, конфликтующими с его операцией. Структура называется справедливой, если она гарантирует, что ожидающий поток получит возможность выполнить свою операцию раньше, чем любая операция, запрошенная позже и конфликтующая с ней.

Определение 2. Поток A выполняет операцию a над структурой данных. При этом операция a не может быть исполнена из-за того, что другая операция, исполняемая в данный момент, не позволяет потоку A получить необходимую для исполнения операции a блокировку. После неудачной попытки получить блокировку потоком A другой поток B пытается выполнить операцию b, причем операция a не может быть выполнена параллельно с операцией b. Тогда структура называется *справедливой*, если она гарантирует, что операция a на потоке A будет исполнена раньше, чем операция b на потоке B, и *несправедливой*, если такой гарантии нет.

Приведем пример: рассмотрим структуру данных, которая хранит в себе одно число, позволяя параллельно выполнять чтение из нескольких потоков, но при этом гарантируя, что запись не выполняется параллельно с чтением или другой записью. Запустим один поток, который будет непрерывно выполнять запись, и на порядок большее число потоков, непрерывно выполняющих чтение. Если такой эксперимент провести на *справедливой* структуре данных, то соотношение выполненных операций чтения к выполненным операциям записи будет примерно соответствовать соотношению числа потоков, в которых выполняется операция записи, к числу потоков, в которых выполняется операция чтения. Если же используемая структура данных будет *несправедливой*, запись может быть выполнена очень малое (близкое к нулю) число раз. Это происходит из-за того, что после захвата блокировки на чтение одним из потоков другие потоки также начнут его выполнять параллельно. Это приведет к тому, что в любой момент времени почти наверняка будет хотя бы один поток, который выполняет чтение, а значит, хотя бы один поток будет владеть блокировкой на чтение, и, следовательно, операция записи не сможет быть исполнена.

На практике блокировки из библиотек языка зачастую поддерживают оба режима. Например, стандартная реализация блокировки чтения—записи (*ReentrantReadWriteLock*) в Java по умолчанию работает в *несправедливом* режиме, а *справедливый* режим включается явно. Предлагаемый в этой работе подход также поддерживает оба режима.

Помимо потокобезопасности от структур чаще всего ожидается *конкурентность*, то есть параллелизация исполнения операций, запрошенных разными потоками, там, где это возможно [5]. В разделе 3.1 мы опишем простейший подход обеспечения потокобезопасности, использующий дополнительную блокировку, запрашиваемую при исполнении каждой операцией. Такой подход лишает структуру данных конкурентности, так как все операции фактически исполняются последовательно. Целью этой работы является, наоборот, как можно большее увеличение конкурентности при использовании предлагаемого решения.

Ключевая метрика, которую мы будем рассматривать как целевую в работе, — *пропускная способность* структуры при заданных параметрах. Интуитивно она означает следующее: предположим, что структура данных рассматривается как черная коробка, с которой работают некоторые потоки, выполняя некоторый набор операций непрерывно. Пропускная способность отражает число операций, которые будут выполнены структурой в такой ситуации за некоторый промежуток времени. Более подробно метрика будет описана в начале раздела 4.

3. Подходы к обеспечению синхронизации операций в потокобезопасных структурах данных

Безусловно, в большинстве языков программирования уже существуют стандартные библиотеки, предлагающие пользователям большое число различных потокобезопасных структур данных. Однако потребность в механизме, помогающем с синхронизацией операций в структурах данных, все равно остается актуальной. Во-первых, такая потребность возникает при разработке собственных специфичных для конкретных задач структур данных. Во-вторых, потребность также возникает при расширении существующей структуры дополнительными операциями. Например, в произвольной структуре данных может возникнуть потребность в выполнении свертки (например, подсчет суммы всех значений) над элементами структуры, а в упорядоченных структурах данных может возникнуть потребность выполнять добавление или установление нового значения на отрезке. В-третьих, оказывается, что в реализациях из стандартных библиотек могут существовать ошибки, приводящие к проблемам с линеаризуемостью, что делает использование таких структур невозможным, если линеаризуемость является критичной для решаемой задачи.

3.1. Использование простой блокировки для синхронизации операций

Простейший способ гарантировать линеаризуемость — использовать одну дополнительную блокировку. В этом случае все операции для исполнения требуют взятия блокировки, а после выполнения ее отпускают. Псевдокод такого подхода приведен на рис. 1.

Многие языки предоставляют стандартный механизм превращения структуры данных в потокобезопасную с помощью такого подхода. Например, в языке программирования Java существует метод *Collections.synchronizedCollection(someStructure)*, который оборачивает все операции в структуре данных в дополнительную блокировку, превращая ее в потокобезопасную.

```
operation():
    lock.lock()
    original_operation()
    lock.unlock()
```

Рис. 1. Синхронизация с помощью дополнительной блокировки

3.2. Использование блокировки чтения—записи для синхронизации операций

Понятно, что такой подход не является оптимальным. Так, рассмотрим операции, которые не модифицируют внутренние данные структуры, а лишь читают их. Такие операции могут быть выполнены параллельно, поскольку выполнение одной операции не может повлиять на данные, а следовательно, и на результат выполнения параллельной операции.

Определение 3. Операции, которые не изменяют внутреннее содержимое структуры данных, будем называть *операциями чтения*.

Определение 4. Операции, которые могут изменить внутреннее содержимое структуры данных, будем называть *операциями записи*.

Операции чаще всего легко и достаточно удобно разделить на операции чтения и записи (при сомнениях можно просто отнести операцию к операциям записи); при этом реализация подхода, при котором операции чтения могут выполняться параллельно, может привести к значительному увеличению производительности, как мы увидим в разделе 4. Это приводит к появлению специального вида блокировок, которые реализованы в большинстве языков программирования в стандартных библиотеках — блокировок чтения—записи [15].

Определение 5. *Блокировка чтения—записи* — специальный вид блокировки, который предоставляет методы для взятия блокировок раздельно на чтение и запись. При этом блокировкой на чтение могут одновременно владеть несколько потоков при условии, что никакой поток не владеет блокировкой на запись, в то время как блокировка на запись может быть взята только одним потоком при условии, что никакие другие потоки не владеют ни блокировкой на чтение, ни блокировкой на запись.

Пример применения блокировки чтения—записи в методах структуры, осуществляющих доступ к одной переменной, приведен на рис. 2.

Блокировка чтения—записи также может быть использована для более эффективной синхронизации при расширении потокобезопасной структуры данных новыми операциями. Поскольку операции, представленные изначально в структуре данных, уже обеспечены потокобезопасностью, их не нужно дополнительно синхронизировать между собой дополнительной блокировкой. Если же операция расширяет структуру данных, то она потенциально может конфликтовать с любой из имеющихся или новых операций. Логика таких зависимостей совпадает с логикой зависимостей между операциями чтения и записи. Это позволяет использовать для дополнительной синхронизации блокировку чтения—записи. Так, все изначально представленные операции будут требовать взятия блокировки чтения перед своим исполнением, что позволит допускать исполнение таких операций из нескольких потоков без дополнительной синхронизации. А все новые операции будут требовать

```
get():
    readWriteLock.lockRead()
    result = value
    readWriteLock.unlockRead()
    return result

set(newValue):
    readWriteLock.lockWrite()
    prev = value
    value = newValue
    readWriteLock.unlockWrite()
    return prev
```

Рис. 2. Использование блокировки чтения—записи

взятия блокировки записи, гарантируя тем самым, что такие операции не будут исполнены параллельно с другими.

Пример применения блокировки чтения—записи при расширении структуры новыми операциями представлен на рис. 3.

```
operation1():
    readWriteLock.lockRead()
    originalOperation1() <- вызов исходного метода
    readWriteLock.unlockRead()

operation2():
    readWriteLock.lockRead()
    originalOperation2() <- вызов исходного метода
    readWriteLock.unlockRead()

newOperation3():
    readWriteLock.lockWrite()
    .. <- логика операции 3, расширяющей исходную структуру
    readWriteLock.unlockWrite()

newOperation4():
    readWriteLock.lockWrite()
    .. <- логика операции 4, расширяющей исходную структуру
    readWriteLock.unlockWrite()
```

Рис. 3. Использование блокировки чтения—записи при расширении структуры

3.3. Анализ зависимостей между операциями для их синхронизации

Использование блокировки чтения—записи позволяет улучшить производительность структуры данных по сравнению с использованием обычной блокировки, однако легко привести пример, в котором зависимости между операциями устроены сложнее: например, некоторые операции записи могут быть выполнены параллельно друг другу.

Рассмотрим структуру данных, представляющую упорядоченный набор *атомарных* (то есть таких, с которыми можно работать из нескольких потоков без дополнительных синхронизаций) чисел, на котором определены три операции: *addOnRange*, которая прибавляет ко всем числам на отрезке некоторое переданное значение, *setOnRange*, которая устанавливает всем числам на отрезке некоторое переданное значение, и *getRangeSum*, которая возвращает сумму чисел на отрезке. Заметим, что выполнение двух операций *addOnRange* параллельно друг с другом не приводит к проблемам с линеаризуемостью, поскольку после завершения обеих операций к каждому элементу структуры будут прибавлены все нужные значения. В то же время для операции *setOnRange* это неверно. У элементов, которые принадлежат обоим отрезкам, могут оказаться в итоге различные значения из-за состояния гонки, что невозможно при последовательном выполнении операций. Очевидно, что операции *getRangeSum* также можно выполнять параллельно друг другу, ведь это просто операции чтения. Получается, что в данном случае более оптимально было бы использовать блокировку, которая допускала бы одновременное исполнение нескольких операций *addOnRange* или *getRangeSum*, но не допускала бы одновременного выполнения операций различного типа или нескольких операций *setOnRange*.

Для того чтобы более эффективно работать с такими структурами данных, в этой работе мы предлагаем собственную *семантическую* блокировку, поведение которой основывается на информации о конфликтах между операциями структуры данных.

Определение 6. Две операции называются *зависимыми* (конфликтующими), если их параллельное исполнение может привести к потери линеаризуемости.

При создании семантическая блокировка принимает на вход информацию о зависимостях между операциями, представляющую из себя матрицу, где на i-й строке j-го столбца стоит логическое значение $True$, если операции j и i зависимы (то есть конфликтуют), и $False$ иначе. В матрице также отражается возможность или невозможность параллельного выполнения операций одного типа с помощью соответствующих значений в i-й строке i-м столбце. Пример матрицы зависимостей для структуры, описанной выше, приведен в табл. 1.

Таблица 1. Пример матрицы зависимостей

	addOnRange	**setOnRange**	**getRangeSum**
addOnRange	False	True	True
setOnRange	True	True	True
getRangeSum	True	True	False

Дальнейшая работа с блокировкой происходит с помощью двух основных методов: *tryLock(operation)* и *unlock(operation)*.

При выполнении каждой операции поток запрашивает разрешение, вызывая метод *tryLock* с идентификатором соответствующей операции в качестве параметра. В случае получения разрешения поток выполняет операцию и в конце вызывает метод *unlock*, сообщая тем самым о завершении операции.

Разберем детали работы перечисленных выше методов. Для этого введем понятие графа зависимостей операций.

Определение 7. Определим *граф зависимостей* операций как граф, построенный с помощью описанной выше матрицы смежности следующим образом: вершины в графе отвечают за операции, а ребра (в том числе петли) проведены между вершинами, соответствующими зависимым операциям.

С точки зрения *графа зависимостей* при вызове *tryLock(a)* необходимо проверить, выполняется ли хотя бы одна операция в данный момент, вершина которой соединена ребром с вершиной операции a.

```
// сохраненные при инициализации номера компонент связанности для каждой операции
operationId2ComponentId = ..
// массив простых блокировок для каждой компоненты связанности
componentLocks = Lock[componentsCount]
// объект, в котором каждой операции по ее operationId сопоставлен список id операций,
//  с которыми она конфликтует
graph = ..
// массив счетчиков (обычных (неатомарных) чисел), отражающих число операций,
//  которые сейчас исполняются для каждого типа операций
inProgress = Int[operationsCount]

tryLock(operationId):
    // взятие блокировки, связанной с компонентой связанности
    componentId = operationId2ComponentId[operationId]
    componentLocks[componentId].lock()

    // проверка отсутствия конфликтов
    for conflictOperationId in graph[operationId]:
        if inProgress[conflictOperationId] > 0:
            return False

    // увеличение соответствующего счетчика
    inProgress[operationId]++
    componentLocks[componentId].unlock()
    return True

unlock(operationId):
    // взятие блокировки, связанной с компонентой связанности,
    //  необходимо из-за неатомарности счетчиков и возможности одновременного
    //  уменьшения счетчика из разных потоков
    componentId = operationId2ComponentId[operationId]
    componentLocks[componentId].lock()

    inProgress[operationId]--
    componentLocks[componentId].unlock()
```

Рис. 4. Проверка конфликтов с помощью отдельной блокировки

Для осуществления такой проверки будем поддерживать внутри семантической блокировки, помимо графа зависимостей, также счетчики для каждой операции, отражающие число операций заданного типа, выполняемых в данный момент. Заметим, что такое число может быть больше 1, если операция не конфликтует сама с собой.

Таким образом, при вызове операции *tryLock(a)* необходимо проверить счетчики всех операций, связанных с операцией a ребром. Если все такие счетчики равны 0, то операция a может быть исполнена.

Проблема, которая возникает с таким подходом, связана с тем, что метод *tryLock* может быть одновременно вызван из нескольких потоков, что может привести к тому, что две конфликтующие операции запустятся одновременно, так как на момент проверки счетчики были равны 0 для обеих из них.

Для того чтобы решить эту проблему, введем также внутренние блокировки для каждой компоненты связанности в графе зависимостей. Это позволит независимо проверять возможность исполнения операций, которые могут конфликтовать, а также безопасно менять счетчики без использования атомарных переменных.

Псевдокод реализации такой проверки приведен на рис. 4.

Другой возможный подход для решения этой проблемы — использование атомарных значений—счетчиков. При этом в начале проверки соответствующее значение увеличивается, а лишь затем проверяются счетчики конфликтующих операций. Псевдокод такой проверки приведен на рис. 5.

```
graph = ..
// массив счетчиков (атомарных чисел), отражающих число операций,
//  которые сейчас исполняются для каждого типа операций
inProgress = AtomicInteger[operationsCount]

tryLock(operationId):
    // увеличение счетчика
    if operationId in graph[operationId] and inProgress[operationId] > 0:
        return False
    inProgress[operationId]++

    // проверка отсутствия конфликтов
    for conflictOperationId in graph[operationId]:
        if conflictOperationId == operationId:
            if inProgress[operationId] > 1:
                inProgress[operationId]--
                return False
        elif inProgress[conflictOperationId] > 0:
            inProgress[operationId]--
            return False

    return True

unlock(operationId):
    inProgress[operationId]--
```

Рис. 5. Проверка конфликтов с помощью атомарных счетчиков

Сравнение эффективности таких подходов будет приведено в разделе 4.3.

3.4. Обеспечение справедливости структуры данных

Выше мы описали разделение структур на справедливые и несправедливые. Для обеспечения поддержки справедливости в структурах, использующих семантическую блокировку, перед первой попыткой взятия блокировки поток добавляет свой идентификатор в специальную очередь, связанную с компонентой связанности текущей операции, после чего метод *tryLock* перед началом исполнения всего остального кода проверяет соответствие первого элемента очереди идентификатору текущего потока. Если идентификаторы не совпали, то метод сразу возвращает отказ во взятии блокировки. Если же идентификаторы совпали, то поток выполняет все необходимые проверки и при отсутствии конфликтов удаляет первый элемент из очереди.

Способы гарантии справедливости структуры данных при использовании семантической блокировки могут быть возможным направлением дальнейших исследований.

Во всех экспериментах в разделе 4 используется справедливый режим работы блокировок.

4. Эксперименты

4.1. Методика проведения экспериментов и целевая метрика

Для проведения сравнений *семантическая* блокировка была реализована на языке Java. Для сравнения с другими подходами был выбран вариант реализации с дополнительными блокировками на компоненты связанности при проверке наличия конфликтов. Сравнение такой реализации с реализацией на основе атомарных счетчиков приведено в разделе 4.3.

Обычная блокировка и блокировки чтения—записи были взяты из стандартной библиотеки Java.

Для проведения экспериментов различные способы синхронизации были применены к тестовой структуре данных. Все реализации были успешно проверены на линеаризуемость с помощью инструмента Lincheck [16].

Исходные коды реализаций семантической блокировки и тестовой структуры данных свободно доступны в репозитории GitHub [17].

Тестовая структура данных представляет собой массив числовых значений со следующими операциями: *addOnRange(left, right, value)* — добавляет значение $value$ ко всем элементам на отрезке с индексами от $left$ (включительно) до $right$ (невключительно); *setOnRange(left, right, value)* — устанавливает значение $value$ всем элементам на отрезке с индексами от $left$ (включительно) до $right$ (невключительно); *getRangeSum(left, right)* — возвращает сумму значений элементов на отрезке с индексами от $left$ (включительно) до $right$ (невключительно).

Позднее мы усложним *граф зависимостей* структуры за счет дополнительных оптимизаций, рассмотренных в разделе 4.4.

В экспериментах использовалась модифицированная версия инструмента для тестирования конкурентных структур данных под нагрузкой — Synchrobench [18]. Модификация инструмента потребовалась из-за того, что исходно Synchrobench поддерживает лишь работу со словарями (интерфейс Map) и множествами (интерфейс Set).

Synchrobench поддерживает достаточно большой набор параметров; основные неизменные в различных запусках параметры перечислены в табл. 2.

Таблица 2. Постоянные параметры запуска Synchrobench

Параметр	Значение
длительность прогрева	10 секунд
длительность эксперимента	10 секунд
размер структуры данных	$2^{26} = 67108864$ (если не указано иное)
количество итераций	10

Помимо перечисленных параметров, при запуске указывались два переменных параметра — число потоков *threads*, используемых в эксперименте, а также дискретное вероятностное распределение P, используемое для выбора очередной операции.

После запуска Synchrobench создает *threads* потоков, каждый из которых в цикле пытается выполнить одну из возможных операций со структурой данных. При этом каждая операция выбирается заново случайным образом с вероятностями, соответствующими дискретному распределению P. Например, если структура поддерживает две операции *read* и *write*, то параметр P должен задавать вероятность выбора *read* и *write* соответственно, причем сумма этих вероятностей должна быть равна 1.

В качестве целевой метрики в экспериментах рассматривается пропускная способность структуры при заданных переменных параметрах (число потоков и распределение операций). Эта метрика отражает число операций, которые были завершены за некоторый фиксированный промежуток времени (обычно, за секунду).

Эксперименты проводились на машине с процессором Xeon Gold 5128 с 16 ядрами и поддержкой гипертрединга и с 64 Гб оперативной памяти. Результаты части экспериментов также были подтверждены на машине с процессором Apple M4 Pro с 14 ядрами (из которых 10 производительных) и с 48 Гб оперативной памяти, что позволяет сделать вывод об отсутствии значимого влияния используемой платформы на эффективность решения.

4.2. Сравнение различных подходов для синхронизации

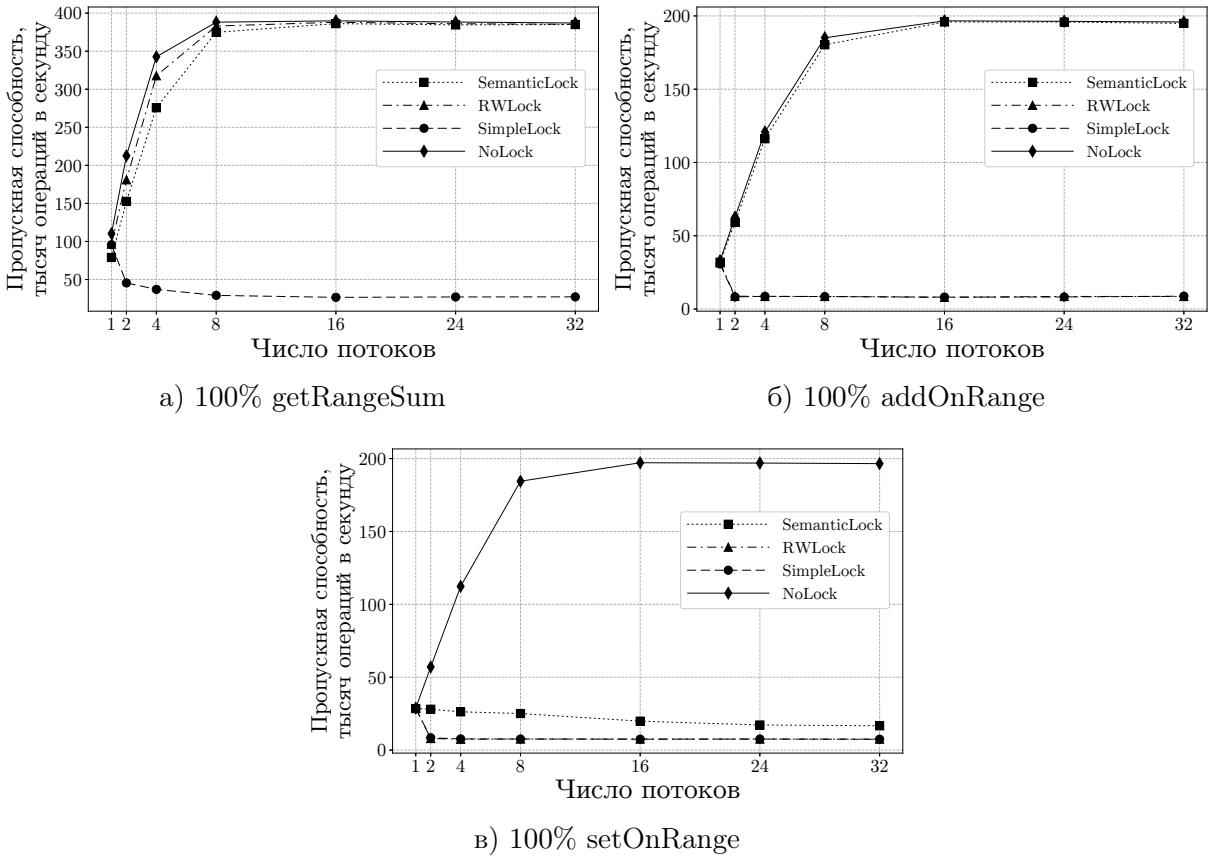
В этом эксперименте рассматриваются три подхода — использование дополнительной обычной блокировки (**SimpleLock** на рисунках ниже); использование дополнительной блокировки чтения—записи (**RWLock** ниже); использование семантической блокировки (в варианте неатомарных счетчиков и проверки конфликтов под дополнительной блокировкой, **SemanticLock** ниже).

Для сравнения также приводятся результаты для структуры, не использующей дополнительные блокировки, результаты работы над которой не гарантируют линеаризуемости (**NoLock** ниже).

Границы отрезков в рамках этого и следующих экспериментов выбирались следующим образом: сначала случайно выбирается левая граница $left$ равномерно по всей длине структуры, после этого выбирается длина отрезка $length$ случайно равномерно от 0 до $min(10^4, size - left)$, затем при помощи сложения левой границы и длины формируется параметр правой границы $right = left + length$.

На рис. 6 показаны зависимости пропускной способности от числа потоков при выполнении всеми потоками операций одного типа.

Видно, что для операции *getRangeSum* реализации и с блокировкой чтения—записи, и с семантической блокировкой показывают рост пропускной способности с ростом числа



а) 100% getRangeSum

б) 100% addOnRange

в) 100% setOnRange

Рис. 6. Результаты в случае выполнения операций одного типа

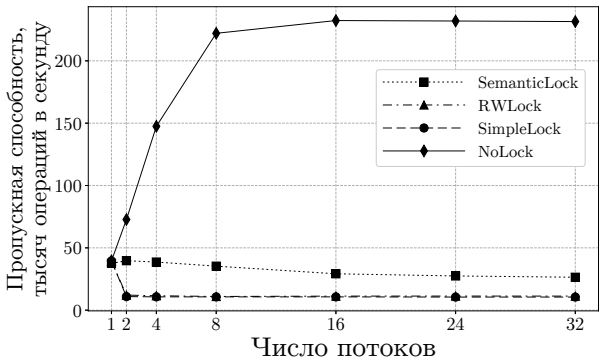
потоков, при этом находясь близко к значению пропускной способности нелинеаризуемой структуры без блокировок. Это свидетельствует, во-первых, о низких накладных расходах на обе блокировки, а во-вторых, об успешном параллельном исполнении операции чтения в этих случаях. Решение, основанное на простой блокировке, не только не показывает роста, но и, наоборот, с переходом к нескольким потокам пропускная способность падает. По остальным графикам видно, что в процентном соотношении падение для операции записи более значительно, чем для операции чтения. Это падение связано с работой кэшей и происходящими синхронизациями потоков.

Результаты для операции *addOnRange* также показывают преимущества семантической блокировки. Там, где блокировка чтения—записи уже не может поддержать корректное с логической точки зрения одновременное исполнение операций, семантическая блокировка позволяет это сделать, тем самым повышая пропускную способность с ростом числа потоков, по-прежнему позволяя поддерживать пропускную способность на уровне реализации без блокировок.

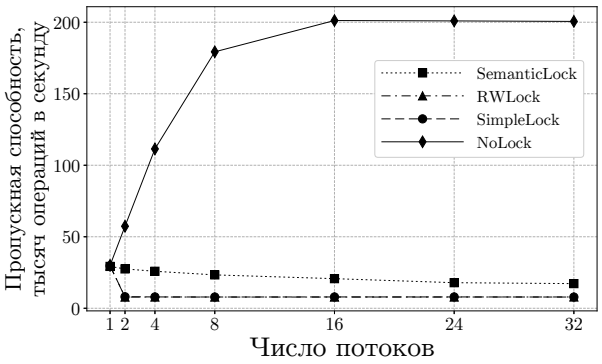
Для операции *setOnRange* пропускная способность всех реализаций падает при переходе к работе с несколькими потоками, аналогично реализации с простой блокировкой в случае чтения. При этом отметим, что использование семантической блокировки в этом случае позволило минимизировать падение производительности.

Рассмотрим теперь смешанные ситуации, в которых распределение операций устроено сложнее.

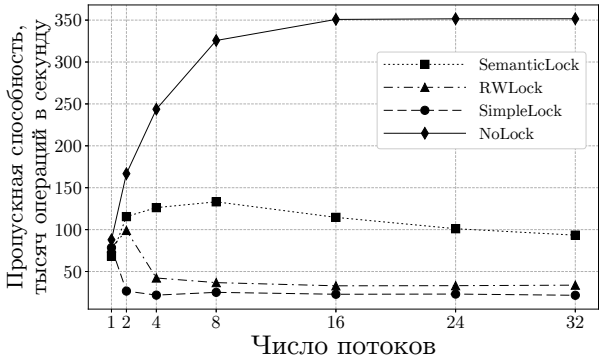
Возьмем несколько вариантов таких распределений: равномерное между всеми типами, доминирование операции чтения (90% *getRangeSum*, по 5% операций записи каждого типа), доминирование операции *addOnRange*, доминирование операции *setOnRange*.



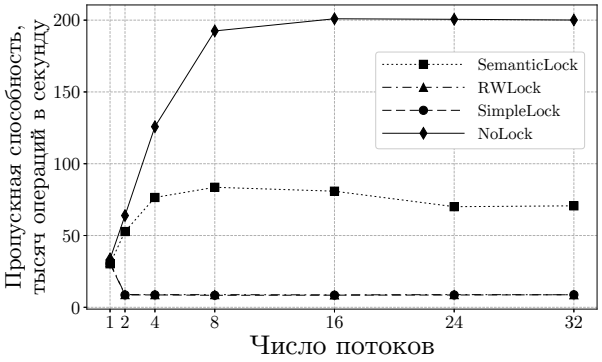
а) равномерное распределение операций



б) 90% setOnRange, 5% addOnRange, 5% getRangeSum



в) 5% setOnRange, 5% addOnRange, 90% getRangeSum



г) 5% setOnRange, 90% addOnRange, 5% getRangeSum

Рис. 7. Результаты в случае выполнения операций различного типа

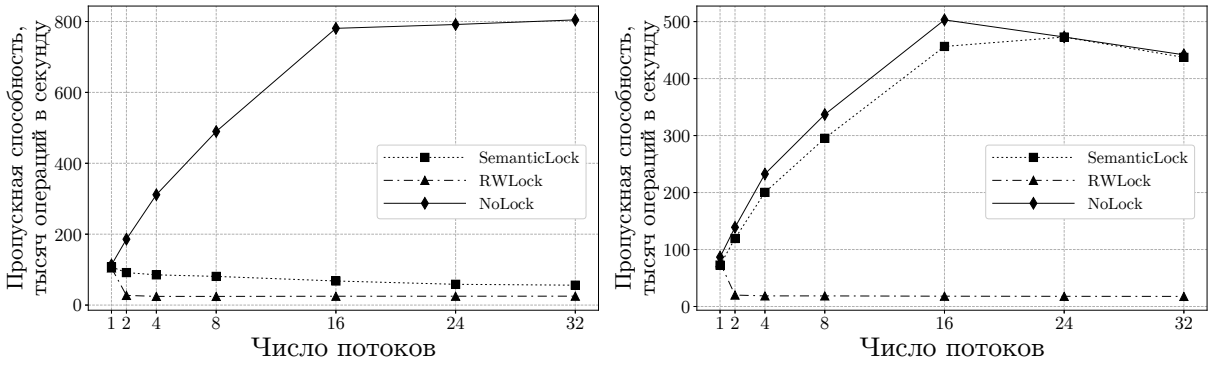
Результаты экспериментов, приведенные на рис. 7, также показывают эффективность предложенного решения.

В случае доминирования операций, которые могут быть выполнены параллельно, семантическая блокировка позволяет увеличить пропускную способность при переходе к нескольким потокам, в то время как даже для операции чтения блокировка чтения—записи показывает лишь небольшой рост при переходе к двум потокам, после чего пропускная способность падает, хоть и остается на уровне, более высоком, чем для обычной блокировки.

В случае же равномерного распределения операций использование семантической блокировки позволило лишь незначительно улучшить пропускную способность при работе из двух потоков; с дальнейшим увеличением числа потоков семантическая блокировка позволила сохранить пропускную способность примерно на том же уровне без значительного падения.

Выход на плато для реализации без блокировок связан с тем, что размер массива не позволяет полностью поместить его в кэш. Эту гипотезу подтверждают результаты подобных экспериментов с меньшим размером массива — 2^{13} элементов, которые приведены на рис. 8. Видно, что рост прекращается лишь после превышения числом потоков числа физических ядер. При этом поддержка гипертрединга в процессоре не помогает увеличить пропуск-

ную способность после превышения числом потоков числа ядер, так как в первую очередь направлена на решение проблемы блокирующих операций, которая в таких примерах не возникает.

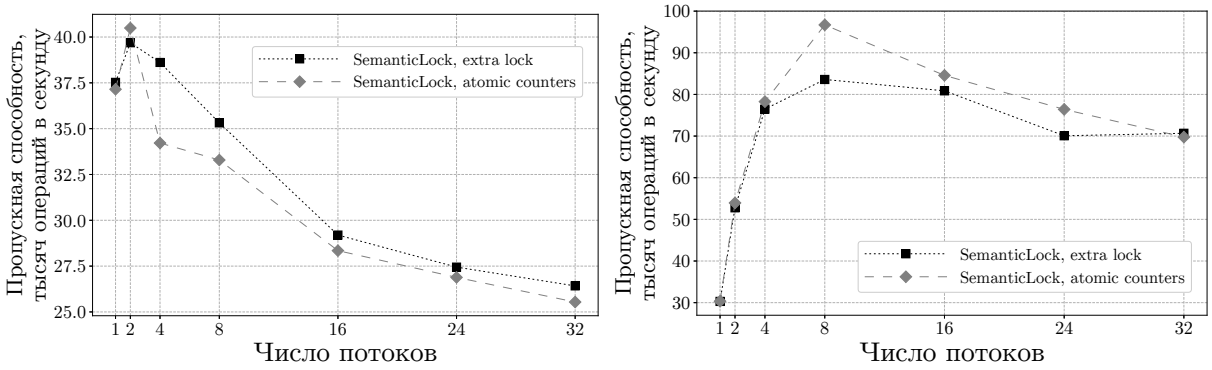


а) равномерное распределение операций б) 100% addOnRange

Рис. 8. Результаты с меньшим числом элементов (2^{13}) в массиве

4.3. Результаты сравнения различных способов проверки конфликта

В этом разделе приводятся результаты сравнения двух способов проверки наличия конфликта, изложенных в разделе 3.3. Использованный в остальных экспериментах подход, основанный на взятии дополнительной блокировки, связанной с компонентой связанности операции, для проверки конфликтов, сравнивается с подходом, основанным на использовании атомарных счетчиков.



а) равномерное распределение операций б) 5% setOnRange, 90% addOnRange, 5% getRangeSum

Рис. 9. Результаты сравнения различных реализаций проверки конфликта

Из результатов, приведенных на рис. 9, сложно сделать выбор в пользу того или иного решения.

Более детальное сравнение подходов может быть направлением дальнейших исследований и улучшений семантической блокировки.

4.4. Разделение операций на длинные и короткие

При работе со структурой значительного размера захват блокировок на всю структуру может быть неоптимальным, если в запросе используются небольшие отрезки. Действи-

тельно, в этом случае было бы эффективнее разделить весь массив на небольшие отрезки и захватывать блокировку лишь на те отрезки, которые пересекаются с отрезком запроса.

Однако накладные расходы на захват большого количества блокировок высоки, поэтому для длинных отрезков такой подход в свою очередь был бы неоптимальным.

Возможная оптимизация с учетом этой идеи состоит в разделении всех операций на *длинные* и *короткие*. Короткие операции не конфликтуют друг с другом (то есть соответствующие вершины не соединены ребром в графе зависимостей) и перед выполнением дополнительно с основной блокировкой (семантической или чтения—записи в зависимости от реализации) захватывают блокировки на все необходимые отрезки массива. Длинные же операции конфликтуют со всеми короткими и не требуют захвата дополнительных блокировок. При этом короткая операция чтения не конфликтует с длинной операцией чтения.

Что считать длинной операцией, а что короткой — гиперпараметр, который может быть предметом дополнительной оптимизации; в нашем примере длинной операцией считаются операции, работающие с отрезком длины больше, чем $2^{26}/10^4 = 6710$ элементов. При этом весь массив был разбит на $2 \cdot 10^4$ отрезков, каждый из которых при выполнении короткой операции защищен отдельной блокировкой. Такой подход позволяет захватывать не более двух отрезков при выполнении одной операции.

Подобная оптимизация приводит к гораздо более сложному графу зависимостей, который изображен на рис. 10.

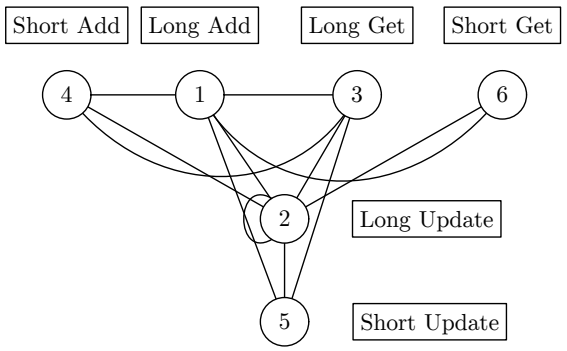


Рис. 10. Граф зависимостей при разделении на долгие и короткие операции

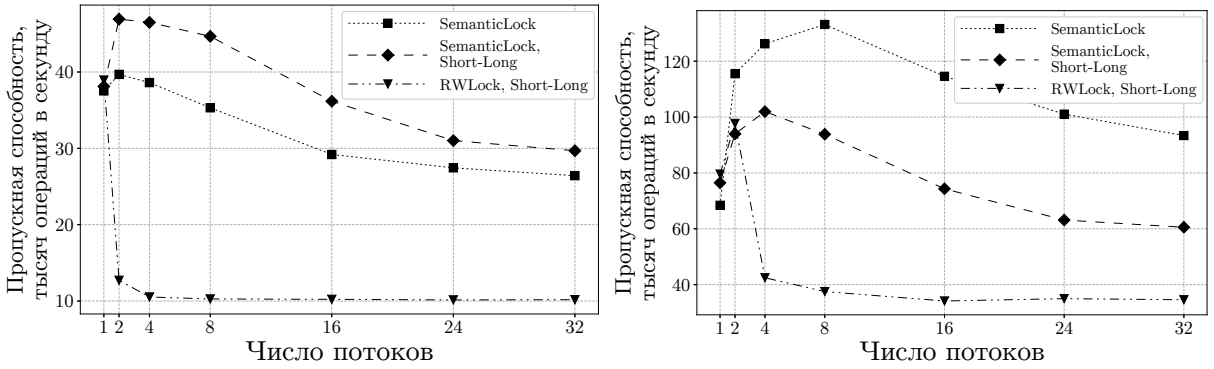
На рис. 11 приведены результаты сравнения трех реализаций: с разбиением на долгие и короткие операции и семантической блокировкой, с разбиением на долгие и короткие операции и блокировкой чтения—записи, без разбиения и с семантической блокировкой.

При равномерном распределении операций использование оптимизации позволило на 10–20% увеличить пропускную способность. Однако при доминировании операции чтения результаты оказались противоположными — использование оптимизации, наоборот, уменьшило пропускную способность на 20–40%.

Отметим при этом следующий результат: использование в таком более сложном графе зависимостей семантической блокировки не привело к заметному усложнению кода, но при этом дало гораздо больший эффект, чем такой же набор операций, синхронизирующийся с помощью блокировки чтения—записи.

4.5. Сравнение различных реализаций массива

В качестве дополнения мы также рассмотрели две возможные реализации атомарности на уровне доступа к элементу в языке Java.



а) равномерное распределение операций

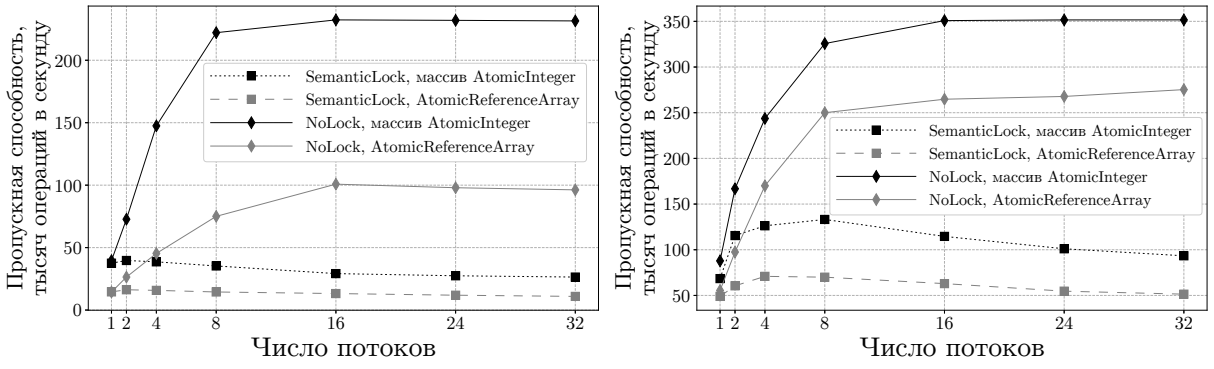
б) 5% setOnRange, 5% addOnRange, 90% getRangeSum

Рис. 11. Использование разбиения на долгие и короткие операции

В экспериментах выше массив представлял собой стандартный массив элементов типа AtomicInteger. Это позволяет обращаться к отдельным элементам, не задумываясь о дополнительных блокировках, что позволяет, например, исполнять параллельные записи во время одновременного исполнения нескольких операций *addOnRange*.

Другой подход основан на использовании класса AtomicReferenceArray, который позволяет атомарно оперировать отдельными элементами памяти; в данном случае, работая с массивом обычных числовых значений, мы позволяем атомарно взаимодействовать с отдельными элементами за счет операций контейнера.

Результаты экспериментов, приведенные на рис. 12, показывают, что подход, основанный на AtomicReferenceArray, оказывается менее эффективным для реализаций с различными видами блокировок.



а) равномерное распределение операций

б) 5% setOnRange, 5% addOnRange, 90% getRangeSum

Рис. 12. Результаты сравнения разных реализаций массива атомарных значений

Заключение

В работе был предложен новый алгоритм семантической блокировки, основанный на анализе графа зависимостей между операциями. Использование такой блокировки показало хорошие результаты в различных экспериментах, моделирующих ситуации, возникающие при реальной работе с потокобезопасными структурами данных. В то же время такая блоки-

ровка предоставляет простой интерфейс для использования, позволяющий без увеличения сложности кода гарантировать корректность синхронизации операций при создании новых структур данных или расширении существующих структур данных новыми операциями.

Дальнейшим направлением исследований по данной теме может стать разработка более эффективных алгоритмов проверки наличия конфликтующей операции в данный момент времени при имеющемся графе зависимостей, автоматизация построения графа зависимостей на основе анализа кода, а также построение математической модели, позволяющей показать преимущество семантической блокировки над стандартными типами блокировок.

Литература

1. Herlihy M.P., Wing J.M. Linearizability: a correctness condition for concurrent objects // ACM Trans. Program. Lang. Syst. 1990. Vol. 12, no. 3. P. 463–492. DOI: 10.1145/78969.78972.
2. Joung Y.-J. Asynchronous group mutual exclusion (extended abstract) // Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing. 1998. P. 51–60. PODC '98. DOI: 10.1145/277697.277706.
3. Yuan He K.G., Gafni E. Group mutual exclusion in linear time and space // Theoretical Computer Science. 2018. Vol. 709. P. 31–47. DOI: 10.1016/j.tcs.2017.05.030.
4. Maor L., Taubenfeld G. Constant RMR Group Mutual Exclusion for Arbitrarily Many Processes and Sessions // 35th International Symposium on Distributed Computing (DISC 2021). Vol. 209. 2021. P. 30:1–30:16. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.DISC.2021.30.
5. Aksenov V., Kuznetsov P., Shalyto A. Parallel Combining: Benefits of Explicit Synchronization // 22nd International Conference on Principles of Distributed Systems (OPODIS 2018). Vol. 125. 2019. P. 11:1–11:16. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.OPODIS.2018.11.
6. Brown T., Prokopec A., Alistarh D. Non-blocking interpolation search trees with doubly-logarithmic running time // Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York, NY, USA, 2020. P. 276–291. PPoPP '20. DOI: 10.1145/3332466.3374542.
7. Blelloch G.E., Wei Y. VERLIB: Concurrent Versioned Pointers // Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. 2024. P. 200–214. PPoPP '24. DOI: 10.1145/3627535.3638501.
8. Lomet D.B. Key Range Locking Strategies for Improved Concurrency // Proceedings of the 19th International Conference on Very Large Data Bases. 1993. P. 655–664. VLDB '93.
9. Eswaran K., Gray J., Lorie R., Traiger I. The Notions of Consistency and Predicate Locks in a Database System // Readings in Artificial Intelligence and Databases. Morgan Kaufmann, 1989. P. 523–532. DOI: 10.1016/B978-0-934613-53-8.50039-X.
10. Badrinath B.R., Ramamritham K. Semantics-Based Concurrency Control: Beyond Commutativity // ACM Trans. Database Syst. 1992. Vol. 17, no. 1. P. 163–199. DOI: 10.1145/128765.128771.
11. Weihl W. Commutativity-based concurrency control for abstract data types // IEEE Transactions on Computers. 1988. Vol. 37, no. 12. P. 1488–1505. DOI: 10.1109/12.9728.

12. Harris T., Larus J., Rajwar R. Software Transactional Memory // Transactional Memory. Springer International Publishing, 2010. P. 101–145. DOI: 10.1007/978-3-031-01728-5_4.

13. Herlihy M., Koskinen E. Transactional boosting: a methodology for highly-concurrent transactional objects // Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2008. P. 207–216. PPoPP '08. DOI: 10.1145/1345206.1345237.

14. Koval N., Fedorov A., Sokolova M., *et al.* Lincheck: A Practical Framework for Testing Concurrent Data Structures on JVM // Computer Aided Verification. 2023. P. 156–169. DOI: 10.1007/978-3-031-37706-8_8.

15. Courtois P.J., Heymans F., Parnas D.L. Concurrent control with "readers" and "writers" // Commun. ACM. 1971. Vol. 14, no. 10. P. 667–668. DOI: 10.1145/362759.362813.

16. Potapov A., Zuev M., Moiseenko E., Koval N. Testing Concurrent Algorithms on JVM with Lincheck and IntelliJ IDEA // Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2024. P. 1821–1825. ISSTA 2024. DOI: 10.1145/3650212.3685301.

17. Коротченко Д.С. SemanticLock. URL: https://github.com/ITMO-PTDC-Team/SemanticLock/tree/PCT-2026 (дата обращения: 28.07.2026).

18. Gramoli V. More than you ever wanted to know about synchronization: synchrobench, measuring the impact of the synchronization on concurrent algorithms // Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2015. P. 1–10. PPoPP 2015. DOI: 10.1145/2688500.2688501.

Коротченко Денис Сергеевич, аспирант, институт прикладных компьютерных наук, Национальный исследовательский университет ИТМО (Санкт-Петербург, Российская Федерация)

Аксенов Виталий Евгеньевич, PhD, доцент, институт прикладных компьютерных наук, Национальный исследовательский университет ИТМО (Санкт-Петербург, Российская Федерация)

DOI: 10.14529/cmse260205

SYNCHRONIZATION IN CONCURRENT DATA STRUCTURES BASED ON ANALYSIS OF DEPENDENCIES BETWEEN OPERATIONS

© 2026 D.S. Korotchenko, V.E. Aksenov

ITMO University (pr. Kronverkskiy 49A, St. Petersburg, 197101 Russia)

E-mail: korotchenkods@gmail.com, aksenov.vitaly@gmail.com

Received: 28.07.2026

This paper examines the synchronization of operations in thread-safe data structures. In different data structures, operations may conflict with each other in different ways in terms of concurrency. For example, a standard Read-Write lock recognizes that concurrent read accesses from different threads are often permissible, while parallel writes from different threads or parallel reads and writes most often require synchronization. This dependency can also be more complex: for example, incrementing all elements in a container by a given value is permissible with synchronization at the element access level, while concurrently assigning one number from one thread and a different number from another thread to all elements of a container requires more complex synchronization. This paper examines various methods for organizing synchronization between operations. The main result is a proposed new approach to ensuring such synchronization, based on the analysis of the dependency graph between operations, which ensures more efficient synchronization of the structure. Experimental results comparing various approaches are also presented, demonstrating the effectiveness of the proposed solution.

Keywords: thread-safe, concurrent data structures, dependency graph, semantic lock, linearizability.

FOR CITATION

Korotchenko D.S., Aksenov V.E. Synchronization in Concurrent Data Structures Based on the Analysis of Dependencies between Operations. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2026. Vol. 15, no. 2. P. 82–103. (in Russian) DOI: 10.14529/cmse260205.

References

1. Herlihy M.P., Wing J.M. Linearizability: a correctness condition for concurrent objects. ACM Trans. Program. Lang. Syst. 1990. Vol. 12, no. 3. P. 463–492. DOI: 10.1145/78969.78972.
2. Joung Y.-J. Asynchronous group mutual exclusion (extended abstract). Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing. 1998. P. 51–60. PODC '98. DOI: 10.1145/277697.277706.
3. Yuan He K.G., Gafni E. Group mutual exclusion in linear time and space. Theoretical Computer Science. 2018. Vol. 709. P. 31–47. DOI: 10.1016/j.tcs.2017.05.030.
4. Maor L., Taubenfeld G. Constant RMR Group Mutual Exclusion for Arbitrarily Many Processes and Sessions. 35th International Symposium on Distributed Computing (DISC 2021). Vol. 209. 2021. P. 30:1–30:16. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.DISC.2021.30.

5. Aksenov V., Kuznetsov P., Shalyto A. Parallel Combining: Benefits of Explicit Synchronization. 22nd International Conference on Principles of Distributed Systems (OPODIS 2018). Vol. 125. 2019. P. 11:1–11:16. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.OPODIS.2018.11.

6. Brown T., Prokopec A., Alistarh D. Non-blocking interpolation search trees with doubly-logarithmic running time. Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York, NY, USA, 2020. P. 276–291. PPoPP '20. DOI: 10.1145/3332466.3374542.

7. Blelloch G.E., Wei Y. VERLIB: Concurrent Versioned Pointers. Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. 2024. P. 200–214. PPoPP '24. DOI: 10.1145/3627535.3638501.

8. Lomet D.B. Key Range Locking Strategies for Improved Concurrency. Proceedings of the 19th International Conference on Very Large Data Bases. 1993. P. 655–664. VLDB '93.

9. Eswaran K., Gray J., Lorie R., Traiger I. The Notions of Consistency and Predicate Locks in a Database System. Readings in Artificial Intelligence and Databases. Morgan Kaufmann, 1989. P. 523–532. DOI: 10.1016/B978-0-934613-53-8.50039-X.

10. Badrinath B.R., Ramamritham K. Semantics-Based Concurrency Control: Beyond Commutativity. ACM Trans. Database Syst. 1992. Vol. 17, no. 1. P. 163–199. DOI: 10.1145/128765.128771.

11. Weihl W. Commutativity-based concurrency control for abstract data types. IEEE Transactions on Computers. 1988. Vol. 37, no. 12. P. 1488–1505. DOI: 10.1109/12.9728.

12. Harris T., Larus J., Rajwar R. Software Transactional Memory. Transactional Memory. Springer International Publishing, 2010. P. 101–145. DOI: 10.1007/978-3-031-01728-5_4.

13. Herlihy M., Koskinen E. Transactional boosting: a methodology for highly-concurrent transactional objects. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2008. P. 207–216. PPoPP '08. DOI: 10.1145/1345206.1345237.

14. Koval N., Fedorov A., Sokolova M., *et al.* Lincheck: A Practical Framework for Testing Concurrent Data Structures on JVM. Computer Aided Verification. 2023. P. 156–169. DOI: 10.1007/978-3-031-37706-8_8.

15. Courtois P.J., Heymans F., Parnas D.L. Concurrent control with "readers" and "writers". Commun. ACM. 1971. Vol. 14, no. 10. P. 667–668. DOI: 10.1145/362759.362813.

16. Potapov A., Zuev M., Moiseenko E., Koval N. Testing Concurrent Algorithms on JVM with Lincheck and IntelliJ IDEA. Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2024. P. 1821–1825. ISSTA 2024. DOI: 10.1145/3650212.3685301.

17. Korotchenko D.S. SemanticLock. URL: https://github.com/ITMO-PTDC-Team/SemanticLock/tree/PCT-2026 (accessed: 28.07.2026) (in Russian).

18. Gramoli V. More than you ever wanted to know about synchronization: synchrobench, measuring the impact of the synchronization on concurrent algorithms. Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2015. P. 1–10. PPoPP 2015. DOI: 10.1145/2688500.2688501.

УДК 004.89 **DOI: 10.14529/cmse260206**

ФОРМАЛИЗАЦИЯ ЗАДАЧИ ПРОЕКТНОГО ПЛАНИРОВАНИЯ ПРИ РАБОТЕ С НЕСТРУКТУРИРОВАННЫМ ОПИСАНИЕМ ЦЕЛИ

© 2026 Т.Р. Рафиков, А.В. Попцов, В.Д. Олисеенко, М.В. Абрамов

Санкт-Петербургский Федеральный исследовательский центр Российской академии наук
(199178 Санкт-Петербург, 14-я линия В.О., д. 39)
E-mail: trr@dscs.pro, avp@dscs.pro, vdo@dscs.pro, mva@dscs.pro

Поступила в редакцию: 08.05.2026

Большие языковые модели (LLM) демонстрируют высокую эффективность в задачах обработки естественного языка. Однако при применении LLM в планировании качество получаемых решений может снижаться, если задача содержит сложные зависимости между подзадачами, ограничения по ресурсам и срокам. Это ограничивает применимость LLM-планировщиков, где корректность плана критична, — в частности, в управлении проектами. Одним из перспективных направлений преодоления данного недостатка является использование LLM как средства построения формальной модели планирования, решение которой затем находится детерминированными алгоритмами. В данной статье рассматривается задача проектного планирования в условиях, когда исходная информация о проекте задана в неструктурированном текстовом виде. Вводится формализация задачи, в которой большая языковая модель используется для извлечения информации о проекте и построение формальной модели на языке PDDL. Такой подход позволяет связать текстовые описания задач, зависимостей, сроков и ресурсов с формальными средствами автоматического планирования. В результате задача проектного планирования переводится в форму, пригодную для проверки, валидации и последующего решения алгоритмическими методами. Кроме того, данное исследование расширяет математический аппарат классической постановки ресурсно-ограниченного проектного планирования. Помимо общего времени завершения проекта, добавляется учет штрафов за нарушение крайних сроков и качества назначения исполнителей. Тем самым классическая постановка дополняется критериями, которые отражают более сложные и практически значимые требования управления проектами. Данные изменения позволяют перейти от упрощенной однокритериальной модели к более реалистичной постановке. Проведенные вычислительные эксперименты показали, что предложенный подход увеличивает долю корректных планов с 0.1–0.9 до 0.95–1.0 для всех четырех протестированных моделей по сравнению с прямым использованием LLM в качестве планировщика.

Ключевые слова: планирование, большие языковые модели, формализация, LLM-агент.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Рафиков Т.Р., Попцов А.В., Олисеенко В.Д., Абрамов М.В. Формализация задачи проектного планирования при работе с неструктурированным описанием цели // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2026. Т. 15, № 2. С. 104–126. DOI: 10.14529/cmse260206.

Введение

Умение составлять планы и достигать поставленной цели — одна из важных способностей больших языковых моделей (англ. Large Language Models, LLM) и агентов на их основе (LLM-агентов). Данное умение требует глубокого понимания сложных зависимостей, способностей к рассуждению и принятию решений [1]. Несмотря на текущее возможности, LLM не демонстрируют высокого качества, особенно при ограничениях во времени или ресурсах [2].

Для описания окружения и задач планирования зачастую используется язык Planning Domain Definition Language (PDDL) [3]. Он позволяет строить формальные модели, ис-

пользуемые автоматическими планировщиками для решения поставленной задачи планирования [4, 5]. Кроме того, PDDL позволяет описывать дискретными терминами ключевые зависимости, предусловия и ограничения окружения, в которых работает LLM-агент [6]. В рамках данной работы PDDL рассматривается как язык формального описания задачи, который может использоваться планировщиками и автоматическими средствами валидации [7].

Особенно актуальным применение данного подхода представляется в предметных областях со сложной внутренней структурой задач и большим количеством ограничений. К их числу относится управление проектами, для которого характерны зависимости между подзадачами, многочисленные условия выполнения, ограничения по срокам, распределение по исполнителям и ресурсы [8]. В классической постановке такие задачи относятся к классу Resource Constrained Project Scheduling Problem (RCPSP) [9]. Однако реальные проекты значительно сложнее базовых академических моделей, поскольку включают неоднородность источников или неполноту данных, неточности в формулировках, мягкие и жесткие крайние сроки [10, 11].

Таким образом, целью данной работы является формализация задачи проектного планирования в условиях, когда исходная информация о проекте задана в неструктурированном текстовом виде. Достижение этой цели обеспечивается за счет извлечения информации о проекте из текстовых данных с помощью большой языковой модели и ее последующего представления на языке PDDL. Научная новизна работы состоит в объединении в рамках одной постановки неструктурированных проектных данных, LLM как средства формализации и PDDL как языка описания задачи планирования, а также в применении подхода LLM-as-formalizer к задаче проектного планирования. Теоретическая значимость работы заключается в расширении классической модели RCPSP за счет учета просрочек задач и качества назначения исполнителей. Практическая значимость работы состоит в создании формальной основы для применения существующих планировщиков и валидаторов к задачам управления проектами, исходно заданным в неструктурированном текстовом виде. Статья организована следующим образом. В разделе 1 проведен обзор существующих подходов использования LLM в задачах планирования. Раздел 2 посвящен языку PDDL, на котором строится дальнейшая формализация. В разделе 3 предлагается формализация задачи проектного планирования и ее перевод в PDDL. Раздел 4 описывает программную реализацию предложенного подхода. В разделе 5 предложенный подход сравнивается с LLM-as-planner на вычислительных экспериментах. В разделе 6 представлено обсуждение результатов работы, а также ограничения формализации. Заключение подводит итоги и обозначает направления дальнейших исследований.

1. Обзор литературы

В работах по LLM-агентам [1, 12, 13] планирование рассматривается как отдельный модуль агентной архитектуры. Декомпозиция задач, рефлексия, выбор действий, использование внешних модулей и памяти в обзоре [14] выделяют как неотъемлемые возможности планирования LLM. Другие работы посвящены использованию LLM в автоматическом планировании и составлении расписания [15]. Обзор [16] показывает, что LLM используются не только для генерации планов, но и для перевода в другой формат, интеграции инструментов, интерактивного планирования и выбора эвристик. При этом именно построение формальных моделей планирования и интеграция с внешними планировщиками рассмат-

риваются как наиболее перспективные направления [6]. В обзоре [6] LLM детально рассматривается как формализатор, то есть как средство перевода с естественного языка в формальное представление, позволяющее решать задачу планирования детерминированными алгоритмами.

При этом исследования подчеркивают и ограничения такого подхода. Работа по LLM-as-formalizer [17] показывает, что качество построения формальной модели зависит от сложности домена и структуры входного текста. Однако данный подход остается более надежным по сравнению с LLM-планировщиками [16]. Именно это лежит в основе дальнейшей формализации, где LLM отвечает за извлечение и структурирование знаний, а план ищет внешний планировщик. Сравнительная характеристика различных подходов по применению LLM к задаче планирования приведена в табл. 1.

Таблица 1. Сравнение подходов применения LLM в планировании

Источник	**Ключевые идеи, выделяемые категории**	**LLM как планировщик**	**LLM как формализатор**	**Формат**
A brain-inspired agentic architecture to improve planning with LLMs [18]	Модульная архитектура планирования с использованием декомпозиции задач и оркестратором	Да	Нет	Естественный язык
The rise and potential of large language model based agents: a survey [14]	Декомпозиция, поиск плана, внешний планировщик, рефлексия и память.	Да	Нет	Естественный и формальный языки
On the Prospects of Incorporating LLMs in APS [15]	Перевод в формальные языки, генерация плана, построение моделей, мультиагентное планирование, интерактивное планирование, оптимизация эвристик, вызов инструментов и когнитивное планирование	Да	Да	Естественный и формальный языки
LLM-as-Formalizer Survey [6]	Обзорная работа о процессе формализации окружения планирования с помощью LLM, рассматривая их генерацию, редактирование и оценку	Нет	Да	Естественный и формальный языки
Classical Planning with LLM-Generated Heuristics [19]	Использование LLM для генерации доменно-зависимых эвристик	Да	Нет	Формальный язык

В строках табл. 1 представлены рассматриваемые работы. В столбцах отражены источник, ключевые идеи, роль LLM и формат представления задачи. Столбцы «LLM как планировщик» и «LLM как формализатор» показывают, используется ли модель для прямого построения плана или для перевода задачи в формальное описание. Столбец «Формат» указывает, в каком виде задается задача: в естественном, формальном или в обоих языках. Анализ данных работ показывает, что существующие подходы к применению LLM в планировании можно условно разделить на два направления. В первом случае LLM используется непосредственно как планировщик. Во втором случае LLM выступает как формализатор для преобразования текстового описания задачи в структуру, поддерживаемую внешними планировщиками. При этом часть работ сочетает оба подхода, однако для задач со строгой формальностью подход LLM-формализатор является более надежным.

Одним из основных языков для формального описания задачи планирования выступает PDDL и его расширения. Следующая версия языка [20] вводит числовые выражения, временные действия и метрики плана, что позволяет использовать язык для временного планирования. Дальнейшее развитие PDDL получил в работе [21], где были добавлены предпочтения и мягкие ограничения, позволяя описывать желаемые свойства плана, что особенно важно в прикладных задачах.

Проектное планирование и задача его оптимизации получили свое развитие в работах [9, 10]. Исследование [11] показывает, что в реальных сценариях существенную роль играют гибкость ресурсов, различия в навыках исполнителей и их распределение. Поэтому минимизации времени завершения проекта как критерия качества в прикладных задачах недостаточно. В работе [22] показано, что соответствие работника требованиям конкретной задачи положительно связано с эффективностью выполнения этой задачи. Кроме того, само распределение исполнителей по задачам существенно влияет на общую эффективность, что подтверждается исследованием [23]. Это дает основание рассматривать качество назначения исполнителей как отдельный фактор для проектного планирования.

2. Формальный язык планирования PDDL

Для того чтобы формально описать перечисленные зависимости, ресурсы и качество назначений в едином представлении, обратимся к Planning Domain Definition Language (PDDL), на котором строится дальнейшая формализация [20]. Данный язык является одним из основных языков описания задач автоматического планирования. Его назначение состоит в том, чтобы построить формальную модель предметной области и конкретного экземпляра задачи. Данная модель далее используется автоматическими планировщиками [4, 5]. Исторически PDDL был введен как единый язык описания задач для International Planning Competition [3]. В дальнейшем он стал фактическим стандартом в исследованиях по классическому планированию.

Описание задачи в PDDL делится на две части: domain (домен) и problem (проблема). В файле домена задаются типы объектов, предикаты, числовые функции и действия. В файле проблемы задаются конкретные объекты, начальное состояние, целевые условия и, при необходимости, метрика качества плана. Такое разделение позволяет многократно использовать одно и то же формальное описание окружения для разных задач.

Синтаксис PDDL основан на префиксной записи в стиле языка программирования Lisp [24]. Основными конструкциями языка являются `:types`, `:predicates`, `:functions`, `:action`, `:preconditions`, `:effects` и `:goal`. Раздел `:types` задает типы объектов предмет-

ной области. Предикаты `:predicates` используются для описания логических свойств объектов и состояний. Функции `:functions` используются для задания числовых параметров, например, длительности или стоимости. Действие `:action` описывается через параметры, предусловия и эффекты. Предусловия `:preconditions` задают, при каких условиях действие может быть выполнено. Эффекты `:effects` задают, как изменяется состояние после его выполнения. Раздел `:goal` задает целевое состояние, которое должно быть достигнуто в результате построения плана. Ниже эти конструкции приведены на минимальном примере домена и связанной с ним проблемы.

В листинге 1 домен объявляется конструкцией `(define (domain mini) ...)`. В блоке `:requirements` указано, что используются типизированные объекты и числовые функции. Блок `:types` вводит типы `task` и `employee`, блок `:predicates` задает состояния задачи `ready`, `done` и отношение квалификации `qualified`. В разделе `:functions` вводится числовая функция `cost`, а действие `do-task` связывает задачу и исполнителя через параметры, проверяет готовность задачи и квалификацию исполнителя в `:precondition`, после чего переводит задачу в состояние `done` в `:effect`.

Листинг 1. Пример домена в PDDL

```
(define (domain mini)
  (:requirements :typing :fluents)      ; типизация и функции
  (:types task employee)                ; типы объектов
  (:predicates                          ; задаваемые отношения (предикаты)
    (ready ?t - task)                   ; задача готова
    (done ?t - task)                    ; задача выполнена
    (qualified ?e - employee ?t - task) ; квалификация
  )
  (:functions
    (cost ?t - task)                    ; функция стоимости задачи
  )
  (:action do-task                      ; задаваемое действие
    :parameters (?t - task ?e - employee)               ; параметры
    :precondition (and (ready ?t) (qualified ?e ?t))    ; предусловия
    :effect (done ?t)                                   ; итог
  )
)
```

В листинге 2 конструкция `(define (problem mini-problem) ...)` объявляет конкретный экземпляр задачи, а строка `(:domain mini)` связывает его с доменом из листинга 1. В блоке `:objects` создаются объект задачи `t1` и объект исполнителя `e1`. В разделе `:init` задается начальное состояние: задача готова к выполнению, исполнитель обладает нужной квалификацией, а значение функции `cost` для задачи равно единице. Блок `:goal` формулирует цель планирования как достижение состояния `(done t1)`.

В классическом варианте действия в PDDL рассматриваются без привязки ко времени. Однако для прикладных задач этого часто недостаточно. По этой причине в языке PDDL 2.1 были введены временные действия, числовые переменные и метрики плана [20]. Временное действие задается конструкцией `:durative-action`. Для него можно определить длительность, условия начала и окончания, а также условия, которые должны сохраняться

Листинг 2. Пример проблемы в PDDL

```
(define (problem mini-problem)
  (:domain mini)                          ; связь с доменом
  (:objects                               ; задаваемые объекты
    t1 - task                             ; задача
    e1 - employee                         ; сотрудник
  )
  (:init                                  ; изначальное состояние
    (ready t1)                            ; готовность задачи t1
    (qualified e1 t1)                     ; сотрудник e1 может исполнить t1
    (= (cost t1) 1)                       ; стоимость задачи 1
  )
  (:goal                                  ; задаваемая цель
    (done t1)                             ; выполнить задачу t1
  )
)
```

на протяжении всего выполнения действия. Дальнейшее развитие язык получил в работе PDDL 3 [21], где появилась возможность использовать мягкие ограничения и предпочтения.

3. Формализация задачи проектного планирования с представлением в PDDL

Как упоминалось выше, на практике задача оптимизации проектного планирования обычно выходит за рамки базовой формализации RCPSP. Реальные проекты часто включают дополнительные крайние сроки, многопроектную структуру, приоритеты задач, неоднородные ресурсы и иные ограничения, которые не всегда удобно выразить в классической постановке [25, 26]. По этой причине дальнейшая формализация в работе опирается на язык PDDL и его расширения. Такой выбор позволяет явно задавать действия, ограничения, ресурсы и целевые условия планирования. Однако перед описанием перевода в PDDL требуется формализовать поступающие на вход неструктурированные данные и задать критерий качества плана.

3.1. Формализация

В данной работе предполагается, что исходное описание проекта и задач задано в неструктурированном текстовом виде с описанием всех зависимостей, ограничений, крайних сроков и т.д. Обозначим множество неструктурированных данных о проекте как

$$D = \{v^{(1)}, v^{(2)}, \ldots, v^{(K)}\}, \tag{1}$$

где элементы $v^{(k)}$ представляют собой элементы текстовой информации: описания и очередь задач, проектную документацию, требования и так далее. Исходя из предоставленной информации в D впоследствии строится формальная модель для планирования.

Из множества D необходимо выделить следующие атрибуты предметной области:

- $T = \{t_1, t_2, \ldots, t_n\}$ — множество задач проекта;
- $E \subseteq T \times T$ — зависимости между задачами;
- $U = \{g_1, g_2, \ldots, g_p\}$ — множество исполнителей;

– $R = \{r_1, r_2, \ldots, r_m\}$ — множество типов ресурсов.

При этом, $(t_1, t_2) \in E \Leftrightarrow$ «t_2 зависит от t_1» и не существует зацикливающихся зависимостей. Кроме того, каждой задаче $t_i \in T$ задаются параметры:

– $d_i \in \mathbb{N}$ — длительность выполнения;
– $R_i \subseteq R$ — требуемые ресурсы;
– $U_i \subseteq U$ — исполнители для этой задачи;
– $\delta_i \in \mathbb{N}$ — крайний срок задачи;
– $w_i \in \mathbb{R}$ — штраф за просрочку задачи.

Для ресурсов введем функцию доступности $c_r(t) : \mathbb{N} \to \mathbb{R}$ в момент времени t, задающую объем ресурса r, доступный в момент t. Стоит отметить, что функция $c_r(t)$ задает ограничение допустимости плана и не входит непосредственно в оптимизируемую функцию, которая будет введена ниже. Кроме того, для каждой задачи t_i введем момент ее завершения $f_i \in \mathbb{N}$. Для описания качества распределения исполнителей по задачам введем параметр

$$p_{iu} \in R, \tag{2}$$

который задает стоимость или штраф назначения исполнителя u на задачу t_i. Также введем переменную

$$x_{iu} \in \{0, 1\}, \tag{3}$$

которая показывает, был ли назначен исполнитель u на задачу t_i. Кроме того, ни один исполнитель не может быть занят на двух задачах одновременно:

$$\sum_{i \in T:\ f_i - d_i \le t < f_i} x_{iu} \le 1, \quad \forall u \in U \text{ — исполнитель},\ \forall t \in \mathbb{N} \text{ — момент времени}. \tag{4}$$

Далее рассмотрим большую языковую модель как отображение

$$\Phi_\theta : D \to M = \langle Dom, Prob \rangle, \tag{5}$$

где Φ_θ — LLM с параметрами θ, а M — представление задачи в PDDL, состоящее из описания предметной области Dom и описания задачи $Prob$. LLM в предлагаемой постановке не выступает непосредственным планировщиком, а выполняет функцию извлечения и структурирования всей необходимой информации.

После построения модели M задача сводится к нахождению плана

$$\pi^* = \arg \min_{\pi \in \Pi(M)} J(\pi), \tag{6}$$

где $\Pi(M)$ — множество всех допустимых планов для модели M формата PDDL, а $J(\pi)$ — целевая функция качества плана. В изначальном варианте RCPSP в качестве целевой функции рассматривается минимизация времени [9]:

$$\min J(\pi) = C_{\max}, \tag{7}$$

где

$$C_{\max} = \max_{i \in T} f_i \tag{8}$$

— момент завершения последней задачи.

Для более реалистичной постановки в данной работе предлагается взять за целевую функцию

$$J(\pi) = \alpha C_{\max} + \beta \sum_{i \in T} w_i L_i + \gamma \sum_{i \in T} \sum_{u \in U} p_{iu} x_{iu}, \tag{9}$$

где $L_i = \max(0, f_i - \delta_i)$ — просрочка задачи t_i, а α, β, $\gamma \in R_{\geq 0}$ обозначают весовые коэффициенты критериев качества плана. Данный выбор обусловлен тем, что в RCPSP базовым критерием является минимизация $C_{\max}$, однако современные расширения проектного планирования рассматривают также задержки относительно крайних сроков и прочие различные характеристики описания как самостоятельные цели [10, 25, 26].

Первый член $\alpha C_{\max}$ был взят из классической постановки задачи оптимизации проектного планирования как стандартный критерий. Кроме времени завершения проекта также нужно учитывать и поставленные промежуточные крайние сроки по задачам. Это подтверждается работами [10, 27, 28], где рассматриваются цели со сроками и штраф за несоблюдение крайнего срока. Поэтому вторым членом $\beta \sum_{i \in T} w_i L_i$ учитывается нарушение сроков отдельных задач. Кроме того, исследования [22, 23, 29—31] показывают, что не менее важным фактором является назначение квалифицированных исполнителей на подходящую им задачу. Чтобы учесть этот критерий, вводится третье слагаемое $\gamma \sum_{i \in T} \sum_{u \in U} p_{iu} x_{iu}$, отвечающее за качество распределения исполнителей по задачам.

3.2. Перевод в PDDL

Кроме формализации необходимо также определить, каким образом выделенные данные отображаются в PDDL. В предлагаемой постановке модель M представляется в виде пары PDDL-представлений (файлов с расширением «.pddl»). Файл *Dom* задает описание предметной области: типы объектов, предикаты, числовые функции и действия. Файл *Prob*, в свою очередь, определяет поставленную задачу: множество задач проекта, исполнителей, ресурсов, начальное состояние и целевые условия.

В рамках данной работы множество задач в T будет представлено в объектах типа `task`, множество исполнителей U — в объектах типа `employee`, а множество ресурсов R — в объектах типа `resource`. Зависимости между задачами могут быть отображены явно через предикаты вида `(depends ?t1 ?t2)`. Состояния выполнения задач описываются предикатами `(ready ?t)` и `(finished ?t)`. Числовые функции в разделе `:functions` PDDL 3 могут быть использованы для длительности, доступности ресурсов и совокупной стоимости плана.

Одним из ключевых элементом в рассматриваемой задаче является продолжительность выполнения действия. Задачи с данным свойством в PDDL 3 задаются с помощью durative actions. В общем виде действие должно учитывать условия начала выполнения задачи, поддерживаемые условия на всем интервале ее выполнения и вклад выполненной задачи. Данная структура поддерживается во временных действиях PDDL 3. Пример такого временного действия приведен в листинге 3.

В листинге 3 конструкция `:durative-action` задает действие, выполнение которого занимает ненулевое время. Блок `:parameters` связывает действие с конкретной задачей `?t` и исполнителем `?u`, а выражение `:duration` определяет длительность через значение числовой функции `(task-duration ?t)`. Условия в разделе `:condition` помечены как `at start`: задача должна быть готова, а исполнитель должен обладать требуемой квалификацией в момент начала действия. В разделе `:effect` эффект `assigned` фиксирует назначение испол-

Листинг 3. Пример временного действия выполнения задачи

```
(:durative-action execute-task                    ; временное действие
  :parameters (?t - task ?u - employee)           ; задача, исполнитель
  :duration (= ?duration (task-duration ?t))      ; длительность задачи
  :condition (and
    (at start (ready ?t))                         ; задача готова
    (at start (qualified ?u ?t)))                 ; нужна квалификация
  :effect (and
    (at start (assigned ?t ?u))                   ; назначение в начале
    (at end (finished ?t))))                      ; завершение в конце
```

нителя в начале выполнения, а эффект `finished` применяется в конце действия. Благодаря этому завершение задачи может использоваться как условие для запуска зависимых работ.

Соответствующий файл *Prob* содержит имеющиеся объекты проекта, изначальные условия и поставленные цели. Например, исполнители, параметры задач и ресурсные ограничения задаются в изначальных условиях. Цели могут быть заданы как завершение всех задач проекта. Критерий качества можно задать как минимизацию целевой функции (9) с помощью `(:metric minimize <numeric_expression>)`.

Если в проекте присутствуют мягкие ограничения, например, нежелательность использования некоторых исполнителей для определенных задач, то их можно задать как `preference` в PDDL 3 [21]. В этом случае допустимость плана определяется ограничениями, а предпочтения влияют на его качество и ранжирование относительно других планов. Данный подход позволяет разделять задачи с безусловными условиями и задачи, допускающие компромиссы в условиях.

Таким образом, PDDL позволяет перейти от неструктурированных данных о проекте к формальной модели, решаемой детерминированными планировщиками. При этом LLM играет ключевую роль в формировании домена Dom и проблемы $Prob$ на основе данных D. Представленная схема соответствует подходу LLM-as-formalizer, демонстрирующему более надежную работу в сравнении с LLM-as-planner.

С помощью введенной формализации задача проектного планирования задается формулой (6) как поиск оптимального плана $\pi^* \in \Pi(M)$. Полученная формальная модель делает возможным применение существующих автоматических планировщиков: LAMA [5], A*-планировщиком [32], жадный поиском [4, 33], а также алгоритмы для решения комбинаторных задач оптимизаций, например, CP-SAT [34]. Для построения формальной модели $M = \langle Dom, Prob \rangle$ из множества неструктурированных проектных данных D при помощи LLM выделяются множества задач T, зависимостей E, исполнителей U и ресурсов R, а также параметры задач d_i, δ_i, w_i и функция доступности ресурсов $c_r(t)$. Тем самым осуществляется переход от неструктурированного описания проекта к математической модели, на которой определяется минимизируемая функция качества $J(\pi)$. Предложенная целевая функция (9) позволяет одновременно учитывать сроки проекта, просроченные задачи и качество распределения исполнителей. За счет этого классическая модель проектного планирования расширяется до более реалистичной постановки.

3.3. Примеры для формализации

Для наглядной иллюстрации применимости предложенной формализации ниже рассмотрены несколько типовых примеров.

Пример 1. Рассмотрим текстовое описание проекта вида: «Необходимо разработать внутренний веб-сервис. Сначала аналитик подготавливает требования, затем архитектор проектирует систему, после этого разработчик реализует модуль, а тестировщик проводит тестирование. Архитектура не может начинаться до завершения требований, разработка — до завершения архитектуры, тестирование — до завершения разработки. Для каждой задачи требуется один исполнитель соответствующей роли». В этом случае D содержит четыре текстовых артефакта или один общий артефакт с четырьмя задачами; множество задач задается как $T = \{t_1, t_2, t_3, t_4\}$, где t_1 — сбор требований, t_2 — проектирование, t_3 — разработка, t_4 — тестирование; зависимости задаются как $E = \{(t_1, t_2), (t_2, t_3), (t_3, t_4)\}$; исполнители $U = \{u_a, u_b, u_c, u_d\}$, ресурсы R могут быть сведены к ролям или рабочему времени; длительности, например, $d_1 = 2$, $d_2 = 3$, $d_3 = 5$, $d_4 = 2$. В PDDL это представляется объектами `req design code test` — `task`, предикатами `(depends req design)`, `(depends design code)`, `(depends code test)`, `(qualified analyst req)`, `(qualified architect design)`, а выполнение задачи задается через `:durative-action execute-task` с `:duration (= ?duration (task-duration ?t))`.

Пример 2. Рассмотрим текстовое описание проекта вида: «Команда готовит релиз мобильного приложения. Прототип интерфейса должен быть готов не позднее 5-го дня, серверная часть — не позднее 10-го дня, итоговая интеграция должна завершиться до 14-го дня. Нарушение промежуточных сроков допустимо, но нежелательно». Здесь кроме стандартных множеств T, E, U, R вводятся крайние сроки δ_i, например $\delta_{ui} = 5$, $\delta_{backend} = 10$, $\delta_{integration} = 14$, а функция качества уже задается не только через $C_{\max}$, но и через просрочки $L_i = \max(0, f_i - \delta_i)$. Если два плана имеют одинаковый $C_{\max}$, но один нарушает крайний срок прототипа, а другой нет, то они различаются именно вторым слагаемым $\sum_{i \in T} w_i L_i$. В PDDL такая постановка может быть задана через стандартные временные действия и метрику `(:metric minimize ...)`, а при необходимости мягкие крайние сроки могут быть выражены через preferences в PDDL 3, где нарушение срока не запрещает план, но ухудшает его качество.

Пример 3. Рассмотрим текстовое описание проекта вида: «В проекте разработки продукта задача «frontend» может быть выполнена либо senior-разработчиком, либо junior-разработчиком, но senior выполняет ее быстрее и с меньшим риском доработок; задача «архитектура» может быть назначена только senior-разработчику; задача «тестирование» может быть назначена двум разным тестировщикам с разной стоимостью времени». В этом случае недостаточно только множеств T и E; дополнительно требуется определить назначения x_{iu}, допустимость исполнителей и стоимость или предпочтительность назначения p_{iu}. Например, для задачи frontend можно задать $x_{front,u1}$, $x_{front,u2} \in \{0, 1\}$, где $u1$ — senior, $u2$ — junior, а значения $p_{front,u1} < p_{front,u2}$ или наоборот определяются выбранной интерпретацией качества назначения. В PDDL это выражается объектами `senior junior` — `employee`, предикатами `(qualified senior architecture)`, `(qualified senior frontend)`, `(qualified junior frontend)`, а также числовыми параметрами, связанными с длительностью или стоимостью выполнения.

4. Реализация предложенного подхода

Основные этапы формирования модели M и плана π^* приведены в Алгоритме 1. На шаге 1 отображение Φ_θ (большая языковая модель) по текстовому описанию задачи D строит формальную модель $M = \langle Dom, Prob \rangle$. Это пара PDDL-файлов: Dom задает словарь типов, предикатов и функций, а $Prob$ кодирует извлеченные из D работы T, зависимости E, исполнителей U, ресурсы R, параметры $\{d_i\}$, $\{\delta_i\}$, $\{w_i\}$, $\{p_{iu}\}$ и веса α, β, γ. Шаг 2 проверяет внутреннюю согласованность M функцией CheckConsistency; если модель некорректна, шаги 3–4 завершают обработку без построения плана. На шаге 6 по модели M с помощью BuildModel строится множество допустимых планов $\Pi(M)$. Шаг 7 задает целевую функцию $J(\pi)$, введенную в разделе 3. На шаге 8 среди допустимых планов $\Pi(M)$ находится план π^*, минимизирующий $J(\pi)$. Шаг 9 возвращает модель M и найденный план π^*.

Алгоритм 1. Формализация задачи проектного планирования при помощи LLM

Вход: D

Выход: $M = \langle Dom, Prob \rangle$, π^*

1: $M \leftarrow \Phi_\theta(D)$, $M = \langle Dom, Prob \rangle$ ▷ Извлечение данных с помощью LLM

2: $valid \leftarrow$ CheckConsistency(M) ▷ Проверка согласованности

3: **if not** $valid$ **then**

4: **return** $\varnothing$ ▷ Ошибка формализации

5: **end if**

6: $\Pi(M) \leftarrow$ BuildModel$(T, E, U, R, d, \delta, w, p)$ ▷ Построение расписания

7: $J(\pi) \leftarrow \alpha C_{\max}(\pi) + \beta \sum\limits_{i \in T} w_i L_i(\pi) + \gamma \sum\limits_{i \in T} \sum\limits_{u \in U} p_{iu} x_{iu}$ ▷ Целевая функция

8: $\pi^* \leftarrow \arg \min\limits_{\pi \in \Pi(M)} J(\pi)$ ▷ Оптимизация плана

9: **return** M, π^* ▷ Результат

Проверка CheckConsistency выполняется на двух уровнях. Первый уровень — внутренняя согласованность M: наличие всех необходимых сущностей T, U, R, корректность ссылок между ними, отсутствие циклов в зависимостях E. Второй уровень — соответствие извлеченных T, E, U, R и параметров d, δ, w, p эталонному ответу задачи. Результаты этого сравнения используются для оценки точности извлечения информации из текстового описания D, приведенной в разделе 5.

Функция BuildModel по извлеченным параметрам T, E, U, R, d, δ, w, p строит множество допустимых планов $\Pi(M)$. Каждой работе из T сопоставляются переменные начала и завершения, зависимости E задаются условием $\mathrm{end}(before) \leq \mathrm{start}(after)$, каждой работе назначается требуемое число исполнителей из допустимых по U с учетом того, что один исполнитель не может выполнять две работы одновременно. Потребление ресурсов из R на любом интервале времени не превышает их вместимости, а превышение крайнего срока δ_i учитывается как неотрицательная переменная просрочки, входящая в целевую функцию с весом w_i. Таким образом, $\Pi(M)$ включает все планы, удовлетворяющие перечисленным ограничениям, среди которых на следующем шаге отыскивается план π^*, минимизирующий $J(\pi)$.

Программная реализация выполнена на языке Python. Большинство классических PDDL-планировщиков не поддерживают модификации в оптимизируемую функцию [4, 5]. Поэтому для шагов 6–8 (построение множества допустимых планов $\Pi(M)$, задание целе-

вой функции и поиск оптимального плана π^*) используется алгоритм для решения задач комбинаторной оптимизации CP-SAT из библиотеки Google OR-Tools [34].

5. Вычислительные эксперименты

Дизайн эксперимента направлен на сравнение предложенного подхода (раздел 4) с базовым способом, при котором LLM планирует самостоятельно (LLM-as-planner). На вход в обоих случаях подается одно и то же текстовое описание D с одинаковой постановкой задачи: минимизировать функцию $J(\pi)$ с заданными параметрами и условиями. Код эксперимента и данные доступны в репозитории [35].

В случае LLM-as-planner текстовое описание D передается большой языковой модели с целью построения плана π в структурированном виде без формализации. Данный план проверяется на соответствие следующим условиям: каждая работа из T выполняется ровно один раз; момент начала каждой работы неотрицателен; длительность и число назначенных исполнителей соответствуют условию задачи; ни один исполнитель не занят одновременно на двух несовместимых по времени работах; зависимости E между работами соблюдены; потребление ресурсов R не превышает их вместимости на любом интервале времени. Для предложенного подхода, подразумевающего формализацию задачи, допустимость плана π^* обеспечивается уже самим построением множества $\Pi(M)$, описанным в разделе 4, поэтому отдельная проверка для него не требуется.

В качестве данных для эксперимента подготовлено 20 примеров, построенных на основе задач из PSPLIB [36]: исходные экземпляры незначительно упрощены и дополнены параметрами из раздела 3 — исполнителями U, крайними сроками δ_i и штрафами w_i, p_{iu}. Текстовые описания D формировались путем подстановки параметров каждого экземпляра PSPLIB — длительностей, зависимостей, требований к ресурсам, а также добавленных исполнителей U, крайних сроков δ_i и штрафов w_i, p_{iu} — в единый текстовый шаблон. Это обеспечивает единообразие и полноту описаний для всех 20 примеров. Оба способа тестировались на моделях GPT-OSS-120B[1], Qwen3-32B[2], Qwen3-235B-A22B-Thinking-2507[2] (ниже сокращенно Qwen3-235B) и DeepSeek V4 Flash[3]

Для сопоставления различных подходов использовались следующие метрики: 1) доля валидных планов; 2) значение целевой функции $J(\pi)$ и ее компоненты — 3) время завершения проекта, 4) суммарная взвешенная просрочка и 5) штраф качества назначения исполнителей; а также метрики качества формализации — F_1 по основным сущностям задачи и точность извлечения числовых параметров. Опишем, как вычисляется каждая из них.

Компоненты целевой функции $J(\pi)$, введенной в разделе 3 формулой (9), вычисляются следующим образом. Время завершения проекта:

$$\max_{i \in T} f_i.$$

Суммарная взвешенная просрочка:

$$\sum_{i \in T_\delta} w_i L_i = \sum_{i \in T_\delta} w_i \max(0, f_i - \delta_i),$$

[1]Agarwal S., Ahmad L., Ai J. и др. gpt-oss-120b & gpt-oss-20b Model Card. 2025. URL: `https://arxiv.org/abs/2508.10925` (дата обращения: 14.07.2026)

[2]Qwen Team. Qwen3 Technical Report. 2025. URL: `https://arxiv.org/abs/2505.09388`. (дата обращения: 14.07.2026)

[3]DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. 2026. URL: `https://arxiv.org/abs/2606.19348` (дата обращения: 14.07.2026)

где $T_\delta \subseteq T$ — подмножество работ с заданным крайним сроком; для работ без крайнего срока вклад в просрочку равен нулю. Штраф качества назначения исполнителей:

$$\sum_{i \in T} \sum_{u \in U} p_{iu} x_{iu}.$$

В проведенных экспериментах веса α, β, γ были независимыми параметрами для каждого из сэмплов. Данные параметры оставались одинаковыми для LLM-as-planner и LLM-as-formalizer в пределах одного примера. Меньшее значение каждой из приведенных метрик является более лучшим показателем.

Таблица 2. Усредненные метрики целевой функции

Модель	**Подход**	**Валидность** (доля)	$J(\pi)\downarrow$ (усл. ед.)	**Время**$\downarrow$ (дни)	**Просрочки**$\downarrow$ (усл. ед.)	**Делегирование**$\downarrow$ (усл. ед.)
GPT-OSS-120B	Планировщик	0.9	91.35	40.00	22.33	26.72
	Формализатор	0.95	53.14	36.89	4.68	27.53
Qwen3-32B	Планировщик	0.1	50.50	44.00	0.00	26.00
	Формализатор	1.0	49.14	35.65	3.40	26.75
Qwen3-235B	Планировщик	0.3	75.50	41.67	13.00	31.33
	Формализатор	1.0	52.00	36.30	4.45	27.20
DeepSeek-Flash	Планировщик	0.7	72.79	37.64	14.29	26.29
	Формализатор	1.0	52.00	36.30	4.45	27.20

Результаты, полученные по описанным метрикам, приведены в табл. 2. Значения $J(\pi)$, просрочки и делегирования выражены в условных единицах, поскольку целевая функция объединяет разнородные по своей природе слагаемые с весовыми коэффициентами α, β, γ. Значения метрик рассчитывались только по корректным планам, поскольку для планов, нарушающих ограничения задачи, величины метрик не имеют содержательной интерпретации.

6. Обсуждение результатов

Доля валидных планов у предложенного подхода составляет от 0.95 до 1.0 для всех четырех моделей, тогда как у способа LLM-as-planner она заметно ниже и колеблется от 0.1 (Qwen3-32B) до 0.9 (GPT-OSS-120B). Значение целевой функции $J(\pi)$ у предложенного подхода ниже для всех моделей: 53.14 против 91.35 у GPT-OSS-120B, 49.14 против 50.50 у Qwen3-32B, 52.00 против 75.50 у Qwen3-235B и 52.00 против 72.79 у DeepSeek V4 Flash. Время выполнения проекта у предложенного подхода в среднем быстрее (35.65–36.89 против 37.64–44.00), штраф за нарушение крайних сроков также в среднем ниже (3.40–4.68 против 13.00–22.33). Исключение — Qwen3-32B в способе планировщика, где просрочка равна 0.00; это значение получено всего по 10% построенных планов, оставшихся допустимыми, и поэтому не может считаться показательным. По качеству назначения исполнителей значения обоих подходов близки и находятся в диапазоне 26.00–31.33.

Помимо качества самого плана, отдельно оценивалось качество формализации по метрикам, описанным выше. Показатели F_1 по работам, зависимостям, исполнителям и ресурсам, а также точность извлечения длительностей и крайних сроков близки к 1.00 для всех четырех моделей. Отклонения от 1.00 наблюдаются только у F_1 по зависимостям для Qwen3-32B (0.95) и у F_1 по длительностям задач для отдельных сотрудников — для Qwen3-32B (0.97) и DeepSeek-Flash (0.94). Кроме того, можно заметить, что метрики у моделей

Qwen3-235B и DeepSeek V4 Flash при подходе «Формализатор» получились идентичными. Это объясняется тем, что обе модели дают на большинстве сэмплов практически безошибочное извлечение. В результате чего построенные по их PDDL-описаниям модели CP-SAT оказываются эквивалентны и приводят к одному и тому же оптимальному расписанию. Полученные результаты подтверждают возможность построения формальной модели задачи планирования при помощи LLM.

Предложенная формализация задачи планирования имеет несколько сильных сторон. Прежде всего, разделяются этап извлечения данных и этап планирования. Это важно для задач, где исходные данные представлены в естественном языке без заданной структуры. В таком случае LLM используется как средство структурирования описания проекта, а не как планировщик. Именно такой подход рассматривается в современных исследованиях [6, 17] как наиболее перспективный и эффективный для LLM-агентов.

Следующее преимущество состоит в том, что предложенная формализация учитывает более широкий набор ограничений, чем это обычно предполагается в базовых постановках проектного планирования. В модели одновременно присутствуют зависимости между задачами, ресурсы, крайние сроки и назначения исполнителей. При этом сохраняется баланс между классическими постановками RCPSP и прикладными задачами, возникающими в реальных проектах. Тем самым предложенный подход не заменяет классическое проектное планирование, а расширяет его за счет новых ограничений, целевой функции качества и использования LLM как формализатора.

Важной особенностью предложенного подхода является использование PDDL как формального средства представления задачи. Это дает возможность задать зависимости, временные параметры, числовые характеристики и критерии качества плана в единой модели, а затем применять к ней автоматические планировщики и средства проверки. Также благодаря данному языку появляется возможность автоматически проверять результат. Если вся информация переведена в PDDL, то корректность плана, домена и проблемы можно оценить с помощью автоматического валидатора [7]. Это упрощает проверку и дополнительную валидацию, а также минимизирует риск получения логически некорректного плана. Именно такой недостаток отмечается в работах [2, 16], где LLM используется как планировщик.

Вместе с тем, предложенная постановка имеет и ограничения. Одно из них связано с качеством исходных данных D. Если текстовое описание проекта неполно или противоречиво, то на этапе извлечения задач и их параметров может возникнуть ошибка. В этом случае планировщик будет работать корректно, однако на модели, которая уже не вполне соответствует реальному проекту. Работа [17] прямо показывает, что качество формализации снижается с ростом сложности и расплывчатости описания. Отметим также, что в проведенных экспериментах текстовые описания D генерировались по единому шаблону на основе PSPLIB, поэтому оценка устойчивости подхода к стилистически более разнородным формулировкам, характерным для реальных проектных текстов, выходит за рамки данной работы.

Следующее ограничение касается целевой функции. Ее выбор мотивирован с прикладной точки зрения, но он остается моделирующим допущением. Минимизация общего времени выполнения соответствует классической постановке RCPSP, а учет просрочек и назначений исполнителей отражает реальные требования управления проектами. Однако веса α, β и γ требуют калибровки в зависимости от приоритетов. Без такой калибровки трудно

гарантировать, что компромисс между общим временем выполнения, крайними сроками и распределением исполнителей соответствует приоритетам команды. Отметим также, что в проведенных экспериментах веса задавались случайным образом независимо для каждого примера. Попарное сравнение между подходами по отдельным метрикам остается корректным, однако усредненное по выборке значение $J(\pi)$ отражает эффект только по случайно распределенным весам.

Отдельного обсуждения заслуживает третье слагаемое в целевой функции. Его достоинство состоит в том, что оно позволяет учитывать различия между исполнителями. Соответствие навыков исполнителя требованиям задачи, а также качественное распределение задач между работниками повышает эффективность и результативность работы, что подтверждается литературой [22, 23, 29]. Однако в модели величина p_{iu} задается как стоимость или штраф назначения. Это удобно с точки зрения вычислений, но данная величина неизбежно агрегирует в одном параметре разные эффекты: квалификацию, опыт, производительность, предпочтительность назначения и, возможно, организационные ограничения.

Описанные недостатки не связаны с внутренней несостоятельностью постановки задачи. При этом выбранная схема опирается на формальную основу RCPSP и использует возможности PDDL 2.1 для задания временных действий, числовых параметров и метрик плана, а при необходимости — возможности PDDL 3 для задания предпочтений и мягких ограничений. Вместе с этим формализация соответствует современному направлению LLM-as-formalizer, которое рассматривается как более надежное, чем LLM-планировщик. В совокупности это позволяет считать предложенную формализацию достаточно обоснованной и масштабируемой для дальнейшего развития.

Заключение

В рамках данной статьи была предложена формализация поставленной задачи проектного планирования. На основе неструктурированных текстовых данных D выделяются задачи проекта, зависимости между ними, исполнители, ресурсы, крайние сроки и другие параметры. Большая языковая модель в данной формализации определяется как отображение $\Phi_\theta : D \to M$, где $M = \langle Dom, Prob \rangle$ — представление задачи на языке PDDL. Далее задача сводится к построению плана $\pi \in \Pi(M)$, минимизирующего заданную целевую функцию качества. Данная постановка позволяет связать неструктурированную информацию о проекте с автоматическими планировщиками посредством перевода в PDDL с помощью LLM.

Предложенная целевая функция учитывает не только время завершения проекта, но и соблюдение крайних сроков задач, а также качество распределения исполнителей. Тем самым постановка RCPSP сохраняется, но расширяется в сторону мультикритериальной постановки. Использование PDDL обосновано тем, что этот язык поддерживает все необходимое для задачи: временные действия, числовые переменные и метрики плана.

Проведенное обсуждение показывает, что введенная формализация имеет как достоинства, так и ограничения. Ее сильные стороны состоят в разделении этапов формализации и планирования, в поддержке сложных ограничений, в возможности формальной проверки результата. Ограничения связаны прежде всего с качеством исходных текстовых данных и трудностью точного восстановления структуры проекта. Однако эти ограничения не свидетельствуют о слабости самой постановки, напротив, они указывают на естественные направления ее дальнейшего развития.

Проведенные эксперименты показали, что в среднем доля корректных планов у предложенного подхода составляет от 0.95 до 1.0 для всех четырех моделей, тогда как у LLM-as-planner эта доля колеблется от 0.1 до 0.9. Время завершения проекта при этом снижается незначительно, тогда как суммарная просрочка относительно крайних сроков уменьшается заметно сильнее. На более сложных и объемных задачах разница в качестве может стать более заметной.

Таким образом, предложенная формализация может рассматриваться как основа для дальнейших исследований применения LLM к задаче оптимизации проектного планирования. Она сохраняет связь с классической постановкой RCPSP, использует возможности PDDL и в то же время применяет LLM как средство формализации неструктурированных данных. Перспективными направлениями дальнейшей работы являются расширение постановки в сторону неопределенности и динамического планирования.

Статья выполнена в рамках научно-исследовательской работы по государственному заданию СПб ФИЦ РАН Mol_Lab (молодежная_лаб) No FFZF-2024-0003.

Литература

1. Wang L., Ma C., Feng X., *et al.* A survey on large language model based autonomous agents // Frontiers of Computer Science. 2024. Vol. 18, no. 6. P. 186345. DOI: 10.1007/s11704-024-40231-1.
2. Valmeekam K., Marquez M., Olmo A., *et al.* PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change // Advances in Neural Information Processing Systems. Vol. 36. 2023. P. 38975–38987. DOI: 10.52202/075280-1693.
3. McDermott D., Ghallab M., Howe A., *et al.* PDDL – The Planning Domain Definition Language: Technical Report / Yale Center for Computational Vision; Control. 1998.. URL: https://ipc08.icaps-conference.org/deterministic/data/mcdermott-et-al-tr-1998.pdf.
4. Helmert M. The Fast Downward Planning System // Journal of Artificial Intelligence Research. 2006. Vol. 26. P. 191–246. DOI: 10.1613/jair.1705.
5. Richter S., Westphal M. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks // Journal of Artificial Intelligence Research. 2010. Vol. 39. P. 127–177. DOI: 10.1613/jair.2972.
6. Tantakoun M., Muise C., Zhu X. LLMs as Planning Formalizers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models // Findings of the Association for Computational Linguistics: ACL 2025 / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 25167–25188. DOI: 10.18653/v1/2025.findings-acl.1291.
7. Howey R., Long D., Fox M. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL // 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 15–17 November 2004, Boca Raton, FL, USA. IEEE Computer Society, 2004. P. 294–301. DOI: 10.1109/ICTAI.2004.120.

8. Atkinson R. Project Management: Cost, Time and Quality, Two Best Guesses and a Phenomenon, Its Time to Accept Other Success Criteria // International Journal of Project Management. 1999. Vol. 17, no. 6. P. 337–342. DOI: 10.1016/S0263-7863(98)00069-6.

9. Brucker P., Drexl A., Moehring R., *et al.* Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods // European Journal of Operational Research. 1999. Vol. 112, no. 1. P. 3–41. DOI: 10.1016/S0377-2217(98)00204-5.

10. Herroelen W., Leus R. Project Scheduling Under Uncertainty: Survey and Research Potentials // European Journal of Operational Research. 2005. Vol. 165, no. 2. P. 289–306. DOI: 10.1016/j.ejor.2004.04.002.

11. Bahroun Z., As'ad R., Tanash M., Athamneh R. The Multi-Skilled Resource-Constrained Project Scheduling Problem: A Systematic Review and an Exploration of Future Landscapes // Management Systems in Production Engineering. 2024. Vol. 32, no. 1. P. 108–132. DOI: 10.2478/mspe-2024-0012.

12. Yehudai A., Eden L., Li A., *et al.* A Survey on Evaluation of LLM-based Agents // Findings of the Association for Computational Linguistics: ACL 2026 / ed. by M. Liakata, V.P. Moreira, J. Zhang, D. Jurgens. San Diego, California, United States: Association for Computational Linguistics, 2026. P. 26690–26714. DOI: 10.18653/v1/2026.findings-acl.1330.

13. Xie J., Zhang K., Chen J., *et al.* TravelPlanner: A Benchmark for Real-World Planning with Language Agents // Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 21–27 Jul 2024. P. 54590–54613. Proceedings of Machine Learning Research.

14. Xi Z., Chen W., Guo X., *et al.* The rise and potential of large language model based agents: a survey // Science China Information Sciences. 2025. Vol. 68, no. 2. P. 121101. DOI: 10.1007/s11432-024-4222-0.

15. Pallagani V., Roy K., Muppasani B., *et al.* On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS) // Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 34. 2024. P. 432–444. DOI: 10.1609/icaps.v34i1.31503.

16. Kambhampati S., Valmeekam K., Guan L., *et al.* Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks // Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 2024. P. 22895–22907. Proceedings of Machine Learning Research. DOI: 10.48550/arXiv.2402.01817.

17. Huang C., Zhang L. On the Limit of Language Models as Planning Formalizers // Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 4880–4904. DOI: 10.18653/v1/2025.acl-long.242.

18. Webb T., Mondal S.S., Momennejad I. A Brain-Inspired Agentic Architecture to Improve Planning with LLMs // Nature Communications. 2025. Vol. 16, no. 1. P. 8633. DOI: 10.1038/s41467-025-63804-5.

19. Corrêa A.B., Pereira A.G., Seipp J. Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code // Advances in Neural Information Processing Systems 38 (NeurIPS 2025). 2025. DOI: 10.48550/arXiv.2503.18809.

20. Fox M., Long D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains // Journal of Artificial Intelligence Research. 2003. Vol. 20. P. 61–124. DOI: 10.1613/jair.1129.

21. Gerevini A., Long D. Preferences and Soft Constraints in PDDL3 // ICAPS Workshop on Planning with Preferences and Soft Constraints. 2006. P. 46–53. URL: https://artificial-intelligence.unibs.it/gerevini/papers/WS-PREF-ICAPS06.pdf.

22. Kristof-Brown A.L., Zimmerman R.D., Johnson E.C. Consequences of Individuals' Fit at Work: A Meta-Analysis of Person-Job, Person-Organization, Person-Group, and Person-Supervisor Fit // Personnel Psychology. 2005. Vol. 58, no. 2. P. 281–342. DOI: 10.1111/j.1744-6570.2005.00672.x.

23. Ruiz-Torres A.J., Ablanedo-Rosas J.H., Mukhopadhyay S., Paletta G. Scheduling Workers: A Multi-Criteria Model Considering Their Satisfaction // Computers & Industrial Engineering. 2019. Vol. 128. P. 747–754. DOI: 10.1016/j.cie.2018.12.070.

24. McCarthy J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I // Communications of the ACM. 1960. Vol. 3, no. 4. P. 184–195. DOI: 10.1145/367177.367199.

25. Browning T.R., Yassine A.A. Resource-Constrained Multi-Project Scheduling: Priority Rule Performance Revisited // International Journal of Production Economics. 2010. Vol. 126, no. 2. P. 212–228. DOI: 10.1016/j.ijpe.2010.03.009.

26. Bold M., Goerigk M. A Compact Reformulation of the Two-Stage Robust Resource-Constrained Project Scheduling Problem // Computers & Operations Research. 2021. Vol. 130. P. 105232. DOI: 10.1016/j.cor.2021.105232.

27. Ranjbar M., Khalilzadeh M., Kianfar F., Etminani K. An Optimal Procedure for Minimizing Total Weighted Resource Tardiness Penalty Costs in the Resource-Constrained Project Scheduling Problem // Computers & Industrial Engineering. 2012. Vol. 62, no. 1. P. 264–270. DOI: 10.1016/j.cie.2011.09.013.

28. Khoshjahan Y., Najafi A.A., Afshar-Nadjafi B. Resource Constrained Project Scheduling Problem with Discounted Earliness-Tardiness Penalties: Mathematical Modeling and Solving Procedure // Computers & Industrial Engineering. 2013. Vol. 66, no. 2. P. 293–300. DOI: 10.1016/j.cie.2013.06.017.

29. Goetz N., Wald A. Employee Performance in Temporary Organizations: The Effects of Person-Environment Fit and Temporariness on Task Performance and Innovative Performance // European Management Review. 2020. Vol. 18, no. 2. P. 25–41. DOI: 10.1111/emre.12438.

30. Sackett P.R. Reflections on a Career Studying Individual Differences in the Workplace // Annual Review of Organizational Psychology and Organizational Behavior. 2021. Vol. 8, no. 1. P. 1–18. DOI: 10.1146/annurev-orgpsych-012420-061939.

31. Chowdhury S., Schulz E., Milner M., Voort D.V.D. Core Employee Based Human Capital and Revenue Productivity in Small Firms: An Empirical Investigation // Journal of Business Research. 2014. Vol. 67, no. 11. P. 2473–2479. DOI: 10.1016/j.jbusres.2014.03.007.

32. Hart P., Nilsson N., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths // IEEE Transactions on Systems Science and Cybernetics. 1968. Vol. 4, no. 2. P. 100–107. DOI: 10.1109/TSSC.1968.300136.

33. Richter S., Helmert M. Preferred Operators and Deferred Evaluation in Satisficing Planning // Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009), Thessaloniki, Greece, September 19–23, 2009. AAAI Press, 2009. P. 273–280. DOI: 10.1609/icaps.v19i1.13345.

34. Perron L., Didier F., Gay S. The CP-SAT-LP Solver // 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. P. 3:1–3:2. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.CP.2023.3.

35. Рафиков Т.Р., Попцов А.В., Олисеенко В.Д., Абрамов М.В. formalizplanning: репозиторий с исходным кодом и данными вычислительных экспериментов. 2026. URL: https://github.com/dscspro/formalizplanning (дата обращения: 14.07.2026).

36. Kolisch R., Sprecher A. PSPLIB – A Project Scheduling Problem Library: OR Software – ORSEP Operations Research Software Exchange Program // European Journal of Operational Research. 1997. Vol. 96, no. 1. P. 205–216. DOI: 10.1016/S0377-2217(96)00170-1.

Рафиков Тимур Ришатович, стажер-исследователь, лаборатория прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

Попцов Александр Владимирович, стажер-исследователь, лаборатория прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

Олисеенко Валерий Дмитриевич, научный сотрудник, лаборатория прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

Абрамов Максим Викторович, к.т.н., доцент, руководитель лаборатории прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

DOI: 10.14529/cmse260206

FORMALIZATION OF THE PROJECT PLANNING PROBLEM WHEN WORKING WITH AN UNSTRUCTURED DESCRIPTION OF THE GOAL

© 2026 T.R. Rafikov, A.V. Poptsov, V.D. Oliseenko, M.V. Abramov

St. Petersburg Federal Research Center of the Russian Academy of Sciences

(Line 14 V.O. 39, St. Petersburg, 199178 Russia)

E-mail: trr@dscs.pro, avp@dscs.pro, vdo@dscs.pro, mva@dscs.pro

Received: 08.05.2026

Large language models demonstrate high efficiency in natural language processing tasks. However, their application to planning as planners under complex dependencies remains limited. One promising direction is to use large language models as a tool for constructing formal planning models. This paper considers the project planning problem in the case where the initial project information is given in an unstructured textual form. A formalization is proposed in which a large language model is used to extract this information and construct a formal model in PDDL. This approach makes it possible to connect textual descriptions of tasks, dependencies, deadlines, and resources with formal tools for automated planning. As a result, the project planning problem is transformed into a form suitable for verification, validation, and subsequent solution by algorithmic methods. In addition, this study extends the mathematical apparatus of the classical resource-constrained project scheduling problem. In addition to the total project completion time, penalties for missed deadlines and the quality of performer assignment are taken into account. Thus, the classical formulation is supplemented with criteria that reflect more complex and practically significant project management requirements. These changes make it possible to move from a simplified single-criterion model to a more realistic formulation. Translation into PDDL makes it possible to use automated tools for validation and planning of the stated problem. The computational experiments demonstrated that the proposed approach increased the proportion of valid plans from 0.1–0.9 to 0.95–1.0 for all four tested models compared with the direct use of an LLM as a planner.

Keywords: planning, large language models, formalization, LLM agent.

FOR CITATION

Rafikov T.R., Poptsov A.V., Oliseenko V.D., Abramov M.V. Formalization of the Project Planning Problem When Working with an Unstructured Description of the Goal. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2026. Vol. 15, no. 2. P. 104–126. (in Russian) DOI: 10.14529/cmse260206.

References

1. Wang L., Ma C., Feng X., *et al.* A survey on large language model based autonomous agents. Frontiers of Computer Science. 2024. Vol. 18, no. 6. P. 186345. DOI: 10.1007/s11704-024-40231-1.
2. Valmeekam K., Marquez M., Olmo A., *et al.* PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. Advances in Neural Information Processing Systems. Vol. 36. 2023. P. 38975–38987. DOI: 10.52202/075280-1693.

3. McDermott D., Ghallab M., Howe A., *et al.* PDDL – The Planning Domain Definition Language: Technical Report / Yale Center for Computational Vision; Control. 1998.. URL: https://ipc08.icaps-conference.org/deterministic/data/mcdermott-et-al-tr-1998.pdf.

4. Helmert M. The Fast Downward Planning System. Journal of Artificial Intelligence Research. 2006. Vol. 26. P. 191–246. DOI: 10.1613/jair.1705.

5. Richter S., Westphal M. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. Journal of Artificial Intelligence Research. 2010. Vol. 39. P. 127–177. DOI: 10.1613/jair.2972.

6. Tantakoun M., Muise C., Zhu X. LLMs as Planning Formalizers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models. Findings of the Association for Computational Linguistics: ACL 2025 / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 25167–25188. DOI: 10.18653/v1/2025.findings-acl.1291.

7. Howey R., Long D., Fox M. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 15–17 November 2004, Boca Raton, FL, USA. IEEE Computer Society, 2004. P. 294–301. DOI: 10.1109/ICTAI.2004.120.

8. Atkinson R. Project Management: Cost, Time and Quality, Two Best Guesses and a Phenomenon, Its Time to Accept Other Success Criteria. International Journal of Project Management. 1999. Vol. 17, no. 6. P. 337–342. DOI: 10.1016/S0263-7863(98)00069-6.

9. Brucker P., Drexl A., Moehring R., *et al.* Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. European Journal of Operational Research. 1999. Vol. 112, no. 1. P. 3–41. DOI: 10.1016/S0377-2217(98)00204-5.

10. Herroelen W., Leus R. Project Scheduling Under Uncertainty: Survey and Research Potentials. European Journal of Operational Research. 2005. Vol. 165, no. 2. P. 289–306. DOI: 10.1016/j.ejor.2004.04.002.

11. Bahroun Z., As'ad R., Tanash M., Athamneh R. The Multi-Skilled Resource-Constrained Project Scheduling Problem: A Systematic Review and an Exploration of Future Landscapes. Management Systems in Production Engineering. 2024. Vol. 32, no. 1. P. 108–132. DOI: 10.2478/mspe-2024-0012.

12. Yehudai A., Eden L., Li A., *et al.* A Survey on Evaluation of LLM-based Agents. Findings of the Association for Computational Linguistics: ACL 2026 / ed. by M. Liakata, V.P. Moreira, J. Zhang, D. Jurgens. San Diego, California, United States: Association for Computational Linguistics, 2026. P. 26690–26714. DOI: 10.18653/v1/2026.findings-acl.1330.

13. Xie J., Zhang K., Chen J., *et al.* TravelPlanner: A Benchmark for Real-World Planning with Language Agents. Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 21–27 Jul 2024. P. 54590–54613. Proceedings of Machine Learning Research.

14. Xi Z., Chen W., Guo X., *et al.* The rise and potential of large language model based agents: a survey. Science China Information Sciences. 2025. Vol. 68, no. 2. P. 121101. DOI: 10.1007/s11432-024-4222-0.

15. Pallagani V., Roy K., Muppasani B., *et al.* On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 34. 2024. P. 432–444. DOI: 10.1609/icaps.v34i1.31503.

16. Kambhampati S., Valmeekam K., Guan L., *et al.* Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 2024. P. 22895–22907. Proceedings of Machine Learning Research. DOI: 10.48550/arXiv.2402.01817.

17. Huang C., Zhang L. On the Limit of Language Models as Planning Formalizers. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 4880–4904. DOI: 10.18653/v1/2025.acl-long.242.

18. Webb T., Mondal S.S., Momennejad I. A Brain-Inspired Agentic Architecture to Improve Planning with LLMs. Nature Communications. 2025. Vol. 16, no. 1. P. 8633. DOI: 10.1038/s41467-025-63804-5.

19. Corrêa A.B., Pereira A.G., Seipp J. Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code. Advances in Neural Information Processing Systems 38 (NeurIPS 2025). 2025. DOI: 10.48550/arXiv.2503.18809.

20. Fox M., Long D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research. 2003. Vol. 20. P. 61–124. DOI: 10.1613/jair.1129.

21. Gerevini A., Long D. Preferences and Soft Constraints in PDDL3. ICAPS Workshop on Planning with Preferences and Soft Constraints. 2006. P. 46–53. URL: https://artificial-intelligence.unibs.it/gerevini/papers/WS-PREF-ICAPS06.pdf.

22. Kristof-Brown A.L., Zimmerman R.D., Johnson E.C. Consequences of Individuals' Fit at Work: A Meta-Analysis of Person-Job, Person-Organization, Person-Group, and Person-Supervisor Fit. Personnel Psychology. 2005. Vol. 58, no. 2. P. 281–342. DOI: 10.1111/j.1744-6570.2005.00672.x.

23. Ruiz-Torres A.J., Ablanedo-Rosas J.H., Mukhopadhyay S., Paletta G. Scheduling Workers: A Multi-Criteria Model Considering Their Satisfaction. Computers & Industrial Engineering. 2019. Vol. 128. P. 747–754. DOI: 10.1016/j.cie.2018.12.070.

24. McCarthy J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. Communications of the ACM. 1960. Vol. 3, no. 4. P. 184–195. DOI: 10.1145/367177.367199.

25. Browning T.R., Yassine A.A. Resource-Constrained Multi-Project Scheduling: Priority Rule Performance Revisited. International Journal of Production Economics. 2010. Vol. 126, no. 2. P. 212–228. DOI: 10.1016/j.ijpe.2010.03.009.

26. Bold M., Goerigk M. A Compact Reformulation of the Two-Stage Robust Resource-Constrained Project Scheduling Problem. Computers & Operations Research. 2021. Vol. 130. P. 105232. DOI: 10.1016/j.cor.2021.105232.

27. Ranjbar M., Khalilzadeh M., Kianfar F., Etminani K. An Optimal Procedure for Minimizing Total Weighted Resource Tardiness Penalty Costs in the Resource-Constrained Project Scheduling Problem. Computers & Industrial Engineering. 2012. Vol. 62, no. 1. P. 264–270. DOI: 10.1016/j.cie.2011.09.013.

28. Khoshjahan Y., Najafi A.A., Afshar-Nadjafi B. Resource Constrained Project Scheduling Problem with Discounted Earliness-Tardiness Penalties: Mathematical Modeling and Solving Procedure. Computers & Industrial Engineering. 2013. Vol. 66, no. 2. P. 293–300. DOI: 10.1016/j.cie.2013.06.017.

29. Goetz N., Wald A. Employee Performance in Temporary Organizations: The Effects of Person-Environment Fit and Temporariness on Task Performance and Innovative Performance. European Management Review. 2020. Vol. 18, no. 2. P. 25–41. DOI: 10.1111/emre.12438.

30. Sackett P.R. Reflections on a Career Studying Individual Differences in the Workplace. Annual Review of Organizational Psychology and Organizational Behavior. 2021. Vol. 8, no. 1. P. 1–18. DOI: 10.1146/annurev-orgpsych-012420-061939.

31. Chowdhury S., Schulz E., Milner M., Voort D.V.D. Core Employee Based Human Capital and Revenue Productivity in Small Firms: An Empirical Investigation. Journal of Business Research. 2014. Vol. 67, no. 11. P. 2473–2479. DOI: 10.1016/j.jbusres.2014.03.007.

32. Hart P., Nilsson N., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics. 1968. Vol. 4, no. 2. P. 100–107. DOI: 10.1109/TSSC.1968.300136.

33. Richter S., Helmert M. Preferred Operators and Deferred Evaluation in Satisficing Planning. Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009), Thessaloniki, Greece, September 19–23, 2009. AAAI Press, 2009. P. 273–280. DOI: 10.1609/icaps.v19i1.13345.

34. Perron L., Didier F., Gay S. The CP-SAT-LP Solver. 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. P. 3:1–3:2. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.CP.2023.3.

35. Rafikov T.R., Poptsov A.V., Oliseenko V.D., Abramov M.V. formalizplanning: repository containing source code and computational experiment data. 2026. URL: https://github.com/dscspro/formalizplanning (accessed: 14.07.2026).

36. Kolisch R., Sprecher A. PSPLIB – A Project Scheduling Problem Library: OR Software – ORSEP Operations Research Software Exchange Program. European Journal of Operational Research. 1997. Vol. 96, no. 1. P. 205–216. DOI: 10.1016/S0377-2217(96)00170-1.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 1.2.3 – Теоретическая информатика, кибернетика (физико-математические науки), 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции: 454080, г. Челябинск, ул. С. Кривой, 79, Издательский центр ЮУрГУ, каб. 2.

Адрес издателя: 454080, г. Челябинск, проспект Ленина, 76.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `https://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, НОЦ ИИКТ,
   зам. главного редактора Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`.
4. **Плата с авторов за публикацию рукописей не взимается, гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
2026 Том 15, № 2

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 22.08.2026. Дата выхода в свет 25.09.2026. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 14,88. Тираж 500 экз. Заказ 125/165. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.