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

2023, том 12, № 2

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2023, volume 12, no. 2

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2023
Т. 12, № 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"

2023
Vol. 12, 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 519.852 **DOI: 10.14529/cmse230201**

О НОВОЙ ВЕРСИИ АПЕКС-МЕТОДА ДЛЯ РЕШЕНИЯ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ*

© 2023 Л.Б. Соколинский, И.М. Соколинская

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: leonid.sokolinsky@susu.ru, irina.sokolinskaya@susu.ru

Поступила в редакцию: 07.04.2023

В статье представлена новая версия масштабируемого итерационного метода линейного программирования, получившего название «апекс-метод». Ключевой особенностью этого метода является построение пути, близкого к оптимальному, на поверхности допустимой области от определенной начальной точки до точного решения задачи линейного программирования. Оптимальный путь — это путь движения по поверхности многогранника в направлении максимального увеличения или уменьшения значения целевой функции в зависимости от того, ее максимум или минимум необходимо найти. Апекс-метод основан на схеме предиктор-корректор и состоит из двух стадий: Quest (предиктор) и Target (корректор). На стадии Quest вычисляется грубое начальное приближение задачи линейного программирования. Основываясь на этом начальном приближении, на стадии Target вычисляется решение задачи линейного программирования с заданной точностью. Основная операция, используемая в апекс-методе, — это операция, которая вычисляет псевдопроекцию, являющуюся обобщением метрической проекции на выпуклое замкнутое множество. Псевдопроекция используется как на стадии Quest, так и на стадии Target. Представлен параллельный алгоритм, использующий фейеровское отображение для вычисления псевдопроекции. Получена аналитическая оценка ресурса параллелизма для этого алгоритма. Также приведен алгоритм, реализующий стадию Target, и доказана его сходимость. Описаны вычислительные эксперименты на кластерной вычислительной системе по применению апекс-метода для решения различных задач линейного программирования.

Ключевые слова: линейное программирование, апекс-метод, итерационный метод, метод проекционного типа, фейеровское отображение, параллельный алгоритм, оценка масштабируемости.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Соколинский Л.Б., Соколинская И.М. О новой версии апекс-метода для решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 5–46. DOI: 10.14529/cmse230201.

Введение

Данная работа является дальнейшим развитием апекс-метода, предложенного нами в статье [1] для решения задач линейного программирования (ЛП). Актуальность этой темы основывается на следующих факторах. Одним из важных классов приложений ЛП являются нестационарные задачи, связанные с оптимизацией нестационарных процессов [2]. В нестационарных задачах ЛП целевая функция и/или ограничения изменяются в течение вычислительного процесса. В качестве примеров можно привести следующие нестационарные задачи: поддержка принятия решений в высокочастотной торговле [3, 4], задачи гидрогазодинамики [5], оптимальное управление технологическими процессами [6–8], транспортные задачи [9–11], оперативное планирование [12, 13].

Один из стандартных подходов к решению нестационарных задач оптимизации состоит в том, чтобы рассматривать каждое изменение как появление новой задачи оптимизации,

*Работа рекомендована к публикации программным комитетом международной научной конференции «Суперкомпьютерные дни в России 2023».

которую необходимо решать с нуля [2]. Однако такой подход часто непрактичен, поскольку решение проблемы с нуля без повторного использования информации из прошлого может занять слишком много времени. Таким образом, желательно иметь алгоритм оптимизации, способный непрерывно адаптировать решение к изменяющейся среде, повторно используя информацию, полученную в прошлом. Этот подход применим для процессов реального времени, если алгоритм достаточно быстро отслеживает траекторию движения оптимальной точки. В случае больших задач ЛП последнее требует разработки масштабируемых методов и параллельных алгоритмов ЛП.

До сих пор одним из наиболее распространенных способов решения задач ЛП был класс алгоритмов, предложенных и разработанных Данцигом на основе симплекс-метода [14]. Было установлено, что симплексный метод эффективен для решения большого класса задач ЛП. В частности, симплексный метод легко использует преимущества любой гиперразреженности в задачах ЛП [15]. Однако симплекс-метод обладает некоторыми фундаментальными особенностями, которые ограничивают его использование для решения больших задач ЛП. Во-первых, в определенных случаях симплексный метод должен выполнять итерации по всем вершинам симплекса, что соответствует экспоненциальной временной сложности [16–18]. Во-вторых, в большинстве случаев симплекс-метод успешно решает задачи ЛП, содержащие до 50 000 переменных. Однако при решении задач бóльших размерностей часто наблюдается потеря точности ЛП [19], которая не может быть компенсирована даже применением таких мощных вычислительных процедур, как «аффинное масштабирование» или «итеративное уточнение» [20]. В-третьих, в общем случае последовательный характер симплексного метода затрудняет распараллеливание в многопроцессорных системах с распределенной памятью [21]. Были предприняты многочисленные попытки создать масштабируемую параллельную реализацию симплексного метода, но все они оказались безуспешными [22]. Во всех случаях граница масштабируемости составляла от 16 до 32 процессорных узлов (см., например, [23]).

Хачиян доказал [24], используя вариант метода эллипсоидов (предложенный в 1970-х годах Шором [25], Юдиным и Немировским [26]), что задачи ЛП могут быть решены за полиномиальное время. Однако попытки применить этот подход на практике оказались безуспешными, поскольку в подавляющем большинстве случаев метод эллипсоида демонстрировал гораздо худшую эффективность по сравнению с симплекс-методом. Позже Кармаркар [27] показал, что алгоритм внутренних точек, предложенный Дикиным [28], имеет полиномиальную временную сложность и применим на практике. Этот алгоритм породил целую область современных методов внутренних точек [29, 30], которые способны решать большие задачи ЛП с миллионами переменных и миллионами уравнений [31–35]. Более того, эти методы являются самокорректирующимися, а следовательно, обеспечивают высокую точность вычислений. Общим недостатком методов внутренних точек является необходимость найти некоторую допустимую точку, удовлетворяющую всем ограничениям задачи ЛП, перед началом вычислений. Нахождение такой внутренней точки может быть сведено к решению дополнительной задачи ЛП [36]. Еще одним методом нахождения внутренней точки является метод псевдопроекции [37], который использует фейеровские отображения [38]. Другим существенным недостатком метода внутренних точек является его плохая масштабируемость в многопроцессорных системах с распределенной памятью. Существует несколько успешных параллельных реализаций метода внутренних точек для частных случаев (см., например, [39]), но, в общем случае, эффективная параллельная реализация на

многопроцессорных системах для этого метода не может быть построена. В соответствии с этим разработка и исследование новых подходов к решению многомерных нестационарных задач ЛП в режиме реального времени является актуальным направлением.

Одним из наиболее перспективных подходов к решению сложных задач в режиме реального времени является использование нейросетевых моделей [40]. Искусственные нейронные сети — это мощный универсальный инструмент, который применим для решения задач практически во всех областях. Самой популярной моделью нейронной сети является нейронная сеть прямого распространения. Обучение и использование таких сетей могут быть очень эффективно реализованы на графических процессорах [41]. Важным свойством нейронной сети прямого распространения является то, что время решения задачи не зависит от ее параметров. Это свойство необходимо для работы в режиме реального времени. Новаторской работой по использованию нейронных сетей для решения задач ЛП является статья Танка и Хопфилда [42]. В этой статье описывается двухслойная рекуррентная нейронная сеть. Число нейронов в первом слое определяется количеством переменных задачи ЛП. Количество нейронов во втором слое совпадает с количеством ограничений задачи ЛП. Первый и второй слои являются полносвязными. Веса и смещения однозначно определяются коэффициентами и правыми частями линейных неравенств, определяющих ограничения, и коэффициентами линейной целевой функции. Таким образом, эта сеть не требует обучения. Состояние нейронной сети описывается дифференциальным уравнением $\dot{x}(t) = \nabla E(x(t))$, где $E(x(t))$ — энергетическая функция специального типа. Первоначально на вход нейронной сети подается произвольная точка допустимой области. Затем сигнал второго слоя рекурсивно подается на первый слой. В итоге процесс приходит в стабильное состояние, в котором выходной сигнал перестает изменяться. Такое состояние соответствует минимуму энергетической функции, а выходной сигнал является решением задачи ЛП. Подход Танка и Хопфилда был развит и усовершенствован в многочисленных работах (см., например, [43–47]). Основным недостатком этого подхода является непредсказуемое количество рабочих циклов нейронной сети. Следовательно, рекуррентная сеть, основанная на энергетической функции, не может использоваться для решения больших задач ЛП в режиме реального времени.

В недавней статье [48] была предложена n-мерная математическая модель визуализации задач ЛП. Эта модель позволяет использовать нейронные сети прямого распространения, включая сверточные сети [49], для решения многомерных задач ЛП, допустимой областью которых является замкнутое ограниченное непустое множество. Однако в научной литературе практически отсутствуют работы, посвященные использованию сверточных нейронных сетей для решения задач ЛП [50]. Причина в том, что сверточные нейронные сети ориентированы на обработку изображений, но до настоящего времени отсутствовали методы построения обучающих наборов данных, основанные на визуальном представлении многомерных задач ЛП.

В данной статье описывается новый масштабируемый итерационный метод для решения многомерных задач ЛП, получивший название «апекс-метод». Апекс-метод позволяет генерировать обучающие наборы данных для разработки нейронных сетей прямого распространения, способных находить решение многомерной задачи ЛП на основе ее визуального представления. Апекс-метод основан на схеме предиктор/корректор. Предиктор вычисляет точку, принадлежащую допустимой области задачи ЛП. Корректор вычисляет последовательность точек, сходящуюся к точному решению задачи ЛП. Статья организована

следующим образом. В разделе 1 представлен обзор итерационных методов и алгоритмов проекционного типа, ориентированных на решение выпуклых неравенств и задач ЛП. Раздел 2 содержит теоретический базис, используемый в описании апекс-метода. В разделе 3 представлено формализованное описание апекс-метода. Раздел 3.1 посвящен разработке алгоритма построения псевдопроекции и аналитическому исследованию масштабируемости его параллельной версии. В разделе 3.2 описывается стадия Quest. Раздел 3.3 содержит описание стадии Target. В разделе 4 представлены информация о программной реализации апекс-метода и результаты вычислительных экспериментов. В разделе 5 обсуждаются научная и практическая значимость полученных результатов, преимущества и недостатки апекс-метода, и способы его использования. В заключении суммируются представленные в статье результаты и намечаются направления дальнейших исследований. В конце статьи приведены основные Обозначения, используемые при описании апекс-метода.

1. Обзор работ по итерационным методам проекционного типа

В этом разделе представлен обзор работ, посвященных итерационным методам проекционного типа, используемым для решения задач совместности выпуклых неравенств и задач ЛП. Задача совместности (допустимости) выпуклых неравенств заключается в нахождении некоторого решения системы выпуклых неравенств. Эта задача возникает в многочисленных приложениях, таких как статистика, параметрическое оценивание, распознавание образов, восстановление изображений, томография и других [51]. В случае линейных неравенств задача совместности может быть сформулирована следующим образом. Имеется система линейных неравенств в матричном виде:

$$Ax \leqslant b, \tag{1}$$

где $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$. Во избежание вырожденности будем предполагать, что $m > 1$. Задача линейной совместности, заключается в нахождении точки $\tilde{x} \in \mathbb{R}^n$, удовлетворяющей матричному неравенству (1). Везде далее мы будем предполагать, что такая точка существует, то есть система (1) является совместной.

Методы проекционного типа основаны на следующей геометрической интерпретации задачи линейной совместности. Обозначим через $a_i \in \mathbb{R}^n$ вектор, состоящий из элементов i-той строки матрицы A. Тогда матричное неравенство $Ax \leqslant b$ может быть представлено в виде системы неравенств

$$\langle a_i, x \rangle \leqslant b_i, i = 1, \ldots, m. \tag{2}$$

Здесь $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение двух векторов. Везде далее мы предполагаем, что

$$a_i \neq \mathbf{0} \tag{3}$$

для всех $i = 1, \ldots, m$. Каждое неравенство $\langle a_i, x \rangle \leqslant b_i$ определяет замкнутое полупространство

$$\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x \rangle \leqslant b_i\} \tag{4}$$

и ограничивающую его гиперплоскость

$$H_i = \{x \in \mathbb{R}^n | \langle a_i, x \rangle = b_i\}. \tag{5}$$

Для любой точки $x \in \mathbb{R}^n$ ортогональная проекция $\pi(x)$ на гиперплоскость H_i может быть вычислена по формуле

$$\pi_i(x) = x - \frac{\langle a_i, x\rangle - b_i}{\|a_i\|^2} a_i. \tag{6}$$

Здесь и далее $\|\cdot\|$ обозначает евклидову норму. Определим допустимый многогранник

$$M = \bigcap_{i=1}^{m} \hat{H}_i, \tag{7}$$

представляющий множество допустимых точек системы (1). Заметим, что M в этом случае будет замкнутым выпуклым множеством. Мы будем предполагать, что $M \neq \emptyset$, то есть система (1) имеет решение. С геометрической точки зрения задача линейной совместности состоит в нахождении точки $\tilde{x} \in M$.

Первыми работами, посвященными задаче линейной совместности, были работы Качмарца (Kaczmarz) и Чиммино (Cimmino). Качмарц в работе [52] (английский перевод [53]) предложил следующий метод последовательных проекций для решения совместной системы линейных неравенств

$$\langle a_i, x\rangle = b_i, i = 1, \ldots, m. \tag{8}$$

Начиная с произвольной точки $x^{(0,m)} \in \mathbb{R}$, этот метод строит последовательность групп точек

$$x^{(k,1)} = \pi_1\left(x^{(k-1,m)}\right), x^{(k,2)} = \pi_2\left(x^{(k,1)}\right), \ldots, x^{(k,m)} = \pi_m\left(x^{(k,m-1)}\right) \tag{9}$$

для $k = 1, 2, 3, \ldots$ Здесь π_i $(i = 1, \ldots, m)$ обозначает ортогональную проекцию на гиперплоскость H_i, вычисляемую по формуле (6). Качмарц показал, что последовательность (9) сходится к решению системы (8). С геометрической точки зрения метод Качмарца может быть описан следующим образом. На первом шаге строится ортогональная проекция начальной точки $x^{(0,m)}$ на гиперплоскость H_1. Полученная точка $x^{(1,1)}$, в свою очередь, проецируется на H_2, что дает нам точку $x^{(1,2)}$. Точка $x^{(1,2)}$ проецируется на H_3, что дает нам точку $x^{(1,3)}$, и так далее. Последней точкой в первой группе будет точка $x^{(1,m)}$, получающаяся в результате ортогональной проекция точки $x^{(1,m-1)}$ на гиперплоскость H_m. Вторая группа точек строится аналогичным образом, используя в качестве начальной точку $x^{(1,m)}$. Процесс продолжается для $k = 3, 4, 5 \ldots$

Чиммино в [54] (английское описание [55]) предложил метод одновременных проекций для решения задачи линейной совместности. В своем методе вместо ортогональных проекций Чиммино использует ортогональные отражения, вычисляемые по формуле

$$\rho_i(x) = x - 2\frac{\langle a_i, x\rangle - b_i}{\|a_i\|^2} a_i. \tag{10}$$

Ортогональное отражение строит точку $\rho_i(x)$, симметричную точке x относительно гиперплоскости H_i. Для текущего приближения $x^{(k)}$ метод Чиммино вычисляет ортогональные отражения сразу относительно всех гиперплоскостей H_i $(i = 1, \ldots, m)$ и затем использует выпуклую комбинацию полученных точек для формирования следующего приближения:

$$x^{(k+1)} = \sum_{i=1}^{m} w_i \rho_i\left(x^{(k)}\right), \tag{11}$$

где $w_i > 0$ $(i = 1, \ldots, m)$, $\sum\limits_{i=1}^{m} w_i = 1$. При $w_i = \frac{1}{m}$ $(i = 1, \ldots, m)$ формула (11) принимает вид

$$x^{(k+1)} = \frac{1}{m} \sum_{i=1}^{m} \rho_i \left(x^{(k)} \right). \tag{12}$$

Агмон (Agmon) [56], Моцкин (Motzkin), Шенберг (Schoenberg) [57] предложили релаксационный метод, являющийся обобщением проекционного метода Качмарца на случай линейных неравенств. Для решения системы (1) они используют следующее релаксационное отображение:

$$\pi_i^{\lambda}(x) = (1 - \lambda)x + \lambda \pi_i(x), \tag{13}$$

где $0 < \lambda < 2$. Очевидно, что $\pi_i^1(x) = \pi_i(x)$, то есть при $\lambda = 1$ релаксационное отображение превращается в ортогональную проекцию. Для вычисления следующего приближения релаксационный метод использует формулу

$$x^{(k+1)} = \pi_l^{\lambda} \left(x^{(k)} \right), \tag{14}$$

где

$$l = \arg\max_i \left\{ \left\| x^{(k)} - \pi_i \left(x^{(k)} \right) \right\| \middle| \, x^{(k)} \notin \hat{H}_i \right\}. \tag{15}$$

С неформальной точки зрения, следующее приближение $x^{(k+1)}$ получается в результате релаксационного отображения предыдущего приближения $x^{(k)}$ относительно самой дальней гиперплоскости H_l, ограничивающей полупространство $\hat{H}_l$, не содержащее точку $x^{(k)}$. Агмон в [56] показал, что последовательность $x^{(k)}$ сходится к граничной точке допустимого многогранника M.

Цензор (Censor) и Эльфвинг (Elfving) в [58] обобщили метод Чиммино на случай линейных неравенств. Они рассматривают ослабленную (relaxed) проекцию на полупространство, определяемую формулой

$$\hat{\pi}_i^{\lambda}(x) = (1 - \lambda)x - \lambda \frac{\max\{0, \langle a_i, x \rangle - b_i\}}{\|a_i\|^2} a_i, \tag{16}$$

и получают следующее итерационное уравнение

$$x^{(k+1)} = \sum_{i=1}^{m} w_i \hat{\pi}_i^{\lambda} \left(x^{(k)} \right). \tag{17}$$

Здесь $0 < \lambda < 2$, $w_i > 0$ $(i = 1, \ldots, m)$, $\sum\limits_{i=1}^{m} w_i = 1$. Де Пьеро (De Pierro) в [59] предложил схему доказательства сходимости этого метода, отличающуюся от схемы цензора и Эльфвинга. Подход де Пьеро также применим для случая, когда исходная система линейных неравенств несовместна. В этом случае при $\lambda = 1$ последовательность (17) сходится к точке минимума функции $f(x) = \sum_{i=1}^{m} w_i \|\hat{\pi}_i(x) - x\|^2$, являющейся взвешенным (с весами w_i) решением системы (1) методом наименьших квадратов.

Проекционные методы, основанные на подходе Чиммино, допускают эффективное распараллеливание, поскольку ортогональные проекции/отражения могут вычисляться одновременно и независимо. Масштабируемость метода Чиммино на многопроцессорных системах с распределенной памятью была исследована в работе [60]. Применимость проекционных методов по схеме Чиммино для решения нестационарных систем линейных неравенств рассматривалась в работе [61].

Решение систем линейных неравенств тесно связано с задачами ЛП, поэтому методы проекционного типа могут быть эффективно использованы для решения этого класса задач. Эквивалентность задачи линейной совместности и задачи ЛП основана на прямо-двойственном методе решения задачи ЛП. Рассмотрим прямую задачу ЛП в матричной форме:

$$\bar{x} = \arg\max_{x} \left\{ \langle c, x \rangle \,|\, Ax \leqslant b, x \geqslant \mathbf{0} \right\}, \tag{18}$$

где $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, $c \neq \mathbf{0}$. Сформулируем двойственную задачу по отношению к задаче (18):

$$\bar{u} = \arg\min_{u} \left\{ \langle b, u \rangle \,|\, A^T u \geqslant c, u \geqslant \mathbf{0} \right\}, \tag{19}$$

где $u \in \mathbb{R}^m$. Для прямой и двойственной задач ЛП справедливо следующее равенство:

$$\langle c, \bar{x} \rangle = \max_{Ax \leqslant b, x \geqslant \mathbf{0}} \langle c, x \rangle = \min_{A^T u \geqslant c, u \geqslant \mathbf{0}} \langle b, u \rangle = \langle b, \bar{u} \rangle . \tag{20}$$

Ерёмин в [38, 62] предложил следующий метод решения задачи ЛП, основанный на прямо-двойственном подходе. Пусть система линейных неравенств

$$A'x \leqslant b' \tag{21}$$

задает допустимую область прямой задачи (18). Указанная система получается из системы $Ax \leqslant b$ путем добавления векторного неравенства $-x \leqslant \mathbf{0}$. В данном случае $A' \in \mathbb{R}^{(m+n) \times n}$ и $b' \in \mathbb{R}^{m+n}$. Пусть a'_i обозначает i-тую строку матрицы A'. Сопоставим каждому неравенству $\langle a'_i, x \rangle \leqslant b'_i$ закрытое полупространство

$$\hat{H}'_i = \left\{ x \in \mathbb{R}^n \middle| \left\langle a'_i, x \right\rangle \leqslant b'_i \right\}, \tag{22}$$

и ограничивающую его гиперплоскость

$$H'_i = \left\{ x \in \mathbb{R}^n \middle| \left\langle a'_i, x \right\rangle = b'_i \right\}. \tag{23}$$

Обозначим через $\pi'_i(x)$ ортогональную проекцию точки x на гиперплоскость H'_i:

$$\pi'_i(x) = x - \frac{\langle a'_i, x \rangle - b'_i}{\|a'_i\|^2} a'_i. \tag{24}$$

Определим проекцию на полупространство $\hat{H}'_i$:

$$\hat{\pi}'_i(x) = x - \frac{\max\{0, \langle a'_i, x \rangle - b'_i\}}{\|a'_i\|^2} a'_i. \tag{25}$$

Указанная проекция обладает следующими двумя свойствами:

$$x \notin \hat{H}'_i \Rightarrow \hat{\pi}'_i(x) = \pi'_i(x); \tag{26}$$

$$x \in \hat{H}'_i \Rightarrow \hat{\pi}'_i(x) = x. \tag{27}$$

Определим отображение $\varphi_1 : \mathbb{R}^n \to \mathbb{R}^n$ следующим образом:

$$\varphi_1(x) = \frac{1}{m+n} \sum_{i=1}^{m+n} \hat{\pi}'_i(x). \tag{28}$$

Аналогичным образом определим допустимую область двойственной задачи (19):

$$D'x \geqslant c', \tag{29}$$

где $D = A^T \in \mathbb{R}^{n\times m}$, $D' \in \mathbb{R}^{(m+n)\times m}$, $c' \in \mathbb{R}^{n+m}$. Обозначим

$$\hat{\eta}'_j(u) = u - \frac{\max\left\{0, \left\langle d'_j, u\right\rangle - c'_j\right\}}{\left\|d'_j\right\|^2} d'_j, \tag{30}$$

и определим отображение $\varphi_2 : \mathbb{R}^m \to \mathbb{R}^m$ следующим образом:

$$\varphi_2(u) = \frac{1}{n+m}\sum_{j=1}^{n+m} \hat{\eta}'_j(x). \tag{31}$$

Далее, определим отображение $\varphi_3 : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$, соответствующее равенству (20):

$$\varphi_3\left([x,u]\right) = [x,u] - \frac{\langle c,x\rangle - \langle b,u\rangle}{\|c\|^2 + \|b\|^2}[c,-b]. \tag{32}$$

Здесь $[\cdot\,,\cdot]$ обозначает конкатенацию двух векторов.

Наконец, определим $\varphi : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ следующим образом:

$$\varphi\left([x,u]\right) = \varphi_3\left([\varphi_1(x), \varphi_2(u)]\right). \tag{33}$$

Если допустимая область прямой задачи является ограниченным непустым множеством, то последовательность, задаваемая формулой

$$\left[x^{(k+1)}, u^{(k+1)}\right] = \varphi\left(\left[x^{(k)}, u^{(k)}\right]\right), \tag{34}$$

будет сходиться к точке $[\bar{x}, \bar{u}]$, где $\bar{x}$ является решением прямой задачи (18), а $\bar{u}$ является решением двойственной задачи (19).

В статье [63] предлагается метод решения невырожденных задач ЛП, основанный на вычислении ортогональной проекции некоторой специальной точки, не зависящей от основной части исходных данных, описывающих задачу ЛП, на проблемно-зависимый конус, порождаемый ограничивающими неравенствами. Фактически, этот метод решает симметричную положительно определенную систему линейных уравнений специального вида. Автор описывает конечный алгоритм на основе метода активных множеств, способный вычислять ортогональные проекции для задач с тысячами строк и столбцов. Основным недостатком этого метода является существенное увеличение размерности исходной задачи.

Цензор в работе [64] описывает применение метода линейной супериоризации (LinSup) для решения задач ЛП. Метод LinSup направляет используемый итерационный алгоритм проекционного типа в сторону точек с увеличивающимся значением целевой функции. При этом LinSup не гарантирует нахождение точного оптимума задачи ЛП. Этот процесс не идентичен тому, который используется ЛП-решателями, но это возможная альтернатива симплекс-методу для решения задач очень большого размера. Основная идея LinSup состоит в том, чтобы добавить дополнительный терм, называемый возмущающим термом, в итерационное уравнение проекционного метода. Возмущающий терм направляет алгоритм

поиска допустимой точки в сторону увеличения значения целевой функции. В контексте задачи ЛП (18) целевая функция имеет вид $f(x) = \langle c, x\rangle$, и LinSup добавляет в итерационное уравнение (17) возмущающий терм вида $\left(-\eta\frac{c}{\|c\|}\right)$:

$$x^{(k+1)} = \left(-\eta\frac{c}{\|c\|}\right) + \sum_{i=1}^{m} w_i\hat{\pi}_i^{\lambda}\left(x^{(k)}\right). \tag{35}$$

Здесь $0 < \eta < 1$ — величина возмущения, являющаяся настраиваемым параметром алгоритма.

В статье [48] предлагается математическая модель для визуального представления многомерных задач ЛП. Для визуализации задачи ЛП вводится целевая гиперплоскость H_c, нормаль к которой совпадает с градиентом целевой функции $f(x) = \langle c, x\rangle$. В случае поиска максимума целевая гиперплоскость располагается так, чтобы значение целевой функции во всех ее точках было больше значения целевой функции в любой точке допустимого многогранника M. Для любой точки $g \in H_c$ определяется целевая проекция этой точки на многогранник M в соответствии со следующей формулой:

$$\gamma_M(g) = \begin{cases} \arg\min\limits_{x} \{\, \|x - g\| \,|\, x \in M, \pi_{H_c}(x) = g\}\,, & \text{если } \exists x \in M : \pi_{H_c}(x) = g; \\ +\infty, & \text{если } \neg\exists x \in M : \pi_{H_c}(x) = g. \end{cases} \tag{36}$$

Здесь, $\pi_{H_c}(x)$ обозначает ортогональную проекцию точки x на гиперплоскость H_c. На целевой гиперплоскости H_c строится прямоугольная решетка точек $\mathfrak{G} \in \mathbb{R}^n \times \mathbb{R}^{K^{(n-1)}}$, где K — число точек по одному измерению. Каждой точке $g \in \mathfrak{G}$ сопоставляется вещественное число $\|\gamma_M(g) - g\|$. В результате получается матрица размерности $(n-1)$, представляющая собой образ задачи ЛП. Этот подход открывает возможность использования нейронных сетей прямого распространения, включая сверточные, для решения многомерных задач ЛП. Основной проблемой для реализации такого подхода на практике является проблема построения обучающего набора данных. Обзор литературы показывает, что в настоящее время не существует методов, позволяющих построить обучающий набор данных, совместимый с представленным подходом. В следующих разделах мы опишем такой метод.

2. Теоретический базис

Данный раздел содержит необходимый теоретических базис, используемый для описания апекс-метода. Рассмотрим задачу ЛП в следующем виде:

$$\bar{x} = \arg\max_{x\in\mathbb{R}^n} \{\langle c, x\rangle \,|\, Ax \leqslant b\}\,, \tag{37}$$

где $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$, $m > 1$, $c \neq \mathbf{0}$. Мы предполагаем, что ограничение

$$-x \leqslant \mathbf{0} \tag{38}$$

также включено в матричное неравенство $Ax \leqslant b$. Обозначим через $\mathcal{P}$ множество индексов, нумерующих строки матрицы A:

$$\mathcal{P} = \{1, \cdots, m\}\,. \tag{39}$$

Пусть $a_i \in \mathbb{R}^n$ обозначает вектор, представляющий i-тую строку матрицы A. Мы предполагаем, что $a_i \neq \mathbf{0}$ для всех $i \in \mathcal{P}$. Обозначим через $\hat{H}_i$ замкнутое полупространство,

определяемое неравенством $\langle a_i, x\rangle \leqslant b_i$, а через H_i — его ограничивающую гиперплоскость:

$$\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}; \tag{40}$$

$$H_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle = b_i\}. \tag{41}$$

Определение 1. Полупространство $\hat{H}_i$ называется доминантным, если

$$\forall x \in \hat{H}_i, \forall \lambda \in \mathbb{R}_{>0} : x + \lambda c \in \hat{H}_i. \tag{42}$$

Геометрический смысл данного определения состоит в том, что луч, исходящий из любой точки доминантного полупространства в направлении вектора c, принадлежит этому полупространству.

Определение 2. Полупространство $\hat{H}_i$ называется рецессивным, если оно не является доминантным, то есть

$$\forall x \in \hat{H}_i, \exists \lambda \in \mathbb{R}_{>0} : x + \lambda c \notin \hat{H}_i. \tag{43}$$

Геометрический смысл этого определения состоит в том, что луч, исходящий из любой точки рецессивного полупространства в направлении вектора c, выходит за пределы этого полупространства.

Утверждение 1. Следующее условие является необходимым и достаточным для того, чтобы полупространство $\hat{H}_i$ было рецессивным:

$$\langle a_i, c\rangle > 0. \tag{44}$$

Доказательство. Сначала докажем необходимость. Пусть условие (43) имеет место. Обозначим

$$x' = \frac{\beta a_i}{\|a_i\|^2}. \tag{45}$$

Имеем

$$\langle a_i, x'\rangle = \left\langle a_i, \frac{\beta a_i}{\|a_i\|^2}\right\rangle = \beta\frac{\langle a_i, a_i\rangle}{\|a_i\|^2} = \beta, \tag{46}$$

то есть $x' \in \hat{H}_i$ в силу (40). Согласно условию (43) существует $\lambda' \in \mathbb{R}_{>0}$ такое, что

$$x' + \lambda' c \notin \hat{H}_i, \tag{47}$$

то есть

$$\langle a_i, x' + \lambda' c\rangle > \beta. \tag{48}$$

Подставляя сюда правую часть равенства (45) вместо x', получаем

$$\left\langle a_i, \frac{\beta a_i}{\|a_i\|^2} + \lambda' c\right\rangle > \beta. \tag{49}$$

Поскольку $\lambda' > 0$, отсюда следует, что

$$\langle a_i, c\rangle > 0. \tag{50}$$

Это доказывает необходимость.

Теперь докажем достаточность. Пусть условие (44) имеет место, но полупространство $\hat{H}_i$ при этом не является рецессивным, то есть

$$\forall x \in \hat{H}_i, \forall \lambda \in \mathbb{R}_{>0} : x + \lambda c \in \hat{H}_i. \tag{51}$$

Так как x', вычисляемый по формуле (45), принадлежит $\hat{H}_i$, отсюда следует

$$x' + \lambda c \in \hat{H}_i \tag{52}$$

для всех $\lambda \in \mathbb{R}_{>0}$, что равносильно

$$\langle a_i, x' + \lambda c \rangle \leqslant \beta. \tag{53}$$

Поставляя сюда правую часть равенства (45) вместо x', получаем

$$\left\langle a_i, \frac{\beta a_i}{\|a_i\|^2} + \lambda c \right\rangle \leqslant \beta. \tag{54}$$

Поскольку $\lambda > 0$, отсюда следует

$$\langle a_i, c \rangle \leqslant 0. \tag{55}$$

Получили противоречие с (44). Таким образом, достаточность также доказана. □

Обозначим

$$e_c = \frac{c}{\|c\|}. \tag{56}$$

Другими словами, e_c обозначает единичный вектор, сонаправленный с вектором c.

Утверждение 2. Пусть полупространство $\hat{H}_i$ является рецессивным. Тогда для любой точки $x' \in \mathbb{R}^n$ и любого положительного числа $\eta > 0$ точка

$$z = x' + \left(\eta + \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \right) e_c \tag{57}$$

не принадлежит полупространству $\hat{H}_i$, то есть

$$\langle a_i, z \rangle > b_i. \tag{58}$$

Доказательство. Так как полупространство $\hat{H}_i$ является рецессивным, то в соответствии с утверждением 1 справедливо следующее неравенство:

$$\langle a_i, c \rangle > 0. \tag{59}$$

В силу (57) мы имеем

$$\langle a_i, z \rangle = \left\langle a_i, x' + \left(\eta + \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \right) e_c \right\rangle = \eta \langle a_i, e_c \rangle + b_i. \tag{60}$$

Подставляя в (60) правую часть равенства (56) вместо e_c, получаем

$$\langle a_i, z \rangle = \frac{\eta}{\|c\|} \langle a_i, c \rangle + b_i. \tag{61}$$

Поскольку $\eta > 1$, из (59) следует $\frac{\eta}{\|c\|} \langle a_i, c \rangle > 0$. Это означает, что из (61) следует $\langle a_i, z \rangle > b_i$, то есть $z \notin \hat{H}_i$. *Утверждение доказано.* □

Определим

$$\mathcal{I} = \{ i \in \mathcal{P} \,|\langle a_i, c\rangle > 0 \}, \tag{62}$$

то есть $\mathcal{I}$ представляет множество индексов, для которых полупространство $\hat{H}_i$ является рецессивным. Поскольку допустимый многогранник M представляет собой ограниченное множество, имеем

$$\mathcal{I} \neq \emptyset. \tag{63}$$

Следствие 1. Пусть имеется произвольная допустимая точка x' задачи ЛП (37):

$$\forall i \in \mathcal{P} : \left\langle a_i, x' \right\rangle \leqslant b_i. \tag{64}$$

Тогда для любого положительного числа $\eta \in \mathbb{R}_{>0}$ точка

$$z = x' + \left(\eta + \max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} \right) e_c \tag{65}$$

не принадлежит ни одному рецессивному пространству $\hat{H}_i$, то есть

$$\forall i \in \mathcal{I} : \langle a_i, z\rangle > b_i. \tag{66}$$

Доказательство. Условие (64) равносильно условию

$$\forall i \in \mathcal{I} : b_i - \left\langle a_i, x' \right\rangle \geqslant 0. \tag{67}$$

Из (62) и (56) получаем

$$\forall i \in \mathcal{I} : \langle a_i, e_c\rangle > 0. \tag{68}$$

Отсюда с учетом (67) следует, что

$$\max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} \geqslant 0 \tag{69}$$

для всех $i \in \mathcal{I}$. Зафиксируем произвольный $j \in \mathcal{I}$ и определим

$$\eta' = \eta + \max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} - \frac{b_j - \langle a_j, x'\rangle}{\langle a_j, e_c\rangle}, \tag{70}$$

где $\eta > 0$. Принимая во внимание (69), отсюда следует $\eta' > 0$. Из (65) и (70) получаем

$$z = x' + \left(\eta + \max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} \right) e_c = x' + \left(\eta' + \frac{b_j - \langle a_j, x'\rangle}{\langle a_j, e_c\rangle} \right) e_c. \tag{71}$$

В силу утверждения 2 это означает, что $\langle a_j, z\rangle > b_j$, то есть точка z, вычисляемая по формуле (65), не принадлежит полупространству $\hat{H}_j$ для всех $j \in \mathcal{I}$. Следствие доказано. □

Следующее утверждение определяет область, где может находиться решение задачи ЛП (37).

Утверждение 3. Пусть $\bar{x}$ является решением задачи ЛП (37). Тогда найдется индекс $i' \in \mathcal{I}$ такой, что

$$\bar{x} \in H_{i'}, \tag{72}$$

то есть существует рецессивное полупространство $\hat{H}_{i'}$ такое, что ограничивающая его гиперплоскость $H_{i'}$ содержит $\bar{x}$.

Доказательство. Обозначим через $\mathcal{J}$ множество индексов, для которых полупространство $\hat{H}_j$ является доминантным:

$$\mathcal{J} = \mathcal{P} \backslash \mathcal{I}. \tag{73}$$

Так как $\bar{x}$ принадлежит допустимой области задачи ЛП (37), справедливы следующие включения:

$$\bar{x} \in \bigcap_{j \in \mathcal{J}} \hat{H}_j, \tag{74}$$

$$\bar{x} \in \bigcap_{i \in \mathcal{I}} \hat{H}_i. \tag{75}$$

Определим луч Y следующим образом:

$$Y = \{\bar{x} + \lambda c \,|\lambda \in \mathbb{R}_{\geqslant 0}\}. \tag{76}$$

В соответствии с определением 1 имеем

$$Y \subset \bigcap_{j \in \mathcal{J}} \hat{H}_j, \tag{77}$$

то есть луч Y принадлежит всем доминантным полупространствам. В силу определения 2

$$\forall i \in \mathcal{I}, \exists \lambda \in \mathbb{R}_{>0} \, : \bar{x} + \lambda c \notin \hat{H}_i. \tag{78}$$

Принимая во внимание (75), это означает, что

$$\forall i \in \mathcal{I} : Y \cap H_i = y_i \in \mathbb{R}^n, \tag{79}$$

то есть луч Y пересекает любую гиперплоскость H_i, ограничивающую рецессивное полупространство $\hat{H}_i$, в единственной точке $y_i \in \mathbb{R}^n$. Положим

$$i' = \arg\min_{i \in \mathcal{I}} \{\|\bar{x} - y_i\| \,|y_i = Y \cap H_i\}, \tag{80}$$

то есть гиперплоскость $H_{i'}$ является ближайшей к точке $\bar{x}$ для всех $i \in \mathcal{I}$. Обозначим через $\bar{y}$ пересечение луча Y и гиперплоскости $H_{i'}$:

$$\bar{y} = Y \cap H_{i'}. \tag{81}$$

В соответствии с (75), (76) и (80)

$$\bar{y} \in \bigcap_{i \in \mathcal{I}} \hat{H}_i, \tag{82}$$

то есть точка $\bar{y}$ принадлежит всем рецессивным полупространствам. В силу (77) отсюда следует, что

$$\bar{y} \in \bigcap_{i \in \mathcal{P}} \hat{H}_i. \tag{83}$$

Это означает, что $\bar{y}$ принадлежит допустимой области задачи ЛП (37).

Положим

$$\lambda' = \|\bar{x} - \bar{y}\|. \tag{84}$$

Тогда в силу (76) имеем

$$\langle c, \bar{y} \rangle = \langle c, \bar{x} + \lambda' e_c \rangle = \langle c, \bar{x} \rangle + \lambda' \frac{\langle c, c \rangle}{\|c\|} = \langle c, \bar{x} \rangle + \lambda' \|c\| . \tag{85}$$

Поскольку $\bar{x}$ является решением задачи ЛП (37), следующее условие имеет место:

$$\forall y \in \bigcap_{i \in \mathcal{P}} \hat{H}_i : \langle c, y \rangle \leqslant \langle c, \bar{x} \rangle . \tag{86}$$

Сопоставляя это с (83), получаем

$$\langle c, \bar{y} \rangle \leqslant \langle c, \bar{x} \rangle . \tag{87}$$

Принимая во внимание, что $\lambda' \geqslant 0$ и $c \neq \mathbf{0}$, в силу (85) и (87) имеем $\lambda' = 0$. В соответствии с (84) отсюда следует, что $\bar{x} = \bar{y}$. В силу (81) это означает, что $\bar{x} \in H_{i'}$, где $\hat{H}_{i'}$ является рецессивным полупространством. *Утверждение доказано.* □

Определение 3. Пусть $M \neq \emptyset$ — выпуклое замкнутое множество. Однозначное отображение $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ называется M-фейеровским отображением [38], если

$$\forall x \in M : \varphi(x) = x, \tag{88}$$

и

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\varphi(x) - y\| < \|x - y\|. \tag{89}$$

Утверждение 4. Пусть $M \neq \emptyset$ — выпуклое замкнутое множество, $x^{(0)}$ — произвольная точка в $\mathbb{R}^n$. Если $\varphi(\cdot)$ является непрерывным M-фейеровским отображением, то последовательность

$$\left\{ x^{(k)} = \varphi^k \left(x^{(0)} \right) \right\}_{k=1}^{\infty},$$

порождаемая этим отображением, сходится к точке, принадлежащей M:

$$x^{(k)} \to \tilde{x} \in M. \tag{90}$$

Доказательство. Сходимость непосредственно следует из теоремы 6.1 и следствия 6.1 в [38]. *Утверждение доказано.* □

Обозначим через $\pi_i(x)$ ортогональную проекцию точки x на гиперплоскость H_i:

$$\pi_i(x) = x - \frac{\langle a_i, x \rangle - b_i}{\|a_i\|^2} a_i. \tag{91}$$

Следующее утверждение дает нам непрерывное M-фейеровское отображение, которое будет использоваться в апекс-методе.

Утверждение 5. Пусть $M \neq \emptyset$ — допустимый многогранник задачи ЛП (37):

$$M = \bigcap_{i=1}^{m} \hat{H}_i. \tag{92}$$

Известно, что в этом случае M является выпуклым замкнутым множеством. Для произвольной точки $x \in \mathbb{R}^n$ определим множество индексов

$$\mathcal{J}_x = \{ i \,|\langle a_i, x \rangle > b_i; i \in \mathcal{P} \} . \tag{93}$$

Другими словами, $\mathcal{J}_x$ — множество индексов полупространств $\hat{H}_i$, которые не содержат точку x. Однозначное отображение $\psi : \mathbb{R}^n \to \mathbb{R}^n$, задаваемое формулой

$$\psi(x) = \begin{cases} x, & \text{если } x \in M; \\ \frac{1}{|\mathcal{J}_x|} \sum\limits_{i \in \mathcal{J}_x} \pi_i(x), & \text{если } x \notin M, \end{cases} \tag{94}$$

является непрерывным M-фейеровским отображением.

Доказательство. Очевидно, что отображение $\psi(\cdot)$ является непрерывным. Покажем, что выполняется условие (89). Доказательство проведем по общей схеме, представленной в [38]. Пусть $y \in M$ и $x \notin M$. Это означает, что

$$\mathcal{J}_x \neq \emptyset. \tag{95}$$

В силу (93) для всех $i \in \mathcal{J}_x$ справедливо неравенство

$$\|\pi_i(x) - x\| > 0. \tag{96}$$

Согласно лемме 3.2 в [38] для всех $i \in \mathcal{J}_x$ также выполняется следующее неравенство:

$$\|\pi_i(x) - y\|^2 \leqslant \|x - y\|^2 - \|\pi_i(x) - x\|^2. \tag{97}$$

Отсюда следует

$$\begin{aligned} &\|y - \psi(x)\|^2 = \left\| y - \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \pi_i(x) \right\|^2 = \left\| \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} (y - \pi_i(x)) \right\|^2 \leqslant \tfrac{1}{|\mathcal{J}_x|^2} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leqslant \\ &\leqslant \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leqslant \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \left(\|x - y\|^2 - \|\pi_i(x) - x\|^2 \right) \leqslant \\ &\leqslant \|x - y\|^2 - \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2. \end{aligned}$$

В соответствии с (95) и (96) следующее неравенство имеет место:

$$\tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2 > 0. \tag{98}$$

Отсюда

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\psi(x) - y\| < \|x - y\|.$$

Утверждение доказано. □

Определение 4. Пусть $M \neq \emptyset$ — допустимый многогранник задачи ЛП (37), $\psi(\cdot)$ — отображение, определяемое формулой (94). *Псевдопроекцией* $\rho_M(x)$ точки x на допустимый многогранник M называется предельная точка последовательности $[x, \psi(x), \psi^2(x), \ldots, \psi^k(x), \ldots]$:

$$\lim_{k \to \infty} \left\| \rho_M(x) - \psi^k(x) \right\| = 0. \tag{99}$$

Корректность этого определения вытекает из утверждений 4 и 5.

3. Описание апекс-метода

В этом разделе мы опишем новый масштабируемый итерационный метод решения задачи ЛП (37), получивший название «апекс-метод». Апекс-метод построен по схеме предиктор/корректор и включает в себя две последовательные стадии: Quest (предиктор) и Target (корректор). Стадия Quest находит грубое начальное приближение для задачи ЛП (37). Стадия Target уточняет это начальное приближение с определенной точностью. Основной операцией, используемой как на стадии Quest, так и на стадии Target, является операция вычисления псевдопроекции (см. определение 4). Следующий раздел посвящен описанию и исследованию алгоритма вычисления псевдопроекции.

3.1. Алгоритм вычисления псевдопроекции

Базовой операцией, используемой в апекс-методе, является операция псевдопроектирования, заключающаяся в последовательном применении отображения $\psi(\cdot)$, задаваемого формулой (94), к исходной точке. В данном разделе мы рассмотрим реализацию операции псевдопроектирования в виде последовательного и параллельного алгоритмов. Согласно определению 4 операция псевдопроектирования $\rho_M(\cdot)$ отображает произвольную точку $x \in \mathbb{R}^n$ в точку $\rho_M(x)$, принадлежащую допустимому многограннику M, представляющему допустимую область задачи ЛП (37). Вычисление $\rho_M(x)$ организуется в виде итерационного процесса с использованием формулы (94). Последовательная реализация этого процесса представлена в виде алгоритма 1. Кратко прокомментируем шаги этого алгоритма. Основной итерационный процесс, вычисляющий последовательность фейеровских приближений, представлен в виде цикла **repeat–until** (шаги 4–20). На шагах 5–10 строится множество $\mathcal{J}$, содержащее индексы полупространств $\hat{H}_i$, которым не принадлежит текущее приближение $x^{(k)}$. На шагах 14–18 вычисляется следующее приближение $x^{(k+1)}$ по формуле (94). Алгоритм завершает свою работу, когда расстояние между соседними приближения станет меньше малой положительной константы ϵ.

Известно, что в случае больших задач ЛП проекционный метод может потребовать значительных временных затрат [65]. Потому мы разработали параллельную версию алгоритма 1, представленную в виде алгоритма 2. Параллельный алгоритм построен на основе модели параллельных вычислений BSF [66], ориентированной на кластерные вычислительные системы. Модель BSF использует схему распараллеливания «мастер–рабочие» и требует представление алгоритма в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*. В качестве второго параметра функции высшего порядка *Map* в алгоритме 2 используется список $\mathcal{L}_{map} = [1, \ldots, m]$, содержащий порядковые номера ограничений задачи ЛП (37), а в качестве первого параметра фигурирует параметризованная функция

$$\mathrm{F}_x : \mathcal{P} \to \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0},$$

определенная следующим образом:

$$\begin{aligned} &\mathrm{F}_x(i) = (u_i, \sigma_i)\,; \\ &u_i = \begin{cases} \pi_i(x), & \text{если } \langle a_i, x\rangle > b_i; \\ \mathbf{0}, & \text{если } \langle a_i, x\rangle \leqslant b_i; \end{cases} \\ &\sigma_i = \begin{cases} 1, & \text{если } \langle a_i, x\rangle > b_i; \\ 0, & \text{если } \langle a_i, x\rangle \leqslant b_i. \end{cases} \end{aligned} \tag{100}$$

Алгоритм 1 Последовательное вычисление псевдопроекции $\rho_M(x)$

Require: $\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$, $M \neq \emptyset$

function $\rho_M(x)$
  $k := 0$
  $x^{(0)} := x$
  repeat
    $\mathcal{J} := \emptyset$
    for $i = 1 \ldots m$ **do**
      if $\left\langle a_i, x^{(k)} \right\rangle > b_i$ **then**
        $\mathcal{J} := \mathcal{J} \cup \{i\}$
      end if
    end for
    if $\mathcal{J} = \emptyset$ **then**
      return $x^{(k)}$
    end if
    $S := 0$
    for all $i \in \mathcal{J}$ **do**
      $S := S + \left(\left\langle a_i, x^{(k)} \right\rangle - b_i\right) a_i / \left\|a_i\right\|^2$
    end for
    $x^{(k+1)} := x^{(k)} - S / \left|\mathcal{J}\right|$
    $k := k + 1$
  until $\left\|x^{(k)} - x^{(k-1)}\right\| < \epsilon$
  return $x^{(k)}$
end function

Таким образом, функция высшего порядка $Map\left(\mathrm{F}_x, \mathcal{L}_{map}\right)$ преобразует список номеров ограничений $\mathcal{L}_{map}$ в список пар (u_i, σ_i):

$$Map\left(\mathrm{F}_x, \mathcal{L}_{map}\right) = \left[\mathrm{F}_x(1), \ldots, \mathrm{F}_x(m)\right] = \left[(u_1, \sigma_1), \ldots, (u_m, \sigma_m)\right]. \tag{101}$$

Здесь u_i является ортогональной проекцией точки x на гиперплоскость H_i в том случае, когда $x \notin \hat{H}_i$, и нулевым вектором в противном; σ_i соответственно принимает значение 1 или 0. Обозначим $\mathcal{L}_{reduce} = [(u_1, \sigma_1), \ldots, (u_m, \sigma_m)]$. Определим бинарную ассоциативную операцию

$$\oplus : \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0} \to \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0},$$

являющуюся первым параметром функции высшего порядка *Reduce*:

$$\left(u', \sigma'\right) \oplus \left(u'', \sigma''\right) = \left(u' + u'', \sigma' + \sigma''\right). \tag{102}$$

Функция высшего порядка $Reduce\left(\oplus, \mathcal{L}_{reduce}\right)$ редуцирует список $\mathcal{L}_{reduce}$ к одной паре путем последовательного применения операции $\oplus$ ко всем элементам списка:

$$Reduce\left(\oplus, \mathcal{L}_{reduce}\right) = (u_1, \sigma_1) \oplus \ldots \oplus (u_m, \sigma_m) = (u, \sigma), \tag{103}$$

Алгоритм 2 Параллельное вычисление псевдопроекции $\rho_M(x)$

	мастер		l**-тый рабочий** $(l = 0, \ldots, L-1)$
1:	**input** $n, x^{(0)}$	1:	**input** n, m, A, b, c
2:		2:	$L :=$ NumberOfWorkers
3:	$k := 0$	3:	$\mathcal{L}_{map(l)} := [lm/L, \ldots, ((l+1)m/L) - 1]$
4:	**repeat**	4:	**repeat**
5:	**Bcast** $x^{(k)}$	5:	**RecvFromMaster** $x^{(k)}$
6:		6:	$\mathcal{L}_{reduce(l)} := Map(\mathrm{F}_{x^{(k)}}, \mathcal{L}_{map(l)})$
7:		7:	$(u_l, \sigma_l) := Reduce(\oplus, \mathcal{L}_{reduce(l)})$
8:	**Gather** $\mathcal{L}_{reduce}$	8:	**SendToMaster** (u_l, σ_l)
9:	$(u, \sigma) := Reduce(\oplus, \mathcal{L}_{reduce})$	9:	
10:	$x^{(k+1)} := u/\sigma$	10:	
11:	$k := k + 1$	11:	
12:	$exit := \left\|x^{(k)} - x^{(k-1)}\right\| < \epsilon$	12:	
13:	**Bcast** $exit$	13:	**RecvFromMaster** $exit$
14:	**until** $exit$	14:	**until** $exit$
15:	**output** $x^{(k)}$	15:	
16:	**stop**	16:	**stop**

где

$$u = \sum_{i=1}^{m} u_i; \tag{104}$$

$$\sigma = \sum_{i=1}^{m} \sigma_i. \tag{105}$$

Параллельная работа алгоритма 2 организована по схеме «мастер–рабочие» и включает в себя $L+1$ процесс: один процесс–мастер и L процессов–рабочих. Процесс–мастер осуществляет общее управление вычислениями, распределяет работу между процессами–рабочими, получает от них результаты и формирует итоговый результат. Для простоты будем предполагать, что количество ограничений m в задаче ЛП (37) кратно количеству рабочих L. На шаге 1 мастер вводит исходные данные: размерность пространства n и начальную точку $x^{(0)}$. На шаге 3 мастер присваивает счетчику итераций k значение 0. Шаги 4–14 реализуют основной цикл **repeat**–**until**, вычисляющий псевдопроекцию. На шаге 5 мастер рассылает текущее приближение $x^{(k)}$ всем рабочим. На шаге 8 он получает от рабочих частичные результаты, которые на шаге 9 редуцируются в пару (u, σ). Последняя используется на шаге 10 для вычисления следующего приближения $x^{(k+1)}$. На шаге 11 мастер увеличивает на единицу счетчик итераций k. На шаге 12 мастер проверяет условие завершения и присваивает результат проверки логической переменной $exit$. На шаге 13 мастер рассылает всем рабочим значение логической переменной $exit$. Если логическая переменная $exit$ принимает значение «истина», цикл **repeat**–**until** завершается на шаге 14. На шаге 15

мастер выводит последнее приближение $x^{(k)}$ в качестве результата псевдопроекции. Шаг 16 завершает работу процесса–мастера.

Все рабочие выполняют один и тот же код, но над различными данными. На шаге 1 l-тый рабочий вводит исходные данные задачи ЛП. Затем он формирует подсписок своих номеров ограничений для обработки (шаги 2–3). Для удобства программирования нумерация ограничений начинается с нуля. Подсписки различных рабочих не пересекаются, и их объединение дает полный список номеров ограничений:

$$\mathcal{L}_{map} = \mathcal{L}_{map(0)} +\!\!+ \ldots +\!\!+ \mathcal{L}_{map(L-1)}. \tag{106}$$

Символ $+\!\!+$ здесь обозначает операцию конкатенации списков. Цикл **repeat–until** рабочего соответствует циклу **repeat–until** мастера (шаги 4–14). На шаге 5 рабочий получает от мастера текущее приближение $x^{(k)}$. На шаге 6 рабочий вызывает функцию высшего порядка Map, которая, в свою очередь, применяет параметризованную функцию $\mathrm{F}_{x^{(k)}}$, определенную по формуле (100), ко всем элементам подсписка $\mathcal{L}_{map(l)}$, формируя на выходе подсписок пар $\mathcal{L}_{reduce(l)}$. Этот подсписок на шаге 7 редуцируется рабочим в единственную пару (u_l, σ_l) с помощью функции высшего порядка $Reduce$, которая последовательно применяет бинарную операцию $\oplus$, определенную по формуле (102), ко всем элементам подсписка $\mathcal{L}_{reduce(l)}$. На шаге 13 рабочий получает от мастера значение логической переменной $exit$. Если эта переменная принимает значение «истина», то рабочий процесс завершается. В противном случае продолжает выполняться цикл **repeat–until**. Операторы обмена **Bcast**, **Gather**, **RecvFromMaster** и **SendToMaster** обеспечивают неявную синхронизацию работы процесса–мастера и процессов–рабочих.

Выполним оценку границы масштабируемости описанного параллельного алгоритма, используя стоимостную метрику модели BSF [66]. Под границей масштабируемости параллельного алгоритма понимается максимальное число процессорных узлов, до которого наблюдается рост ускорения. Стоимостная метрика модели BSF включает в себя следующие параметры.

- m : длина списка $\mathcal{L}_{map}$;
- D : латентность (время, необходимое мастеру, чтобы послать одному рабочему сообщение длиной в один байт);
- t_c : время, необходимое мастеру, чтобы переслать одному рабочему текущее приближение $x^{(k)}$ и получить от него пару (u_l, σ_l) с учетом латентности;
- t_{Map} : время, требуемое одному рабочему, чтобы выполнить функцию высшего порядка Map для всех элементов списка $\mathcal{L}_{map}$;
- t_a : время, необходимое для выполнения бинарной операции $\oplus$, определяемой по формуле (102).

Согласно формуле (14) из [66], граница масштабируемости L_{max} параллельного алгоритма 2 может быть оценена следующим образом:

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{t_c}{t_a \ln 2}\right)^2 + \frac{t_{Map}}{t_a} + 4m} - \frac{t_c}{t_a \ln 2}. \tag{107}$$

Вычислим временные параметры в формуле (107). Введем следующие обозначения для одной итерации цикла **repeat–until** (шаги 4–14 алгоритма 2):

c_c : количество чисел, пересылаемых от мастера рабочему и обратно в ходе одной итерации;

c_F : количество арифметических операций и операций сравнения, необходимых для вычисления функции F_x, определяемой по формуле (100);

$c_\oplus$: количество арифметических операций и операций сравнения, необходимых для выполнения бинарной операции $\oplus$, определяемой по формуле (102).

На шаге 5 мастер посылает l-тому рабочему вектор размерности n. Затем на шаге 8 мастер получает от l-того рабочего пару, состоящую из вектора размерности n и одного вещественного числа. Кроме этого, на шаге 13 мастер посылает l-тому рабочему одно логическое значение. Последняя пересылка состоит в пересылке одного бита и равносильна одному добавлению латентности D, что будет сделано позже. Следовательно,

$$c_c = 2n + 1. \tag{108}$$

Принимая во внимание формулы (91), (100) и предполагая, что значения $\|a_i\|^2$ для всех $i = 1, \ldots, m$ вычислены заранее, получаем

$$c_F = 3n + 2. \tag{109}$$

Исходя из (102), для $c_\oplus$ справедлива следующая формула:

$$c_\oplus = 2n + 1. \tag{110}$$

Обозначим через τ_{op} время выполнения одной арифметической операции или операции сравнения. Обозначим через τ_{tr} время пересылки одного вещественного числа без учета латентности. Тогда на основе (108), (109) и (110) имеем

$$t_c = c_c\tau_{tr} + 3D = (2n + 1)\tau_{tr} + 3D; \tag{111}$$

$$t_{Map} = c_F m\tau_{op} = (3n + 2)m\tau_{op}; \tag{112}$$

$$t_a = c_\oplus\tau_{op} = (2n + 1)\tau_{op}. \tag{113}$$

Подставляя правые части этих формул в (107) и добавляя латентность D, получаем

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{(2n+1)\tau_{tr} + 3D}{(2n+1)\tau_{op}\ln 2}\right)^2 + \left(\frac{n+1}{2n+1} + 5\right)m} - \frac{(2n+1)\tau_{tr} + 3D}{(2n+1)\tau_{op}\ln 2},$$

где n — размерность пространства, m — количество ограничений. Для больших значений n и m отсюда вытекает следующая приближенная оценка:

$$L_{max} \approx O(\sqrt{m}). \tag{114}$$

Полученная оценка свидетельствует о том, что параллельный алгоритм 2 обладает слабой масштабируемостью[1].

[1]Если граница масштабируемости определяется формулой $L_{max} = O\left(m^\alpha\right)$, то мы полагаем, что параллельный алгоритм обладает сильной масштабируемостью при $\alpha \geqslant 1$, слабой масштабируемостью при $0 < \alpha < 1$, и масштабируемость отсутствует при $\alpha \leqslant 0$.

3.2. Стадия Quest

Стадия Quest играет роль предиктора и состоит из следующих шагов.

1. Найти допустимую точку $\tilde{x} \in M$.
2. Вычислить точку апекса z.
3. Построить начальное приближение $u^{(0)}$, являющееся псевдопроекцией точки апекса z на допустимый многогранник M.

Допустимая точка $\tilde{x}$ на шаге 1 может быть вычислена с помощью формулы

$$\tilde{x} = \begin{cases} \mathbf{0}, & \text{если } \mathbf{0} \in M; \\ \rho_M(\mathbf{0}), & \text{если } \mathbf{0} \notin M, \end{cases} \tag{115}$$

где $\rho_M(\cdot)$ — операция псевдопроектирования на допустимый многогранник M (см. определение 4).

Точка апекса z на шаге 2 может быть вычислена следующим образом:

$$z = \tilde{x} + \left(\eta + \max \left\{ \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \,\middle|\, i \in \mathcal{I} \right\} \right) e_c, \tag{116}$$

где $\mathcal{I}$ — множество индексов, для которых полупространство $\hat{H}_i$ является c-рецессивным; $\eta \in \mathbb{R}_{>0}$ — положительный параметр, определяющий удаление точки z от точки $\tilde{x}$. Следствие 1 гарантирует, что при любом $\eta > 0$ точка z, вычисленная по формуле (116), не принадлежит никакому рецессивному полупространству $\hat{H}_i$. Подобный выбор точки апекса z основывается на эвристике, согласно которой псевдопроекция такой точки будет находиться «не очень далеко» от точного решения задачи ЛП. Данная эвристика основана на утверждении 3, в котором говорится, что решение задачи ЛП (37) лежит на некоторой гиперплоскости H_i, ограничивающей рецессивное полупространство $\hat{H}_i$. При этом значение параметра η может существенно влиять на близость точки $\rho_M(z)$ к точному решению. Оптимальное значение η может быть получено путем нахождения максимума целевой функции с использованием метода последовательной дихотомии.

На шаге 3 вычисляется точка $u^{(0)}$ по формуле

$$u^{(0)} = \rho_M(z). \tag{117}$$

Эта точка служит начальным приближением на стадии Target. Многочисленные вычислительные эксперименты, выполненные нами на искусственных и реальных невырожденных задачах ЛП, показывают, что итерационный процесс вычисления псевдопроекции, стартуя с произвольной внешней точки, всегда сходится к точке на границе допустимого многогранника M. Однако, в настоящий момент у нас отсутствует строгое доказательство этого факта.

3.3. Стадия Target

Стадия Target играет в апекс-методе роль корректора и вычисляет последовательность точек

$$\left\{ u^{(0)}, u^{(1)}, \ldots, u^{(k)}, \ldots \right\}, \tag{118}$$

обладающую следующими свойствами:

$$u^{(k)} \in \Gamma_M; \tag{119}$$

Алгоритм 3 Стадия Target

Require: $\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$, $M \neq \emptyset$

1: **input** $u^{(0)}$
2: $k := 0$
3: $v := u^{(k)} + \delta e_c$
4: $w := \rho_M(v)$
5: **while** $\left\langle c, w - u^{(k)}\right\rangle > \epsilon_f$ **do**
6: **assert** $\exists i \in \mathcal{I} : w, u^{(k)} \in H_i$ ▷ Если не выполняется, уменьшить δ
7: $d := w - u^{(k)}$
8: $\lambda' = \max\left\{\lambda \in \mathbb{R}_{>0} | \, u^{(k)} + \lambda d \in M\right\}$
9: $u^{(k+1)} := u^{(k)} + \lambda' d$
10: $k := k + 1$
11: $v := u^{(k)} + \delta e_c$
12: $w := \rho_M(v)$
13: **end while**
14: **output** $u^{(k)}$
15: **stop**

$$\left\langle c, u^{(k)}\right\rangle < \left\langle c, u^{(k+1)}\right\rangle; \tag{120}$$

$$\lim_{k\to\infty} \left\| u^{(k)} - \bar{x}\right\| = 0 \tag{121}$$

для всех $k \in \{0, 1, 2, \ldots\}$. Здесь Γ_M обозначает множество граничных точек допустимого многогранника M. Условие (119) означает, что все точки последовательности (118) лежат на границе допустимого многогранника M. Условие (120) говорит о том, что значение целевой функции в каждой точке последовательности (118) больше, чем в предыдущей. Согласно условию (121) последовательность (118) сходится к точному решению задачи ЛП (37).

Реализация стадии Target приведена в виде алгоритма 3. Дадим краткие комментарии по шагам алгоритма 3. На шаге 1 осуществляется ввод начального приближения $u^{(0)}$, полученного на стадии Quest. На шаге 2 счетчику итераций k присваивается значение 0. На шаге 3 вычисляется внешняя точка v как сумма векторов δe_c и $u^{(k)}$. Здесь e_c обозначает единичный вектор, сонаправленный с вектором c. На шаге 4 вычисляется точка w, являющаяся псевдопроекцией точки v на допустимый многогранник M. Шаги 5–13 реализуют основной цикл стадии Target, проиллюстрированный на рис. 1. Этот цикл выполняется, пока справедливо условие

$$\left\langle c, w - u^{(k)}\right\rangle > \epsilon_f. \tag{122}$$

Здесь ϵ_f — малый положительный параметр. На шаге 6 проверяется требование, в соответствии с которым точки w и $u_{(k)}$ должны лежать на некоторой гиперплоскости H_i, ограничивающей рецессивное полупространство $\hat{H}_i$. Это необходимо для того, чтобы перемещение от точки $u^{(k)}$ к точке $u^{(k+1)}$ происходило по поверхности многогранника M, а не через его внутреннюю часть. Если это требование не выполняется, необходимо уменьшить параметр δ. На шаге 7 вычисляется вектор d, задающий направление перемещения. На шаге 8 вычисляется

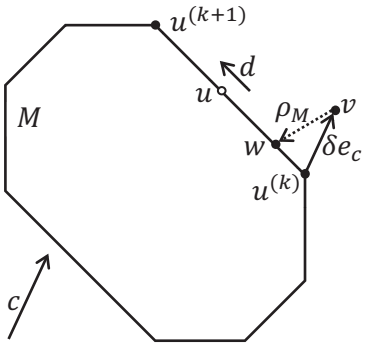


Рис. 1. Итерация основного цикла стадии Target

максимальное положительное число λ', для которого точка $\left(u^{(k)} + \lambda' d\right)$ принадлежит многограннику M. На шаге 9 вычисляется следующее приближение $u^{(k+1)}$. На шаге 10 счетчик итераций k увеличивается на 1. На шагах 11 и 12 вычисляются новая внешняя точка v и ее псевдопроекция w, используемые на следующей итерации основного цикла. После выхода из основного цикла на шаге 14 точка $u^{(k)}$ выводится в качестве приближенного решения задачи ЛП (37).

Следующее утверждение гарантирует сходимость алгоритма 3.

Утверждение 6. Пусть допустимый многогранник M задачи ЛП (37) является непустым ограниченным множеством. Тогда последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3, завершается через конечное число итераций $K \geqslant 0$ в некоторой допустимой точке, причем

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \ldots < \left\langle c, u^{(K)} \right\rangle. \tag{123}$$

Доказательство. Случай $K = 0$ является тривиальным. Пусть $K > 0$, либо $K = \infty$. Сначала покажем, что для любого $k < K$ выполняется следующее неравенство:

$$\left\langle c, u^{(k)} \right\rangle < \left\langle c, u^{(k+1)} \right\rangle. \tag{124}$$

Действительно, из (122) следует, что

$$\left\langle c, u^{(k)} \right\rangle < \langle c, w \rangle. \tag{125}$$

Принимая во внимание шаг 7 алгоритма 3, это означает, что

$$d \neq \mathbf{0}. \tag{126}$$

В соответствии с шагами 8, 9 имеем

$$u^{(k+1)} = u^{(k)} + \lambda' d, \tag{127}$$

где $\lambda' > 0$. Принимая во внимание неравенство (122) и шаг 7 алгоритма 3, отсюда следует

$$\begin{aligned} &\left\langle c, u^{(k+1)} \right\rangle = \left\langle c, u^{(k)} + \lambda' d \right\rangle = \left\langle c, u^{(k)} + \lambda' \left(w - u^{(k)} \right) \right\rangle = \\ &= \left\langle c, u^{(k)} \right\rangle + \lambda' \left\langle c, w - u^{(k)} \right\rangle > \left\langle c, u^{(k)} \right\rangle. \end{aligned}$$

Теперь покажем, что $K < \infty$. Предположим противное, то есть алгоритм 3 генерирует бесконечную последовательность точек. В таком случае мы получаем бесконечную монотонно возрастающую числовую последовательность

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \dots \tag{128}$$

Поскольку допустимый многогранник M является ограниченным множеством, последовательность (128) ограничена сверху. Согласно теореме Вейерштрасса монотонно возрастающая ограниченная числовая последовательность имеет конечный предел, равный ее супремуму. Это означает, что существует $K' \in \mathbb{N}$ такой, что

$$\forall k > K' : \left\langle c, u^{(k+1)} \right\rangle - \left\langle c, u^{(k)} \right\rangle < \epsilon_f. \tag{129}$$

Отсюда следует

$$\forall k > K' : \langle c, w \rangle - \left\langle c, u^{(k)} \right\rangle < \epsilon_f, \tag{130}$$

что равносильно

$$\forall k > K' : \left\langle c, w - u^{(k)} \right\rangle < \epsilon_f. \tag{131}$$

Получили противоречие с условием (122) выполнения цикла, используемом на шаге 5 алгоритма 3. *Утверждение доказано.* □

Покажем, что последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3, сходится к точному решению задачи ЛП (37) при $\epsilon_f \to 0$. Для этого заметим, что при $\delta \to 0$ псевдопроекция сводится к метрической проекции. Следуя [38], дадим определение метрической проекции.

Определение 5. Пусть Q является замкнутым выпуклым множеством в $\mathbb{R}^n$, и $Q \neq \emptyset$. Метрическая проекция $P_Q(x)$ точки $x \in \mathbb{R}^n$ на множество Q определяется формулой

$$P_Q(x) = \arg\min \left\{ \|x - q\| \,|q \in Q \right\}. \tag{132}$$

Следующее утверждение имеет место.

Утверждение 7. Последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3 с метрической проекцией $P_M(\cdot)$ вместо псевдопроекции $\rho_M(\cdot)$, завершается через конечное число итераций $K \geqslant 0$ в некоторой допустимой точке, причем

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \dots < \left\langle c, u^{(K)} \right\rangle. \tag{133}$$

Доказательство. Данное утверждение доказывается по той же схеме, что и утверждение 6. □

Следующее утверждение доказывает сходимость алгоритма 3 к точному решению задачи ЛП (37) для случая метрической проекции.

Утверждение 8. При замене псевдопроекции $\rho_M(\cdot)$ метрической проекцией $P_M(\cdot)$ алгоритм 3 завершается через конечное число итераций в точке $\bar{x}$, являющейся точным решением задачи ЛП (37).

Доказательство. Обозначим через $\bar{u}$ конечную точку последовательности $\left\{u^{(k)}\right\}$, генерируемой алгоритмом 3 с использованием метрической проекции $P_M(\cdot)$. Такая точка существует в силу утверждения 7. Предположим противное, то есть $\bar{u} \neq \bar{x}$. Это равносильно

$$\langle c, \bar{u} \rangle < \langle c, \bar{x} \rangle . \tag{134}$$

Обозначим с помощью $S_\delta(v)$ открытый n-мерный шар радиуса δ с центром в точке v, где

$$v = \bar{u} + \delta e_c. \tag{135}$$

В силу (134) имеем

$$S_\delta(v) \cap M \neq \emptyset. \tag{136}$$

Положим

$$w = \arg\min \{ \|x - v\| \, | x \in S_\delta(v) \cap M \} . \tag{137}$$

Последнее эквивалентно

$$w = P_M(v). \tag{138}$$

Легко видеть, что справедливо неравенство

$$\langle c, w \rangle > \langle c, \bar{u} \rangle . \tag{139}$$

Сопоставляя формулу (135) с шагом 11 алгоритма 3, формулу (138) с шагом 12 (где псевдопроекция заменена на метрическую проекцию), и формулу (139) с условием на шаге 5, мы видим, что $\bar{u}$ не может быть конечной точкой последовательности $\left\{u^{(k)}\right\}$, генерируемой алгоритмом 3. Получили противоречие. *Утверждение доказано.* □

На практике заменить псевдопроекцию $\rho_M(v)$ в алгоритме 3 на метрическую проекцию $P_M(v)$ не представляется возможным, так как неизвестен алгоритм вычисления метрической проекции на выпуклый замкнутый многогранник в общем случае. Таким образом, утверждение 8 в строгом смысле не доказывает сходимость алгоритма 3 к точному решению задачи ЛП (37), хотя на практике такая сходимость наблюдалась нами во всех случаях.

4. Программная реализация и вычислительные эксперименты

Мы реализовали параллельную версию апекс-метода на языке C++ с использованием программного BSF-каркаса [67], базирующегося на модели параллельных вычислений BSF [66]. BSF-каркас инкапсулирует все аспекты, связанные с распараллеливанием программы на основе библиотеки MPI. Исходные коды апекс-метода свободно доступны в репозитории GitHub по адресу `https://github.com/leonid-sokolinsky/Apex-method`. С помощью этой программы мы исследовали масштабируемость апекс-метода. Масштабные вычислительные эксперименты проводились на вычислительном кластере «Торнадо ЮУрГУ» [68], характеристики которого представлены в табл. 1. В качестве тестов мы использовали искусственные задачи, полученные с помощью генератора случайных задач линейного программирования FRaGenLP [69]. Верификация решений, выдаваемых апекс-методом, осуществлялась программой VaLiPro [70]. Была выполнена серия вычислительных экспериментов, в которой для задач ЛП различной размерности исследовались ускорение и параллельная эффективность в зависимости от количества используемых рабочих узлов. Результаты этих экспериментов представлены на рис. 2. В данном контексте ускорение $\alpha(L)$ опре-

Таблица 1. Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 cores, 3.33 GHz)
Число процессоров на узел	2
Память на узел	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

делялось как отношение времени $T(1)$ решения задачи на конфигурации с узлом-мастером и единственным узлом-рабочим ко времени $T(L)$ решения той же задачи на конфигурации с узлом-мастером и L узлами-рабочими:

$$\alpha(L) = \frac{T(1)}{T(L)}. \tag{140}$$

Параллельная эффективность $\epsilon(L)$ вычислялась как отношение ускорения $\alpha(L)$ к числу L используемых узлов-рабочих:

$$\epsilon(L) = \frac{\alpha(L)}{L}. \tag{141}$$

Вычисления проводились для следующих размерностей: 5 000, 7 500 и 10 000. Число ограничений соответственно составило 10 002, 15 002 и 20 002.

Эксперименты показали, что граница масштабируемости параллельной реализации апекс-метода существенно зависит от размера задачи. Для $n = 5\,000$ граница масштабируемости составила приблизительно 55 рабочих узлов. Для задачи размерности $n = 7\,500$ эта граница увеличилась до 80 узлов, а для задачи размерности $n = 10\,000$ она оказалась близкой к 100 узлам. Дальнейшее увеличение размерности задачи приводило к ошибке компилятора «недостаточно памяти». Необходимо отметить, что вычисления проводились с двойной точностью, при которой число с плавающей точкой занимает в оперативной памяти 64 бита. Попытка использовать одинарную точность, требующую 32 бита для хранения числа с плавающей точкой, оказалась неудачной, так как при этом апекс-метод переставал сходиться к точному решению задачи ЛП.

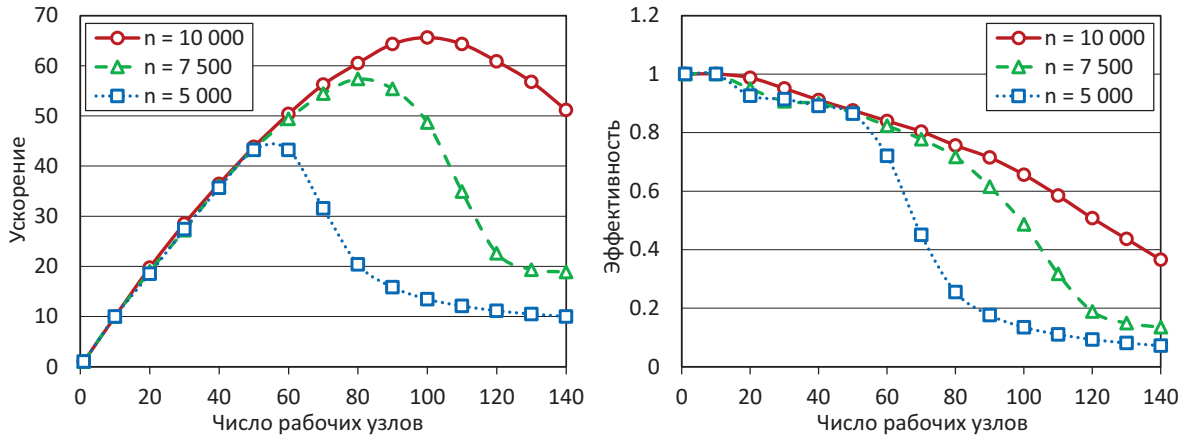


Рис. 2. Ускорение и параллельная эффективность апекс-метода

Параллельная эффективность также продемонстрировала существенную зависимость от размера задачи ЛП. При $n = 5\,000$ эффективность показала падение ниже 50% уже на 70 рабочих узлах. Для $n = 7\,500$ и $n = 10\,000$ падение на 50% наблюдалось на 110 и 130 рабочих узлах соответственно.

Кроме этого, эксперименты показали, что параметр η в формуле (116), используемой на стадии Quest для вычисления точки апекса z, оказывает незначительное влияние на общее время решения задачи ЛП в случае, когда этот параметр принимает большие значения (более 100 000). Если точка апекса располагается недостаточно далеко от допустимого многогранника, то ее псевдопроекция может оказаться на одной из его граней. Если же точка апекса располагается далеко от допустимого многогранника (в экспериментах использовалось значение $\eta = 20000n$), то ее псевдопроекция всегда оказывается в одной из его вершин. Также стоит отметить, что для искусственных задач, сгенерированных программой FRaGenLP, все точки последовательности $\left\{u^{(k)}\right\}$ оказывались на пути, близком к оптимальному[2].

Проведенные вычислительные эксперименты на искусственных задачах показали, что более 99% времени апекс-метод тратил на вычисление псевдопроекций (шаг 18 алгоритма 3). При этом вычисление одного приближения $u^{(k)}$ для задачи размерности $n = 10\,000$ на 100 рабочих узлах занимало 44 минуты.

Мы также протестировали апекс-метод на задачах из репозитория Netlib-LP [71], доступного по адресу `https://netlib.org/lp/data`. Набор задач линейной оптимизации Netlib-LP включает в себя множество реальных приложений, таких как оценка лесных ресурсов, задачи нефтепереработки, проектирование закрылков самолетов, модели пилотирования, планирование работы аудиторского персонала, расчет мостовых ферм, планирование расписаний авиакомпаний, расчет моделей промышленного производства и распределения ресурсов, восстановление изображений и задачи многосекторального экономического планирования. Netlib-LP содержит задачи ЛП размером от 32 переменных и 27 ограничений до 15 695 переменных и 16 675 ограничений [72]. Точные решения (оптимальные значения целевых функций) для всех задач были заимствованы из работы [73]. Результаты представлены в табл. 2. Эксперименты показали, что относительная ошибка грубого приближения, вычисляемого на стадии Quest, не превосходила 0.2 для всех задач, кроме *adlittle*, *blend*, и *fit1d*. Относительная ошибка уточненного приближения, получаемого на стадии Target, оказалась менее 10^{-3}, за исключением задач *kb2* и *sc105*, для которых ошибка составила 0.035 и 0.007 соответственно. Время решения указанных задач варьировалось от нескольких секунд для *afiro* до десятков часов для *blend*. Одним из главных параметров, влияющих на скорость сходимости апекс-метода, был параметр ϵ, используемый на шаге 12 параллельного алгоритма 2, вычисляющего псевдопроекцию. Прогоны всех задач доступны на GitHub по адресу `https://github.com/leonid-sokolinsky/Apex-method/tree/master/Runs`.

5. Обсуждение полученных результатов

В этом разделе мы обсудим научную и практическую значимость апекс-метода, его сильные и слабые стороны, и дадим ответы на следующие вопросы.

1. В чем состоит научная значимость полученных результатов?
2. В чем заключается практическое значение апекс-метода?

[2]Под оптимальным путем понимается путь движения по поверхности допустимого многогранника в направлении максимального, в данном случае, увеличения значения целевой функции.

Таблица 2. Применение апекс-метода для решения задач из Netlib-LP

№	Задача из Netlib-LP		Стадия Quest		Стадия Target	
	Наименование	Точное решение	Грубое приближение	Ошибка	Уточненное приближение	Ошибка
1	adlittle	2.25494963E5	3.67140280E5	6.28E-1	2.2571324E5	9.68E-4
2	afiro	-4.64753142E2	-4.55961488E2	1.89E-2	-4.6475310E2	8.61E-9
3	blend	-3.08121498E1	-3.60232513E0	8.83E-1	-3.0811018E1	3.19E-5
4	fit1d	-9.14637809E3	-3.49931014E3	6.17E-1	-9.1463386E3	8.77E-7
5	kb2	-1.74990012E3	-1.39603193E3	2.02E-1	-1.6879152E3	3.54E-2
6	recipe	-2.66616000E2	-2.66107349E2	1.91E-3	-2.6660404E2	2.23E-5
7	sc50a	-6.45750770E1	-5.58016335E1	1.36E-1	-6.4568167E1	1.06E-4
8	sc50b	-7.00000000E1	-6.92167246E1	1.12E-2	-6.9990792E1	1.32E-4
9	sc105	-5.22020612E1	-4.28785710E1	1.79E-1	-5.1837995E1	6.97E-3
10	share2b	-4.15732240E2	-4.28792528E2	3.14E-2	-4.1572001E2	2.40E-5

3. На сколько мы можем быть уверены, что апекс-метод всегда сходится к точному решению задачи ЛП?
4. Как мы можем ускорить сходимость апекс-метода в целом и выполнение операции псевдопроектирования в частности?

Основной научный вклад этой работы заключается в том, что разработан апекс-метод, впервые позволяющий построить на поверхности допустимого многогранника близкий к оптимальному путь от начальной точки до точки решения задачи ЛП. Под оптимальным путем мы понимаем путь движения по поверхности многогранника в направлении максимального увеличения или уменьшения значения целевой функции в зависимости от того, ее максимум или минимум необходимо найти.

Практическая значимость апекс-метода состоит в том, что он открывает возможность использования искусственных нейронных сетей прямого распространения, включая сверточные нейронные сети, для решения многомерных задач ЛП. В недавней работе [48] был предложен оригинальный метод визуализации n-мерных задач ЛП. Этот метод строит образ допустимого многогранника M в виде матрицы I размерности $(n-1)$ на основе техники растеризации. В качестве луча зрения используется вектор, противоположно направленный вектору градиента целевой функции. Каждый пиксель представляется в матрице I вещественным числом, пропорциональным значению целевой функции в соответствующей точке на поверхности допустимого многогранника M. Подобные образы подаются на вход нейронной сети прямого распространения для нахождения оптимального пути к решению задачи ЛП. Более точно, нейронная сеть прямого распространения непосредственно вычисляет вектор d в алгоритме 3, делая ненужным ресурсоемкие вычисления псевдопроекций. Главным преимуществом такого подхода является то, что нейронная сеть прямого распространения работает в режиме реального времени, актуальном для задач робототехники. В настоящее время нам не известны другие методы решения задач ЛП, работающие в режиме реального времени. Однако применение нейронных сетей прямого распространения для решения задач ЛП предполагает подготовку обучающих наборов данных. Апекс-метод впервые предоставляет возможность конструировать такие обучающие наборы.

В утверждении 6 говорится, что алгоритм 3 сходится за конечное число итераций к некоторой точке на поверхности допустимого многогранника M, однако вопрос о том, будет ли эта точка решением задачи ЛП, остается открытым. Согласно утверждению 8 ответ на этот вопрос оказывается положительным, если в алгоритме 3 заменить псевдопроекцию на метрическую проекцию. Однако не существует методов построения метрической проекции для произвольного выпуклого замкнутого многогранника. Поэтому мы вынуждены использовать псевдопроекцию. Многочисленные эксперименты показывают, что апекс-метод всегда сходится к решению задачи ЛП, однако этот факт нуждается в формальном доказательстве. Мы планируем получить такое доказательство в рамках наших дальнейших исследований.

Основным недостатком апекс-метода является его медленная сходимость к решению задачи ЛП. Задача ЛП, которая занимает несколько секунд для нахождения оптимального решения с использованием одного из стандартных решателей, может потребовать нескольких часов для нахождения решения с помощью апекс-метода. Вычислительные эксперименты показывают, что более 99% времени, затрачиваемого апекс-методом на решение задачи ЛП, приходится на вычисление псевдопроекций. Поэтому вопрос ускорения процесса вычисления псевдопроекций является актуальным. В апекс-методе псевдопроекции вычисляются с помощью алгоритма 1, принадлежащего к семейству проекционных методов, рассмотренных в разделе 1. Известно [74], что в случае, когда выпуклое замкнутое множество является многогранником $M \neq \emptyset$, методы проекционного типа имеют низкую линейную скорость сходимости:

$$\left\| x^{(k+1)} - \rho_M \left(x^{(0)} \right) \right\| \leqslant C q^k, \tag{142}$$

где $0 < C < \infty$ — некоторая константа, а $q \in (0, 1)$ — параметр, зависящий от углов между гиперплоскостями, соответствующими граням многогранника M. Это означает, что расстояние между соседними приближениями с каждой итерацией уменьшается в геометрической прогрессии со знаменателем, меньшим единицы. Для малых углов скорость сходимости может падать до значений, близких к нулю. Это фундаментальное ограничение методов проекционного типа не может быть преодолено. Однако, мы можем уменьшить количество гиперплоскостей, вовлекаемых в вычисление псевдопроекции. В соответствии с утверждением 3 решение задачи ЛП (37) находится на границе некоторого рецессивного полупространства. Следовательно, при вычислении псевдопроекции с помощью алгоритма 1, нам достаточно использовать только гиперплоскости, ограничивающие рецессивные полупространства. Это уменьшает количество ограничений в среднем в два раза. Другой способ сократить время вычисления псевдопроекций — распараллелить алгоритм 1, как это было сделано в алгоритме 2. Однако в этом случае степень параллелизма будет ограничена теоретической оценкой (114).

Заключение

В статье предложен новый масштабируемый итерационный метод линейного программирования, получивший название «апекс-метод». Ключевой особенностью этого метода является построение максимального пути на поверхности допустимого многогранника от начальной токи к решению задачи линейного программирования. Под максимальным путем понимается путь движения по поверхности допустимого многогранника в направлении максимального увеличения значения целевой функции. Практическая значимость предложенного метода состоит в том, что он открывает возможность применения искусственных

нейронных сетей прямого распространения для решения многомерных задач линейного программирования.

В работе описан оригинальный теоретический базис, лежащий в основе апекс-метода. Рассмотрены полупространства, порождаемые ограничениями задачи линейного программирования. Пересечение этих полупространств образует замкнутый выпуклый многогранник M, называемый допустимым. Указанные полупространства делятся на две группы, в зависимости от градиента c линейной целевой функции: доминантные и рецессивные. Получено достаточное и необходимое условие для того, чтобы полупространство было рецессивным.

Доказано, что решение задачи линейного программирования всегда лежит на границе некоторого рецессивного полупространства. Получена формула вычисления точки апекса, которая не принадлежит ни одному рецессивному полупространству. Точка апекса используется для получения начального приближения на поверхности допустимого многогранника M. Для построения оптимального пути к решению задачи линейного программирования апекс-метод использует параллельный алгоритм построения псевдопроекции, являющейся обобщением метрической проекции. Для параллельного алгоритма построения псевдопроекции получена аналитическая оценка границы его масштабируемости на кластерной вычислительной системе. Эта граница не превышает $O\left(\sqrt{m}\right)$ процессорных узлов, где m — количество ограничений задачи линейного программирования. Описан алгоритм, строящий на границе допустимого многогранника оптимальный путь от начального приближения до точки решения задачи линейного программирования. Доказана сходимость этого алгоритма.

Параллельная версия апекс-метода реализована на языке C++ с использованием программного BSF-каркаса, основанного на модели параллельных вычислений BSF. Проведены эксперименты по исследованию масштабируемости апекс-метода на кластерной вычислительной системе. Вычислительные эксперименты показали, что для задачи линейного программирования с 10 000 переменными и 20 002 ограничениями граница масштабируемости не превышает 100 процессорных узлов. В то же время эксперименты показали, что более 99% времени, затрачиваемого на решение задачи линейного программирования апекс-методом, приходилось на вычисление псевдопроекций.

В дополнение, апекс-метод был протестирован на 10 задачах из репозитория Netlib-LP. Относительная ошибка на этих задачах составила от $3.5{\cdot}10^{-3}$ до $8.6{\cdot}10^{-9}$. Время вычислений варьировалось от нескольких секунд до нескольких десятков часов. Точность вычисления псевдопроекции оказалась основным параметром, влияющим на скорость сходимости апекс-метода.

В качестве направлений дальнейших исследований выделим следующие. Мы планируем разработать новый, более эффективный метод вычисления псевдопроекций на допустимый многогранник. Основная идея состоит в сокращении количества полупространств, используемых в рамках одной итерации. В то же время количество оставшихся в рассмотрении полупространств должно быть достаточным для эффективного распараллеливания. Мы рассчитываем, что новый метод превзойдет алгоритм 2 по скорости сходимости. Также мы собираемся доказать, что новый метод сходится к точке, лежащей на границе допустимого многогранника. Кроме этого мы планируем исследовать полезность использования в апекс-методе техники линейной супериоризации, предложенной в работе [64].

Обозначения

$\mathbb{R}^n$	вещественное евклидово пространство
$\|\cdot\|$	евклидова норма
$\langle\cdot,\cdot\rangle$	скалярное произведение двух векторов
$[\cdot\,,\cdot]$	конкатенация двух векторов
$f(x)$	линейная целевая функция
c	градиент целевой функции $f(x)$
e_c	единичный вектор, сонаправленный с вектором c
$\bar{x}$	решение задачи ЛП
M	допустимый многогранник
Γ_M	множество граничных точек допустимого многогранника M
a_i	i-тая строка матрицы A
$\hat{H}_i$	полупространство, определяемое формулой $\langle a_i, x\rangle \leqslant b_i$
H_i	гиперплоскость, определяемая формулой $\langle a_i, x\rangle = b_i$
$\mathcal{P}$	множество индексов строк матрицы A
$\mathcal{I}$	множество индексов, для которых полупространство $\hat{H}_i$ является рецессивным
$\pi_i(\cdot)$	ортогональная проекция на гиперплоскость H_i
$\rho_M(\cdot)$	псевдопроекция на допустимый многогранник M
$P_M(\cdot)$	метрическая проекция на допустимый многогранник M

Исследование выполнено при финансовой поддержке РНФ (проект №23-21-00356).

Литература

1. Соколинская И.М., Соколинский Л.Б. Об одном итерационном методе решения задач линейного программирования на кластерных вычислительных системах // Вычислительные методы и программирование. 2020. Т. 21, № 4. С. 329–340. DOI: 10.26089/NumMet.v21r328.

2. Branke J. Optimization in Dynamic Environments // Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Boston, MA: Springer, 2002. P. 13–29. DOI: 10.1007/978-1-4615-0911-0_2.

3. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery // Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.

4. Deng S., Huang X., Wang J., *et al.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion // IEEE Access. 2021. Vol. 9. P. 1271–1285. DOI: 10.1109/ACCESS.2020.3047138.

5. Seregin G. Lecture notes on regularity theory for the Navier-Stokes equations. Singapore: World Scientific Publishing Company, 2014. 268 p. DOI: 10.1142/9314.

6. Demin D.A. Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state // Eastern-European Journal of Enterprise Technologies. 2017. Vol. 3, 4(87). P. 51–63. DOI: 10.15587/1729-4061.2017.105294.

7. Kazarinov L.S., Shnayder D.A., Kolesnikova O.V. Heat load control in steam boilers // 2017 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2017 - Proceedings. IEEE, 2017. DOI: 10.1109/ICIEAM.2017.8076177.

8. Zagoskina E.V., Barbasova T.A., Shnaider D.A. Intelligent Control System of Blast-furnace Melting Efficiency // SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings. IEEE, 2019. P. 710–713. DOI: 10.1109/SIBIRCON48586.2019.8958221.

9. Fleming J., Yan X., Allison C., *et al.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements // IET Intelligent Transport Systems. 2021. Vol. 15, no. 4. P. 573–583. DOI: 10.1049/ITR2.12047.

10. Scholl M., Minnerup K., Reiter C., *et al.* Optimization of a thermal management system for battery electric vehicles // 14th International Conference on Ecological Vehicles and Renewable Energies, EVER 2019. IEEE, 2019. DOI: 10.1109/EVER.2019.8813657.

11. Meisel S. Dynamic Vehicle Routing // Anticipatory Optimization for Dynamic Decision Making. Operations Research/Computer Science Interfaces Series, vol. 51. New York, NY: Springer, 2011. P. 77–96. DOI: 10.1007/978-1-4614-0505-4_6.

12. Cheng A.M.K. Real-Time Scheduling and Schedulability Analysis // Real-Time Systems: Scheduling, Analysis, and Verification. John Wiley, Sons, 2002. P. 41–85. DOI: 10.1002/0471224626.CH3.

13. Kopetz H. Real-Time Scheduling // Real-Time Systems. Real-Time Systems Series. Boston, MA: Springer, 2011. P. 239–258. DOI: 10.1007/978-1-4419-8237-7_10.

14. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.

15. Hall J., McKinnon K. Hyper-sparsity in the revised simplex method and how to exploit it // Computational Optimization and Applications. 2005. Vol. 32, no. 3. P. 259–283. DOI: 10.1007/s10589-005-4802-0.

16. Klee V., Minty G. How good is the simplex algorithm? // Inequalities - III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. New York-London: Academic Press, 1972. P. 159–175.

17. Jeroslow R. The simplex algorithm with the pivot rule of maximizing criterion improvement // Discrete Mathematics. 1973. Vol. 4, no. 4. P. 367–377. DOI: 10.1016/0012-365X(73)90171-4.

18. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms // Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.

19. Bartels R., Stoer J., Zenger C. A Realization of the Simplex Method Based on Triangular Decompositions // Handbook for Automatic Computation. Volume II: Linear Algebra. Berlin, Heidelberg: Springer, 1971. P. 152–190. DOI: 10.1007/978-3-642-86940-2_11.

20. Tolla P. A Survey of Some Linear Programming Methods // Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: John Wiley, Sons, 2014. Chap. 7. P. 157–188. DOI: 10.1002/9781119005216.ch7.

21. Hall J. Towards a practical parallelisation of the simplex method // Computational Management Science. 2010. Vol. 7, no. 2. P. 139–170. DOI: 10.1007/s10287-008-0080-5.

22. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method // Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307. DOI: 10.1007/978-3-319-24024-4_17.

23. Lubin M., Hall J.A.J., Petra C.G., Anitescu M. Parallel distributed-memory simplex for large-scale stochastic LP problems // Computational Optimization and Applications. 2013. Vol. 55, no. 3. P. 571–596. DOI: 10.1007/s10589-013-9542-y.

24. Хачиян Л. Полиномиальные алгоритмы в линейном программировании // Журнал вычислительной математики и математической физики. 1980. Т. 20, № 1. С. 51–68.

25. Шор Н.З. Метод отсечения с растяжением пространства для решения задач выпуклого программирования // Кибернетика. 1977. № 1. С. 94–95.

26. Юдин Д.Б., Немировский А.С. Информационная сложность и эффективные методы решения выпуклых экстремальных задач // Экономика и математические методы. 1976. № 2. С. 357–369.

27. Karmarkar N. A new polynomial-time algorithm for linear programming // Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.

28. Дикин И.И. Итеративное решение задач линейного и квадратичного программирования // Доклады Академии наук СССР. 1967. Т. 174, № 4. С. 747–748. URL: https://www.mathnet.ru/rus/dan33112.

29. Gondzio J. Interior point methods 25 years later // European Journal of Operational Research. 2012. Vol. 218, no. 3. P. 587–601. DOI: 10.1016/j.ejor.2011.09.017.

30. Зоркальцев В.И., Мокрый И.В. Алгоритмы внутренних точек в линейной оптимизации // Сибирский журнал индустриальной математики. 2018. Т. 21, 1 (73). С. 11–20.

31. Fathi-Hafshejani S., Mansouri H., Reza Peyghami M., Chen S. Primal–dual interior-point method for linear optimization based on a kernel function with trigonometric growth term // Optimization. 2018. Vol. 67, no. 10. P. 1605–1630. DOI: 10.1080/02331934.2018.1482297.

32. Asadi S., Mansouri H. A Mehrotra type predictor-corrector interior-point algorithm for linear programming // Numerical Algebra, Control and Optimization. 2019. Vol. 9, no. 2. P. 147–156. DOI: 10.3934/naco.2019011.

33. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs // Proc. SPIE, vol. 8285. International Conference on Graphic and Image Processing (ICGIP 2011). International Society for Optics, Photonics, 2011. DOI: 10.1117/12.913019.

34. Kheirfam B., Haghighi M. A full-Newton step infeasible interior-point method for linear optimization based on a trigonometric kernel function // Optimization. 2016. Vol. 65, no. 4. P. 841–857. DOI: 10.1080/02331934.2015.1080255.

35. Xu Y., Zhang L., Zhang J. A full-modified-newton step infeasible interior-point algorithm for linear optimization // Journal of Industrial and Management Optimization. 2016. Vol. 12, no. 1. P. 103–116. DOI: 10.3934/jimo.2016.12.103.

36. Roos C., Terlaky T., Vial J.-P. Interior Point Methods for Linear Optimization. New York: Springer, 2005. 500 p. DOI: 10.1007/b100325.

37. Sokolinskaya I. Parallel Method of Pseudoprojection for Linear Inequalities // Parallel Computational Technologies. PCT 2018. Communications in Computer and Information Science, vol. 910 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2018. P. 216–231. DOI: 10.1007/978-3-319-99673-8_16.

38. Васин В.В., Ерёмин И.И. Операторы и итерационные процессы фейеровского типа. Теория и приложения. Екатеринбург: УрО РАН, 2005. 211 с.

39. Gondzio J., Grothey A. Direct Solution of Linear Systems of Size 109 Arising in Optimization with Interior Point Methods // Parallel Processing and Applied Mathematics. PPAM 2005. Lecture Notes in Computer Science, vol. 3911. 3911 LNCS / ed. by R. Wyrzykowski, J. Dongarra, N. Meyer, J. Wasniewski. Berlin, Heidelberg: Springer, 2006. P. 513–525. DOI: 10.1007/11752578_62.

40. Prieto A., Prieto B., Ortigosa E.M., *et al.* Neural networks: An overview of early research, current frameworks and new challenges // Neurocomputing. 2016. Vol. 214. P. 242–268. DOI: 10.1016/j.neucom.2016.06.014.

41. Raina R., Madhavan A., Ng A.Y. Large-scale deep unsupervised learning using graphics processors // Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). New York, NY, USA: ACM Press, 2009. P. 873–880. DOI: 10.1145/1553374.1553486.

42. Tank D.W., Hopfield J.J. Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit // IEEE transactions on circuits and systems. 1986. Vol. CAS-33, no. 5. P. 533–541. DOI: 10.1109/TCS.1986.1085953.

43. Kennedy M.P., Chua L.O. Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin // IEEE Transactions on Circuits and Systems. 1987. Vol. 34, no. 2. P. 210–214. DOI: 10.1109/TCS.1987.1086095.

44. Rodriguez-Vazquez A., Dominguez-Castro R., Rueda A., *et al.* Nonlinear Switched-Capacitor "Neural" Networks for Optimization Problems // IEEE Transactions on Circuits and Systems. 1990. Vol. 37, no. 3. P. 384–398. DOI: 10.1109/31.52732.

45. Zak S.H., Upatising V. Solving Linear Programming Problems with Neural Networks: A Comparative Study // IEEE Transactions on Neural Networks. 1995. Vol. 6, no. 1. P. 94–104. DOI: 10.1109/72.363446.

46. Malek A., Yari A. Primal–dual solution for the linear programming problems using neural networks // Applied Mathematics and Computation. 2005. Vol. 167, no. 1. P. 198–211. DOI: 10.1016/J.AMC.2004.06.081.

47. Liu X., Zhou M. A one-layer recurrent neural network for non-smooth convex optimization subject to linear inequality constraints // Chaos, Solitons and Fractals. 2016. Vol. 87. P. 39–46. DOI: 10.1016/j.chaos.2016.03.009.

48. Ольховский Н.А., Соколинский Л.Б. Визуальное представление многомерных задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 31–56. DOI: 10.14529/cmse220103.

49. LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. 2015. Vol. 521, no. 7553. P. 436–444. DOI: 10.1038/nature14539.

50. Lachhwani K. Application of Neural Network Models for Mathematical Programming Problems: A State of Art Review // Archives of Computational Methods in Engineering. 2020. Vol. 27. P. 171–182. DOI: 10.1007/s11831-018-09309-5.

51. Поляк Б.Т. Рандомизированные алгоритмы решения выпуклых неравенств // Стохастическая оптимизация в информатике / под ред. О.Н. Граничин. СПб.: Издательство С.-Петербургского университета, 2005. С. 123–127. URL: https://www.math.spbu.ru/user/gran/sb1/polyak.pdf.

52. Kaczmarz S. Angenherte Auflsung von Systemen linearer Gleichungen // Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres. Classe des Sciences Mathmatiques et Naturelles. Srie A, Sciences Mathmatiques. 1937. Vol. 35. P. 355–357.

53. Kaczmarz S. Approximate solution of systems of linear equations // International Journal of Control. 1993. Vol. 57, no. 6. P. 1269–1271. DOI: 10.1080/00207179308934446.

54. Cimmino G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari // La Ricerca Scientifica, XVI, Series II, Anno IX, 1. 1938. P. 326–333.

55. Gastinel N. Linear Numerical Analysis. New York: Academic Press, 1971. ix+341 p.

56. Agmon S. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. DOI: 10.4153/CJM-1954-037-2.

57. Motzkin T.S., Schoenberg I.J. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 393–404. DOI: 10.4153/CJM-1954-038-x.

58. Censor Y., Elfving T. New methods for linear inequalities // Linear Algebra and its Applications. 1982. Vol. 42. P. 199–211. DOI: 10.1016/0024-3795(82)90149-5.

59. De Pierro A.R., Iusem A.N. A simultaneous projections method for linear inequalities // Linear Algebra and its Applications. 1985. Vol. 64. P. 243–253. DOI: 10.1016/0024-3795(85)90280-0.

60. Соколинская И., Соколинский Л. Исследование масштабируемости алгоритма Чиммино для решения систем линейных неравенств на кластерных вычислительных системах // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 1. С. 20–35. DOI: 10.14529/cmse190102.

61. Соколинский Л.Б., Соколинская И.М. Параллельный алгоритм решения нестационарных систем линейных неравенств // Параллельные вычислительные технологии – XIV международная конференция, ПаВТ'2020, г. Пермь, 31 марта–2 апреля 2020 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2020. С. 275–286. DOI: 10.14529/pct2020.

62. Ерёмин И.И., Попов Л.Д. Фейеровские процессы в теории и практике: обзор последних результатов // Известия вузов. Математика. 2009. № 1. С. 44–65. URL: https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ivm&paperid=1253.

63. Nurminski E.A. Single-projection procedure for linear optimization // Journal of Global Optimization. 2016. Vol. 66, no. 1. P. 95–110. DOI: 10.1007/S10898-015-0337-9.

64. Censor Y. Can linear superiorization be useful for linear optimization problems? // Inverse Problems. 2017. Vol. 33, no. 4. P. 044006. DOI: 10.1088/1361-6420/33/4/044006.

65. Gould N.I. How good are projection methods for convex feasibility problems? // Computational Optimization and Applications. 2008. Vol. 40, no. 1. P. 1–12. DOI: 10.1007/S10589-007-9073-5.

66. Sokolinsky L.B. BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems // Journal of Parallel and Distributed Computing. 2021. Vol. 149. P. 193–206. DOI: 10.1016/j.jpdc.2020.12.009.

67. Sokolinsky L.B. BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems // MethodsX. 2021. Vol. 8. Article number 101437. DOI: 10.1016/j.mex.2021.101437.

68. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University // Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

69. Соколинский Л.Б., Соколинская И.М. О генерации случайных задач линейного программирования на кластерных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 38–52. DOI: 10.14529/cmse210103.

70. Соколинский Л.Б., Соколинская И.М. О валидации решений задач линейного программирования на кластерных вычислительных системах // Вычислительные методы и программирование. 2021. Т. 22, № 4. С. 252–261. DOI: 10.26089/NUMMET.V22R416.

71. Gay D.M. Electronic mail distribution of linear programming test problems // Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

72. Keil C., Jansson C. Computational experience with rigorous error bounds for the Netlib linear programming library // Reliable Computing. 2006. Vol. 12, no. 4. P. 303–321. DOI: 10.1007/S11155-006-9004-7/METRICS.

73. Koch T. The final NETLIB-LP results // Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI: 10.1016/S0167-6377(03)00094-4.

74. Deutsch F., Hundal H. The rate of convergence for the cyclic projections algorithm I: Angles between convex sets // Journal of Approximation Theory. 2006. Vol. 142, no. 1. P. 36–55. DOI: 10.1016/J.JAT.2006.02.005.

Соколинский Леонид Борисович, д.ф.-м.н., профессор, заведующий кафедрой системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинская Ирина Михайловна, к.ф.-м.н., доцент кафедры математического обеспечения информационных технологий, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse230201

ON NEW VERSION OF THE APEX METHOD FOR SOLVING LINEAR PROGRAMMING PROBLEMS

© 2023 L.B. Sokolinsky, I.M. Sokolinskaya

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: leonid.sokolinsky@susu.ru, irina.sokolinskaya@susu.ru

Received: 07.04.2023

The article presents a new scalable iterative method for linear programming, called the apex method. The key feature of this method is constructing a path close to optimal on the surface of the feasible region from a certain starting point to the exact solution of the linear programming problem. The optimal path refers to a path of minimum length according to the Euclidean metric. The apex method is based on the predictor-corrector framework and proceeds in two stages: Quest (predictor) and Target (corrector). The Quest stage calculates a rough initial approximation of the linear programming problem. The Target stage refines the initial approximation with a given precision. The main operation used in the apex method is an operation that calculates the pseudoprojection, which is a generalization of the metric projection to a convex closed set. This operation is used both in the Quest stage and in the Target stage. A parallel algorithm using a Fejér mapping to compute the pseudoprojection is presented. An analytical estimation of the parallelism degree of this algorithm is obtained. Also, an algorithm implementing the Target stage is given. The convergence of this algorithm is proven. The results of applying the apex method for solving various linear programming problems are presented.

Keywords: linear programming, apex method, iterative method, projection-type method, Fejér mapping, parallel algorithm, scalability evaluation.

FOR CITATION

Sokolinsky L.B., Sokolinskaya I.M. On New Version of the Apex Method for Solving Linear Programming Problems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 2. P. 5–46. (in Russian) DOI: 10.14529/cmse230201.

References

1. Sokolinsky L.B., Sokolinskaya I.M. Scalable Method for Linear Optimization of Industrial Processes. Proceedings - 2020 Global Smart Industry Conference, GloSIC 2020. IEEE, 2020. 20–26. Article number 9267854. DOI: 10.1109/GloSIC50886.2020.9267854.
2. Branke J. Optimization in Dynamic Environments. Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Boston, MA: Springer, 2002. P. 13–29. DOI: 10.1007/978-1-4615-0911-0_2.
3. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery. Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.
4. Deng S., Huang X., Wang J., *et al.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion. IEEE Access. 2021. Vol. 9. P. 1271–1285. DOI: 10.1109/ACCESS.2020.3047138.

5. Seregin G. Lecture notes on regularity theory for the Navier-Stokes equations. Singapore: World Scientific Publishing Company, 2014. 268 p. DOI: 10.1142/9314.

6. Demin D.A. Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state. Eastern-European Journal of Enterprise Technologies. 2017. Vol. 3, 4(87). P. 51–63. DOI: 10.15587/1729-4061.2017.105294.

7. Kazarinov L.S., Shnayder D.A., Kolesnikova O.V. Heat load control in steam boilers. 2017 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2017 - Proceedings. IEEE, 2017. DOI: 10.1109/ICIEAM.2017.8076177.

8. Zagoskina E.V., Barbasova T.A., Shnaider D.A. Intelligent Control System of Blast-furnace Melting Efficiency. SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings. IEEE, 2019. P. 710–713. DOI: 10.1109/SIBIRCON48586.2019.8958221.

9. Fleming J., Yan X., Allison C., *et al.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements. IET Intelligent Transport Systems. 2021. Vol. 15, no. 4. P. 573–583. DOI: 10.1049/ITR2.12047.

10. Scholl M., Minnerup K., Reiter C., *et al.* Optimization of a thermal management system for battery electric vehicles. 14th International Conference on Ecological Vehicles and Renewable Energies, EVER 2019. IEEE, 2019. DOI: 10.1109/EVER.2019.8813657.

11. Meisel S. Dynamic Vehicle Routing. Anticipatory Optimization for Dynamic Decision Making. Operations Research/Computer Science Interfaces Series, vol. 51. New York, NY: Springer, 2011. P. 77–96. DOI: 10.1007/978-1-4614-0505-4_6.

12. Cheng A.M.K. Real-Time Scheduling and Schedulability Analysis. Real-Time Systems: Scheduling, Analysis, and Verification. John Wiley, Sons, 2002. P. 41–85. DOI: 10.1002/0471224626.CH3.

13. Kopetz H. Real-Time Scheduling. Real-Time Systems. Real-Time Systems Series. Boston, MA: Springer, 2011. P. 239–258. DOI: 10.1007/978-1-4419-8237-7_10.

14. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.

15. Hall J., McKinnon K. Hyper-sparsity in the revised simplex method and how to exploit it. Computational Optimization and Applications. 2005. Vol. 32, no. 3. P. 259–283. DOI: 10.1007/s10589-005-4802-0.

16. Klee V., Minty G. How good is the simplex algorithm?. Inequalities - III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. New York-London: Academic Press, 1972. P. 159–175.

17. Jeroslow R. The simplex algorithm with the pivot rule of maximizing criterion improvement. Discrete Mathematics. 1973. Vol. 4, no. 4. P. 367–377. DOI: 10.1016/0012-365X(73)90171-4.

18. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms. Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.

19. Bartels R., Stoer J., Zenger C. A Realization of the Simplex Method Based on Triangular Decompositions. Handbook for Automatic Computation. Volume II: Linear Algebra. Berlin, Heidelberg: Springer, 1971. P. 152–190. DOI: 10.1007/978-3-642-86940-2_11.

20. Tolla P. A Survey of Some Linear Programming Methods. Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: John Wiley, Sons, 2014. Chap. 7. P. 157–188. DOI: 10.1002/9781119005216.ch7.

21. Hall J. Towards a practical parallelisation of the simplex method. Computational Management Science. 2010. Vol. 7, no. 2. P. 139–170. DOI: 10.1007/s10287-008-0080-5.

22. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method. Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307. DOI: 10.1007/978-3-319-24024-4_17.

23. Lubin M., Hall J.A.J., Petra C.G., Anitescu M. Parallel distributed-memory simplex for large-scale stochastic LP problems. Computational Optimization and Applications. 2013. Vol. 55, no. 3. P. 571–596. DOI: 10.1007/s10589-013-9542-y.

24. Khachiyan L. Polynomial algorithms in linear programming. USSR Computational Mathematics and Mathematical Physics. 1980. Vol. 20, no. 1. P. 53–72. DOI: 10.1016/0041-5553(80)90061-0.

25. Shor N.Z. Cut-off method with space extension in convex programming problems. Cybernetics and Systems Analysis. 1977. Vol. 13, no. 1. P. 94–96. DOI: 10.1007/BF01071394.

26. Yudin D., Nemirovsky A. Information complexity and efficient methods for solving convex extremal problems. Economics and mathematical methods (Ekonomika i matematicheskie metody). 1976. No. 2. P. 357–369. (in Russian).

27. Karmarkar N. A new polynomial-time algorithm for linear programming. Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.

28. Dikin I. Iterative solution of problems of linear and quadratic programming. Soviet Mathematics. Doklady. 1967. Vol. 8. P. 674–675.

29. Gondzio J. Interior point methods 25 years later. European Journal of Operational Research. 2012. Vol. 218, no. 3. P. 587–601. DOI: 10.1016/j.ejor.2011.09.017.

30. Zorkaltsev V., Mokryi I. Interior point algorithms in linear optimization. Journal of applied and industrial mathematics. 2018. Vol. 12, no. 1. P. 191–199. DOI: 10.1134/S1990478918010179.

31. Fathi-Hafshejani S., Mansouri H., Reza Peyghami M., Chen S. Primal–dual interior-point method for linear optimization based on a kernel function with trigonometric growth term. Optimization. 2018. Vol. 67, no. 10. P. 1605–1630. DOI: 10.1080/02331934.2018.1482297.

32. Asadi S., Mansouri H. A Mehrotra type predictor-corrector interior-point algorithm for linear programming. Numerical Algebra, Control and Optimization. 2019. Vol. 9, no. 2. P. 147–156. DOI: 10.3934/naco.2019011.

33. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs. Proc. SPIE, vol. 8285. International Conference on Graphic and Image Processing (ICGIP 2011). International Society for Optics, Photonics, 2011. DOI: 10.1117/12.913019.

34. Kheirfam B., Haghighi M. A full-Newton step infeasible interior-point method for linear optimization based on a trigonometric kernel function. Optimization. 2016. Vol. 65, no. 4. P. 841–857. DOI: 10.1080/02331934.2015.1080255.

35. Xu Y., Zhang L., Zhang J. A full-modified-newton step infeasible interior-point algorithm for linear optimization. Journal of Industrial and Management Optimization. 2016. Vol. 12, no. 1. P. 103–116. DOI: 10.3934/jimo.2016.12.103.

36. Roos C., Terlaky T., Vial J.-P. Interior Point Methods for Linear Optimization. New York: Springer, 2005. 500 p. DOI: 10.1007/b100325.

37. Sokolinskaya I. Parallel Method of Pseudoprojection for Linear Inequalities. Parallel Computational Technologies. PCT 2018. Communications in Computer and Information Science, vol. 910 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2018. P. 216–231. DOI: 10.1007/978-3-319-99673-8_16.

38. Vasin V.V., Eremin I.I. Operators and Iterative Processes of Fejér Type. Theory and Applications. Berlin, New York: Walter de Gruyter, 2009. 155 p. Inverse and III-Posed Problems Series. DOI: 10.1515/9783110218190.

39. Gondzio J., Grothey A. Direct Solution of Linear Systems of Size 109 Arising in Optimization with Interior Point Methods. Parallel Processing and Applied Mathematics. PPAM 2005. Lecture Notes in Computer Science, vol. 3911. 3911 LNCS / ed. by R. Wyrzykowski, J. Dongarra, N. Meyer, J. Wasniewski. Berlin, Heidelberg: Springer, 2006. P. 513–525. DOI: 10.1007/11752578_62.

40. Prieto A., Prieto B., Ortigosa E.M., *et al.* Neural networks: An overview of early research, current frameworks and new challenges. Neurocomputing. 2016. Vol. 214. P. 242–268. DOI: 10.1016/j.neucom.2016.06.014.

41. Raina R., Madhavan A., Ng A.Y. Large-scale deep unsupervised learning using graphics processors. Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). New York, NY, USA: ACM Press, 2009. P. 873–880. DOI: 10.1145/1553374.1553486.

42. Tank D.W., Hopfield J.J. Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. IEEE transactions on circuits and systems. 1986. Vol. CAS-33, no. 5. P. 533–541. DOI: 10.1109/TCS.1986.1085953.

43. Kennedy M.P., Chua L.O. Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin. IEEE Transactions on Circuits and Systems. 1987. Vol. 34, no. 2. P. 210–214. DOI: 10.1109/TCS.1987.1086095.

44. Rodriguez-Vazquez A., Dominguez-Castro R., Rueda A., *et al.* Nonlinear Switched-Capacitor "Neural" Networks for Optimization Problems. IEEE Transactions on Circuits and Systems. 1990. Vol. 37, no. 3. P. 384–398. DOI: 10.1109/31.52732.

45. Zak S.H., Upatising V. Solving Linear Programming Problems with Neural Networks: A Comparative Study. IEEE Transactions on Neural Networks. 1995. Vol. 6, no. 1. P. 94–104. DOI: 10.1109/72.363446.

46. Malek A., Yari A. Primal–dual solution for the linear programming problems using neural networks. Applied Mathematics and Computation. 2005. Vol. 167, no. 1. P. 198–211. DOI: 10.1016/J.AMC.2004.06.081.

47. Liu X., Zhou M. A one-layer recurrent neural network for non-smooth convex optimization subject to linear inequality constraints. Chaos, Solitons and Fractals. 2016. Vol. 87. P. 39–46. DOI: 10.1016/j.chaos.2016.03.009.

48. Olkhovsky N., Sokolinsky L. Visualizing Multidimensional Linear Programming Problems. Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 172–196. DOI: 10.1007/978-3-031-11623-0_13.

49. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521, no. 7553. P. 436–444. DOI: 10.1038/nature14539.

50. Lachhwani K. Application of Neural Network Models for Mathematical Programming Problems: A State of Art Review. Archives of Computational Methods in Engineering. 2020. Vol. 27. P. 171–182. DOI: 10.1007/s11831-018-09309-5.

51. Polyak B.T. Random Algorithms for Solving Convex Inequalities. Studies in Computational Mathematics. Vol. 8 / ed. by C. Brezinski, L. Wuytack. Amsterdam, London, New York, Oxford, Paris, Shannon, Tokyo: Elsevier, 2001. P. 409–422. DOI: 10.1016/S1570-579X(01)80024-0.

52. Kaczmarz S. Angenherte Auflsung von Systemen linearer Gleichungen. Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres. Classe des Sciences Mathmatiques et Naturelles. Srie A, Sciences Mathmatiques. 1937. Vol. 35. P. 355–357.

53. Kaczmarz S. Approximate solution of systems of linear equations. International Journal of Control. 1993. Vol. 57, no. 6. P. 1269–1271. DOI: 10.1080/00207179308934446.

54. Cimmino G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. La Ricerca Scientifica, XVI, Series II, Anno IX, 1. 1938. P. 326–333.

55. Gastinel N. Linear Numerical Analysis. New York: Academic Press, 1971. ix+341 p.

56. Agmon S. The relaxation method for linear inequalities. Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. DOI: 10.4153/CJM-1954-037-2.

57. Motzkin T.S., Schoenberg I.J. The relaxation method for linear inequalities. Canadian Journal of Mathematics. 1954. Vol. 6. P. 393–404. DOI: 10.4153/CJM-1954-038-x.

58. Censor Y., Elfving T. New methods for linear inequalities. Linear Algebra and its Applications. 1982. Vol. 42. P. 199–211. DOI: 10.1016/0024-3795(82)90149-5.

59. De Pierro A.R., Iusem A.N. A simultaneous projections method for linear inequalities. Linear Algebra and its Applications. 1985. Vol. 64. P. 243–253. DOI: 10.1016/0024-3795(85)90280-0.

60. Sokolinskaya I.M., Sokolinsky L.B. Scalability Evaluation of Cimmino Algorithm for Solving Linear Inequality Systems on Multiprocessors with Distributed Memory. Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 2. P. 11–22. DOI: 10.14529/jsfi180202.

61. Sokolinsky L.B., Sokolinskaya I.M. Scalable parallel algorithm for solving non-stationary systems of linear inequalities. Lobachevskii Journal of Mathematics. 2020. Vol. 41, no. 8. P. 1571–1580. DOI: 10.1134/S1995080220080181.

62. Eremin I.I., Popov L.D. Fejér processes in theory and practice: Recent results. Russian Mathematics. 2009. Vol. 53, no. 1. P. 36–55. DOI: 10.3103/S1066369X09010022.

63. Nurminski E.A. Single-projection procedure for linear optimization. Journal of Global Optimization. 2016. Vol. 66, no. 1. P. 95–110. DOI: 10.1007/S10898-015-0337-9.

64. Censor Y. Can linear superiorization be useful for linear optimization problems?. Inverse Problems. 2017. Vol. 33, no. 4. P. 044006. DOI: 10.1088/1361-6420/33/4/044006.

65. Gould N.I. How good are projection methods for convex feasibility problems?. Computational Optimization and Applications. 2008. Vol. 40, no. 1. P. 1–12. DOI: 10.1007/S10589-007-9073-5.

66. Sokolinsky L.B. BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems. Journal of Parallel and Distributed Computing. 2021. Vol. 149. P. 193–206. DOI: 10.1016/j.jpdc.2020.12.009.

67. Sokolinsky L.B. BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems. MethodsX. 2021. Vol. 8. Article number 101437. DOI: 10.1016/j.mex.2021.101437.

68. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University. Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

69. Sokolinsky L.B., Sokolinskaya I.M. FRaGenLP: A Generator of Random Linear Programming Problems for Cluster Computing Systems. Parallel Computational Technologies. PCT 2021. Communications in Computer and Information Science, vol. 1437 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2021. P. 164–177. DOI: 10.1007/978-3-030-81691-9_12.

70. Sokolinsky L.B., Sokolinskaya I.M. VaLiPro: Linear Programming Validator for Cluster Computing Systems. Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 3. P. 51–61. DOI: 10.14529/jsfi210303.

71. Gay D.M. Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

72. Keil C., Jansson C. Computational experience with rigorous error bounds for the Netlib linear programming library. Reliable Computing. 2006. Vol. 12, no. 4. P. 303–321. DOI: 10.1007/S11155-006-9004-7/METRICS.

73. Koch T. The final NETLIB-LP results. Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI: 10.1016/S0167-6377(03)00094-4.

74. Deutsch F., Hundal H. The rate of convergence for the cyclic projections algorithm I: Angles between convex sets. Journal of Approximation Theory. 2006. Vol. 142, no. 1. P. 36–55. DOI: 10.1016/J.JAT.2006.02.005.

УДК 004.272.25, 004.421, 004.032.24 DOI: 10.14529/cmse230202

ПОИСК АНОМАЛИЙ В СЕНСОРНЫХ ДАННЫХ ЦИФРОВОЙ ИНДУСТРИИ С ПОМОЩЬЮ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ*

© 2023 Я.А. Краева

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: kraevaya@susu.ru

Поступила в редакцию: 20.09.2022

В статье представлены результаты исследований по поиску аномалий в сенсорных данных из различных приложений цифровой индустрии. Рассматриваются временные ряды, полученные при эксплуатации деталей машин, показания датчиков, установленных на металлургическом оборудовании, и показания температурных датчиков в системе умного управления отоплением зданий. Аномалии, найденные в таких данных, свидетельствуют о нештатной ситуации, отказах, сбоях и износе технологического оборудования. Аномалия формализуется как диапазонный диссонанс — подпоследовательность временного ряда, расстояние от которой до ее ближайшего соседа не менее наперед заданного аналитиком порога. Ближайшим соседом данной подпоследовательности является такая подпоследовательность ряда, которая не пересекается с данной и имеет минимальное расстояние до нее. Поиск диссонансов выполняется с помощью параллельного алгоритма для графического процессора, ранее разработанного автором данной статьи. Для визуализации найденных аномалий предложены метод построения тепловой карты диссонансов, имеющих различные длины, и алгоритм нахождения в построенной тепловой карте наиболее значимых диссонансов независимо от их длин.

Ключевые слова: временной ряд, сенсорные данные, поиск аномалий, диссонанс, параллельный алгоритм, графический процессор, CUDA.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Краева Я.А. Поиск аномалий в сенсорных данных цифровой индустрии с помощью параллельных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 47–61. DOI: 10.14529/cmse230202.

Введение

В настоящее время поиск аномалий временных рядов является одной из актуальных задач в широком спектре предметных областей, связанных с обработкой сенсорных данных [1]. В приложениях цифровой индустрии [2] и Интернета вещей [3, 4] датчики киберфизических систем имеют высокую дискретность снятия показаний (десятки–сотни раз в секунду) и за короткое время продуцируют временные ряды, состоящие из сотен миллионов элементов. Аномалии, найденные в данных сенсоров, свидетельствуют о нештатной ситуации, отказах, сбоях и износе технологического оборудования. Обнаружение аномалий может использоваться для заблаговременного уведомления оператора технологического процесса и организации предиктивного технического обслуживания и ремонта оборудования, что в конечном итоге увеличивает остаточный ресурс этого оборудования.

В данном исследовании поиск аномалий предполагает нахождение подпоследовательностей временного ряда, которые наименее похожи на все остальные подпоследовательности ряда. Одним из наиболее эффективных подходов к поиску аномалий является концепция

*Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2023».

диапазонного диссонанса (range discord) [5, 6]. Диапазонный диссонанс (далее для краткости — диссонанс) представляет собой подпоследовательность ряда, расстояние от которой до ее ближайшего соседа не менее заданного порога, являющегося параметром алгоритма. Ближайшим соседом данной подпоследовательности является такая подпоследовательность ряда, которая не пересекается с данной и имеет минимальное расстояние до нее. Применение концепции диссонансов для поиска аномалий требует от аналитика интуитивно понятных параметров (длина подпоследовательности и порог расстояния), в отличие от альтернативных подходов, требующих более трех не всегда интуитивно понятных параметров [7].

Авторы концепции диссонанса предложили последовательный алгоритм DRAG (Discord Range Aware Gathering) [6] поиска диссонансов временного ряда. Ранее автором настоящей статьи был разработан алгоритм PD3 (Parallel DRAG-based Discord Discovery) [8], представляющий собой параллельную версию алгоритма DRAG. В данной статье показано, как алгоритм PD3 может быть применен для поиска аномалий в приложениях цифровой индустрии. Рассматриваются сенсорные данные, полученные при эксплуатации деталей машин, показания датчиков, установленных на металлургическом оборудовании, и показания температурных датчиков в системе умного управления отоплением зданий. Для визуализации найденных аномалий предлагается метод построения тепловой карты диссонансов, имеющих различные длины, и предложен способ нахождения в построенной тепловой карте наиболее значимых диссонансов независимо от их длин.

Статья организована следующим образом. В разделе 1 приводятся формальные определения и краткое описание параллельного алгоритма PD3. В разделе 2 представлены метод построения тепловой карты диссонансов и алгоритм нахождения наиболее значимых диссонансов. В разделе 3 описаны результатов исследований по поиску аномалий в сенсорных данных цифровой индустрии. Заключение подводит итоги исследования.

1. Параллельный алгоритм поиска диссонансов PD3

1.1. Формальные определения и обозначения

В данном разделе приводятся обозначения и определения используемых терминов в соответствии с работами [5, 6].

Временной ряд (time series) T представляет собой последовательность хронологически упорядоченных вещественных значений:

$$T = \{t_i\}_{i=1}^{n}, \quad t_i \in \mathbb{R}. \tag{1}$$

Число n обозначается $|T|$ и называется длиной ряда.

Подпоследовательность (subsequence) $T_{i,\,m}$ временного ряда T представляет собой непрерывный промежуток из m элементов, начиная с позиции i:

$$T_{i,m} = \{t_k\}_{k=i}^{i+m-1}, \quad 1 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n - m + 1. \tag{2}$$

Множество всех подпоследовательностей ряда T, имеющих длину m, обозначим как S_T^m, а мощность такого множества за N, $N = |S_T^m| = n - m + 1$.

Подпоследовательности $T_{i,\,m}$ и $T_{j,\,m}$ ряда T называются *непересекающимися (non-self match)*, если $|i - j| \geqslant m$. Подпоследовательность, которая является непересекающейся к данной подпоследовательности C, будем обозначать как M_C.

Подпоследовательность D ряда T является *диапазонным диссонансом (range discord)*, если

$$\forall M_D \in T \ \min\big(\mathrm{Dist}(D, M_D)\big) > r, \tag{3}$$

где $\mathrm{Dist}(\cdot\,,\cdot\,)$ представляет собой неотрицательную симметричную функцию, порог расстояния r — наперед заданный параметр. Другими словами, некая подпоследовательность ряда является диапазонным диссонансом, если ее ближайший сосед (ближайшая и не пересекающаяся с ней подпоследовательность) находится на расстоянии не менее чем r. Далее для краткости будем использовать термин «диссонанс», подразумевая диапазонный диссонанс, если не указано обратное.

Алгоритм PD3 предполагает, что обрабатываемые подпоследовательности временного ряда предварительно подвергнуты z-нормализации. *Z-нормализация* подпоследовательности (ряда) T представляет собой подпоследовательность (ряд) $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_m)$, элементы которого вычисляются следующим образом:

$$\hat{t}_i = \frac{t_i - \mu}{\sigma}, \quad \mu = \frac{1}{m}\sum_{i=1}^{m} t_i, \quad \sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m} t_i^2 - \mu^2}. \tag{4}$$

В данном исследовании в качестве функции $\mathrm{Dist}(\cdot\,,\cdot\,)$ используется квадрат нормированного евклидова расстояния, обозначаемого как $\mathrm{ED}^2_{\mathrm{norm}}(\cdot\,,\cdot\,)$. Вычисление указанного расстояния с помощью формулы, предложенной в работе [9], выполняется быстрее, чем с использованием формулы (4):

$$\mathrm{ED}^2_{\mathrm{norm}}(T_{i,m}, T_{j,m}) = \mathrm{ED}^2(\hat{T}_{i,m}, \hat{T}_{j,m}) = 2m\left(1 - \frac{< T_{i,m}, T_{j,m} > - m\mu_i\mu_j}{m\sigma_i\sigma_j}\right), \tag{5}$$

где подпоследовательности $T_{i,m}$ и $T_{j,m}$ рассматриваются как вектора в евклидовом пространстве $\mathbb{R}^m$, μ_i и μ_j, σ_i и σ_j — среднее арифметическое и стандартное отклонение указанных векторов соответственно.

1.2. Принципы реализации алгоритма

Алгоритм PD3 распараллеливает вычислительную схему алгоритма DRAG [6], которая заключается в следующем. DRAG сначала выполняет отбор кандидатов в диссонансы, а затем очищает полученное множество от ложноположительных кандидатов. На этапе отбора множество кандидатов $\mathcal{C}$ инициализируется первой подпоследовательностью ряда T. Далее алгоритм сканирует ряд с помощью скользящего окна длины m и для каждой подпоследовательности $s \in S_T^m$ выполняет проверку, что каждый кандидат $c \in \mathcal{C}$ является диссонансом. Кандидат c, не прошедший проверку, удаляется из $\mathcal{C}$. По завершении проверки s либо добавляется в множество кандидатов, либо удаляется из него. На этапе очистки сначала для каждого кандидата из $\mathcal{C}$ расстояние до его ближайшего соседа полагается $+\infty$. Затем алгоритм сканирует ряд с помощью скользящего окна длины m, вычисляя расстояние между каждой подпоследовательностью $s \in S_T^m$ и каждым кандидатом c. Если расстояние меньше r, то кандидат является удаляется из $\mathcal{C}$ как ложноположительный. Если указанное расстояние меньше текущего минимального расстояния до ближайшего соседа, то текущий минимум расстояния до ближайшего соседа обновляется этим значением.

Алгоритм PD3 [8], используя параллелизм по данным, распараллеливает на графическом процессоре этапы отбора и очистки алгоритма DRAG. Временной ряд сегментируется,

и отдельный блок нитей GPU обрабатывает свой сегмент. На этапе отбора схема работы алгоритма PD3 выглядит следующим образом. Блок нитей полагает все подпоследовательности своего сегмента кандидатами и обрабатывает те из них, которые расположены справа от него и не пересекаются с кандидатами. Если расстояние от кандидата до подпоследовательности меньше r, то они заведомо не являются диссонансами и отбрасываются. Если все кандидаты отброшены, блок завершает работу. Блок нитей обрабатывает подпоследовательности ряда порциями, количество элементов в которых равно длине сегмента. При этом первая порция таких элементов начинается с m-го элемента в сегменте, что позволяет избежать избыточных проверок на пересечение кандидатов и подпоследовательностей обрабатываемой порции. Обработка порции заключается в вычислении расстояний от всех подпоследовательностей сегмента, назначенного блоку, до всех подпоследовательностей данной порции, на основе формулы (5), и выполняется следующим образом. Сперва нити блока вычисляют скалярные произведения между первой подпоследовательностью сегмента и всеми подпоследовательностями текущей порции, сохраняя результат в массиве в разделяемой памяти GPU. Далее вычисляются расстояния между первой подпоследовательностью порции и всеми подпоследовательностями сегмента, при этом используются полученные ранее результаты. Вычисленное расстояние используется для отбрасывания ложноположительных кандидатов в сегменте и текущей порции. Очистка найденных кандидатов выполняется в алгоритме PD3 аналогично описанной выше процедуре отбора. В очистке задействуются сегменты ряда с непустыми множествами кандидатов, при этом обрабатываются подпоследовательности ряда, не пересекающиеся с кандидатами и находящиеся слева от сегмента. Если расстояние от кандидата до подпоследовательности меньше r, то кандидат отбрасывается.

2. Визуализация диссонансов

Визуализация является важной и неотъемлемой частью решения задач интеллектуального анализа данных, поскольку обеспечивает аналитику наглядное представлением исследуемых данных и результатов анализа, являющееся основой выявления скрытых закономерностей для принятия стратегически важных решений [10].

В данном разделе представлен новый метод визуализации диссонансов временного ряда, который предполагает построение тепловой карты, где степень аномальности диссонансов отражается посредством интенсивности цвета. Далее в разделах 2.1 и 2.2 описаны детали построения тепловой карты и способ нахождения наиболее значимых аномалий на основе построенной карты соответственно.

2.1. Построение тепловой карты диссонансов

Пусть имеется временной ряд T длины n, в котором с помощью алгоритма PD3 осуществляется поиск диссонансов длины m, принимающей значения в заданном диапазоне $minL \leqslant m \leqslant maxL$ ($minL \leqslant maxL \ll n$). Последовательно запуская алгоритм PD3 $maxL - minL + 1$ раз для каждого значения длины из указанного диапазона, получим итоговое множество диссонансов $\mathcal{D} = \bigcup\limits_{m=minL}^{maxL} D_m$, где D_m — подмножество диссонансов, имеющих длину m.

Далее рассмотрим матричный профиль временного ряда T для длины подпоследовательности m. *Матричный профиль (matrix profile)* [11] временного ряда T для длины подпоследовательности m представляет собой временной ряд $MP_m \in \mathbb{R}^{n-m+1}$, элементами кото-

рого являются расстояния от соответствующей подпоследовательности ряда до ближайшего не пересекающегося с ней соседа:

$$MP_m(i) = \mathrm{Dist}(T_{i,\,m},\, Neighbor), \quad Neighbor = \arg \min_{T_{j,\,m} \in S_T^m} \mathrm{Dist}(T_{i,\,m},\, T_{j,\,m}), \quad |i-j| \geqslant m. \quad (6)$$

Затем возьмем матричные профили ряда T для всех длин подпоследовательности из диапазона $minL..maxL$ в порядке возрастания длин и построим из них матрицу профилей $MMP \in \mathbb{R}^{(maxL-minL+1)\times(n-minL)}$. В качестве строки указанной матрицы фигурирует отдельный матричный профиль ряда, где обнулены элементы, соответствующие подпоследовательностям ряда, которые не являются диссонансами (индексы строки и столбца матрицы показывают соответственно длину диссонанса и его индекс в ряде):

$$MMP(m,\, i) = \begin{cases} MP_{minL+m-1}(i), & T_{i,\,minL+m-1} \in D_{minL+m-1} \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Отметим, что матрица MMP является естественным следствием запусков алгоритма PD3 и не требует отдельных вычислений матричных профилей ряда для рассматриваемых длин подпоследовательности (например, с помощью алгоритма [12]). Далее определим *тепловую карту диссонансов (discord heatmap)* как матрицу $heatmap \in \mathbb{R}^{(maxL-minL+1)\times(n-minL)}$, получаемую нормированием элементов в каждой строке матрицы $MMP(m, \cdot)$ с помощью множителя $\frac{1}{2m}$:

$$heatmap(m,\, i) = \frac{MMP(m,\, i)}{2m}. \quad (8)$$

Указанный нормирующий множитель обеспечивает приведение значений элементов тепловой карты к диапазону от 0 до 1, поскольку доказано [9], что положительная корреляция по Пирсону двух векторов $x,\, y \in \mathbb{R}^m$ и нормализованное евклидово расстояние между ними связаны следующим соотношением:

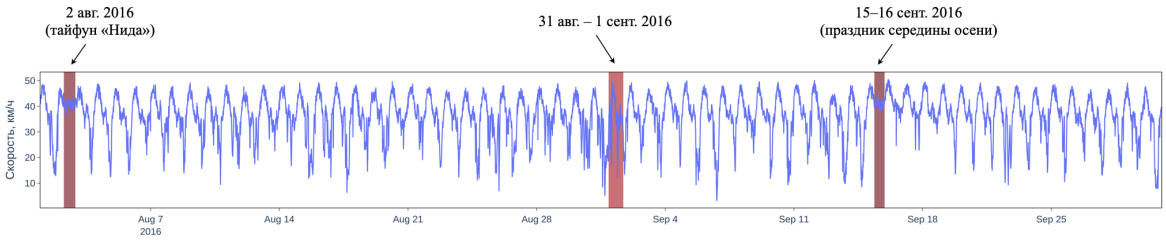
$$PearsonCorr(x, y) = 1 - \frac{\mathrm{ED}^2_{\mathrm{norm}}(x,\, y)}{2m}. \quad (9)$$

Таким образом, каждый элемент $heatmap(m,\, i)$ представляет собой оценку аномальности диссонанса $T_{i,\,m} \in D_m$, нормированную по всем диссонансам множества $\mathcal{D}$. В тепловой карте диссонансов используется один цвет (например, красный), и интенсивность цвета в пикселе (m, i) прямо пропорциональна оценке диссонанса $T_{i,\,m} \in D_m$. При этом нормирование обеспечивает на одной тепловой карте корректную визуализацию оценок аномальности диссонансов различной длины.

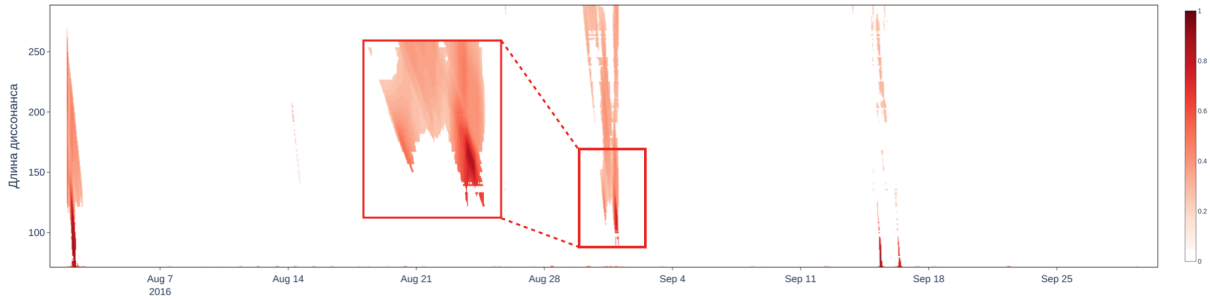
2.2. Ранжирование диссонансов различной длины

Тепловая карта диссонансов, описанная выше, представляет собой удобный для аналитика инструмент визуализации диссонансов заданного временного ряда, имеющих различную длину. Тем не менее, для аналитика также важно ранжирование найденных диссонансов по их практической значимости вне зависимости от их длины. Далее представлен алгоритм нахождения top-k наиболее значимых диссонансов, который основан на простой идее отдавать предпочтение диссонансам, которые имеют бóльшую оценку, исключая при этом пересекающиеся диссонансы (см. Алг. 1).

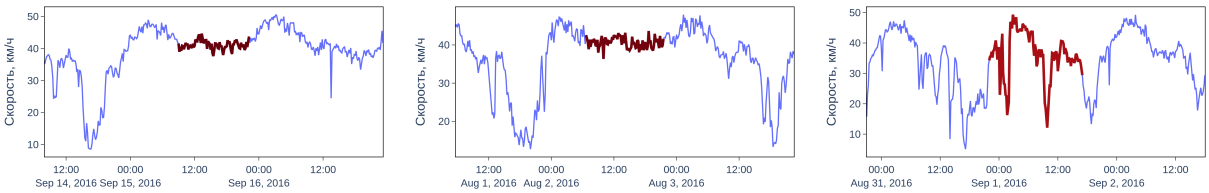
Алгоритм получает на входе тепловую карту диссонансов $heatmap \in \mathbb{R}^{(maxL-minL+1)\times(n-minL)}$ и число K искомых диссонансов, а на выходе формирует



(a) Временной ряд



(b) Тепловая карта диссонансов



Top-1 диссонанс, $m = 78$ Top-2 диссонанс, $m = 87$ Top-3 диссонанс, $m = 116$

(c) Найденные top-3 интересных диссонанса различных длин

Рис. 1. Результаты визуализации и подхода нахождения наиболее интересных диссонансов

Алг. 1 TopInterestDiscords (in: $heatmap$, K; out: $InterestD$)

$\overline{Scores} \leftarrow \{score_i\}_{i=1}^{n-minL+1}$, где $score_i = \max\limits_{minL \leqslant m \leqslant maxL} heatmap(m, i)$

$\overline{Lengths} \leftarrow \{length_i\}_{i=1}^{n-minL+1}$, где $length_i = \arg\max\limits_{minL \leqslant m \leqslant maxL} heatmap(m, i)$

$\overline{Indexes} \leftarrow \{i\}_{i=1}^{n-minL+1}$

Sort($\overline{Scores}, \overline{Lengths}, \overline{Indexes}$) ▷ Сортировка векторов по убыванию значений в $\overline{Scores}$

$InterestD \leftarrow \{(T_{\overline{Indexes}(1), \overline{Lengths}(1)}, \overline{Scores}(1))\}$; $k \leftarrow 2$

while ($|InterestD| < K$) **or** ($k < N$) **do**

  $score \leftarrow \overline{Scores}(k)$; $length \leftarrow \overline{Lengths}(k)$; $index \leftarrow \overline{Indexes}(k)$

  for each $T_{i,m} \in InterestD$ **do**

    if $\min(length, m) < |i - index| < \max(length, m)$ **then** ▷ Проверка пересечения

      $InterestD \leftarrow InterestD \cup \{(T_{index,\, length}, score)\}$

    else

      break

  $k \leftarrow k + 1$

return $InterestD$

множество $Interest\mathcal{D}$, элементами которого являются пары вида $(T_{i,m}, score)$, где диссонанс $T_{i,m} \in D_m$, $score$ — оценка диссонанса. Алгоритм выполняется следующим образом. Сначала формируются вектора $Scores, Lengths \in \mathbb{R}^{n-minL+1}$, в которых i-й элемент хранит соответственно максимальную оценку диссонанса с индексом i и длину такого диссонанса. Вектор $Indexes \in \mathbb{R}^{n-minL+1}$ хранит индексы диссонансов. Далее вектор $Scores$ сортируется по убыванию, а вектора $Lengths$ и $Indexes$ упорядочиваются на основе результата такой сортировки. В итоге указанные три вектора совместно представляют сведения о диссонансах множества $\mathcal{D}$ в порядке убывания оценок входящих в него диссонансов (строки 1–4 в Алг. 1). Искомое множество диссонансов инициализируется диссонансом с максимальной оценкой (строка 5 в Алг. 1). Затем выполняется просмотр диссонансов множества $\mathcal{D}$ в полученном порядке (цикл в строках 6–13 в Алг. 1). Если рассматриваемый диссонанс не пересекается ни с одним элементом из $Interest\mathcal{D}$, то он добавляется в него. Сканирование продолжается до тех пор, пока не найдено K наиболее значимых диссонансов или исчерпано множество $\mathcal{D}$.

Проиллюстрируем работу описанного алгоритма на следующем реальном примере. На рис. 1a представлен пример временного ряда [13], который содержит данные о скорости городского трафика на одном из участков городской автомагистрали Гуанчжоу (Китай), измерявшиеся каждые 10 мин. с 1 августа по 30 сентября 2016 г. В ряде выделены красным цветом диссонансы различной длины, которые были найдены с помощью алгоритма PD3. На рис. 1b показана тепловая карта найденных диссонансов. Далее, на рис. 1c показаны три наиболее важных диссонанса (имеющие различную длину), отобранные с помощью описанного выше алгоритма TopInterestDiscords. Можно видеть, что у двух из указанных диссонансов время возникновения совпадает с нетипичными событиями: тайфун Нида и отмечаемый в Китае Праздник середины осени.

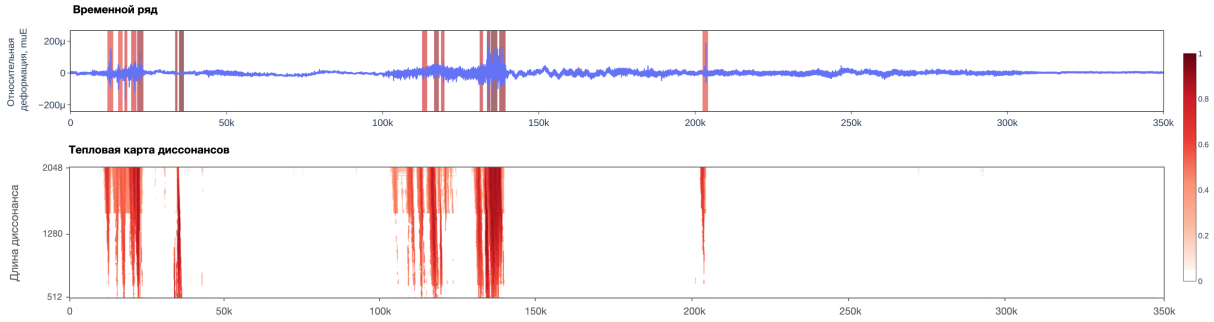
3. Поиск диссонансов в сенсорных данных цифровой индустрии

В данном разделе приведены результаты исследований по применению разработанного автором данной статьи алгоритма для поиска диссонансов в сенсорных данных из различных областей цифровой индустрии. Указанные исследования проведены на оборудовании Лаборатории суперкомпьютерного моделирования ЮУрГУ [14]: графический процессор NVIDIA Tesla V100 SXM2 (5 120 ядер с тактовой частотой 1.3 GHz, пиковая производительность 7 TFLOPS для чисел с двойной точностью).

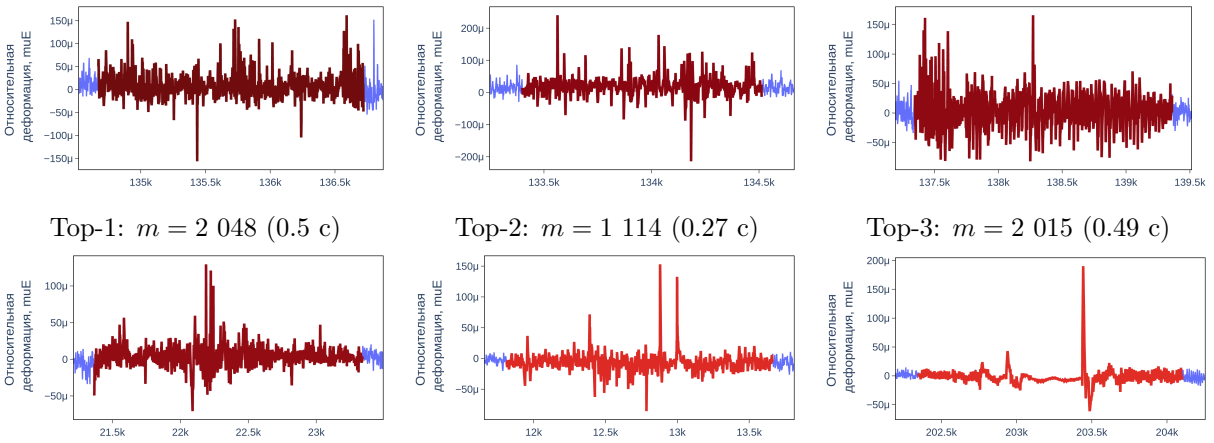
3.1. Деформации в механизме сцепки вагонов трамвая

Первое исследование связано с поиском диссонансов в сенсорных данных, полученных с тензометрического датчика, установленного на механизме сцепки вагонов трамвая. Данные указанного датчика представляют собой значения относительной деформации, связанные со значениями напряжений, возникающими в процессе эксплуатации механизма сцепки вагонов. Использованные в исследовании данные сняты в течение одной рабочей смены трамвая, следующего по одному маршруту, с частотой дискретизации 4 096 Гц в течение 1.5 мин. (350 000 точек).

На рис. 2a представлены временной ряд и тепловая карта найденных с помощью разработанного автором данной статьи параллельного алгоритма диссонансов, имеющих длину в диапазоне 512..2 048. На рис. 2b представлены примеры шести наиболее значимых дис-



(a) Временной ряд и тепловая карта диссонансов



Top-1: $m = 2\,048$ (0.5 с) Top-2: $m = 1\,114$ (0.27 с) Top-3: $m = 2\,015$ (0.49 с)

Top-4: $m = 1\,975$ (0.48 с) Top-5: $m = 1\,852$ (0.45 с) Top-6: $m = 1\,756$ (0.43 с)

(b) Примеры найденных top-k диссонансов различных длин

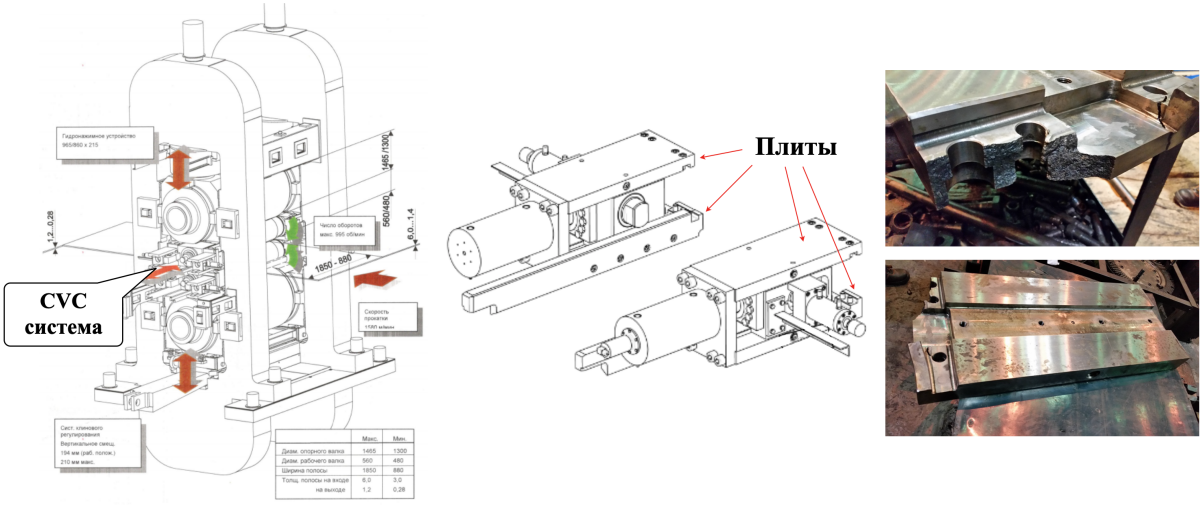
Рис. 2. Обнаружение аномалий во временном ряде относительных деформаций механизма стыковки вагонов трамвая

сонансов. Найденные диссонансы позволяют видеть, что при эксплуатации узла сцепки вагонов имеет место работа в экстремальных условиях с высокими амплитудами отклонения знакопеременной нагрузки. Очевидно, что при длительной работе в подобных условиях остаточный ресурс узла сцепки вагонов сокращается.

3.2. Разрушение плит системы профилировки валков стана холодной прокатки

Второе исследование связано с поиском диссонансов во временных рядах, полученных с сенсоров, установленных на стане холодной прокатки металлургического завода. Холодная прокатка сопровождается воздействием больших усилий и напряжений на основные рабочие и вспомогательные элементы стана, поэтому они быстро изнашиваются и часто ломаются, что негативно влияет на качество выпускаемой продукции. Для повышения качества металлопродукции и совершенствования технологии производства холоднокатанных полос каждая клеть стана оснащается системой регулировки профиля полос посредством осевой сдвижки выпукло-вогнутых рабочих валков CVC (Continuously Variable Curvature) [15] (см. рис. 3а). Система CVC позволяет добиться уменьшения разнотолщинности по профилю, по краям и по центру холоднокатанных полос и высокой плоскостности по всей ширине листа. Однако при эксплуатации системы CVC имеют место частые поломки плит, по которым происходит движение системы CVC совместно с рабочими валками клети в горизонтальном

направлении (см. рис. 3b). Причиной подобных поломок, предположительно, являются высокие значения знакопеременных нагрузок изгибающих моментов, вызванных изменениями напряженности клети. Указанные нагрузки приводят к появлению в наиболее нагруженных частях плит CVC трещин, которые ведут к разрушению плит (см. рис. 3c).



(a) Клеть стана холодной прокатки

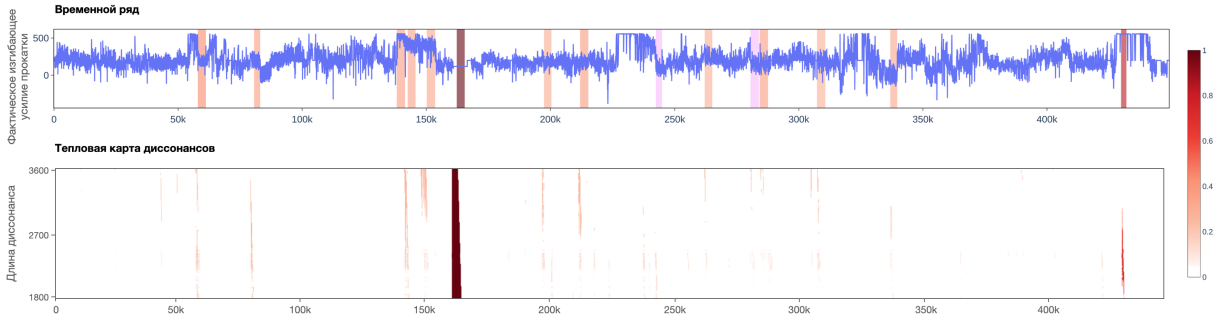
(b) CVC система рабочего валка клети стана

(c) Разрушение плит CVC системы

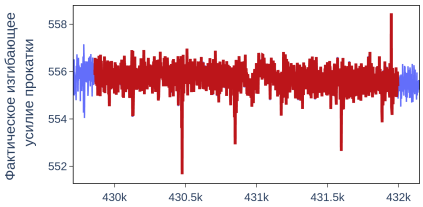
Рис. 3. Проблема поломки плит CVC системы стана холодной прокатки

В исследовании использовались данные сенсоров, установленные на одной из клетей стана холодной прокатки, которые измеряли различные показатели с частотой 1 раз в секунду в течение 6 месяцев, а также сведения о простоях клетей стана вследствие технического обслуживания или ремонта плит CVC за указанный период. В силу ранее сделанного предположения о высокой напряженности клети как о наиболее вероятной причине поломок для исследования был выбран временной ряд показаний датчика, измеряющего фактическое изгибающее усилие прокатки, в котором исключены периоды простоев стана, не связанных с поломками плит. Полученный ряд представлен на рис. 4a. В данном временном ряде выполнялся поиск диссонансов различных длин из диапазона от 30 минут до 1 часа (т.е. $minL = 1\ 800$ и $maxL = 3\ 600$).

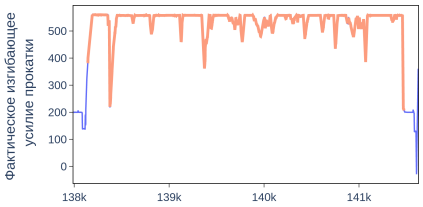
На рис. 4a представлены результаты работы алгоритма PD3 по поиску top-15 диссонансов, которые выделены красным цветом. Далее из них были отобраны 4 следующих диссонанса, которые наиболее точно отражают причины возникновения критических ситуаций при работе плит CVC системы (см. рис. 4b). Первый диссонанс показывает, что на плиту длительное время действуют напряжения, намного превышающие допустимые значения, вызванные напряженностью клети стана в отсутствии прокатываемого металла. Остальные три диссонанса соответствуют напряженному состоянию в плитах при знакопеременных нагрузках и критических значениях давлений на клеть при обработке проката. Отобранные диссонансы показывают, что в межремонтные периоды плиты, подвергаемые экстремальным знакопеременным нагрузкам, с большой вероятностью могут получить повреждения в виде трещин, приводящих к поломками и выходу плит из строя. Найденные диссонансы могут в дальнейшем быть использованы как паттерны снижения остаточного ресурса плит CVC системы и предсказания их поломок.



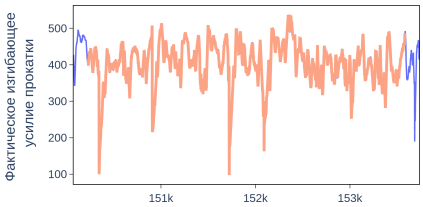
(a) Временной ряд фактического изгибающего усилия прокатки



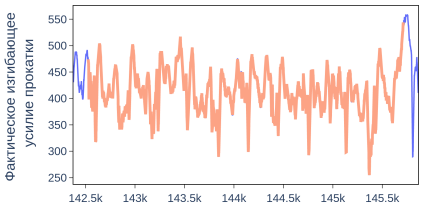
Top-3: $m = 2\ 898$ (48 мин. 18 с)



Top-5: $m = 3\ 345$ (55 мин. 45 с)



Top-7: $m = 3\ 576$ (59 мин. 36 с)



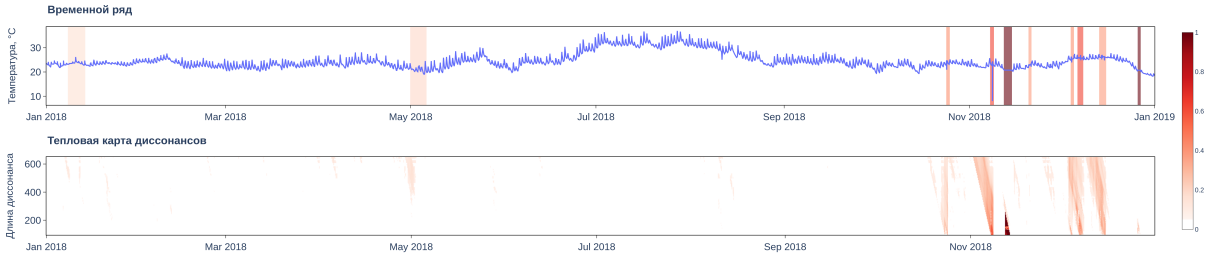
Top-10: $m = 3\ 187$ (53 мин. 6 с)

(b) Примеры найденных top-k диссонансов различных длин

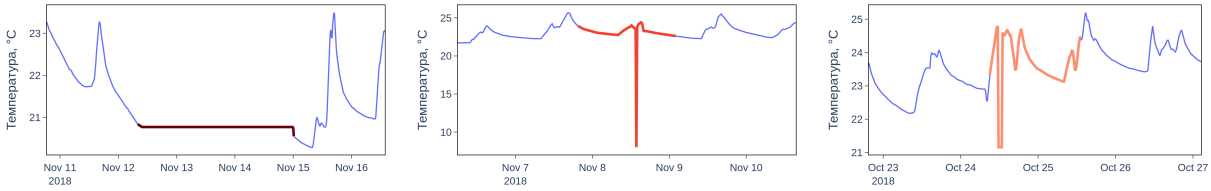
Рис. 4. Обнаружение аномалий в показаниях датчиков, установленных на металлургическом оборудовании

3.3. Нештатные ситуации в системе умного управления отоплением зданий

Третье исследование направлено на поиск диссонансов в показаниях температурных датчиков, которые являются частью интеллектуальной системы ПолиТЭР [16] управления теплоснабжением, установленной в Южно-Уральском государственном университете (ЮУрГУ). ПолиТЭР выполняет мониторинг инженерных систем зданий кампуса ЮУрГУ и управляет режимами их работы на основе анализа показаний проводных и беспроводных датчики. Для исследования были взяты показания беспроводного датчика, установленного в одной из учебных аудиторий ЮУрГУ (см. рис. 5a), за 2018 год при частоте снятия показаний 1 раз в 15 мин. В данном временном ряде выполнялся поиск диссонансов различных длин из диапазона от 1 дня до 1 недели (т.е. $minL = 96$ и $maxL = 672$). Примеры некоторых аномалий из списка top-10, полученного с помощью алгоритм PD3, представлены на рис. 5b. Первая аномалия показывает, что датчик, скорее всего, вышел из строя, поскольку в течение определенного времени передавалась постоянная температура. В остальных случаях аномалии свидетельствуют о влиянии человеческого фактора на энергоэффективность системы теплоснабжения зданий университета: вероятно, резкое изменение температуры вызвано сознательно либо случайно открытыми окнами в помещении.



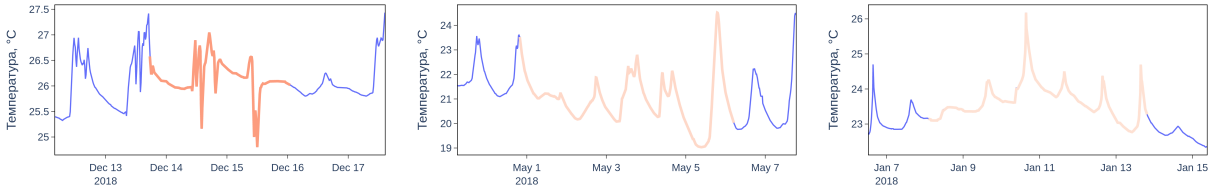
(a) Временной ряд и тепловая карта диссонансов



Top-1: $m = 258$
(2 дня 16 час. 30 мин.)

Top-3: $m = 122$
(1 день 6 час. 30 мин.)

Top-6: $m = 113$
(1 день 4 час. 15 мин.)



Top-7: $m = 219$
(2 дня 6 час. 45 мин.)

Top-9: $m = 504$
(5 дней 6 час.)

Top-10: $m = 650$
(6 дней 18 час. 30 мин.)

(b) Примеры найденных top-k диссонансов различных длин

Рис. 5. Обнаружение аномалий в показаниях температурных датчиков в системе умного управления отоплением зданий

Заключение

В статье рассмотрена проблема поиска аномалий в сенсорных данных, которая в настоящее время возникает в широком спектре предметных областей: цифровая индустрия, Интернет вещей и др. Аномалии, найденные в данных сенсоров, свидетельствуют о нештатной ситуации, отказах, сбоях и износе технологического оборудования. Оперативное обнаружение аномалий может использоваться для заблаговременного уведомления оператора технологического процесса и организации профилактического обслуживания и ремонта оборудования. В данном исследовании аномалия формализуется как диапазонный диссонанс [5, 6] — подпоследовательность ряда, расстояние от которой до ее ближайшего соседа не менее наперед заданного аналитиком порога. Ближайшим соседом данной подпоследовательности является такая подпоследовательность ряда, которая не пересекается с данной и имеет минимальное расстояние до нее.

В статье представлены результаты исследований по поиску диссонансов во временных рядах, которые представляют собой показания сенсоров из реальных предметных областей: деформации механизма стыковки вагонов трамвая, деформации плиты системы регулировки профиля полос стана холодной прокатки металлургического завода и показания температурных датчиков интеллектуальной системы управления отоплением зданий ЮУрГУ. Поиск диссонансов выполнен с помощью ранее разработанного автором статьи параллель-

ного алгоритма PD3 (Parallel DRAG-based Discord Discovery) [8] для графического процессора. Для визуализации найденных аномалий предложен метод построения тепловой карты диссонансов, имеющих различные длины, и алгоритм TopInterestDiscords, позволяющий выбрать в построенной тепловой карте наиболее значимые диссонансы независимо от их длин. Алгоритм подбирает диссонансы, которые имеют бóльшую оценку аномальности (независимо от длин диссонансов), исключая при этом пересекающиеся диссонансы. Для каждого рассмотренного случая представлены тепловая карта найденных диссонансов и наиболее значимые из них.

Работа выполнена при финансовой поддержке Российского научного фонда (грант № 23-21-00465).

Литература

1. Blázquez-García A., Conde A., Mori U., Lozano J.A. A Review on Outlier/Anomaly Detection in Time Series Data // ACM Comput. Surv. 2021. Vol. 54, no. 3. 56:1–56:33. DOI: 10.1145/3444690.
2. Kumar S., Tiwari P., Zymbler M.L. Internet of Things is a revolutionary approach for future technology enhancement: a review // J. Big Data. 2019. Vol. 6. P. 111. DOI: 10.1186/s40537-019-0268-2.
3. Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др. Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 16–36. DOI: 10.14529/cmse210302.
4. Иванов С.А., Никольская К.Ю., Радченко Г.И. и др. Концепция построения цифрового двойника города // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 4. С. 5–23. DOI: 10.14529/cmse200401.
5. Keogh E.J., Lin J., Fu A.W. HOT SAX: efficiently finding the most unusual time series subsequence // Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), Houston, Texas, USA, November 27-30, 2005. IEEE Computer Society, 2005. P. 226–233. DOI: 10.1109/ICDM.2005.79.
6. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets // Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA. 2007. P. 381–390. DOI: 10.1109/ICDM.2007.61.
7. Chandola V., Cheboli D., Kumar V. Detecting anomalies in a time series database. Retrieved from the University of Minnesota Digital Conservancy. 2009. URL: https://hdl.handle.net/11299/215791 (дата обращения: 12.04.2022).
8. Kraeva Y., Zymbler M. A parallel discord discovery algorithm for a graphics processor // Pattern Recognition and Image Analysis. 2023. Vol. 33, no. 2. P. 101–113. DOI: 10.1134/S1054661823020062.
9. Mueen A., Nath S., Liu J. Fast approximate correlation for massive time-series data // Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010. ACM, 2010. P. 171–182. DOI: 10.1145/1807167.1807188.

10. Han Z., Gao P., Wan F. Research on Data Mining and Visualization Technology // CONF-CDS 2021: The 2nd International Conference on Computing and Data Science, Stanford, CA, USA, January 28-30, 2021. ACM, 2021. 71:1–71:4. DOI: 10.1145/3448734.3450801.

11. Yeh C.M., Zhu Y., Ulanova L., *et al.* Time series joins, motifs, discords and shapelets: A unifying view that exploits the matrix profile // Data Min. Knowl. Discov. 2018. Vol. 32, no. 1. P. 83–123. DOI: 10.1007/s10618-017-0519-9.

12. Zimmerman Z., Kamgar K., Senobari N.S., *et al.* Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond // Proceedings of the ACM Symposium on Cloud Computing, SoCC 2019, Santa Cruz, CA, USA, November 20-23, 2019. ACM, 2019. P. 74–86. DOI: 10.1145/3357223.3362721.

13. Chen X., Chen Y., He Z. Urban Traffic Speed Dataset of Guangzhou, China. 2018. DOI: 10.5281/zenodo.1205229.

14. Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

15. Rosenthal D. CVC technology on hot and cold strip rolling mills // Rev. Met. Paris. 1988. Vol. 85, no. 7. P. 597–606. DOI: 10.1051/metal/198885070597.

16. Басалаев А.А. Автоматизированный энергоменеджмент теплоэнергетического комплекса университетского городка // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2015. Т. 15, № 4. С. 26–32. DOI: 10.14529/ctcr150403.

Краева Яна Александровна, старший преподаватель, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse230202

ANOMALY DETECTION IN DIGITAL INDUSTRY SENSOR DATA USING PARALLEL COMPUTING

© 2023 Ya.A. Kraeva

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: kraevaya@susu.ru

Received: 20.09.2022

The article presents the results of case studies on the anomaly discovery in sensor data from various applications of the digital industry. The time series data obtained from the sensors installed on machine parts and metallurgical equipment, and from the temperature sensors in the smart building heating control system are considered. The anomalies discovered in such data indicate an abnormal situation or failures in the technological equipment. In this study, the anomaly is formalized as a range discord, namely a subsequence, the distance from which to its nearest neighbor is not less than the threshold prespecified by an analyst. The nearest neighbor of the given subsequence is a subsequence that does not overlap with this one and has a minimum distance to it. The discord discovery is performed through the parallel algorithm for GPU developed by the author. To visualize the anomalies found, a discord heatmap method and an algorithm for selection the most interesting discords regardless of their lengths are proposed.

Keywords: time series, sensor data, anomaly detection, discord, parallel algorithm, GPU, CUDA.

FOR CITATION

Kraeva Ya.A. Anomaly Detection in Digital Industry Sensor Data Using Parallel Computing. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 2. P. 47–61. (in Russian) DOI: 10.14529/cmse230202.

References

1. Blázquez-García A., Conde A., Mori U., Lozano J.A. A Review on Outlier/Anomaly Detection in Time Series Data. ACM Comput. Surv. 2021. Vol. 54, no. 3. 56:1–56:33. DOI: 10.1145/3444690.
2. Kumar S., Tiwari P., Zymbler M.L. Internet of Things is a revolutionary approach for future technology enhancement: a review. J. Big Data. 2019. Vol. 6. P. 111. DOI: 10.1186/s40537-019-0268-2.
3. Zymbler M.L., Kraeva Y.A., Latypova E.A., *et al.* Cleaning Sensor Data in Intelligent Heating Control System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 16–36. (in Russian) DOI: 10.14529/cmse210302.
4. Ivanov S.A., Nikolskaya K.Y., Radchenko G.I., *et al.* Digital Twin of a City: Concept Overview. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2020. Vol. 9, no. 4. P. 5–23. (in Russian) DOI: 10.14529/cmse200401.
5. Keogh E.J., Lin J., Fu A.W. HOT SAX: efficiently finding the most unusual time series subsequence. Proceedings of the 5th IEEE International Conference on Data Mining (ICDM

2005), Houston, Texas, USA, November 27-30, 2005. IEEE Computer Society, 2005. P. 226–233. DOI: 10.1109/ICDM.2005.79.

6. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA. 2007. P. 381–390. DOI: 10.1109/ICDM.2007.61.
7. Chandola V., Cheboli D., Kumar V. Detecting anomalies in a time series database. Retrieved from the University of Minnesota Digital Conservancy. 2009. URL: https://hdl.handle.net/11299/215791 (accessed: 12.04.2022).
8. Kraeva Y., Zymbler M. A parallel discord discovery algorithm for a graphics processor. Pattern Recognition and Image Analysis. 2023. Vol. 33, no. 2. P. 101–113. DOI: 10.1134/S1054661823020062.
9. Mueen A., Nath S., Liu J. Fast approximate correlation for massive time-series data. Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010. ACM, 2010. P. 171–182. DOI: 10.1145/1807167.1807188.
10. Han Z., Gao P., Wan F. Research on Data Mining and Visualization Technology. CONFCDS 2021: The 2nd International Conference on Computing and Data Science, Stanford, CA, USA, January 28-30, 2021. ACM, 2021. 71:1–71:4. DOI: 10.1145/3448734.3450801.
11. Yeh C.M., Zhu Y., Ulanova L., *et al.* Time series joins, motifs, discords and shapelets: A unifying view that exploits the matrix profile. Data Min. Knowl. Discov. 2018. Vol. 32, no. 1. P. 83–123. DOI: 10.1007/s10618-017-0519-9.
12. Zimmerman Z., Kamgar K., Senobari N.S., *et al.* Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond. Proceedings of the ACM Symposium on Cloud Computing, SoCC 2019, Santa Cruz, CA, USA, November 20-23, 2019. ACM, 2019. P. 74–86. DOI: 10.1145/3357223.3362721.
13. Chen X., Chen Y., He Z. Urban Traffic Speed Dataset of Guangzhou, China. 2018. DOI: 10.5281/zenodo.1205229.
14. Bilenko R.V., Dolganina N.Y., Ivanova E.V., Rekachinsky A.I. High-performance Computing Resources of South Ural State University. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 15–30. (in Russian) DOI: 10.14529/cmse220102.
15. Rosenthal D. CVC Technology on Hot and Cold Strip Rolling Mills. Rev. Met. Paris. 1988. Vol. 85, no. 7. P. 597–606. DOI: 10.1051/metal/198885070597.
16. Basalaev A.A. Automated Energy Management for Heat and Power System of University Campus. Bulletin of the South Ural State University. Series: Computer Technologies, Automatic Control, Radio Electronics. 2015. Vol. 15, no. 4. P. 26–32. (in Russian) DOI: 10.14529/ctcr150403.

УДК 004.021, 004.032.24, 004.051 DOI: 10.14529/cmse230203

ПРИМЕНЕНИЕ МЕТОДА ПРОЕКТИРОВАНИЯ Q-ЭФФЕКТИВНЫХ ПРОГРАММ ДЛЯ АЛГОРИТМА ДЕЙКСТРЫ

© 2023 В.Н. Алеева, П.А. Манатин

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: aleevavn@susu.ru, manatinpa@ya.ru

Поступила в редакцию: 21.10.2022

Проблема повышения эффективности параллельных вычислений чрезвычайно актуальна. В статье впервые продемонстрировано применение концепции Q-детерминанта для эффективной реализации алгоритма на графах. Концепция Q-детерминанта основана на унифицированном представлении численных алгоритмов в форме Q-детерминанта. Q-детерминант позволяет выразить и оценить внутренний параллелизм алгоритма, а также показать способ его параллельного исполнения. В работе приведены основные понятия концепции Q-детерминанта, необходимые для понимания приведенного исследования. Также описан основанный на концепции Q-детерминанта метод проектирования эффективных программ для численных алгоритмов. Результатом применения метода является программа, полностью использующая ресурс параллелизма алгоритма. Такая программа называется Q-эффективной. В качестве первого применения метода проектирования Q-эффективных программ для алгоритмов на графах описано проектирование программ для реализации алгоритма Дейкстры на параллельных вычислительных системах с общей и распределенной памятью. Приведены также результаты экспериментального исследования разработанных программ, проведенного с помощью суперкомпьютера «Торнадо ЮУрГУ». На основе анализа результатов экспериментального исследования определяются динамические характеристики разработанных программ и выявляются особенности их выполнения. Проведенные в статье исследования дают возможность сделать вывод, что применение концепции Q-детерминанта с целью разработки эффективных программ возможно не только для численных алгоритмов, но и для алгоритмов на графах.

Ключевые слова: повышение эффективности параллельных вычислений, Q-детерминант алгоритма, представление алгоритма в форме Q-детерминанта, Q-эффективная реализация алгоритма, ресурс параллелизма алгоритма, Q-эффективная программа, алгоритм Дейкстры.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Алеева В.Н., Манатин П.А. Применение метода проектирования Q-эффективных программ для алгоритма Дейкстры // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 62–77. DOI: 10.14529/cmse230203.

Введение

В настоящее время широко распространены параллельные вычислительные системы (далее — ПВС), к которым относятся, в том числе, персональные компьютеры с многоядерными процессорами. Архитектура ПВС позволяет сократить общее время решения задач с помощью последовательных программ за счет их одновременного выполнения. Однако, подобная организация работы программ во многих случаях не использует все доступные ресурсы вычислительной системы на всем промежутке времени выполнения данных программ. Для дальнейшего ускорения работы необходима разработка параллельных программ, то есть программ, производящих одновременные вычисления на нескольких ядрах или вычислительных узлах, причем эффективных программ, которые имеют наибольшую эффективность по сравнению с программами, выполняющими другие реализации данного алгоритма.

Целью данной работы является разработка эффективных программ для алгоритма Дейкстры [1] с помощью метода проектирования Q-эффективных программ на основе концепции Q-детерминанта. Для достижения цели решаются следующие задачи:

1) построение Q-детерминанта алгоритма Дейкстры;
2) описание Q-эффективной реализации алгоритма Дейкстры;
3) разработка параллельных программ для выполнимой Q-эффективной реализации алгоритма Дейкстры и исследование их динамических характеристик.

Статья относится к направлению исследований, целью которых является разработка эффективных параллельных программ, представленному работами [2–4], и вносит вклад в развитие этого направления. В данной статье описано проектирование эффективных программ, реализующих алгоритм Дейкстры на ПВС с общей и распределенной памятью.

Статья организована следующим образом. Она состоит из введения, четырех разделов и заключения. Раздел 1 содержит обзор работ по теме исследования. В разделе 2 приведены некоторые основные понятия концепции Q-детерминанта, используемые в статье, и изложен метод проектирования Q-эффективных программ [5]. В разделе 3 описано применение метода проектирования Q-эффективных программ для алгоритма Дейкстры. В разделе 4 представлены результаты экспериментального исследования динамических характеристик разработанных Q-эффективных программ, реализующих алгоритм Дейкстры. Заключение содержит краткое изложение полученных результатов и выводы об их применении.

1. Обзор работ по теме исследования

Проблеме эффективной реализации алгоритмов на ПВС посвящено много научных исследований. Приведем краткий обзор некоторых из них.

Очень важным и развитым направлением по исследованию параллельной структуры алгоритмов и программ с целью их реализации на ПВС является направление, основы которого изложены в [6, 7]. Результаты исследований данного направления используются при создании Интернет-энциклопедии AlgoWiki [8, 9]. В энциклопедии описываются свойства, особенности, статические и динамические характеристики конкретных алгоритмов. Это помогает реализовывать описанные алгоритмы эффективно. Однако в рамках данного направления программное исследование ресурса параллелизма алгоритмов не приводится. Кроме того, не предлагается технология создания параллельных программ, использующих весь ресурс параллелизма алгоритмов.

За время развития параллельных вычислений предложены различные подходы к разработке параллельных программ, созданы десятки языков параллельного программирования и множество различных инструментальных средств. Среди таких разработок отметим T-систему [10, 11]. Она представляет среду программирования с поддержкой автоматического динамического распараллеливания программ. Однако нет оснований полагать, что создаваемые с помощью T-системы параллельные программы используют ресурс параллелизма алгоритмов полностью.

Еще одним подходом к созданию параллельных программ является синтез параллельных программ. Метод синтеза параллельных программ заключается в том, чтобы из базы знаний параллельных алгоритмов конструировать новые параллельные алгоритмы для решения более крупных задач. На основе метода синтеза параллельных программ разработана технология фрагментированного программирования и реализующие ее язык и система программирования LuNA. В настоящее время это направление исследований развивает-

ся [12, 13]. Подход является универсальным, но при его применении не исследуется использование ресурса параллелизма алгоритмов.

Для преодоления ресурсных ограничений в статье [14] предлагаются методы построения параллельных архитектурно-независимых программ с использованием функционального языка программирования. Однако отсутствуют исследования касательно использования создаваемыми программами полного ресурса параллелизма алгоритмов.

Существует много исследований, в которых при разработке параллельных программ учитывается специфика алгоритмов и архитектуры ПВС. В качестве примеров таких исследований можно привести [15–20]. Эти исследования повышают эффективность реализации конкретных алгоритмов или реализации алгоритмов на ПВС конкретной архитектуры. Однако они не предлагают универсального подхода. Важно отметить, что в таких исследованиях нет информации о степени использования ресурса параллелизма алгоритма.

Реализации алгоритма Дейкстры с помощью параллельных программ посвящено несколько работ, например, [21–23]. Существенным отличием этой работы является применение для алгоритма на графах концепции Q-детерминанта, которая позволяет разработать программы, называемые Q-эффективными. Такие программы являются эффективными. В данном исследовании мы применяем для них часто используемые вычислительные инфраструктуры, т.е. условия разработки и выполнения программ. Динамические характеристики программы зависят от ресурса параллелизма реализуемого алгоритма и ее вычислительной инфраструктуры [4]. Так как вычислительных инфраструктур существует потенциально бесконечное множество, то для алгоритма не существует Q-эффективной программы с лучшими динамическими характеристиками. Однако, Q-эффективная программа наиболее эффективна для той вычислительной инфраструктуры, для которой она создавалась [4]. Для представления входных данных при вычислении кратчайших расстояний от выделенной вершины графа до всех остальных вершин в данном исследовании используется матрица смежности. Применение другого представления данных влечет то, что будет другой алгоритм для решения той же задачи. Новый алгоритм будет иметь другой Q-детерминант и, возможно, другой ресурс параллелизма алгоритма. Поэтому для сравнения наших результатов с другими нужно использовать те же алгоритм и вычислительную инфраструктуру, которые используются в нашем случае. Конечно, для данного алгоритма более хорошие динамические характеристики могут получиться при использовании другой инфраструктуры, но это будет лишь отражением выбора инфраструктуры. Следует отметить, что найти аналогичный результат практически невозможно, так как, как правило, работы не содержат описания всех характеристик вычислительной инфраструктуры.

2. Некоторые основные понятия концепции Q-детерминанта

Пусть α — алгоритм для решения алгоритмической проблемы $\bar{y} = F(N, B)$, где $N = \{n_1, n_2, \ldots, n_k\}$ — множество параметров размерности или пустое множество, B — множество входных данных, $\bar{y} = \{y_1, y_2, \ldots, y_k\}$ — множество выходных данных, $\bar{N}$ — вектор $(\bar{n}_1, \bar{n}_2, \ldots, \bar{n}_k)$, где $\bar{n}_i$ — некоторое значение параметра n_i, $\{\bar{N}\}$ — множество всевозможных векторов $\bar{N}$, Q — множество операций, используемых алгоритмом α.

Определение 1. Любое однозначное отображение $w\colon \{\bar{N}\} \to V$, где V — множество всех выражений над B и Q, называется *безусловным Q-термом.*

Определение 2. Если при любом $\bar{N} \in \{\bar{N}\}$ и любой интерпретации переменных B $w\left(\bar{N}\right)$ принимает значение логического типа, то w называется *безусловным логическим Q-термом.*

Определение 3. Пусть $u_1, u_2, ..., u_l$ — безусловные логические Q-термы, $w_1, w_2, \ldots, w_l$ — безусловные Q-термы. Множество пар (u_i, w_i), где i=$1, 2, \ldots, l$, обозначается $(\bar{u},\ \bar{w}) = \{(u_i, w_i)\}_{i=1,...,l}$ и называется *условным Q-термом длины* l. Счетное множество пар безусловных Q-термов $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,...}$ называется *условным бесконечным Q-термом*, если $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,...,l}$ является условным Q-термом для любого конечного l.

Определение 4. Под *вычислением безусловного Q-терма w при интерпретации B* следует понимать вычисление выражения $w\left(\bar{N}\right)$ при некотором $\bar{N} \in \{\bar{N}\}$. Для вычисления при заданной интерпретации B и $\bar{N} \in \{\bar{N}\}$ условного Q-терма $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,...,l}$ необходимо найти такие $u_{i_0}\left(\bar{N}\right), w_{i_0}\left(\bar{N}\right)$, что $u_{i_0}\left(\bar{N}\right)$ принимает значение *true*, а значение $w_{i_0}\left(\bar{N}\right)$ определено. В качестве значения $(\bar{u}, \bar{w})$ нужно взять $w_{i_0}\left(\bar{N}\right)$. Если установлено, что выражений $u_{i_0}\left(\bar{N}\right)$, $w_{i_0}\left(\bar{N}\right)$ не существует, то значение $(\bar{u}, \bar{w})$ для данной интерпретации B и N не определено. Вычисление условного бесконечного Q-терма определяется аналогично.

Определение 5. Предположим, что I_1, I_2, I_3 — подмножества множества $I = (1, ..., m)$ такие, что: одно или два из множеств I_i $(i = 1, 2, 3)$ могут быть пустыми, $I_1 \cup I_2 \cup I_1 = I,\ I_i \cap I_j = \varnothing$, где $i \neq j$. Множество Q-термов $\{f_i\}_{i\in I}$ удовлетворяет условиям: $f_{i_1} = w^{i_1}\ (i_1 \in I_1)$ — безусловный Q-терм, $f_{i_2} = (\bar{u}, \bar{w}) = \{(u_j^{i_2}, w_j^{i_2})\}_{\text{j=1,...,l}}\ (i_2 \in I_2)$ — условный Q-терм, $f_{i_3} = (\bar{u}, \bar{w}) = \{(u_j^{i_3}, w_j^{i_3})\}_{i_3=1,2,...}\ (i_3 \in I_3)$ — условный бесконечный Q-терм. Если алгоритм α состоит в том, что для определения y_i $(i \in I)$ требуется вычислить Q-терм f_i, то множество Q-термов f_i $(i \in I)$ называется *Q-детерминантом алгоритма*, а представление алгоритма в виде $y_i = f_i$ $(i \in I)$ — *представлением в форме Q-детерминанта* [2].

Определение 6. *Реализацией алгоритма, представленного в форме Q-детерминанта*, называется вычисление Q-термов при заданной интерпретации входных данных. *Реализация алгоритма* называется *Q-эффективной*, если выражения вида: $W\left(\bar{N}\right) = \left\{w^{i_1}\left(i_1 \in I_1\right); u_j^{i_2}\left(\bar{N}\right), w_j^{i_2}\left(\bar{N}\right)\left(i_2 \in I_2; j = 1, 2, \ldots, l_{i_2}\right); u_j^{i_3}\left(\bar{N}\right), w_j^{i_3}\left(\bar{N}\right)\left(i_3 \in I_3; j = 1, 2, \ldots, \right)\right\}$ вычисляются одновременно и операции при этом выполняются по мере готовности. Такая реализация лучше всего использует ресурс параллелизма [3].

Определение 7. *Реализация алгоритма* называется *выполнимой*, если одновременно должно выполняться конечное (непустое) множество операций. Существуют алгоритмы, Q-эффективная реализация которых невыполнима [4].

Существует метод проектирования Q-эффективных программ, то есть программ, выполняющих Q-эффективную реализацию алгоритма. Он основан на концепции Q-детерминанта и использует то, что, во-первых, любой численный алгоритм представим в форме Q-детерминанта, а во-вторых, Q-детерминант содержит Q-эффективную реализацию этого алгоритма [3].

Метод состоит из следующих этапов:

1) построение Q-детерминанта алгоритма;
2) описание Q-эффективной реализации алгоритма;

3) разработка параллельной программы для выполнимой Q-эффективной реализации алгоритма (см. определение 7).

Этот метод возможно применять для разработки Q-эффективных программ как для общей памяти, так и для распределенной памяти с применением модели «Master–Slave».

Применение данного метода подробно описано в следующем разделе.

3. Применение метода проектирования Q-эффективных программ для эффективной реализации алгоритма Дейкстры

Опишем постановку решаемой задачи. В качестве представления графа будем использовать матрицу смежности A размера $n \times n$ простого ориентированного взвешенного графа без дуг отрицательного веса, содержащего n вершин. Для отметки посещения вершины в ходе выполнения алгоритма применяется вектор посещенных вершин $\vec{p}$ размера n. Для хранения результатов промежуточных вычислений и вывода конечного результата применяется вектор расстояний $\vec{r}$ размера n.

Алгоритм реализуется следующим образом.

1. Инициализируется вектор посещенных вершин $\vec{p}$ и вектор расстояний $\vec{r}$.
2. Заполняется вектор расстояний $\vec{r}$ из матрицы смежности A значениями расстояний от начальной вершины до всех остальных вершин.
3. Если есть непосещенные вершины в векторе посещенных вершин $\vec{p}$, то среди них выбирается вершина с наименьшим значением в векторе расстояний $\vec{r}$, иначе алгоритм переходит к шагу 7.
4. Выбранная вершина помечается как посещенная в векторе посещенных вершин $\vec{p}$.
5. Для каждой вершины, посещенной ранее на предыдущих итерациях, значение в векторе расстояний $\vec{r}$ принимает значение расстояния от помеченной вершины до данной, если подобный путь существует и эта сумма меньше значения в векторе расстояний $\vec{r}$.
6. Алгоритм возвращается к шагу 3.
7. Конец алгоритма.

Таким образом, критерием завершения алгоритма является выполнение такого условия, что все компоненты вектора посещенных вершин $\vec{p}$ являются помеченными. Очевидно, что цикл из шагов 3–6 повторяется n раз.

3.1. Представление алгоритма в форме Q-детерминанта

Опишем представление в форме Q-детерминанта для алгоритма Дейкстры. В данном алгоритме используется граф, представленного в виде матрицы смежности A с количеством вершин n, где $A = [a_{ij}]_{i,j=1,\ldots,n}$ — матрица смежности простого ориентированного взвешенного графа. Пусть $\vec{p} = (p_1, \ldots, p_n)$ — вектор посещенных вершин, а $\vec{r} = (r_1, \ldots, r_n)$ — вектор расстояний от вершины r_1 до вершины r_i, где $i = 1, \ldots, n$. Процесс работы алгоритма можно представить как определенный процесс отметки вершин в векторе $\vec{p}$, за которым следует процесс заполнения по определенному правилу вектора расстояний $\vec{r}$. Таким образом, получаем алгоритм δ, реализующий алгоритм Дейкстры.

Процесс работы алгоритма δ на k-ом этапе можно представить в виде итераций:

$$r_j^k = \begin{cases} r_j^{k-1}, \text{ если } r_j^{k-1} \le r_i^k + a_{ij} \\ r_i^k + a_{ij}, \text{ если } r_j^{k-1} > r_i^k + a_{ij} \end{cases} \text{, где } j = 1, \ldots, n, j \ne i, a_{ij} \ne 0, \quad (1)$$

$\forall i \in (1, \ldots, n) : r_i^k = \min\left(r_i^{k-1}\right) \wedge p_i = \mathit{false}.$

Критерием завершения алгоритма является выполнение такого условия, что все вершины вектора $\vec{p}$ являются помеченными. Очевидно, что для достижения данного условия необходимо выполнить n этапов, после чего работа алгоритма завершается.

Этап 1. Система уравнений

$$r_j = \left\{ \left(u_1, w_1^j\right), \ldots, \left(u_{g(n)}, w_{g(n)}^j\right) \right\}_{j \in \{1,\ldots,n\}} \quad (2)$$

является представлением алгоритма δ в форме Q-детерминанта, состоящий из n условных Q-термов длины $g(n)$, где $g(n)$ — функция, определяющая длину Q-терма в зависимости от количества вершин n.

Этап 2. Опишем Q-эффективную реализацию алгоритма δ. В рамках реализации алгоритма необходимо поочередно выполнять вычисление r_j^k, $k = 1, \ldots, n$. Когда будет выполнен последний шаг, будет получено решение поставленной задачи. Q-эффективная реализация алгоритма является выполнимой, так как при реализации одновременно необходимо выполнять конечное число операций.

Этап 3. Данную реализацию можно использовать для разработки Q-эффективной программы для общей памяти. Опишем процесс реализации алгоритма δ для распределенной памяти с использованием модели «Master–Slave», при котором используется один вычислительный узел «Master» (обозначается буквой M) и несколько вычислительных узлов «Slave» (обозначаются буквой S). При подобной организации общие данные для узлов S на очередной итерации рассылаются и затем результаты вычислений от них обрабатываются только узлом M, что во многих случаях сокращает частоту обмена данными между узлами.

1. Каждая компонента вектора $\vec{r}$ вычисляется на отдельном узле S. Если количество узлов S меньше n, то узлы S должны выполнять вычисления для нескольких компонент вектора. Перед вычислением r_i^0, где $i \in \{1, 2, \ldots, n\}$, узел S получает от узла M строку матрицы A с номером i.
2. Полученные на узлах S результаты, являющиеся компонентами вектора $\vec{r}^0$, передаются на узел M для вычисления поэлементных минимумов вектора $\vec{r}^0$.
3. Аналогично шагу 1, производится вычисление $\vec{r}^1$ на узлах S.
4. Полученные на узлах S результаты, являющиеся компонентами вектора $\vec{r}^1$, передаются на узел M для вычисления поэлементных минимумов вектора $\vec{r}^1$.
5. Если компоненты вектора $\vec{r}^0$ не равны компонентам вектора $\vec{r}^1$, итерации алгоритма δ продолжаются аналогично шагам 3–5.

4. Разработка и экспериментальное исследование Q-эффективных программ

В нашем исследовании при разработке Q-эффективных программ используются:

1) язык программирования $C{+}{+}11$;

2) компилятор GCC версии 10.1.0 с поддержкой стандарта $OpenMP$ без использования опций оптимизации;
3) библиотека $OpenMPI$ версии 2.1.0, реализующая технологию MPI для ПВС с распределенной памятью.

Технология $OpenMP$ позволяет распределить итерации циклов между нитями при инициализации входных данных и обновлении значений промежуточных данных на очередном этапе выполнения алгоритма. Технология MPI используется для распределения элементов данных по вычислительным узлам, решения на данных узлах локальных задач и использования полученных результатов для решения исходной задачи по некоторым правилам.

Для выполнения Q-эффективной реализации алгоритма δ были разработаны Q-эффективные программы для общей памяти и для распределенной памяти с применением модели «Master–Slave», проектирование которых описано в разделе 3.

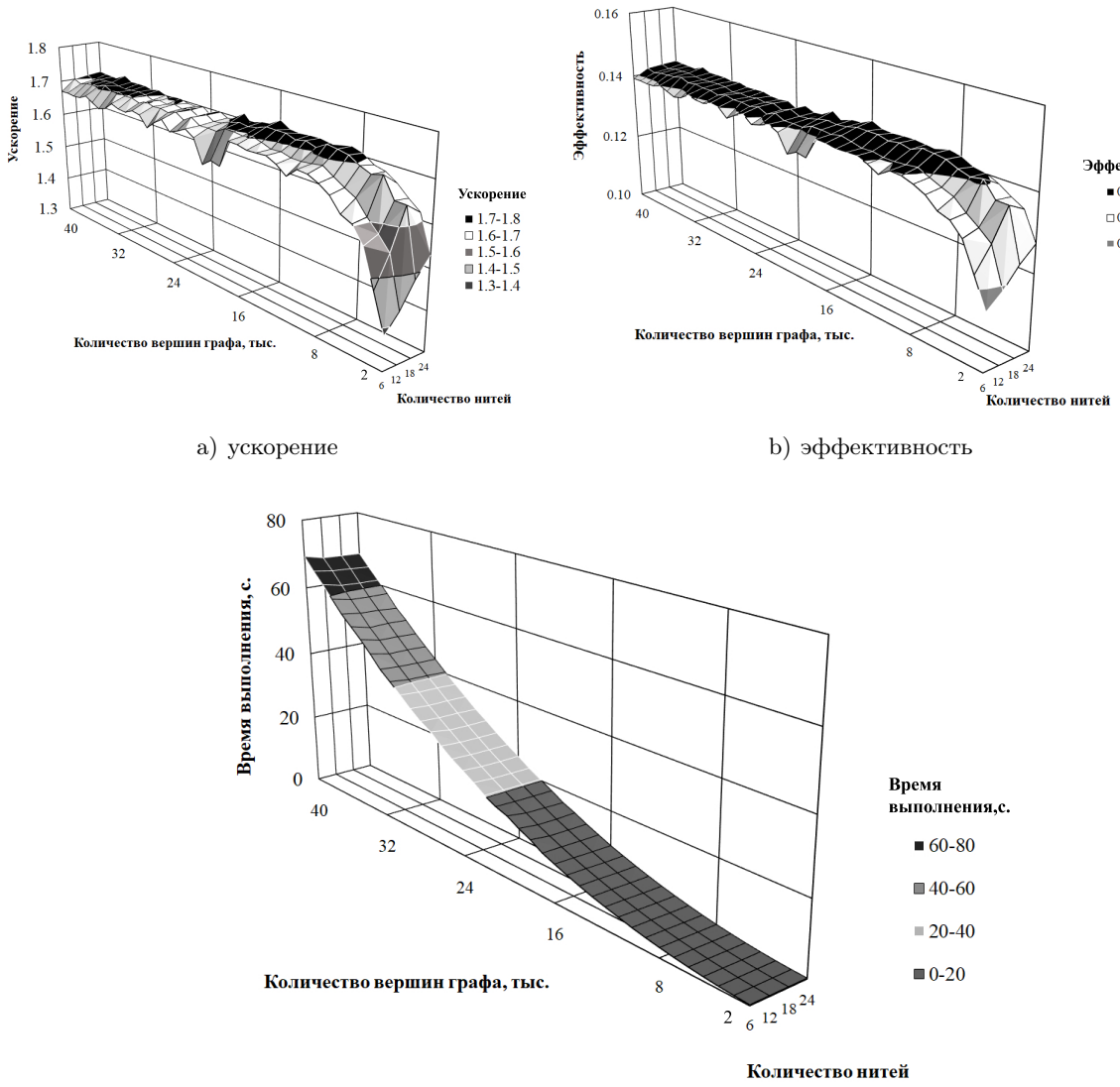
Для всех разработанных программ было проведено экспериментальное исследование динамических характеристик. Оно проводилось на суперкомпьютере «Торнадо» Южно-Уральского государственного университета. Для программы для общей памяти был использован один вычислительный узел, для распределенной памяти — несколько вычислительных узлов. В экспериментах у каждого из использованных вычислительных узлов были задействованы все имеющиеся два центральных процессора Intel Xeon X5680 с частотой 3.33 ГГц, каждый из которых имеет 6 ядер и поддерживает 12 нитей [24]. Таким образом, на каждом узле может одновременно выполняться до 24 нитей.

В экспериментах исследуются динамические характеристики программы в зависимости от количества нитей и количества вычислительных узлов (только в случае программы для распределенной памяти) на графах с разным количеством вершин. В ходе экспериментов на каждом вычислительном узле выполняется только один процесс, соответствующий рассматриваемой программе. Кроме того, в программе для общей памяти приведены результаты для разного числа нитей, исполняемых на каждом вычислительном узле. В программе для распределенной памяти на каждом вычислительном узле всегда выполняются 24 нити, что соответствует максимально возможному числу нитей, которые могут выполняться одновременно на узле. При экспериментальном исследовании находится вектор расстояний $\vec{r}$. Компоненты матрицы A формировались путем генерации случайных чисел. Измерения для каждой конфигурации проводились несколько раз.

На основании анализа полученных рядов измерений времени выполнения для определенных сочетаний конфигурации и размерности задачи можно сделать вывод об отсутствии оснований для опровержения гипотезы о подчинении этих рядов нормальному закону распределения по критерию Пирсона. На подчинение рядов нормальному закону распределения косвенно указывает тот факт, что в данных рядах выборочная средняя почти равна выборочной медиане, а мода отсутствует, так как значения в рядах не повторяются.

На основании полученных измерений времени выполнения определяются следующие динамические характеристики программ [7].

1. Ускорение — отношение времени выполнения программы для общей памяти, выполняемой на одном ядре одного вычислительного узла с использованием одной нити (назовем ее последовательной) ко времени выполнения параллельной программы.
2. Эффективность — отношение ускорения к количеству используемых параллельной программой вычислителей. Количество вычислителей, используемых программой для об-



a) ускорение

b) эффективность

c) время выполнения

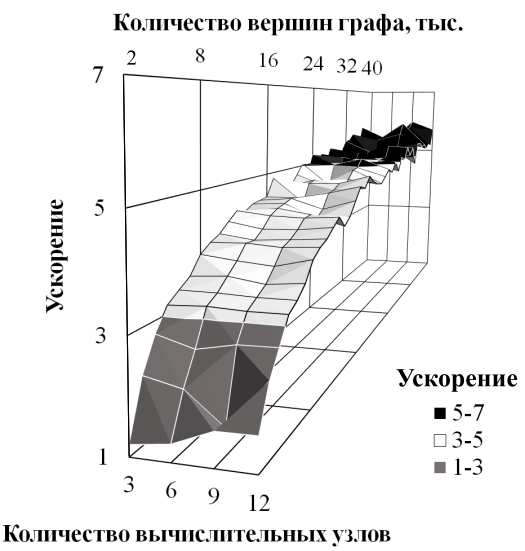
Рис. 1. Динамические характеристики программы для общей памяти

щей памяти, соответствует количеству нитей, а используемых программой для распределенной памяти — количеству вычислительных узлов.

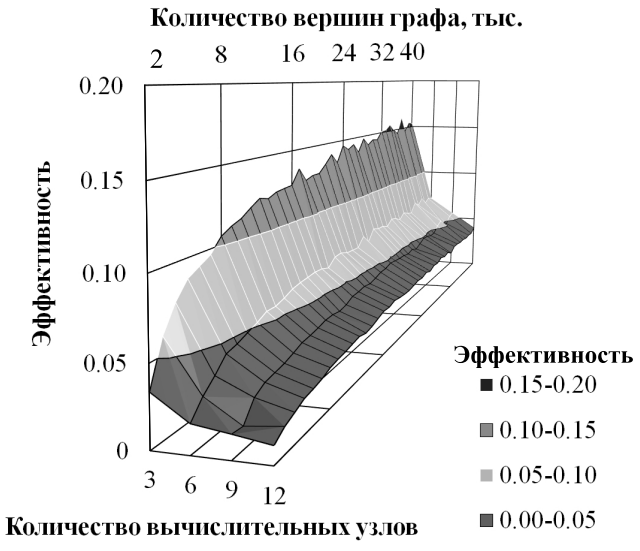
На рис. 1 показаны графики времени выполнения, ускорения и эффективности Q-эффективной программы для общей памяти. Рисунок 2 содержит графики времени выполнения, ускорения и эффективности Q-эффективной программы для распределенной памяти. На данных графиках в качестве значений указаны соответствующие значения выборочных средних.

Охарактеризуем результаты экспериментального исследования.

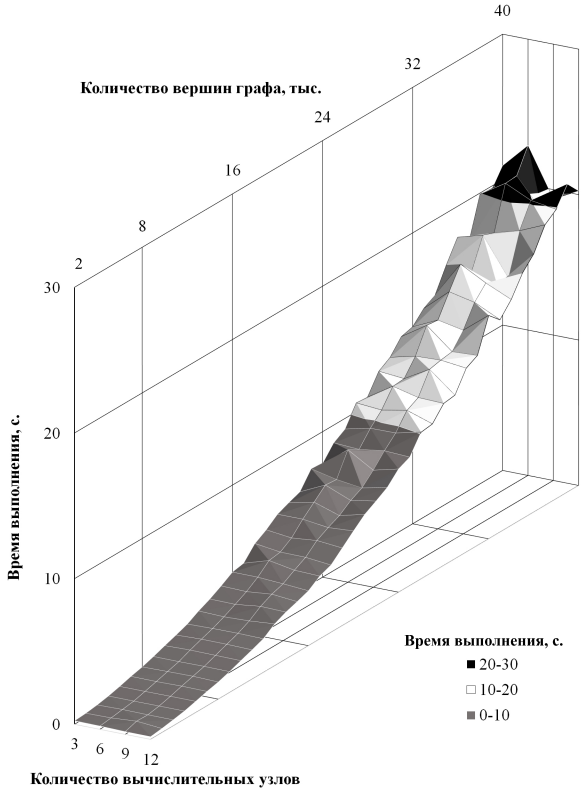
1. Величина ускорения Q-эффективной программы для общей памяти в зависимости от количества вычислителей и вершин графа варьируется в диапазоне 1.3–1.8. Минимальное ускорение достигнуто при минимальном числе вершин графа (2 тыс.) и минимальном количестве вычислителей (6 нитей), также при данном сочетании достигнута наименьшая разница во времени выполнения последовательной и параллельной программ. Ускорение возрастает по мере роста числа вершин графа, затем достигает максималь-



a) ускорение



b) эффективность



c) время выполнения

Рис. 2. Динамические характеристики Q-эффективной программы для распределенной памяти

ного значения, равного около 1.8, при работе 18 вычислителей и наличии 9 тыс. вершин графа. При таком сочетании параметров достигается наибольший эффект от использования многопоточности. Далее значение ускорения плавно уменьшается до значения около 1.6 по мере дальнейшего роста числа вершин графа, что указывает на ухудшение локальности данных алгоритма.

2. Q-эффективная программа для распределенной памяти при большинстве сочетаний параметров может продемонстрировать большее ускорение относительно Q-эффективной программы для общей памяти. Величина ускорения относительно последовательной программы в зависимости от количества вычислителей и числа вершин графа варьируется в диапазоне 1.2–6.0. Зависимость ускорения Q-эффективной программы для распределенной памяти от изучаемых факторов имеет сложный характер. Описание этой зависимости приведены в следующих пунктах.
3. Наименьшее ускорение (меньше 2) достигается в области малых по числу вершин графов. Это означает, что на малых задачах организация работы нитей отнимает большее время, чем межпроцессное взаимодействие и косвенно указывает на хорошую локальность данных алгоритма, реализованного для систем с распределенной памятью [7].
4. Наибольшее ускорение (больше 5) достигается в области больших по числу вершин графов. Так как рассматриваемый алгоритм использует матрицу смежности, то объем входных данных зависит от квадрата количества вершин графа. В таком случае возрастающее ускорение косвенно указывает на факт того, что при существенном увеличении объема входных данных накладные расходы увеличиваются незначительно, и тогда их доля в общем времени выполнения уменьшается.
5. Ускорение программы для распределенной памяти, равно как и программы для общей памяти, не зависит линейно от количества вычислителей. Это связано с тем, что по мере увеличения количества вычислителей растут и накладные расходы. В случае программы для общей памяти они связаны с организацией работы нитей, а в случае программы для распределенной памяти, кроме того — с увеличением в общем времени выполнения доли задержек в коммуникационной среде по мере роста общего числа передаваемых узлами сообщений, хоть и становящихся меньшими по объему.
6. Наиболее эффективными с точки зрения распараллеливания являются конфигурации Q-эффективной программы для распределенной памяти, использующие небольшое число узлов и большое количество нитей, т.к. при несущественном росте количества вычислителей и использовании многопоточности достигаются наименьшие накладные расходы и, как следствие, максимальный прирост ускорения.

Заключение

В статье впервые показано применение метода проектирования Q-эффективных программ, основанного на концепции Q-детерминанта, для эффективной реализации алгоритма на графах, сводящегося к численному алгоритму, на примере алгоритма Дейкстры. Для этого были решены следующие задачи:

1) построение Q-детерминанта алгоритма Дейкстры;
2) описание Q-эффективной реализации алгоритма Дейкстры;
3) разработка Q-эффективных программ для алгоритма Дейкстры, предназначенных для ПВС с общей и распределенной памятью, и их экспериментальное исследование.

Данное исследование показывает, что описание эффективной реализации возможно не только для численных алгоритмов, но и для алгоритмов других разновидностей, в частности, для алгоритмов на графах. Подобная реализация существует, если алгоритмы можно описать как численные.

Проведенное исследование пополнило коллекцию алгоритмов, для которых разработаны Q-эффективные программы и оценены их динамические характеристики. Применение

метода проектирования Q-эффективных программ решает проблему наиболее полного использования ресурса параллелизма численных алгоритмов и тем самым делает возможной эффективную реализацию численных алгоритмов на ПВС. Программная Q-система для исследования ресурса параллелизма численных алгоритмов [25, 26] метод проектирования Q-эффективных программ и технология Q-эффективного программирования [5] в комплексе являются одним из решений проблемы повышения эффективности параллельных вычислений, использующих численные алгоритмы. Их могут применять разработчики программного обеспечения для проектирования эффективных программ, предназначенных для любых ПВС — от персональных компьютеров с многоядерными процессорами до суперкомпьютеров. Это приведет к уменьшению времени выполнения программного обеспечения и более полному использованию ресурсов вычислительных систем.

Литература

1. Dijkstra E.W. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1, no. 1. P. 269–271. DOI: 10.1007/BF01386390.

2. Алеева В.Н., Алеев Р.Ж. Применение Q-детерминанта численных алгоритмов для параллельных вычислений // Параллельные вычислительные технологии (ПаВТ'2019): Труды международной научной конференции, Калининград, 2–4 апреля 2019. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2019. С. 133–145.

3. Алеева В.Н. Основные положения технологии Q-эффективного программирования // Наука ЮУрГУ. Секции технических наук: материалы 71-й научной конференции, Челябинск, апрель 2019. Челябинск: Издательский центр ЮУрГУ, 2019. С. 334–342.

4. Алеева В.Н., Шатов М.Б. Применение концепции Q-детерминанта для эффективной реализации численных алгоритмов на примере метода сопряженных градиентов для решения систем линейных уравнений // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 56–71. DOI: 10.14529/cmse210304.

5. Aleeva V.N. Improving Parallel Computing Efficiency // Proceedings – 2020 Global SmartIndustry Conference, GloSIC 2020, Chelyabinsk, Russia, November 17–19, 2020. IEEE, 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.

6. Voevodin V.V., Voevodin Vl.V. The V-Ray technology of optimizing programs to parallel computers // Numerical Analysis and Its Applications. WNAA 1996. Vol. 1196 / ed. by L.G. Vulkov, P. Yalamov, J. Wasniewski. Springer, 1997. P. 546–556. LNCS. DOI: 10.1007/3-540-62598-4_136.

7. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

8. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features // Supercomputing Frontiers and Innovations. 2015. Vol. 2, no. 1. P. 4–18. DOI: 10.14529/jsfi150101.

9. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki Project as an Extension of the Top500 Methodology // Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 1. P. 4–10. DOI: 10.14529/jsfi180101.

10. Абрамов С.М., Адамович А.И., Коваленко М. Р. Т-система – среда программирования с поддержкой автоматического динамического распараллеливания программ. Пример реализации алгоритма построения изображений методом трассировки лучей // Программирование. 1999. Т. 25, № 2. С. 100–107.

11. Абрамов С.М., Васенин В.А., Мамчиц Е.Е. и др. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы // Научная сессия МИФИ-2001, 22–26 января 2001: Сборник научных трудов. Т. 2. 2001. С. 234.

12. Malyshkin V.E., Perepelkin V.A., Schukin G.F. Distributed Algorithm of Data Allocation in the Fragmented Programming // Parallel Computing Technologies (PaCT'2015): Proceedings of the 13th International Scientific Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Proceedings. Vol. 9251 / ed. by V. Malyshkin. Springer, 2015. P. 80–85. LNCS. DOI: 10.1007/978-3-319-21909-7_8.

13. Malyshkin V.E., Perepelkin V.A., Tkacheva A.A. Control Flow Usage to Improse Performanse of Fragmented // Parallel Computing Technologies (PaCT'2015): Proceedings of the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Proceedings. Vol. 9251 / ed. by V. Malyshkin. Springer, 2015. P. 86–90. LNCS. DOI: 10.1007/978-3-319-21909-7_9.

14. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. 2005. Т. 1, № 10. С. 71–89.

15. Gurieva Y.L., Il'in V.P. On Parallel Computational Technologies of Augmented Domain Decomposition Methods // Parallel Computing Technologies (PaCT'2015): Proceedings of the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Proceedings. Vol. 9251. / ed. by V. Malyshkin. Springer, 2015. P. 35–46. LNCS. DOI: 10.1007/978-3-319-21909-7_4.

16. Schlueter M., Munetomo M. Parallelization strategies for evolutionary algorithms for MINLP // IEEE Congress on Evolutionary Computation, Cancun, Mexico, June 23–25, 2013. P. 635–641. DOI: 10.1109/CEC.2013.6557628.

17. Wang Q., Liu J., Tang X., *et al.* Accelerating embarrassingly parallel algorithm on Intel MIC // IEEE International Conference on Progress in Informatics and Computing, Shanghai, China, May 16–18, 2014. P. 213–218. DOI: 10.1109/PIC.2014.6972327.

18. Li Y., Dou W., Yang K., *et al.* Optimized Data I/O Strategy of the Algorithm of Parallel Digital Terrain Analysis // 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Xi'an, China, November 24–27, 2014. P. 34–37. DOI: 10.1109/DCABES.2014.10.

19. Prifti V., Bala R., Tafa I., *et al.* The time profit obtained by parallelization of quicksort algorithm used for numerical sorting // Science and Information Conference (SAI), London, UK, July 28–30, 2015. P. 897–901. DOI: 10.1109/SAI.2015.7237248.

20. Rajashri A. Parallelization of shortest path algorithm using OpenMP and MPI // International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, February 10–11, 2017. P. 304–309. DOI: 10.1109/I-SMAC.2017.8058360.

21. Fazio M., Buzachis A., Galletta A., *et al.* A Map-Reduce Approach for the Dijkstra Algorithm in SDN Over Osmotic Computing Systems // International Journal of Parallel Programming. 2021. Vol. 49, no. 3. P. 347–375. DOI: 10.1007/s10766-021-00693-3.

22. Zhang W., Zhang L., Chen Y. Asynchronous Parallel Dijkstra's Algorithm on Intel Xeon Phi Processor // International Conference on Algorithms and Architectures for Parallel Processing. 2018. P. 337–357. DOI: 10.1007/978-3-030-05051-1_24.

23. Jasika N., Alispahic N., Elma A.,*et al.* Dijkstra's shortest path algorithm serial and parallel execution performance analysis // 2012 Proceedings of the 35th International Convention MIPRO, Opatija, Croatia, May 21–25, 2012. P. 1811–1815. URL: `https://ieeexplore.ieee.org/document/6240942/` (дата обращения: 14.03.2023).

24. Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

25. Алеева В.Н., Зотова П.С., Склезнев Д.С. Расширение возможностей исследования ресурса параллелизма численных алгоритмов с помощью программной Q-системы // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 66–81. DOI: 10.14529/cmse210205.

26. Aleeva V., Bogatyreva E., Skleznev A., *et al.* Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. Vol. 1129 / ed. by V. Voevodin, S. Sobolev. Springer, 2019. P. 641–652. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_52.

Алеева Валентина Николаевна, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Манатин Павел Андреевич, студент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse230203

APPLICATION OF THE DESIGN METHOD FOR Q-EFFICIENT PROGRAMS IMPLEMENTING DIJKSTRA'S ALGORITHM

© 2023 V.N. Aleeva, P.A. Manatin

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: aleevavn@susu.ru, manatinpa@ya.ru

Received: 21.10.2022

The problem of improving the efficiency of parallel computing is extremely relevant. The article demonstrates the initial application of the concept of Q-determinant for the effective implementation of graph algorithms. The concept of the Q-determinant is based on a unified representation of numerical algorithms in the form of the Q-determinant. The Q-determinant allows to express and evaluate the internal parallelism of the algorithm, as well as to show the method of its parallel execution. The article gives the main notions of the Q-determinant concept necessary for better understanding of our research. Also, we describe a method of designing effective programs for numerical algorithms on the base of the concept of the Q-determinant. As a result, we obtain the program, which uses the parallelism resource of the algorithm completely, and this program is called Q-effective. As the initial application of the method for design of Q-effective programs implementing graph algorithms, we describe the designing programs for Dijkstra's algorithm implementation on parallel computing systems with shared and distributed memory. Finally, for the developed programs, we present the results of experiments on the Tornado SUSU supercomputer. Analyzing the results of the experimental study, we determine the effectiveness of the developed programs and identify features of their execution. The research described in the article leads to the conclusion that the application of the Q-determinant concept for the development of effective programs is possible not only for numerical algorithms, but also for graph algorithms.

Keywords: improving parallel computing efficiency, Q-determinant of algorithm, representation of algorithm in the form of Q-determinant, Q-effective implementation of algorithm, parallelism resource of algorithm, Q-effective program, Dijkstra's algorithm.

FOR CITATION

Aleeva V.N., Manatin P.A. Application of the Design Method for Q-effective Programs Implementing Dijkstra's Algorithm. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 2. P. 62–77. (in Russian) DOI: 10.14529/cmse230203.

References

1. Dijkstra E.W. A note on two problems in connexion with graphs. Numerische Mathematik. 1959. Vol. 1, no. 1. P. 269–271. DOI: 10.1007/BF01386390.

2. Aleeva V.N., Aleev R.Zh. Application of the Q-determinant of numerical algorithms for parallel computing. Parallel Computational Technologies (PCT'2019): Proceedings of the International Conference, Kaliningrad, Russia, April 2–4, 2019. Short articles and poster descriptions. Chelyabinsk: SUSU Publishing Center, 2019. P. 133–145. (in Russian)

3. Aleeva V.N. The fundamentals of Q-effective programming technology. Science of SUSU. Sections of Technical Sciences: materials of the 71st Scientific Conference, Chelyabinsk,

Russia, April 2019. Chelyabinsk: SUSU Publishing Center, 2019. P. 334–342. (in Russian)

4. Aleeva V.N., Shatov M.B. Application of the Q-determinant Concept for Efficient Implementation of Numerical Algorithms by the Example of the Conjugate Gradient Method for Solving Systems of Linear Equations. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 56–71. (in Russian) DOI: 10.14529/cmse210304.
5. Aleeva V.N. Improving Parallel Computing Efficiency. Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020, Chelyabinsk, Russia, November 17–19, 2020. IEEE, 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.
6. Voevodin V.V., Voevodin Vl.V. The V-Ray technology of optimizing programs to parallel computers. Numerical Analysis and Its Applications. WNAA 1996. Vol. 1196 / ed. by L.G. Vulkov, P. Yalamov, J. Wasniewski. Springer, 1997. P. 546–556. LNCS. DOI: 10.1007/3-540-62598-4_136.
7. Voevodin V.V., Voevodin Vl.V. Parallel Computing. St. Petersburg: BHV-Petersburg, 2002. 608 p. (in Russian)
8. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. Supercomputing Frontiers and Innovations. 2015. Vol. 2, no. 1. P. 4–18. DOI: 10.14529/jsfi150101.
9. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki Project as an Extension of the Top500 Methodology. Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 1. P. 4–10. DOI: 10.14529/jsfi180101.
10. Abramov S.M., Adamovich A.I., Kovalenko M. R. T-system programming environment with support for automatic dynamic parallelization of programs. An example of the implementation of an algorithm for constructing images by ray tracing. Programming. 1999. Vol. 25, no. 2. P. 100–107. (in Russian)
11. Abramov S.M., Vasenin V.A., Mamchits E.E., *et al.* Dynamic parallelization of programs based on parallel graph reduction. Software architecture of the new version of the T-system. Scientific session of MEPhI–2001, January 22–26, 2001: Collection of scientific papers. Vol. 2. 2001. P. 234. (in Russian)
12. Malyshkin V.E., Perepelkin V.A., Schukin G.F. Distributed Algorithm of Data Allocation in the Fragmented Programming. Parallel Computing Technologies (PaCT'2015): Proceedings of the 13th International Scientific Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Proceedings. Vol. 9251 / ed. by V. Malyshkin. Springer, 2015. P. 80–85. LNCS. DOI: 10.1007/978-3-319-21909-7_8.
13. Malyshkin V.E., Perepelkin V.A., Tkacheva A.A. Control Flow Usage to Improse Performanse of Fragmented. Parallel Computing Technologies (PaCT'2015): Proceedings of the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Proceedings. Vol. 9251 / ed. by V. Malyshkin. Springer, 2015. P. 86–90. LNCS. DOI: 10.1007/978-3-319-21909-7_9.
14. Legalov A.I. Functional language for architecturally independent parallel programs creating. Computational Technologies. 2005. Vol. 1, no. 10. P. 71–89. (in Russian)
15. Gurieva Y.L., Il'in V.P. On Parallel Computational Technologies of Augmented Domain Decomposition Methods. Parallel Computing Technologies (PaCT'2015): Proceedings of

the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Proceedings. Vol. 9251. / ed. by V. Malyshkin. Springer, 2015. P. 35–46. LNCS. DOI: 10.1007/978-3-319-21909-7_4.

16. Schlueter M., Munetomo M. Parallelization strategies for evolutionary algorithms for MINLP. IEEE Congress on Evolutionary Computation, Cancun, Mexico, June 23–25, 2013. P. 635–641. DOI: 10.1109/CEC.2013.6557628.

17. Wang Q., Liu J., Tang X., *et al.* Accelerating embarrassingly parallel algorithm on Intel MIC. IEEE International Conference on Progress in Informatics and Computing, Shanghai, China, May 16–18, 2014. P. 213–218. DOI: 10.1109/PIC.2014.6972327.

18. Li Y., Dou W., Yang K., *et al.* Optimized Data I/O Strategy of the Algorithm of Parallel Digital Terrain Analysis. 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Xi'an, China, November 24–27, 2014. P. 34–37. DOI: 10.1109/DCABES.2014.10.

19. Prifti V., Bala R., Tafa I., *et al.* The time profit obtained by parallelization of quicksort algorithm used for numerical sorting. Science and Information Conference (SAI), London, UK, July 28–30, 2015. P. 897–901. DOI: 10.1109/SAI.2015.7237248.

20. Rajashri A. Parallelization of shortest path algorithm using OpenMP and MPI. International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, February 10–11, 2017. P. 304–309. DOI: 10.1109/I-SMAC.2017.8058360.

21. Fazio M., Buzachis A., Galletta A., *et al.* A Map-Reduce Approach for the Dijkstra Algorithm in SDN Over Osmotic Computing Systems. International Journal of Parallel Programming. 2021. Vol. 49, no. 3. P. 347–375. DOI: 10.1007/s10766-021-00693-3.

22. Zhang W., Zhang L., Chen Y. Asynchronous Parallel Dijkstra's Algorithm on Intel Xeon Phi Processor International Conference on Algorithms and Architectures for Parallel Processing. 2018. P. 337–357. DOI: 10.1007/978-3-030-05051-1_24.

23. Jasika N., Alispahic N., Elma A., *et al.* Dijkstra's shortest path algorithm serial and parallel execution performance analysis. 2012 Proceedings of the 35th International Convention MIPRO, Opatija, Croatia, May 21–25, 2012. P. 1811–1815. URL: `https://ieeexplore.ieee.org/document/6240942/` (accessed: 14.03.2023).

24. Bilenko R.V., Dolganina N.Yu., Ivanova E.V., Rekachinsky A.I. High-performance computing resources of South Ural State University. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 15–30. DOI: 10.14529/cmse220102.

25. Aleeva V.N., Zotova P.S., Skleznev D.S. Advancement of research for the parallelism resourceof numerical algorithms with help of software Q-system. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 2. P. 66–81. (in Russian) DOI: 10.14529/cmse210205.

26. Aleeva V., Bogatyreva E., Skleznev A., *et al.* Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism. Supercomputing. Vol. 1129 / ed. by V. Voevodin, S. Sobolev. Springer, 2019. P. 641–652. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_52.

DOI: 10.14529/cmse230204

SOLVING GRID EQUATIONS USING THE ALTERNATING-TRIANGULAR METHOD ON A GRAPHICS ACCELERATOR*

© 2023 A.I. Sukhinov[1], V.N. Litvinov[1,2], A.E. Chistyakov[1], A.V. Nikitina[1,3], N.N. Gracheva[1,2], N.B. Rudenko[1,2]

[1]*Don State Technical University (Gagarin Sq. 1, Rostov-on-Don, 344003 Russia),*
[2]*Azov-Black Sea Engineering Institute of Don State Agrarian University (Lenina 21, Zernograd, 347740 Russia),*
[3]*Southern Federal University (Bolshaya Sadovaya 105/42, Rostov-on-Don, 344006 Russia)*
E-mail: sukhinov@gmail.com, litvinovvn@rambler.ru, cheese_05@mail.ru, nikitina.vm@gmail.com, 79286051374@yandex.ru, nelli-rud@yandex.ru
Received: 15.03.2023

The paper describes a parallel-pipeline implementation of solving grid equations using the modified alternating-triangular iterative method (MATM), obtained by numerically solving the equations of mathematical physics. The greatest computational costs at using this method are on the stages of solving a system of linear algebraic equations (SLAE) with lower triangular and upper non-triangular matrices. An algorithm for solving the SLAE with a lower triangular matrix on a graphics accelerator using NVIDIA CUDA technology is presented. To implement the parallel-pipeline method, a three-dimensional decomposition of the computational domain was used. It is divided into blocks along the y coordinate, the number of which corresponds to the number of GPU streaming multiprocessors involved in the calculations. In turn, the blocks are divided into fragments according to two spatial coordinates — x and z. The presented graph model describes the relationship between adjacent fragments of the computational grid and the pipeline calculation process. Based on the results of computational experiments, a regression model was obtained that describes the dependence of the time for calculation one MATM step on the GPU, the acceleration and efficiency for SLAE solution with a lower triangular matrix by the parallel-pipeline method on the GPU were calculated using the different number of streaming multiprocessors.

Keywords: mathematical modeling, parallel algorithm, graphics accelerator.

FOR CITATION

Sukhinov A.I., Litvinov V.N., Chistyakov A.E., Nikitina A.V., Gracheva N.N., Rudenko N.B. Solving Grid Equations Using the Alternating-triangular Method on a Graphics Accelerator. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 2. P. 78–92. DOI: 10.14529/cmse230204.

Introduction

Modeling of any physical processes occurring in the environment and their mathematical description leads to the necessity to solve differential equations in private derivatives. To study dynamic processes in hydrophysics and hydrodynamics, the diffusion-convection-reaction equation is used [1, 2].

The solution to the equations of mathematical physics is based on the approximation of equations of end-and-character schemes. In the case of the use of an implicit, non-exposure scheme, the solution of equation is reduced to solving the system of linear algebraic equations of a large dimension. The largest computational costs in solving differential equations are at solution to the indicated SLAE, therefore, various iterative methods and algorithms are developed and

*The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2023".

applied [3–6]. One of the effective iterative methods for solving SLAE is the alternating-triangular method. This method is applicable to the high-dimensional SLAE with self-adjoint and non-self-adjoint operators, and has a high convergence rate. The iterative alternating-triangular method is used to solve ill-conditioned SLAE.

The dimensions of resulting SLAE are such that they require large computing performance. Problems of computing performance lack are solved in several ways, in particular, using graphic accelerators (GPUs) for calculation the resources and computing processes [3, 7–12].

Russian scientists applied computational grid decomposition in solving a three-dimensional boundary value problem. The parallelization algorithm was implemented in a heterogeneous computing environment. Due to the use of graphic accelerations, the calculation time was reduced in 60 times, compared to the calculations on the CPU [7]. The three-phase filtration problem was also solved in heterogeneous computing environment using the decomposition of computational domain. The parallel algorithm is performed on C++ with using the CUDA and MPI technology. Due to the use of GPU for calculating the specified problem, the computational costs reduces in several tens of times [3]. Scientists of China University of Petroleum propose a parallel algorithm for modeling hardening in two dimensions based on the domain decomposition. This algorithm was implemented on GPU, which significantly reduces the calculation time [8]. Researchers of Altai State University have developed a parallel algorithm for a numerical solution to the problem of electromagnetic impulse spreading in two-dimensional rectangular field. The algorithm was implemented on the basis of CUDA technology. An analysis of the performance of the developed algorithm showed that the GPU performance is several times higher than the CPU performance at solving the problem [9]. The effectiveness of the use of graphic accelerators for numerical modeling of the tasks of applied hydrodynamics has been proved. The calculation rate when solving the problem of numerical modeling of the hydrodynamic characteristics of mushroom screws was increased 1.4–3 times [10]. The exact calculation of heat transfer coefficients requires powerful computing resources that are not available. The parallelization algorithm for such calculations with implementation in heterogeneous computing environment increases productivity, compared to the CPU, more than ten times [12].

In this paper, the decomposition method of three-dimensional calculated circle to implement the parallel algorithm on the GPU is proposed. The developed parallel-conveyor method for SLAE solving allows to effectively use the GPU resources and reduce the calculation time.

1. Method of Solving Grid Equations

Solving the equations of mathematical physics can be reduced to solving a system of linear algebraic equations of the form:

$$Ax = f,\ A : H \to H, \tag{1}$$

where A is the linear, positive definite operator.

For the grid equation (1), the iterative methods are used, which in canonical form can be represented by the equation [1, 2]:

$$B\frac{x^{m+1} - x^m}{\tau_{m+1}} + Ax^m = f,\ B : H \to H, \tag{2}$$

where m is the iteration number; $\tau_{m+1} > 0$ is the iteration parameter; B is the preconditioner.

The resulting grid equations will be solved using the modified alternating-triangular method of variational type. The preconditioner is formed as follows:

$$B = (D + \omega R_1)\, D^{-1}\, (D + \omega R_2)\,,\ D = D^* > 0,\ \omega > 0, \tag{3}$$

where D is the diagonal operator; R_1, R_2 are the lower- and upper-triangular operators, respectively.

The calculation algorithm of grid equations by the modified alternating-triangular method of the variational type is written in the form:

$$r^m = Ax^m - f,$$

$$(D + \omega R_1)y^m = r^m,\ (D + \omega R_2)w^m = Dy^m,$$

$$\tilde{\omega}_m = \sqrt{\frac{(Dw^m, w^m)}{(D^{-1}R_2 w^m, R_2 w^m)}},$$

$$s_m^2 = 1 - \frac{(A_0 w^m, w^m)^2}{(B^{-1}A_0 w^m)\,(Bw^m, w^m)},\ k_m^2 = \frac{\left(B^{-1}A_1 w^m, A_1 w^m\right)}{(B^{-1}A_0 w^m, A_0 w^m)}, \tag{4}$$

$$\theta_m = \frac{1 - \sqrt{\frac{s_m^2\, k_m^2}{(1+k_m^2)}}}{1 + k_m^2\,(1 - s_m^2)},\ \tau_{m+1} = \theta_m \frac{(A_0 w^m, w^m)}{(B^{-1}A_0 w^m, A_0 w^m)},$$

$$x^{m+1} = x^m - \tau_{m+1} w^m,\ \omega_{m+1} = \tilde{\omega}_m,$$

where r^m is the residual vector; w^m is the correction vector; the parameter s_m describes the convergence rate of the method; k_m describes the ratio of the norm of the skew-symmetric operator part to the norm of the symmetric part.

The most labors part of the algorithm is the calculation of the correction vector w^m and reduced to the solution of two SLAE with the lower-triangular and upper-triangular matrix:

$$(D + \omega R_1)y^m = r^m,\ (D + \omega R_2)w^m = Dy^m.$$

The algorithm fragment for solving SLAE with the lower-triangular matrix is given in Algorithm 1. The residual vector is calculated in $14N$ arithmetic operations. The total number of

Algorithm 1 matm(IN: $n_1, n_2, n_3, a_0, a_2, a_4, a_6, \omega$; IN/OUT: r)

```
1: for k ∈ [1; n3 − 2] do
2:     for i ∈ [1; n1 − 2] do
3:         for j ∈ [1; n2 − 2] do
4:             p0 ← i + n1 · j + n1 · n2 · k
5:             if a0[p0] > 0 then
6:                 p2 ← p0 − 1; p4 ← p0 − n1; p6 ← p0 − n1 · n2
7:                 r[p0] ← (ω · (a2[p0] · r[p2] + a4[p0] · r[p4] + a6[p0] · r[p6]) + r[p0])/((0.5 · ω +
                   1) · a0[p0])
```

arithmetic operations required to solve the SLAE with the seven-diagonal matrix using MATM in the case of known iterative parameters τ_{m+1}, ω_{m+1} is $35N$, where $N = n_1 n_2 n_3$ is SLAE dimension.

2. Decomposition Model of Computational Domain

Let Q be the set of technical characteristics of the video adapter, then we will present the characteristics of the video adapter in the form of a tuple.

$$Q = \left\langle q^1, q^2 \right\rangle, \tag{5}$$

where q^1 is the amount of video memory of the video adapter, GB; q^2 is the number of streaming multiprocessors.

If S is a set of program threads involved in the computational process, then

$$S = \left\{ s_k, k = \overline{1, q^2} \right\}, \tag{6}$$

where s_k is a CUDA streaming block that implements the calculation process on GPU streaming multiprocessor with index k.

Let us take the computational domain with the following parameters: l_x is the characteristic size on the axis Ox, l_y — on the axis Oy, l_z — on the axis Oz.

Let us compare the specified area with a uniform computational grid of the following type:

$$\begin{aligned} W = \{ & x_i = ih_x, y_j = jh_y, z_k = kh_z; \\ & i = \overline{0, n_x - 1}, j = \overline{0, n_y - 1}, k = \overline{0, n_z - 1}; \\ & (n_x - 1)h_x = l_x, (n_y - 1)h_y = l_y, (n_z - 1)h_z = l_z \}, \end{aligned} \tag{7}$$

where h_x, h_y, h_z are the steps of computational grid at the corresponding spatial directions; n_x, n_y, n_z are the number of grid nodes at the corresponding spatial directions.

Then the set of nodes of the computational grid can be represented as

$$\begin{aligned} G = \left\{ g_{i,j,k}, i = \overline{0, n_x - 1}, j = \overline{0, n_y - 1}, k = \overline{0, n_z - 1} \right\}, \\ g_{i,j,k} = \left\langle x_i, y_j, z_k \right\rangle, \end{aligned} \tag{8}$$

where $g_{i,j,k}$ is the grid node.

The number of nodes of the computational grid N_G is calculated by the formula:

$$N_G = n_x \cdot n_y \cdot n_z, \tag{9}$$

Under the block of the computational grid $G^{k_1} \subset G$ (further – the block) we will understand the sub-set of nodes of the computational grid G.

$$G = \bigcup_{k_1 \in K_{k_1}} G^{k_1} = \{ g^{k_1} | \exists k_1 \in K_{k_1}, g^{k_1} \in G^{k_1} \}, \bigcap_{k_1 \in K_{k_1}} G^{k_1} = \oslash, \tag{10}$$

where $K_{k_1} = \{1, ..., N_{k_1}\}$ is the set of block indices G^{k_1} of the computational grid G ; N_{k_1} is the number of blocks G_{k_1}, $N_{k_1} = d^2$; $K_{k_1}, N_{k_1} \subset N$; N is the set of natural numbers; k_1 is the block index G^{k_1}.

Since $G^{k_1} \subset G$, then

$$G^{k_1} = \left\{ g^{k_1}_{\widetilde{i},\widetilde{j},k}, i = \overline{0, n_x - 1}, j = \overline{0, n_y^{k_1} - 1}, k = \overline{0, n_z - 1} \right\}, \tag{11}$$

where $g^{k_1}_{i,\widetilde{j},k}$ block node k_1; the $\sim$ sign denotes belonging to the block; $\widetilde{j}$ is the block node index k_1 at coordinate y; $n^{k_1}_y$ is the number of nodes in block k_1 at coordinate y.

$$g^{k_1}_{i,\widetilde{j},k} = \langle x_i, y_j, z_k \rangle \,,\; x_i = ih_x, y_j = \left(\sum_{b=1}^{k_1-1} n^b_y + \widetilde{j} \right) \cdot h_y, z_k = kh_z, \tag{12}$$

where n^b_y is the number of nodes at coordinate y of the b-th block.

Under the fragment of the computational grid G^{k_1,k_2} (further — the fragment) we will understand a subset of the nodes of the computational grid of block G^{k_1}.

$$G^{k_1} = \bigcup_{k_2 \in K_{k_1,k_2}} G^{k_1,k_2} = \{ g^{k_1,k_2} | \exists k_2 \in K_{k_1,k_2}, g^{k_1,k_2} \in G^{k_1,k_2} \},$$

$$\bigcap_{k_2 \in K_{k_1,k_2}} G^{k_1,k_2} = \oslash, \tag{13}$$

where $K_{k_1,k_2} = \{1, ...N_{k_1,k_2}\}$ is a plurality of fragment indexes G^{k_1,k_2} of block G^{k_1}; N_{k_1,k_2} is the number of fragments G^{k_1,k_2}; $K_{k_1,k_2}, N_{k_1,k_2} \subset N$; k_2 is the index of fragment G^{k_1,k_2} of block G^{k_1}.

Since $G^{k_1,k_2} \subset G^{k_1}$ then

$$G^{k_1,k_2} = \left\{ \breve{g}_{\breve{i},\widetilde{j},\breve{k}}, \breve{i} = \overline{0, \breve{n}_x - 1}, \widetilde{j} = \overline{0, \widetilde{n}_y - 1}, \breve{k} = \overline{0, \breve{n}_z - 1} \right\}, \tag{14}$$

where $\breve{g}_{\breve{i},\widetilde{j},\breve{k}}$ is the fragment node; the sign $\smile$ denotes belonging to a fragment; $\breve{i}, \breve{k}$ are indexes of fragment node by coordinates x, z; $\breve{n}_x, \breve{n}_z$ is the number of nodes of the computational grid in the fragment along the coordinates x, z.

Each index k_2 of fragment G^{k_1,k_2} is associated with a tuple of indices $\langle k_3, k_4 \rangle$, designed to store fragment coordinates in plane xOz, where k_3 is the fragment index at coordinate x, k_4 is the fragment index at coordinate z.

$$k_2 = k_3 + K_{k_3} \cdot k_4, \tag{15}$$

where k_3 is the fragment index along the x coordinate; k_4 is the fragment index along the z coordinate; K_{k_3} is the number of fragments along the Ox axis.

The number of fragments G^{k_1,k_2} block G^{k_1} is calculated by the formula

$$K_{k_2} = K_{k_3} \cdot K_{k_4}, \tag{16}$$

where K_{k_4} is the number of fragments at coordinate z.

$$\breve{g}_{\breve{i},\widetilde{j},\breve{k}} = \langle x_i, y_j, z_k \rangle$$

$$x_i = \left(\sum_{b=1}^{k_3-1} \breve{n}_b + \breve{i} \right) \cdot h_x, y_j = \widetilde{j} h_y, z_k = \left(\sum_{b=1}^{k_4-1} \breve{n}_b + \breve{k} \right) \cdot h_z, \tag{17}$$

where $\breve{n}_b$ is the number of nodes in the b-th fragment.

Let us introduce a set of comparisons of computational grid blocks with program currents M.

$$M = \left\{ m_{k_1} = \left\langle G^{k_1}, s_{k_1} \right\rangle, k_1 \in K_{k_1} \right\}, \tag{18}$$

where $s_{k_1} \in S$ — program flow, calculating block G^{k_1}.

For the domain decomposition, it is necessary to take into account the computing performance of device, involved in calculations. Performance refers to the number of nodes of the computational grid, calculated using a given algorithm, per unit of time.

To calculate the number of nodes along the coordinate y in the blocks of the computational grid processed by GPU streaming multiprocessors, we use the formulas

$$n_{yGT} = \left\lfloor \frac{n_y}{N_{k_1} - 1} \right\rfloor, n_{yGTL} = n_y - \sum_{b=1}^{N_{k_1}-1} n_{yGT}^b, \tag{19}$$

where n_{yGT} is the number of computational grid nodes along coordinate y in blocks processed by GPU streaming multiprocessors, except for the last block; n_{yGTL} is the number of nodes at coordinate y in the last block of the computational grid processed by GPU streaming multiprocessors.

The number of the computational grid fragments along the coordinate y is equal to

$$N_y^f = N_{k_1}. \tag{20}$$

Let the number of fragments be N_x^f and N_z^f at coordinates x and z, respectively. Then, the number of nodes of the computational grid along the coordinate x is calculated by the formulas:

$$n_x^f = \left\lfloor \frac{n_x}{N_x^f - 1} \right\rfloor, \; n_x^{fL} = n_x - n_x^f \cdot (N_x^f - 1), \tag{21}$$

where n_x^f is the number of nodes of the computational grid along the coordinate x in all fragments except the last one; n_x^{fL} — the number of nodes of the computational grid along the coordinate x in the last fragment.

Similarly, the number of nodes of the computational grid along the coordinate z is calculated

$$n_z^f = \left\lfloor \frac{n_z}{N_z^f - 1} \right\rfloor, \; n_z^{fL} = n_z - n_z^f \cdot (N_z^f - 1), \tag{22}$$

where n_z^f is the number of nodes of the computational grid along the coordinate z in all fragments except the last one; n_z^{fL} — the number of nodes of the computational grid along the coordinate z in the last fragment.

Let on M it is necessary to organize a parallel process for calculation some function F, and the calculations in each fragment G^{k_1,k_2} depend on the values in neighboring fragments, each of which has at least one of the indices at coordinates x, y and z one less than the current one.

To organize a parallel-pipelining method, let us introduce a set of tuples A that define correspondences a between program flows s_k, processing fragments G^{k_1,k_2}, and the numbers of steps of the parallel-pipelining method r.

$$\forall s_k \in S \; \exists a \in A : a = \left\langle s_k, G^{k_1,k_2}, r \right\rangle, \tag{23}$$

where $r = \overline{1, N_r}$ is the step number of the parallel-pipeline method, N_r is the number of steps of the parallel-pipeline method, calculated by the formula

$$N_r = N_x^f N_z^f + N_y^f - 1. \tag{24}$$

Full download of all calculators in the proposed parallel-pipeline method starts from step $r_{100START} = N_y^f$ and ends at the step $r_{100STOP} = N_x^f N_z^f$. In this case, the total number of steps

with a full load of N_{rPAR} calculators will be

$$N_{rPAR} = r_{100STOP} - r_{100START} + 1 = N_x^f N_z^f - N_y^f + 1. \quad (25)$$

The calculation time of some function F by the parallel-pipeline method can be written in the form

$$T_M = \sum_{r=1}^{N_r} max(T_a), \quad (26)$$

where T_a is a vector of time values for fragment processing in parallel mode.

3. Parallel Implementation

The numerical implementation of the MATM for solving SLAE with the high dimension is based on the developed parallel algorithms that implement the pipeline computing process. The use of these algorithms allows to fully utilize all available streaming multiprocessors of graphics accelerator.

A class library was developed in C++ for describing the domain decomposition. The class library contains the following classes:

- Grid3D, describes the parameters of the computational grid (number of nodes n_x, n_y, n_z, and step sizes h_x, h_y, h_z in spatial coordinates) and contains an array of objects of the GridBlock3D class.
- GridBlock3D, describes the parameters of the computational grid block and contains an array of objects of the GridFragment3D class.
- GridFragment3D, describes the parameters of a computational grid fragment and contains data arrays.

The organization of calculations is performed by an algorithm that controls all available streaming multiprocessors of GPU (calculators). Each calculator performs calculations only for its own block of the computational domain. For this, the computational domain is divided into blocks that are assigned to individual calculators (Fig. 1). Next, each block is divided into fragments. Notations in Fig. 1: SM_1, SM_2, SM_3 are streaming multiprocessors of GPU.

A graph model was used to describe the relationships between adjacent fragments of the computational grid and the organization of the pipeline calculation process (Fig. 2). Each graph node is an object of a class GridFragment3D that describes a fragment of the computational domain. This class contains the following fields: the dimensions of the fragment along the Ox, Oy, and Oz axes; the index of the zero node of the fragment in the global computational domain; pointers to adjacent fragments of the computational grid; pointers to objects that describe the parameters of calculators. The computational process is a graph traversal from the root node with parallel launch of calculators that process the graph nodes in accordance with the value of the calculation step counter r.

An algorithm and its program implementation in the CUDA C language are developed to improve the calculation efficiency of the computational grid fragments assigned to the graphics accelerator [13–17].

We present an algorithm for searching the solution for the system of equations with the lower-triangular matrix (straight line) on CUDA C.

The input parameters of the algorithm are the vectors of the coefficients of grid equations a_0, a_2, a_4, a_6 and the constant ω. The output parameter is the vector of the water flow velocity v. Before running the algorithm, it is necessary to programmatically set the dimensions of the

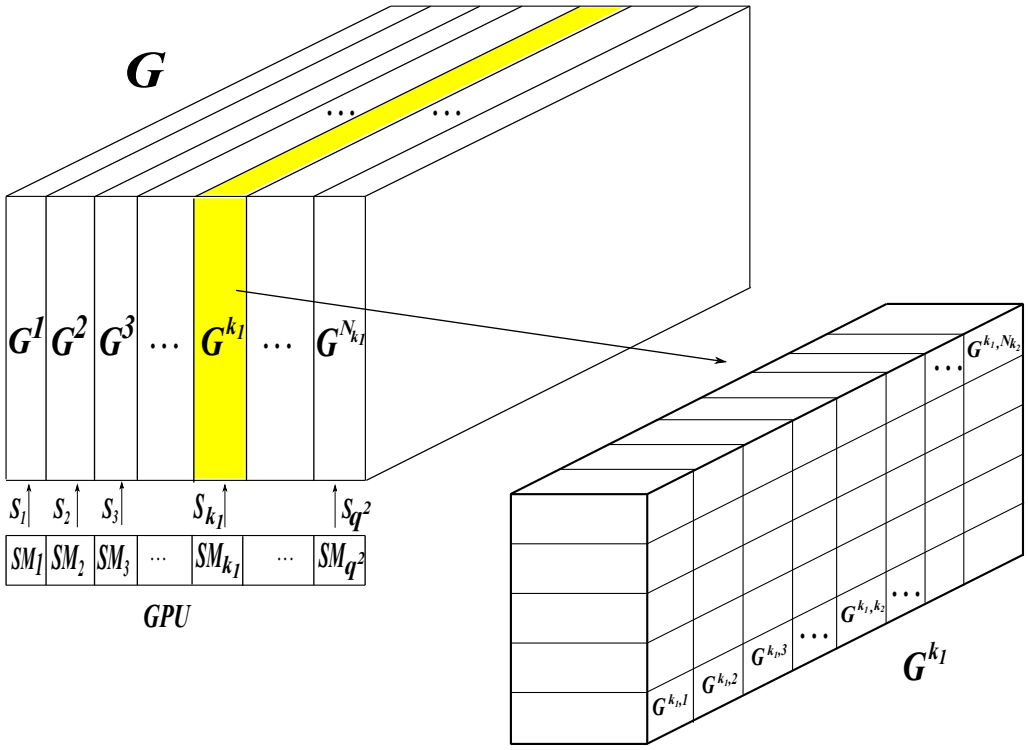


Fig. 1. Decomposition of the third-dimensional computational domain

CUDA computing block $blockDim.x$, $blockDim.z$ according to the spatial coordinates x, z, respectively. The CUDA framework runs this algorithm for each thread, and the variable values $threadIdx.x$, $threadIdx.z$, $blockIdx.x$, $blockIdx.z$ are automatically initialized by the indexes of the corresponding threads and blocks. Global thread indexes are calculated in rows 1 and 2. The row index i and the layer index k that the current thread processes are calculated in row 3. A variable j is initialized that represents a counter by coordinate y. The calculation pipeline is organized as a loop in line 4. The indexes of the central node of the grid pattern p_0 and the surrounding nodes p_2, p_4, p_6 are calculated in line 8. The two-dimensional array *cache* is located in the GPU shared memory and designed to store the calculation results on the current layer by the coordinate y. This allows us to reduce the number of reads from slow global memory and accelerate the calculation process by up to 30 %.

The performed researches show a significant dependence of the algorithm implementation time for calculation the preconditioner on the ratio of threads in spatial coordinates. A series of experiments is preperformed to calculate the performance of calculators, which is the 95th percentile of the calculation time in terms of 1000 nodes of the computational grid.

GeForce GTX 1650 video adapter was used in experimental researches. The GeForce GTX 1650 video adapter has 4 GB of video memory, core and memory clock frequency of 1485 MHz and 1665 MHz, and a video memory bus bit rate of 128 bits. The computing part consists of 14 streaming multiprocessors (SM).

The purpose of the experiment is to determine the distribution of flows along the Ox and Oz axes of the computational grid at different values of its nodes along the Oy axis so that the implementation time on the GPU of one MATM step is minimal. Two values are taken as factors: $k = X/Z$ is the ratio of the number of threads on the Ox (X) axis to the number of threads on the Oz (Z) axis; Y is the number of threads on the axis Oy. Values of the objective

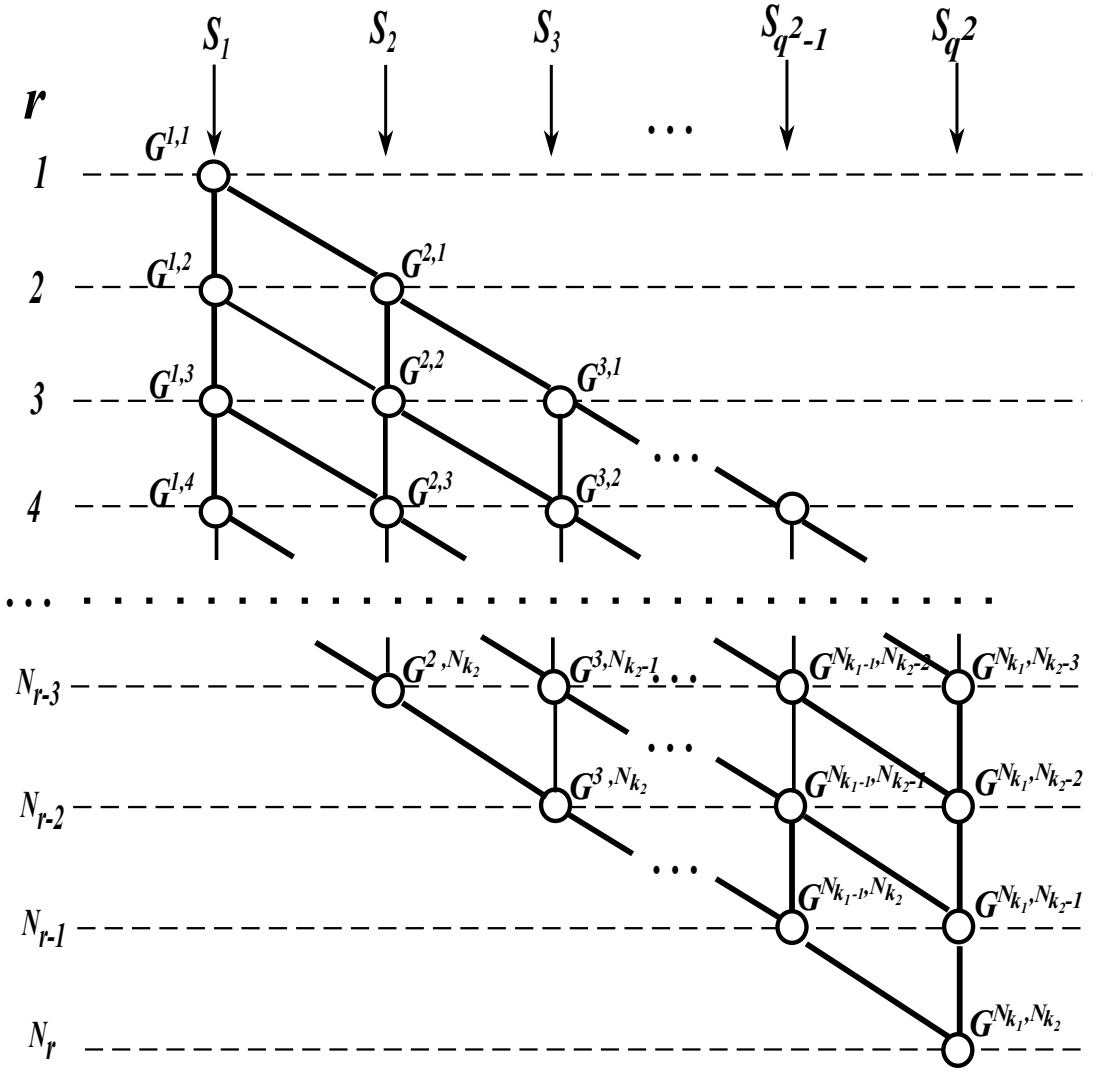


Fig. 2. A graph model that describes the relationships between adjacent fragments of the computational grid and the process of pipeline calculation

function: T_{GPU} is the calculation time of one MATM step on the GPU in terms of 1000 nodes of the computational grid, ms.

The regression equation was obtained as a result of the experimental data processing (Fig. 3):

$$T_{GPU} = a - b \cdot Y - c \cdot ln(k) - d \cdot ln(Y), \tag{27}$$

where T_{GPU} is the implementation time of one MATM step on the GPU in terms of 1000 nodes of the computational grid, ms. The determination coefficient was 0.86; $a = 0.026$; $b = 2 \cdot 10^{-7}$; $c = 16 \cdot 10^{-5}$; $d = 77 \cdot 10^{-5}$.

To evaluate the effectiveness of the parallel-pipeline method for solving SLAE with a lower triangular matrix, a numerical experiment was performed. The dimensions of the three-dimensional uniform computational grid along the spatial coordinates x, y, and z were respectively set equal to 640, 224, and 448, respectively. The amount of video memory was 3.8 GB. In the course of experimental researches, we changed the number of streaming multiprocessors N_{k_1} involved in the calculations and fixed the computation time T_M.

Algorithm 2 matmKernel(IN: $a_0, a_2, a_4, a_6, \omega$ IN/OUT: v;)

1: $thX \leftarrow blockDim.x \cdot blockIdx.x + threadIdx.x$
2: $thZ \leftarrow blockDim.z \cdot blockIdx.z + threadIdx.z$
3: $i \leftarrow thX + 1$; $j \leftarrow 1$; $k \leftarrow thZ + 1$
4: **for** $s \in [3; n_1 + n_2 + n_3 - 3]$ **do**
5: **if** $(i + j + k = s) \wedge (s < i + n_2 + k)$ **then**
6: $p_0 \leftarrow i + (blockDim.x + 1) \cdot j + n_1 \cdot n_2 \cdot k$
7: **if** $a0[p0] > 0$ **then**
8: $p_2 \leftarrow p_0 - 1; p_4 \leftarrow p_0 - n_1; p_6 \leftarrow p_0 - n_1 \cdot n_2$
9: $vp4 \leftarrow 0$
10: **if** $(s > 3 + thX + thZ)$ **then**
11: $vp4 \leftarrow cache[thX][thZ]$
12: **else**
13: $vp4 \leftarrow v[p_4]$
14: $vp2 \leftarrow 0$
15: **if** $(thX \neq 0) \wedge (s > 3 + thX + thZ)$ **then**
16: $vp2 \leftarrow cache[thX - 1][thZ]$
17: **else**
18: $vp2 \leftarrow v[p_2]$
19: $vp6 \leftarrow 0$;
20: **if** $(thZ \neq 0) \wedge (s > 3 + thX + thZ)$ **then**
21: $vp6 \leftarrow cache[thX][thZ - 1]$
22: **else**
23: $vp6 \leftarrow v[p_6]$
24: $vp0 \leftarrow (\omega \cdot (a2[p_0] \cdot vp2 + a4[p_0] \cdot vp4 + a6[p_0] \cdot vp6) + v[p_0]) / ((0.5 \cdot \omega + 1) \cdot a_0[p_0])$
25: $cache[thX][thZ] \leftarrow vp0$
26: $v[p_0] \leftarrow vp0$
27: $j \leftarrow j + 1$

For each experiment, the computational grid was divided into three-dimensional blocks and fragments. In this case, the number of blocks was set equal to the number of streaming multiprocessors. The number of fragments in blocks along spatial coordinates x, y, and z was set equal to 4, 1, and 7, respectively. The sizes of fragments along spatial coordinates x ($n_x^{k_1}$) and z ($n_z^{k_1}$) were set equal to 160 and 64, respectively. The acceleration $S_p = T_M(1)/T_M(i)$ and efficiency $E_p = S_p(i)/N_{k_1}(i)$ were calculated from the experimental data. The results of numerical experiments are shown in Tab. 1.

Conclusions

To solve grid equations using the MPTM method on the graphics accelerator, the decomposition model of computational domain has been developed. The computational domain is divided into blocks along the spatial coordinate y, and then the blocks are divided into fragments along the spatial coordinates x and z. This model allows each GPU streaming multiprocessor to map a computational domain block and organize a parallel-pipelined computational process. The graph model was proposed that describes the relationship between adjacent fragments of

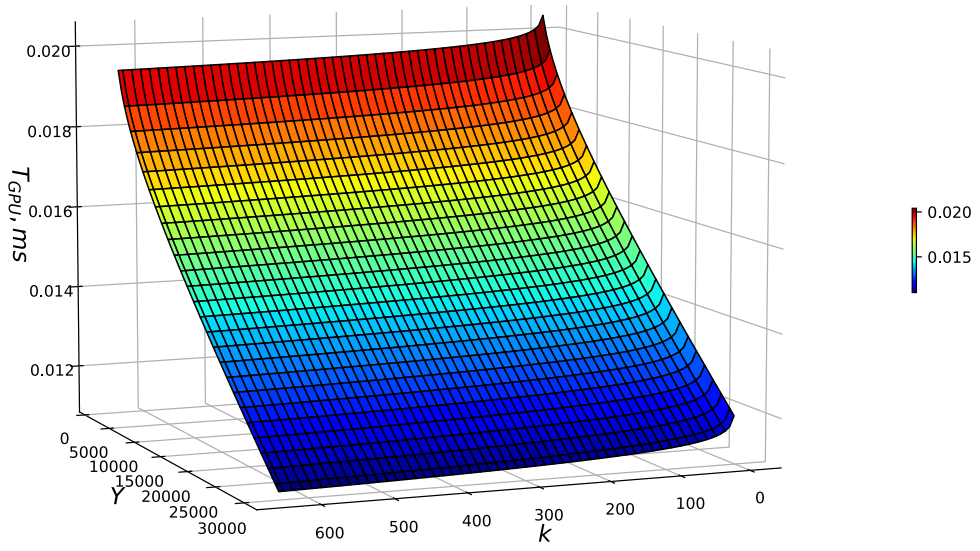


Fig. 3. Surface of the response function $T_{GPU} = f(k, Y)$

Table 1. Results of SLAE calculations with a lower triangular matrix by a parallel-pipeline method on GPU

N_{k_1}	$n_y^{k_1}$	N_r	T_M	S_p	E_p
1	224	–	580	1.0	1.00
2	112	29	304	1.9	0.95
7	32	34	102	5.7	0.81
14	16	41	61	9.5	0.68

the computational grid and the process of conveyor calculation. The algorithm for solving the system of equations with a lower triangular matrix in the CUDA C language was described.

As a result of the experiment, a regression model was obtained: it describes the dependence of the time for calculation one step of the MATM on the GPU. According to the regression model, at $k < 10$ and $Y < 1000$, the calculation velocity slows down, which is explained by the inefficient use of the distributed memory of the graphics accelerator.

The results of calculations of SLAE with the lower triangular matrix by the parallel-pipeline method on the GPU with using the different number of streaming multiprocessors are presented. At $N_{k_1} = 14$, the acceleration S_p was 9.5, and the efficiency E_p was 0.668.

The reported study was funded by the Russian Science Foundation (project No. 21-71-20050).

References

1. Sukhinov A.I., Atayan A.M., Belova Y.V., *et al.* Data processing of field measurements of expedition research for mathematical modeling of hydrodynamic processes in the Azov Sea. Computational Continuum Mechanics. 2020. Vol. 13, no. 2. P. 161–174. DOI: 10.7242/1999-6691/2020.13.2.13.

2. Sukhinov A.I., Litvinov V.N., Chistyakov A.E., *et al.* Computational aspects of solving grid

equations in heterogeneous computing systems. Parallel Computing Technologies. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 166–177. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_13.

3. Lyupa A., Morozov D., Trapeznikova M., *et al.* Three-phase filtration modeling by explicit methods on hybrid computer systems. Mathematical Models and Computer Simulations. 2014. Vol. 6. P. 551–559. DOI: 10.1134/S2070048214060088.

4. Mat Ali N.A., Rahman R., Sulaiman J., Ghazali K. Solutions of reaction-diffusion equations using similarity reduction and HSSOR iteration. Indonesian Journal of Electrical Engineering and Computer Science. 2019. Vol. 16, no. 3. P. 1430–1438. DOI: 10.11591/ijeecs.v16.i3.pp1430-1438.

5. Kittisopaporn A., Chansangiam P. The steepest descent of gradient-based iterative method for solving rectangular linear systems with an application to Poisson's equation. Advances in Difference Equations. 2020. Vol. 2020. Article number 259. DOI: 10.1186/s13662-020-02715-9.

6. Yifen K., Ma C. Adaptive parameter based matrix splitting iteration method for the large and sparse linear systems. Computers & Mathematics with Applications. 2022. Vol. 122. P. 19–27. DOI: 10.1016/j.camwa.2022.07.010.

7. Klimonov I.A., Korneev V.D., Sveshnikov V.M. Parallelization technologies for solving three-dimensional boundary value problems on quasi-structured grids using the CPU+GPU hybrid computing environment. Numerical Methods and Programming. 2016. Vol. 17, no. 1. P. 65–71. DOI: 10.26089/NumMet.v17r107.

8. Ding P., Liu Z. Accelerating phase-field modeling of solidification with a parallel adaptive computational domain approach. International Communications in Heat and Mass Transfer. 2020. Vol. 111. P. 104452. DOI: 10.1016/j.icheatmasstransfer.2019.104452.

9. Molostov I., Scherbinin V. Application of NVIDIA CUDA Technology for Numerical Simulation of Electromagnetic Pulses Propagation. Izvestiya of Altai State University. 2015. Vol. 1, no. 1/1(85). DOI: 10.14258/izvasu(2015)1.1-06.

10. Krasnopolsky B., Medvedev A., Chulyunin A. On application of GPUs for modelling of hydrodynamic characteristics of screw marine propellers in OpenFOAM package. Proceedings of the Institute for System Programming of RAS. 2014. Vol. 26, no. 5. P. 155–172. DOI: 10.15514/ISPRAS-2014-26(5)-8.

11. Egorov M., Egorov S., Egorov D. Using graphics accelerator to improve computing performance in the numerical modeling of complex technical systems functioning. Perm National Research Polytechnic University Aerospace Engineering Bulletin. 2015. No. 40. P. 81–91. DOI: 10.15593/2224-9982/2015.40.05.

12. Szenasi S. Solving the inverse heat conduction problem using NVLink capable Power architecture. PeerJ Computer Science. 2017. Vol. 3. P. 138. DOI: 10.7717/peerj-cs.138.

13. Zheng L., Gerya T., Knepley M., *et al.* GPU Implementation of Multigrid Solver for Stokes Equation with Strongly Variable Viscosity. GPU Solutions to Multi-scale Problems in Science and Engineering. Springer, 2013. P. 321–333. Lecture Notes in Earth System Sciences. DOI: 10.1007/978-3-642-16405-7_21.

14. Konovalov A. The steepest descent method with an adaptive alternating-triangular preconditioner. Differential Equations. 2004. Vol. 40. P. 1018–1028.

15. Sukhinov A.I., Chistyakov A.E., Litvinov V.N., *et al.* Computational Aspects of Mathematical Modeling of the Shallow Water Hydrobiological Processes. Numerical methods and programming. 2020. Vol. 21, no. 4. P. 452–469. DOI: 10.26089/NumMet.v21r436.

16. Samarskii A.A., Vabishchevich P.N. Numerical methods for solving convection-diffusion problems. Moscow: URSS, 2009. (in Russian).

17. Browning J.B., Sutherland B. C++20 Recipes. A Problem-Solution Approach. Berkeley, CA: Apress, 2020. 630 p.

УДК 519.6 **DOI: 10.14529/cmse230204**

РЕШЕНИЕ СЕТОЧНЫХ УРАВНЕНИЙ ПОПЕРЕМЕННО-ТРЕУГОЛЬНЫМ МЕТОДОМ НА ГРАФИЧЕСКОМ УСКОРИТЕЛЕ

© 2023 А.И. Сухинов[1], В.Н. Литвинов[1,2], Ф.Е. Чистяков[1], А.В. Никитина[1,3], Н.Н. Грачева[1,2], Н.Б. Руденко[1,2]

[1]*Донской государственный технический университет (344003 Ростов-на-Дону, пл. Гагарина, д. 1),*

[2]*Азово-Черноморский инженерный институт ФГБОУ ВО Донской ГАУ (347740 Зерноград, ул. Ленина, д. 21),*

[3]*Южный федеральный университет (344006 Ростов-на-Дону, ул. Большая Садовая, д. 105/42)*

E-mail: sukhinov@gmail.com, litvinovvn@rambler.ru, cheese_05@mail.ru, nikitina.vm@gmail.com, 79286051374@yandex.ru, nelli-rud@yandex.ru

Поступила в редакцию: 15.03.2023

В статье описана параллельно-конвейерная реализация решения сеточных уравнений модифицированным попеременно-треугольным итерационным методом (МПТМ), получаемых при численном решении уравнений математической физики. Наибольшие вычислительные затраты при использовании указанного метода приходятся на этапы решения системы линейных алгебраических уравнений (СЛАУ) с нижнетреугольной и верхнетреугольной матрицами. Представлен алгоритм решения СЛАУ с нижнетреугольной матрицей на графическом ускорителе с использованием технологии NVIDIA CUDA. Для реализации параллельно-конвейерного метода использовалась трехмерная декомпозиция расчетной области. Она делится по координате y на блоки, количество которых соответствует количеству потоковых мультипроцессоров GPU, задействованных в вычислениях. В свою очередь, блоки разделяются на фрагменты по двум пространственным координатам — x и z. Представленная графовая модель описывает взаимосвязь между соседними фрагментами расчетной сетки и процессом конвейерного расчета. По результатам проведенных вычислительных экспериментов получена регрессионная модель, описывающая зависимость времени расчета одного шага МПТМ на GPU, вычислены ускорение и эффективность расчетов СЛАУ с нижнетреугольной матрицей параллельно-конвейерным методом на GPU при задействовании различного количества потоковых мультипроцессоров.

Ключевые слова: математическое моделирование, параллельный алгоритм, графический ускоритель.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Sukhinov A.I., Litvinov V.N., Chistyakov A.E., Nikitina A.V., Gracheva N.N., Rudenko N.B. Solving Grid Equations Using the Alternating-triangular Method on a Graphics Accelerator // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 78–92. DOI: 10.14529/cmse230204.

Литература

1. Sukhinov A.I., Atayan A.M., Belova Y.V., *et al.* Data processing of field measurements of expedition research for mathematical modeling of hydrodynamic processes in the Azov Sea // Computational Continuum Mechanics. 2020. Vol. 13, no. 2. P. 161–174. DOI: 10.7242/1999-6691/2020.13.2.13.

2. Sukhinov A.I., Litvinov V.N., Chistyakov A.E., *et al.* Computational aspects of solving grid equations in heterogeneous computing systems // Parallel Computing Technologies. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 166–177. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_13.

3. Lyupa A., Morozov D., Trapeznikova M., *et al.* Three-phase filtration modeling by explicit methods on hybrid computer systems // Mathematical Models and Computer Simulations. 2014. Vol. 6. P. 551–559. DOI: 10.1134/S2070048214060088.

4. Mat Ali N.A., Rahman R., Sulaiman J., Ghazali K. Solutions of reaction-diffusion equations using similarity reduction and HSSOR iteration // Indonesian Journal of Electrical Engineering and Computer Science. 2019. Vol. 16, no. 3. P. 1430–1438. DOI: 10.11591/ijeecs.v16.i3.pp1430-1438.

5. Kittisopaporn A., Chansangiam P. The steepest descent of gradient-based iterative method for solving rectangular linear systems with an application to Poisson's equation // Advances in Difference Equations. 2020. Vol. 2020. Article number 259. DOI: 10.1186/s13662-020-02715-9.

6. Yifen K., Ma C. Adaptive parameter based matrix splitting iteration method for the large and sparse linear systems // Computers & Mathematics with Applications. 2022. Vol. 122. P. 19–27. DOI: 10.1016/j.camwa.2022.07.010.

7. Klimonov I.A., Korneev V.D., Sveshnikov V.M. Parallelization technologies for solving three-dimensional boundary value problems on quasi-structured grids using the CPU+GPU hybrid computing environment // Numerical Methods and Programming. 2016. Vol. 17, no. 1. P. 65–71. DOI: 10.26089/NumMet.v17r107.

8. Ding P., Liu Z. Accelerating phase-field modeling of solidification with a parallel adaptive computational domain approach // International Communications in Heat and Mass Transfer. 2020. Vol. 111. P. 104452. DOI: 10.1016/j.icheatmasstransfer.2019.104452.

9. Молостов И.П., Щербинин В.В. Применение технологии NVIDIA CUDA для численного моделирования распространения электромагнитных импульсов // Известия Алтайского государственного университета. 2015. Т. 1, № 1/1(85). DOI: 10.14258/izvasu(2015)1.1-06.

10. Краснопольский Б.И., Медведев А.В., Чулюнин А.Ю. Применение графических ускорителей для расчета гидродинамических характеристик гребных винтов в пакете OpenFOAM // Труды Института системного программирования РАН. 2014. Т. 26, № 5. С. 155–172. DOI: 10.15514/ISPRAS-2014-26(5)-8.

11. Егоров М.Ю., Егоров С.М., Егоров Д.М. Применение графических ускорителей для повышения производительности вычислений при численном моделировании функциони-

рования сложных технических систем // Вестник ПНИПУ. Аэрокосмическая техника. 2015. № 40. С. 81–91. DOI: 10.15593/2224-9982/2015.40.05.

12. Szenasi S. Solving the inverse heat conduction problem using NVLink capable Power architecture // PeerJ Computer Science. 2017. Vol. 3. P. 138. DOI: 10.7717/peerj-cs.138.
13. Zheng L., Gerya T., Knepley M., *et al.* GPU Implementation of Multigrid Solver for Stokes Equation with Strongly Variable Viscosity // GPU Solutions to Multi-scale Problems in Science and Engineering. Springer, 2013. P. 321–333. Lecture Notes in Earth System Sciences. DOI: 10.1007/978-3-642-16405-7_21.
14. Konovalov A. The steepest descent method with an adaptive alternating-triangular preconditioner // Differential Equations. 2004. Vol. 40. P. 1018–1028.
15. Sukhinov A.I., Chistyakov A.E., Litvinov V.N., *et al.* Computational Aspects of Mathematical Modeling of the Shallow Water Hydrobiological Processes // Numerical methods and programming. 2020. Vol. 21. P. 452–469. DOI: 10.26089/NumMet.v21r436.
16. Самарский А.А., Вабищевич П.Н. Численные методы решения уравнений конвекции-диффузии. Москва: УРСС, 2009.
17. Browning J.B., Sutherland B. C++20 Recipes. A Problem-Solution Approach Berkeley, CA: Apress, 2020. 630 p.

Сухинов Александр Иванович, чл.-корр. РАН, д.ф.-м.н., профессор, кафедра математики и информатики, Донской государственный технический университет (Ростов-на-Дону, Российская Федерация)

Литвинов Владимир Николаевич, к.т.н., доцент, кафедра математики и информатики, Донской государственный технический университет (Ростов-на-Дону, Российская Федерация); кафедра математики и биоинформатики, Азово-Черноморский инженерный институт ФГБОУ ВО Донской ГАУ (Зерноград, Российская Федерация)

Чистяков Александр Евгеньевич, д.ф.-м.н., кафедра программного обеспечения вычислительной техники и автоматизированных систем, Донской государственный технический университет (Ростов-на-Дону, Российская Федерация)

Никитина Алла Валерьевна, д.т.н., доцент, кафедра программного обеспечения вычислительной техники и автоматизированных систем, Донской государственный технический университет (Ростов-на-Дону, Российская Федерация); кафедра интеллектуальных и многопроцессорных систем, Южный федеральный университет (Ростов-на-Дону, Российская Федерация)

Грачева Наталья Николаевна, к.т.н., доцент, кафедра математики и биоинформатики, Азово-Черноморский инженерный институт ФГБОУ ВО Донской ГАУ (Зерноград, Российская Федерация); кафедра проектирования и технического сервиса транспортно-технологических систем, Донской государственный технический университет (Ростов-на-Дону, Российская Федерация)

Руденко Нелли Борисовна, к.т.н., доцент, кафедра математики и биоинформатики, Азово-Черноморский инженерный институт ФГБОУ ВО Донской ГАУ (Зерноград, Российская Федерация); кафедра медиатехнологий, Донской государственный технический университет (Ростов-на-Дону, Российская Федерация)

УДК 004.43, БК 22.18 DOI: 10.14529/cmse230205

РАБОТА С ДАННЫМИ В УЧЕБНОМ ЯЗЫКЕ ПРОГРАММИРОВАНИЯ СИНХРО*

© 2023 Л.В. Городняя[1,2]

[1]*Институт систем информатики СО РАН*
(630090, Новосибирск, пр. им. М.А. Лаврентьва, д. 6),
[2]*Новосибирский государственный университет*
(630090, Новосибирск, ул. Пирогова, д. 2)
E-mail: gorod@iis.nsk.su

Поступила в редакцию: 04.11.2022

Статья является продолжением собственных предыдущих исследований автора в рамках многолетней работы по созданию учебного языка программирования СИНХРО, предназначенного для ознакомления с параллелизмом. Основное направление работ — уточнение понятий, способствующих подготовке небольших многопоточных программ при обучении параллельному программированию. Главный результат последнего года заключается в развитии механизма взаимодействия локальной и общей памяти. Дан приоритет парадигме функционального программирования, популярной при подготовке прототипов многопоточных программ. Это помогло преодолеть зависимость порядка вычислений от последовательности вхождения выражений в текст программы и размещения данных в памяти. Описаны отличия от привычных понятий программирования, сдерживающих решение задач организации параллельных вычислений и предельно распределенных систем из ряда потоков, взаимодействующих в терминах доступа к значениям переменных, возможно расположенных в общей памяти. Повышен базовый уровень воздействий на память. Часть из них укрупнены для предотвращения неожиданностей из-за асинхронности и ослабления императивности элементов распределенных систем. Добавлено понятие команд-двойников для управления императивной синхронизацией взаимодействующих устройств, полезное при решении вопросов освобождения памяти.

Ключевые слова: дисциплина доступа к памяти, функциональное программирование, многопоточные программы, неизменяемость данных, восстановление данных, освобождение памяти.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Городняя Л.В. Работа с данными в учебном языке программирования СИНХРО // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 93–107. DOI: 10.14529/cmse230205.

Введение

В лаборатории информационных систем ИСИ СО РАН разрабатывается учебный язык программирования СИНХРО, предназначенный для ознакомления с явлениями параллелизма, мета-программированием и особенностями много-поточного программирования [1, 2]. Прагматика языка соответствует парадигме функционального программирования (ФП), дополненной некоторыми, сравнительно безопасными, методами императивного программирования. При создании языка учтен опыт применения языков Lisp, Робик, SETL, БАРС, Sisal, Haskell, mpC, Oz. Приняты во внимания основные модели организации параллельных вычислений [3, 4]. и уточнен ряд понятий и методов, изменяющихся при переходе от обычного программирования к представлению много-поточных и многопроцессорных программ. Прежде всего это преодоление зависимости порядка вычислений от последовательности вхождения выражений в текст программы, а также повышение базового

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2022».

уровня воздействий на память и взаимодействия с устройствами. Часть из них укрупнены для предотвращения неожиданностей из-за асинхронности и ослабления императивности элементов распределенных систем. Добавлено понятие команд-двойников для управления императивной синхронизацией взаимодействующих устройств.

Язык СИНХРО ориентирован на ознакомление с причинами неожиданностей при подготовке многопоточных программ, а также на формирование навыков профилактики неудачных взаимодействий процессов, учета равноправия независимых потоков, использования укрупненных воздействий на общую память, прогнозирования результатов автоматизированных преобразований программ и данных, включая оптимизацию и компиляцию программ. Учтены особенности отладки программ решений небольших учебных задач. В данной статье описаны решения, принятые на уровне ядра системы программирования. Рассмотрены подходы к решению проблем работы многопоточных программ над общей памятью. Язык СИНХРО позволяет управлять компиляцией многопоточных программ в многопроцессорные. Обучение метапрограммированию предполагается выполнять в форме подготовки сценариев отладки и преобразования программ, а также измерения вклада программируемых решений в производительность программ, что выходит за пределы данной статьи.

Изложение начинается с описания адаптации принципов ФП к подготовке многопоточных программ (раздел 1) с акцентом на равноправие параметров. Затем уточнены некоторые понятия программирования, требующие пересмотра в случае многопоточных программ (раздел 2), включая переход от потоков к процессам, достаточный для этого перехода уровень абстрактного процессорного комплекса, команды взаимодействия с устройствами и организации многопроцессорных комплексов. Далее рассмотрены механизмы повышения производительности многопроцессорных программ (раздел 3), включая восстановление данных и решения по организации работы с общей памятью. В заключении отмечены образовательные проблемы параллельных вычислений, перечислены представленные в статье результаты и намечены направления дальнейших исследований.

1. Функциональное программирование и параллелизми

Функциональное программирование — одна из первых парадигм, направленных не столько на получение эффективной реализации заранее созданных и хорошо изученных алгоритмов, сколько на продуктивность решения новых и исследовательских задач, изучение которых продолжается в процессе программирования [5–11]. Для таких задач правильность и полнота решений важнее эффективности и производительности полученных программ. Семантические принципы ФП, такие как всеобщность (универсальность), самоприменение (рекурсия) и равноправие параметров функций, позволяют функции и значения представлять такими же символами, что и любые данные для компьютерной обработки. Представления функций и значений могут использовать рекурсивные символьные формы. Порядок вычисления параметров не важен, они не зависят друг от друга и вычисляются в одной области видимости. Прагматические принципы реализации ФП, такие как гибкость ограничений на размеры блоков памяти, неизменяемость обрабатываемых данных, однозначность и строгость (чистота) результата функции, позволяют системам ФП предотвращать простои памяти автоматизацией оперативного анализа достижимости данных, освобождения и повторного использования памяти, хранящей недостижимые данные. Представление каждого данного помещается в новую часть свободной памяти без искажения аргументов

функции, они могут быть полезны для других функций. Любое количество результатов функции может быть представлено в виде единой символьной формы, причем значения одинаковых форм в одной области видимости совпадают.

Представление алгоритмов в виде чисто функциональных программ дает важные следствия. Из семантических принципов вытекает возможность метапрограммирования, верификации и факторизации программ на автономно развиваемые модули. Прагматические принципы допускают интуитивные модели непрерывности процессов, обратимости действий и линейных потоков из унарных функций, служащих основой для разработки прототипов на ранних стадиях жизненного цикла многопоточных программ. Таким образом, комплекс семантических и прагматических принципов чистого ФП обеспечивает поддержку подготовки программ при организации параллельных вычислений благодаря сведению к наборам независимых потоков. Кроме того, любой фрагмент, выполнение которого маловероятно или несвоевременно, может быть перемещен из представления функции в отложенное или опережающие действие. Такие «ленивые» вычисления дают возможность оперативно перераспределять нагрузку процессоров. Любое конечное множество потоков может работать как пространство итераций для определенной на нем функции при автоматическом распараллеливании [12].

При переходе к многократно применяемым программам и суперкомпьютерным вычислениям успех приложения и производительность программ становятся важнее, чем их полная формальная корректность и предельная эффективность. В дополнение к принципам и следствиям в реальные языки и системы программирования для производственного ФП обычно включают механизмы практичного компромисса, которые, внешне не нарушая функциональный стиль представления программ, встраиваются как специальные функции. Например, Lisp 1.5, Clisp, Cmucl, Clojure и другие члены семейства языка Lisp обычно предоставляют возможности такого рода:

- контроль типов данных, несколько ограничивающий принцип универсальности механизмом статического и динамического анализа типов данных, допускает частичные функции;
- схемы циклов, позволяющие преодолеть типичные опасения по поводу сложности реализации принципа самоприменения, уменьшают рекурсивность определений и наследуют привычные шаблоны управления вычислениями;
- возможность восстановления данных, исключающая чрезмерное потребление и обременительное освобождение памяти, аккуратно отклоняется от принципа неизменности данных в пользу программируемого повторного использования памяти;
- программируемые прогнозы объема памяти и скорости выполнения программ, поддерживают управление выбором моментов включения механизмов автоматизации, смягчая несколько затратный принцип гибкости ограничений («сборщик мусора» — garbage collector);
- псевдофункции, выполняя нагрузку по взаимодействию с устройствами, ввод-вывод данных и доступ к файлам, внешне сохраняют общую функциональную схему управления вычислениями, что несколько уводит от принципов равноправия параметров и неизменяемости данных;
- мемоизация, снижая сложность многократно повторяемых вычислений, сохраняет доступ к успешному опыту ранее выполненных вычислений, слегка модифицируя принцип чистого результата.

Высокопроизводительное программирование требует учета перспектив многократного использования и улучшения программ [13]. Эксперименты на суперкомпьютерах показали, что системные решения могут вносить значительный вклад в производительность параллельных вычислений, важно, что такой вклад может превышать теоретические оценки. Квалификация разработчика программных систем обычно включает в себя умение изобретать новые решения задач и навыки ответственного улучшения готовых решений.

Представленные на посвященной современным тенденциям ФП конференции (Trends in Functional programming) доклады убедительно показали нацеленность ФП на решение задач организации параллельных вычислений [14]. Строго говоря, ФП способно выполнять роль проектно-конструкторского отдела для производственного программирования. В некоторых источниках по измерению трудоемкости программирования и производительности программ утверждается, что опыт ФП повышает качество программирования в любой парадигме [15].

2. Уточнение семантики

При переходе к параллельному программированию многие привычные понятия претерпевают изменения, начиная с понятия «данное». Данными являются уже не только представления значений и функций, но еще и представления процессоров и разнообразных устройств. Такое обобщение позволяет сферу успешного программирования обработки данных распространить на параллельные вычисления и распределенные системы.

2.1. Равноправие элементов структур данных

Многопоточная программа допускает асинхронное выполнение потоков и действий, что означает в соответствии с принципом равноправия параметров функций независимость порядка вычисления выражений от порядка их вхождения в структуры данных, рассматриваемые как функции над этими выражениями. Для приобретения навыков учета такой независимости в языке СИНХРО предлагается в записи структур данных разнести смысл скобок и разделителей. Круглые скобки означают, что доступ к данным возможен последовательно, а квадратные символизируют произвольный доступ. Если разделителем в перечне данных является точка с запятой «;», то элементы перечня вычисляются последовательно, в порядке вхождения в запись выражения, а в случае запятой «,» — в произвольном порядке, независимо от порядка записи (табл. 1).

Часть проблем взаимодействия потоков алгоритмически не разрешима, поэтому в задачи ознакомления с параллелизмом входит научиться предвидеть такие опасности и отлаживать программы при обнаружении неудачных взаимодействий потоков. Наполнение многопоточной программы может развиваться независимо от схем управления вычислениями в отдельных потоках, а схемы можно реорганизовывать без дополнительной отладки наполнения, используя факторизацию на автономно развиваемые модули. Схемы работают подобно макросам, но их применение нагружено дополнительным контролем соответствия параметров объявленным видам фрагментов. Выражение может использовать размещенные в общей памяти данные, но не изменяет их. Директива обрабатывает память и при благополучном исходе дает результат подобно выражению.

Таблица 1. Структуры данных

№ п/п	Синтаксис	Семантика
1	’(’Выр(’;’Выр)...’)’	Список, заполняемый последовательно вычисляемыми элементами, в порядке записи в программе
2	’(’Выр(’,’Выр)...’)’	Список, заполняемый в порядке записи в программе асинхронно вычисляемыми элементами, возможно в другом порядке
3	’[’Выр (’;’Выр)...’]’	Вектор, заполняемый последовательно вычисляемыми элементами, в порядке записи в программе
4	’[’Выр (’,’Выр)...’]’	Вектор, заполняемый в порядке записи в программе асинхронно вычисляемыми элементами, возможно в другом порядке

2.2. От потоков к процессам

Компилятор многопоточной программы строит функционально эквивалентную ей многопроцессорную программу для размещения на многопроцессорном комплексе. Набор потоков при компиляции преобразуется в набор процессов, каждый из которых выполняется отдельным процессором, но может быть иначе при решении проблем балансировки нагрузки в соответствии с принципом гибкости ограничений, распространенному с памяти на процессоры.

Контекст исполнения программы при переходе к процессам — общая память, данные из нее доступны всем процессам многопроцессорной программы. Общая память содержит представления глобальных, возможно изменяемых, переменных. Существует и локальная память, соответствующая процессам, созданным по определениям потоков, функций и циклов, подчиненная иерархии определений в программе — области видимости. Обработка общей и локальной памяти на уровне много-поточной программы выглядит одинаково, что позволяет при отладке перемещать фрагменты в разные позиции программы. Возможно хранение в общей памяти определений и имен. Типы именуемых данных можно устанавливать по виду их значения или функции. Имеются распознающие функции-предикаты ATOM, NUMBER, LIST, ARRAY и др.

Процесс выполняется как непустой ряд команд процессора, допускающий наращивание при исполнении. Очередная команда может начинать выполнение строго после начала предыдущей команды, но не обязана дожидаться ее завершения. Каждый процессор работает по шагам, соответствующим выполнению одной команды. После шага происходит переход к общему механизму управления многопроцессорной конфигурацией.

Компилятор по каждому потоку строит корректный ряд команд процесса, согласованных для выполнения действий потока. Завершение набора процессов может включать выполнение сверток для получения единого результата, что несколько влияет на их синхронизацию. Между двумя соседними командами процесса может быть выполнена команда другого процесса, а это требует особого внимания при организации воздействий на общую память.

Решение учебной задачи строится в два шага. Сначала программа решения выглядит как чисто функциональная схема, наполняемая выражениями — фрагментами вычислений без действий над общей памятью. Потом возможен гладкий переход к более эффектив-

ной версии программы методом замены некоторых выражений на директивы, т.е. чистых функций на функционально эквивалентные им псевдофункции или процедурные аналоги с воздействиями на память и синхронизацией потоков по мере необходимости — принцип независимости параметров и факторизация. Для простоты изложения здесь не рассматривается разнообразие категорий систем команд процессоров и видов используемой памяти с различной дисциплиной доступа.

2.3. Абстрактный процессорный комплекс

Абстрактный процессорный комплекс (АПК) способен выполнять вычисления несколько более широкого класса, чем обычно задано семантикой языка программирования. Кроме собственно процессоров к выполнению программы привлекаются дополнительные устройства, которые рассматриваются как особые процессоры, система команд которых не определена на уровне языка. Они способны выполнять некоторые действия по запросам от других процессоров. Система команд локального процессора поддерживает обработку данных, их размещение и реорганизацию в своей локальной или общей памяти, и управление ходом выполнения процессов, включая взаимодействие процессоров и устройств, и резервирование данных для их защиты от случайных изменений, подобно средствам операционных систем [16]. Программа использует общую память, данные из которой доступны отдельным процессорам, выполняющим программу. Шаги разных процессоров асинхронны, но могут происходить одновременно.

При спецификации команд АПК для языка СИНХРО использовалась предложенная П. Лендиным машина SECD [7], дополненная командами наиболее известных виртуальных машин, включая JVM [17–20]. Машина SECD работает над четырьмя регистрами: S — стек для промежуточных результатов, E — контекст для размещения именованных значений, C — управляющая вычислениями программа, D — резервная память (Stack, Environment, Control-list, Dump). Регистры приспособлены для хранения выражений в форме символов или списков. К регистрам системы команд машины SECD добавлен регистр «M» (Memory), что дает уточненное обозначение многопроцессорного комплекса: M(SECD)+ — абстрактный многопроцессорный комплекс над общей памятью. Это обозначение символизирует, что M — общая память для всех процессоров, (SECD)+ — что хотя бы одни процессор обязателен, общее число процессоров произвольно и не исключено изменение их числа в динамике. M — общая память, условно подчинена принципу неизменяемости данных, расширенному механизмом восстановления данных. Она состоит из регистров произвольного доступа, хранящих счетчик числа потоков, использующих эти регистры, имя переменной, ее текущее значение и протокол произошедших изменений, устроенный как вектор или список. Поддержана возможность восстановления значений, оттесненных присваиваниями. Состояние машины АПК полностью определяется содержимым этих пяти регистров.

m s e c d $\rightarrow$ m' s' e' c' d' — переход от старого состояния машины к новому.

Разница между SECD и M(SECD)+ сводится к операциям над данными в общей памяти. Регистр M является списком списков или стеков, каждый элемент которого начинается с имени глобальной переменной, вслед за которым расположено ее текущее значение, а дальше следует протокол изменения значений, выглядящий как последовательность номеров процессов с каждым последующим установленным этим процессом значением. В определенном смысле состояние общей памяти можно рассматривать как непременное дополнение к формальному результату работы программы, технически похожее на мемоизацию.

Фактически исполняются команды по очереди, последовательно, начиная с первых в регистре управляющей программы, хотя можно считать, что они исполняются в произвольном порядке по мере готовности. Такое определение может быть машинно-независимым и переносимым. Размер стека не ограничен. Каждая команда абстрактного многопроцессорного комплекса «знает» число используемых при ее работе элементов стека S и их форматы, элементы она удаляет из стека S и вместо них размещает один выработанный результат для обычных команд или набор результатов для многопроцессорных команд, размещая число процессов в верхнем элементе стека. Каждая команда АПК «знает» число используемых при ее работе элементов стека S и их форматы, элементы она удаляет из стека S и вместо них размещает один выработанный результат для обычных команд или набор результатов для многопроцессорных команд, размещая число процессов в верхнем элементе стека. Всегда известно число текущих результатов в стеке S, которые можно явно свернуть в единственный строгий результат специальной сверткой, необходимость в которой выясняет компилятор и размещает свертку в регистре C абстрактной машины. Свертка может быть встроенной в язык или программируемой, позволяющей указанное число элементов в стеке свести в одно данное. Это может быть список, структура, сумма, произведение, максимум, минимум, последний и т.п. Свертки обычно коммутативны, за редким исключением — первый или последний по времени вычислений или позиции вхождения в программу. Фильтр можно рассматривать как частный случай свертки.

Суммарно комплект команд АПК включает в себя действия, часть из которых определены в книге П. Хендерсона (LD, LDC, LDF, AP, RTN, RAP, DUM, SEL, JOIN, CONS, CAR, CDR, CONS, ATOM, EQ, SUB, ADD, MUL, DIV, STOP) [7]. Этот набор команд в языке СИНХРО пополнен для решения проблем работы с общей памятью и внешними устройствами, особенности которых проявляются на командах воздействия на общую память, взаимодействия общей и локальной памяти (табл. 2), доступа к устройствам и организации параллельных процессов (табл. 3) [17–20]:

Таблица 2. Команды работы с памятью

№ п/п	Синтаксис	Семантика
1	LDM	копирование данного из общей памяти в стек
2	SET	запись в общую память комплекса из стека
3	LET	сохранение локальных значений в общей памяти
4	DEL	удаление верхнего элемента стека
5	MLL	пересылка в головной элемент списка из другого списка
6	MLV	пересылка и списка в указанный элемент вектора
7	MVL	пересылка из вектора в головной элемент списка
8	MVV	пересылка в указанный элемент вектора из другого вектора
9	CHNG	обмен данными в общей памяти комплекса

Усложненные, точнее укрупненные, команды пересылки и обмена данными в общей памяти (MLL, MLV, MVL, MVV, CHNG) нужны, чтобы исключить возникновение временных интервалов между взаимосвязанными присваиваниями в общей памяти. Иначе может возникать так называемая «фантомная» память или доступ к формально уже удаленным данным, что иногда обнаруживается при использовании JVM в сетях, стандарт на которую был утвержден более 20 лет назад. Асинхронное выполнение воздействий на память при ее

освобождении не исключает, что некоторое время ссылка из программы на память удалена, а адрес блока памяти не помечен как свободный, или, наоборот, блок памяти помечен как свободный, а ссылка на него из программы не удалена. Технические детали определения этих и остальных команд приведены в статье [21].

2.4. Внешние устройства и процессоры

При работе с устройствами через конечное время вырабатывается сигнал, говорящий или об успешном выполнении действия, или о причине отказа в его завершении. Выработка сигнала об отказе внешнего устройства приводит к запоминанию в протоколе информации об ущербно выработанных результатах, что можно учесть при отладке, и/или к выполнению остаточной программы в стиле смешанных вычислений (табл. 3).

Таблица 3. Команды работы с памятью

№ п/п	Синтаксис	Семантика
1	WRT	вывод данных из стека на внешнее устройство и сигнал успеха-провала
2	RD	ввод данных от внешнего устройства в стек и сигнал успеха-провала
3	ANY	выравнивание стека, если данное с устройства или на устройство не введено из-за провала
Для организации параллельных вычислений требуются еще команды:		
4	KIT	комплекс = из процессоров для выполнения действий
5	ROW	поток или ряд действий на одном процессоре
6	REZ	размещение заданного числа результатов процесса в свой стек
7	WAIT	приостановка с ожиданием сигнала от указанного процесса (завершения или сообщения)
8	SEND	сообщение указанному процессу
9	NEXT	ожидание события или завершения действия указанного процесса

Так обеспечивается подключение комплексов процессоров для неупорядоченных наборов потоков, мощность комплекса известна. На каждом процессоре происходит размещение процессов в виде ряда действий, длина ряда известна в каждый момент. Предполагается выполнение последовательности действий ряда на одном процессоре, завершаемых передачей результатов процесса в стек с локализацией воздействий на транзакционную общую память с явной обработкой ошибок. Реализация передачи сообщений подобна рандеву в языке ADA — обмен происходит между двумя активными процессами и каждый знает что и кому он передает или от кого сообщение получает. Процесс-отправитель перед выполнением SEND может дожидаться готовности процесса-получателя, то есть проверять, что получатель сейчас выполняет команду WAIT, иначе ожидать ее, чтобы произошло рандеву. Процесс может не знать, что его завершения кто-то ждет.

Многопроцессорная программа считается идеальной, компилятор ее строит без учета возможных аварийных ситуаций, считает, что все данные на устройстве ввода расположены в соответствии с запросами программы, именно те, что нужны. Состояния стеков к моменту выполнения команд сформированы вполне корректно. Регистры локальной памяти подчи-

нены принципу неизменяемости данных. Новые данные размещаются в первом элементе списка, а к прежним значениям, хранимым в общей памяти, можно вернуться при необходимости, например, при отладке посмотреть историю изменения данных. При выполнении обратимых действий поддержана и обратимость воздействий на общую память. Идеальная конфигурация многопроцессорной программы работает на как бы бесконечной общей памяти, не решает проблемы ее исчерпания и необходимости освобождения от ставших ненужными данных. Поскольку процессоры рассматриваются как одна из категорий данных, то можно от понимания памяти как одного из регистров машины перейти к отдельному процессору общей памяти, обладающему своей системой команд.

3. Уточнение прагматики

Уточнение семантики влечет некоторое усложнение реализационных решений в системе программирования для подготовки многопоточных программ, допускающих взаимодействия процессов и использование общей памяти.

3.1. Восстановление данных

В проекте системы программирования для языка СИНХРО уточнен принцип неизменяемости данных, характерный для ФП. Работа с памятью поддержана в транзакционном стиле, подобно обработке записей в базах данных, т.е. каждое воздействие на память или выполняется полностью, или восстанавливается состояние до начала выполнения не завершившейся команды [22]. Каждый элемент общей памяти в любой момент времени обрабатывается только в одном процессе программы, подобно захвату-освобождению файла. Хранение данных в общей памяти сопровождается протоколом изменений, чтобы при отладке можно было видеть какой процесс внес изменения и, при необходимости, вернуть утраченное значение. Поэтому не так уж опасен и возможен побочный эффект присваивания, если допускается восстановление прежних значений переменных. Такая реализация позволяет поддержать транзакции и обратимость обработки памяти при отладке учебных программ. В стеке размещается список результатов однократных присваиваний (SSA-формы1). К верхнему элементу контекста прицепляется этот список в хвост, вслед за аргументами. Предполагается, что имена формальных параметров и локальных данных различны.

3.2. Шкала доступа к общей памяти

Освобождение общей памяти требует механизмов, подобных опробованным в практике операционных систем при решении вопросов управления распределенными системами с совмещением работы независимых программ [16]. Можно рассмотреть вариант многопроцессорного комплекса, допускающее управление доступом к общей памяти с использованием паспортов, хранящих шкалу необходимых регистров общей памяти для доступа к глобальным переменным. Паспорт одновременно доступен и локальному процессору, и процессору общей памяти. В этом случае локальный процесс при «сборке мусора» формирует новое состояние такой шкалы, доступной процессору общей памяти. Локальные процессы вместо произвольного доступа к общей памяти в таком случае обладают лишь шкалой для доступа к определенным регистрам. При освобождении общей памяти появляется возможность частичного освобождения памяти, не задействованной в объединении шкал регистров от локальных процессов, без приостановки всех процессоров. В частности, при отказе отдель-

ного процессора можно по шкале глобальных переменных выполнить уменьшение счетчиков доступа к этим переменным для решений по освобождению или восстановлению памяти.

3.3. Команды-двойники доступа к общей памяти

Несколько проще выглядит модель комплекса со специальным процессором общей памяти, поддерживающим императивную синхронизацию. Такой процессорный комплекс состоит из ряда обычных процессоров и одного пассивного процессора общей памяти, с которым могут взаимодействовать активные локальные процессоры. Между собой они взаимодействуют только через общую память. Каждый процессор выполняет одну последовательность команд, среди которых встречаются команды запросов к процессору общей памяти. Это активные команды, выполняемые без особенностей, за исключением того, что команды запросов к общей памяти имеют двойников среди команд процессора общей памяти. Это пассивные команды, работающие как ленивые вычисления, возбуждаемые при выполнении запросов от локальных процессоров. Пассивные команды процессора общей памяти собраны в ряд очередей, каждая из которых соответствует конкретному локальному процессору и содержит двойников в том же порядке, что и последовательность активных команд. Пара запрос и его двойник выполняются неразрывно в стиле рандеву языка Ada. Как и при пересылках, механизм рандеву исключает случайное вмешательство сторонних процессов. При этом происходит обмен данными между локальной памятью активного процессора и общей памятью пассивного процессора (см. табл. 4).

Таблица 4. Дублеты — парные команды (активные + пассивные)

№ п/п	Синтаксис	Семантика	Синтаксис	Семантика
1	GIVE	запрос регистра для переменной в общей памяти	TAKE	выбор регистра для переменной, счетчик надо увеличить на 1
2	SAFE	запрос на хранение данного в переменной в общей памяти	WRITE	размещение данного, в протокол — указатель на данное и имя процесса
3	READ	запрос на чтение данного	VAR	доступ процесса к данному из общей памяти
4	UNDO	запрос на получение предыдущего значения переменной	UNDO	доступ к предыдущему значению переменной, состояние регистров переменной не меняется
5	FREE	объявление, что переменная не нужна	FREE	освобождение памяти, счетчик кратности уменьшается на 1

Для локальных процессоров возможны запросы на регистр для переменной, на хранение данного в переменной по ее регистру, на чтение данного из переменной по регистру, на получение предыдущего данного из переменной по адресу и на объявление, что больше переменная не нужна.

Команды-двойники процессора общей памяти, пассивно ждущие запросов от локальных процессоров или отладчика, одновременно с выполнением активных команд выполняют выбор регистра для переменной с указанным номером (счетчик кратности доступа к нему надо увеличить на 1), размещение данного (если данное — указатель на вектор или список, то они копируются полностью в общую память и в протокол вносится копия указателя на данное и номер активного процесса), чтение данного для передачи локальному процессору (состояние регистра не меняется), чтение предыдущего данного, освобождение памяти (счетчик кратности доступа уменьшается на 1).

Процессор общей памяти может функционировать как элемент распределенной системы, включающийся по мере поступления запросов от локальных процессоров, и не требующий приостановки всех процессоров на время решения проблемы освобождения памяти.

Представленный здесь метод использует ряд процессоров со своей локальной памятью и процессор общей памяти. Локальные процессоры могут работать автономно, по мере необходимости обращаясь к общей памяти и время от времени освобождая свою локальную память. В локальной памяти хранится шкала регистров доступа к общей памяти, которая может то расширяться, то сужаться по мере инструкций из программы или сужаться в результате «сборки мусора». Общая память состоит из двух частей — регистров прямого доступа и кучи для хранения протоколов, а также структур данных и старых значений после присваивания на случай необходимости восстановления состояний памяти. Время от времени при исчерпании той или иной области общей памяти или просто по графику запускается алгоритм, похожий на «Stop-copy» в системах ФП. Система команд абстрактного комплекса образует два уровня, на каждом из которых можно видеть подсистемы организации вычислений, работы с памятью, управления процессами и структурирования данных. Один уровень реализует работу отдельного потока над локальной памятью, другой поддерживает взаимодействие потоков над общей памятью и внешними устройствами. Напоминает «рассредоточенное представление сосредоточенных действий» А.Л. Фуксмана.

Заключение

На этапе ознакомления с проблемами параллелизма полезно научиться строить и отлаживать небольшие многопоточные программы взаимодействия процессов, выполняемых на модели произвольной многопроцессорной конфигурации над общей памятью. При обучении параллельному программирования опыт такой работы поможет видеть и понимать разные явления, происходящие при взаимодействии потоков, предвидеть проблемы и накапливать рецепты решения проблем, возникающих при подготовке и отладке многопоточных программ, что особенно ярко может влиять на переход к суперкомпьютерам [23]. Главный результат ознакомления — формирование интуитивных понятий, соответствующих реальной сложности параллельного программирования.

При создании языка СИНХРО для этих целей принципы ФП адаптированы к учебным задачам подготовки многопоточных программ. Более конкретно, принцип неизменяемости данных, характерный для ФП, распространен на работу с переменными в общей памяти. Этот механизм дополнен средствами восстановления данных на основе протоколов изменения значений переменных. Предложено решение проблемы освобождения общей памяти как композиции методов «сборки мусора» и учета кратности доступа к переменным из потоков, взаимодействующих через использование данных, хранящихся в общей памяти.

В данной статье описана адаптация принципов ФП к подготовке многопоточных программ (раздел 1), позволившая уточнить семантику структурных выражений, требующую пересмотра в случае многопоточных программ из-за различий в пространствах допустимых процессов (раздел 2) с учетом принципа равноправия параметров. Рассмотрены особенности перехода от потоков к процессам и представлен достаточный для этого перехода уровень абстрактного процессорного комплекса, в систему команд которого включены команды обмена данными и работы с общей памятью, описаны команды взаимодействия с периферийными устройствами и организации многопроцессорных комплексов. Определены дополнительные механизмы для повышения производительности многопоточных программ (раздел 3), включая восстановление данных и решения по организации работы с общей памятью, использующие шкалы доступа к памяти и команды-двойники.

Направления дальнейших исследований связаны с расширением реализации языка СИНХРО в форме рада диалектов, поддерживающих последовательность целей обучения, таких как ознакомление с феноменами параллелизма, приобретение опыта метапрограммирования и навыков подготовки многопоточных программ, изучение безопасных методов представления взаимодействующих процессов и эксперименты по многопроцессорному программированию. Диалекты можно использовать автономно и совместно.

Автор выражает благодарность Дмитрию Владимировичу Мажуге, выполнившему для языка СИНХРО реализацию виртуального многопроцессорного комплекса на языке Clojure.

Литература

1. Городняя Л.В. Язык параллельного программирования СИНХРО, предназначенный для обучения. Новосибирск, Препринт ИСИ СО РАН № 180, 2016. 30 с.

2. Городняя Л.В. О курсе «Начала параллелизма» // Ершовская конференция по информатике, секция «Информатика образования», Новосибирск, 27 июля 2011. С. 51–54.

3. Воеводин В.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

4. Хоар Ч. Взаимодействующие последовательные процессы. Издательство «Мир», 1989. 264 с.

5. McCarthy J. LISP 1.5 Programming Manual. Cambridge: The MIT Press, 1963. 106 p. DOI: 10.7551/mitpress/5619.001.0001.

6. Backus J. Can programming be liberated from the von Neumann style? A functional stile and its algebra of programs // Commun. ACM. 1978. Vol. 21, no. 8. P. 613–641. DOI: 10.1145/359576.359579.

7. Хендерсон П. Функциональное программирование. М.: Мир, 1983. 349 с.

8. Лавров С.С. Функциональное программирование // Компьютерные инструменты в образовании. 2002. № 2-4. С. 42–52.

9. Лавров С.С., Городняя Л.В. Функциональное программирование. Принципы реализации языка Лисп // Компьютерные инструменты в образовании. 2002. № 5. С. 49–58.

10. Городняя Л.В. Основы функционального программирования. М.: Интернет-Университет Информационных технологий, 2004. 272 с.

11. Городняя Л.В. Первые реализации языка Lisp в СССР // Материалы второй Международной конференции «Развитие вычислительной техники и ее программного обеспечения в России и странах бывшего СССР», SoRuCom-2011, Великий Новгород, 12–16 сентября 2011. С. 95–100.

12. Cann D.C. SISAL 1.2: A Brief Introduction and tutorial. Preprint UCRL-MA-110620. Lawrence Livermore National Lab., Livermore, California, May, 1992. 128 p.

13. Бурдонов И.Б., Косачев А.С. Семантики взаимодействия с отказами, дивергенцией и разрушением. Часть 2. Условия конечного полного тестирования // Вестник Томского государственного университета. 2011. Т. 2, № 15.

14. Koopman P., Michels S., Plasmeijer R. Dynamic Editors for Well-Typed Expressions // Trends in Functional programming - 22nd International Symposium, 22nd International Symposium, TFP 2021, Virtual Event, February 17–19, 2021. Revised Selected Papers. Vol. 12834. Springer, 2021. P. 44–66. LNCS. DOI: 10.1007/978-3-030-83978-9_3.

15. Erann Gat Lisp as an Alternative to Java. URL: `https://flownet.com/gat/papers/lisp-java.pdf` (дата обращения: 05.04.2023).

16. Иртегов Д.В. Введение в операционные системы. СПб.: БХВ-Петербург, 2008. 1040 с.

17. Кнут Д.Э. Искусство программирования, том 1, выпуск 1. MMIX — RISC-компьютеры нового тысячелетия. М.: Вильямс, 2017. 160 с.

18. Вирт Н. Построение компиляторов. М.: ДМК Пресс, 2010.

19. Айлиф Дж. Принципы построения базовой машины. М.: Мир, 1973. 119 с.

20. Эванс Б., Гоф Дж., Ньюленд К. Java: оптимизация программ. Практические методы повышения производительности приложений в JVM. М.: Диалектика, 2019. 448 с.

21. Городняя Л.В. Абстрактная машина языка программирования учебного назначения СИНХРО // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 16–35.

22. Грабер М. Введение в SQL. М.: Лори, 1996. 337 с.

23. Левин В.К. Отечественные суперкомпьютеры семейства МВС. URL: `http://parallel.ru/mvs/levin.html` (дата обращения: 05.04.2023).

Городняя Лидия Васильевна, к.ф.-м.н., доцент, кафедра программирования, Новосибирский государственный университет (Новосибирск, Российская Федерация); с.н.с., Лаборатория информационных систем, Институт систем информатики СО РАН (Новосибирск, Российская Федерация)

DOI: 10.14529/cmse230205

WORKING WITH DATA IN THE SYNHRO EDUCATIONAL PROGRAMMING LANGUAGE

© 2023 L.V. Gorodnyaya[1,2]

[1] *A.P. Ershov Institute of Informatics Systems (pr. Acad. Lavrentjev 6, Novosibirsk, 630090 Russia),*

[2] *Novosibirsk State University (Pirogova 2, Novosibirsk-90, 630090 Russia)*

E-mail: gorod@iis.nsk.su

Received: 04.11.2022

The article is devoted to clarifying the concepts that are useful in preparing small multi-threaded programs for teaching parallel programming. The approach was formed in the process of creating the SYNHRO language. The priority is given to the paradigm of functional programming, which is popular in the preparation of prototypes of multi-threaded programs. A description of differences from the usual ideas that hinder the solution of problems of organizing parallel computing and extremely distributed systems from a number of threads interacting in terms of access to the values of variables, possibly located in a shared memory is given. A mechanism for the interaction of local and shared memory is proposed. The main models of organizing parallel computing are taken into account and a number of concepts and methods are refined, which changes during the transition from conventional programming to the representation of multi-threaded and multiprocessor programs. First of all, this is the overcoming of the dependence of the order of calculations on the sequence of occurrence of expressions in the program text. Further, the basic mechanism of influences on memory changes. Some of them are made more complex to prevent surprises due to asynchrony and weakening the imperativeness of executing elements of distributed systems. The concept of twin commands has been added to control the imperative synchronization of interacting devices, useful in resolving memory freeing issues.

Keywords: memory access discipline, functional programming, multi-threaded programs, data immutability, data recovery, memory freeing.

FOR CITATION

Gorodnyaya L.V. Working with Data in the SYNHRO Educational Programming Language. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 2. P. 93–107. (in Russian) DOI: 10.14529/cmse230205.

References

1. Gorodnyaya L.V. Parallel programming language SYNHRO for learning. Novosibirsk. Preprint of ISI SB RAS No. 180, 2016. 30 p.

2. Gorodnyaya L.V. About the course “The Beginnings of Parallelism”. Ershov Conference on Informatics. Section “Informatics of education”, Novosibirsk, Russia, July 27, 2011. P. 51–54. (in Russian).

3. Voevodin V.V. Parallel Computing. St. Petersburg: BHV-Petersburg, 2002. 608 p. (in Russian).

4. Hoare C.A.R. Interacting Sequential Processes. Mir Publishing House, 1989. 264 p. (in Russian).

5. McCarthy J. LISP 1.5 Programming Manual. Cambridge: The MIT Press, 1963. 106 p. DOI: 10.7551/mitpress/5619.001.0001.

6. Backus J. Can programming be liberated from the von Neumann style? A functional stile and its algebra of programs. Commun. ACM. 1978. Vol. 21, no. 8. P. 613–641. DOI: 10.1145/359576.359579.

7. Henderson P. Functional programming. M.: Mir, 1983. 349 p. (in Russian).

8. Lavrov S.S. Functional programming. Computer tools in education. 2002. No. 2-4. (in Russian).

9. Lavrov S.S., Gorodnyaya L.V. Functional programming. Lisp language implementation principles. Computer tools in education. 2002. No. 5. P. 49–58. (in Russian).

10. Gorodnyaya L.V. Fundamentals of functional programming. M.: Internet University of Information Technologies, 2004. 272 p.

11. Gorodnyaya L.V. First implementations of the Lisp language in the USSR. Proceedings of the Second International Conference Development of Computing Technology and Its Software in Russia and the Former USSR Countries, SoRuCom-2011. P. 95–100. (in Russian).

12. Cann D.C. SISAL 1.2: A Brief Introduction and tutorial. Preprint UCRL-MA-110620. Lawrence Livermore National Lab., Livermore, California, May, 1992. 128 p.

13. Burdonov I.B., Kosachev A.S. Semantics of interaction with failures, divergence and destruction. Part 2. Conditions for final full testing. Bulletin of Tomsk State University. 2011. Vol. 2, no. 15. (in Russian).

14. Koopman P., Michels S., Plasmeijer R. Dynamic Editors for Well-Typed Expressions. Trends in Functional programming - 22nd International Symposium, 22nd International Symposium, TFP 2021, Virtual Event, February 17–19, 2021. Revised Selected Papers. Vol. 12834. Springer, 2021. P. 44–66. LNCS. DOI: 10.1007/978-3-030-83978-9_3.

15. Erann Gat Lisp as an Alternative to Java. URL: `https://flownet.com/gat/papers/lisp-java.pdf` (accessed: 05.04.2022).

16. Irtegov D.V. Introduction to operating systems St. Petersburg: BHV-Petersburg, 2008. 1040 p. (in Russian).

17. Donald Knuth D.E. The Art of Programming, volume 1, issue 1. MMIX – RISC Computers of the New Millennium. M.: Williams, 2017. 160 p. (in Russian).

18. Wirth N. Compiler Construction. Moscow: DMK Press, 2010. (in Russian).

19. Ailiffe J.K. Principles of building a base machine. M.: Mir, 1973. 119 p. (in Russian).

20. Evans B., Gough J., Newland K. Java: program optimization. Practical methods for improving application performance in the JVM. M.: Dialectics, 2019. 448 p. (in Russian).

21. Gorodnyaya L.V. Abstract machine of the programming language for educational purposes SYNHRO. Bulletin of NGU. Series: Information technologies. 2021. Vol. 19, no. 4. P. 16–35. (in Russian).

22. Graber M. Introduction to SQL. M.: Lori, 1996. 377 p. (in Russian).

23. Levin V.K. National Family of MVS Supercomputers. URL: `http://parallel.ru/mvs/levin.html` (accessed: 05.04.2022) (in Russian).

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, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

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, кафедра СП,
   зам. главного редактора Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
2023 Том 12, № 2

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 25.05.2023. Дата выхода в свет 31.05.2023. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 12,55. Тираж 500 экз. Заказ 97/183. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.