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

2020, том 9, № 3

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2020, volume 9, no. 3

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2020
Т. 9, № 3

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"

2020
Vol. 9, no. 3

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

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

South Ural State University

The scope of the journal:

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

Содержание

Contents

УДК 004.932.4 DOI: 10.14529/cmse200301

О СВОЙСТВАХ АЛГОРИТМА СГЛАЖИВАНИЯ ЦВЕТНЫХ ИЗОБРАЖЕНИЙ НА ОСНОВЕ АНАЛИЗА ГРАДИЕНТА

© 2020 В.Ю. Гудков, И.Ю. Моисеев
Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: diana@sonda.ru, villeman.5@yandex.ru
Поступила в редакцию: 26.05.2020

В данной работе исследуются свойства и возможные приложения алгоритма сглаживания, который позволяет сохранять выраженные структуры на изображении и подавлять слабозаметные текстуры. Он основан на анализе двух компонент векторов градиента, отражающих изменение интенсивности цвета в окрестности определенной точки. Эти компоненты — длина и угол наклона вектора градиента. В основе теории, которая служит основанием изучаемому методу лежит различие между двумя видами границ на изображении, которые отличаются поведением векторов градиента. Предполагается, что близость углов градиента в точках окрестности говорит о принадлежности двух точек к одной границе, а значит, при сглаживании они должны иметь большее влияние на результат. Также принимается во внимание обратное значение длины вектора градиента как фактор формирования веса, который позволяет выделять края объектов. Мы ставим своей целью сфокусироваться на результатах применения алгоритма в качестве предобработки в задачах выделения контуров и подобных им, сглаживая лишние детали, которые не важны при формировании изображения контуров. Мы также выявили интересные свойства последствий применения алгоритма несколько итераций подряд и изучили его поведение в задаче борьбы с шумом.

Ключевые слова: сглаживание изображений, фильтр, градиент.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Гудков В.Ю., Моисеев И.Ю. О свойствах алгоритма сглаживания цветных изображений на основе анализа градиента // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 3. С. 5–16. DOI: 10.14529/cmse200301.

Введение

Сглаживание изображений является фундаментальной задачей в области компьютерной графики и компьютерного зрения. Его цель — устранение незначительных деталей изображения и сохранения его общей структуры. Но при решении этих задач всегда присутствует необходимость идти на компромисс между сглаживанием и сохранением. Сглаживание имеет широкий спектр применений в задачах компьютерной графики и других прикладных областях. К сферам применения относятся задачи выделения границ, сегментации, абстракции, улучшения изображений, где сглаживание выступает в качестве стадии предобработки и подобные. В данной работе мы изучаем свойства и возможный спектр приложений алгоритма сглаживания изображений, основанного на анализе двух компонент векторов градиента: их угла и длины.

В данной статье принята следующая структура. Проводится обзор существующих методов сглаживания изображений в разделе 1, после чего описывается теория и детали реализации метода, подлежащего исследованию, в разделе 2. Экспериментальные данные приводятся в разделе 3, где исследуются свойства алгоритма для различных приложений. В заключении приводится сводка полученных результатов, и предлагаются направления дальнейших исследований.

1. Обзор существующих подходов

В этом разделе будут приведены наиболее интересные подходы в области. В общих чертах рассматриваются как фундаментальные методы, так и последние достижения.

Долгая история развития подходов к развитию задачи свидетельствует о ее фундаментальности. Классическим и самым простым, например, является линейный усредняющий фильтр. Такой фильтр производит сглаживание, но сильно искажает структуру изображения. Позже появился билатеральный фильтр (bilateral filter, BF) [9]: такой нелинейный фильтр, который сглаживает изображение, сохраняя резкие контуры, а также фильтры, основанные на анизотропной диффузии. Стоит также упомянуть направляемый фильтр (guided filter), использующий специальное направляющее изображение, для сохранения контуров исходного. Анизотропная диффузия [8], как алгоритм, который позволяет подавлять шум, сохраняя при этом детали изображения, стала основой для большого количества исследований, направленных на более глубокое понимание модели, таких как, например, применение робастной статистики и вейвлетов.

В последние годы развитие получили методы, основанные на применении глобальной оптимизации [1], и таких методов машинного обучения как сверточные нейронные сети [3] и обучение без учителя [5]. Такие подходы хорошо себя показывают в задачах сглаживания изображений, но чаще всего являются неэффективными вычислительно и в этих случаях исследования направлены на ускорение существующих методов и уменьшение затрат оперативной памяти.

В целях упрощения дальнейшего описания, в данной работе будет принята следующая система категорий. Подходы к задаче сглаживания можно разделить на две категории [11]. Выделяются *явные* методы — такие, которые работают с явно выраженными фильтрами и *неявные* — такие, которые чаще всего формируются в виде задачи минимизации (взвешенной) суммы двух слагаемых: слагаемого данных и слагаемого сглаживания. Такая формулировка дается им не всегда: основная суть методов из этой категории это именно «неявность» — отсутствие явно выраженного фильтра с конкретными числовыми значениями. Метод, рассматриваемый в этой работе, относится к категории явных методов.

1.1. Явные методы сглаживания

Методы данной категории являются, как правило, менее затратными вычислительно, а возможно по этой причине, более распространенными. Подавляющее большинство ранней литературы по теме сглаживания посвящено именно этим подходам. Многие явные алгоритмы применяют фильтрацию маской (и рассматриваемый не исключение). Фильтрацию изображения маской можно описать выражением:

$$S(p) = \sum_{q \in N(p)} w(p,q) I(q), \tag{1}$$

где $S(p)$ — значение пикселя выходного изображения, $w(p,q)$ — значение веса маски фильтра (значения маски подчиняются условию нормировки — складываются в единицу), $I(q)$ — значение пикселя исходного изображения, $N(p)$ — окрестность пикселя p.

Это выражение является взвешенной суммой значений пикселей окрестности. Исходное изображение расширяется (классический подход — нулями) и для каждого пикселя вычисляется указанная взвешенная сумма.

Практически все методы этой категории можно, так или иначе, выразить формулой такого вида. Самые простые из явных подходов часто приводят к потере деталей изображения, размытию структуры. Одним из самых известных и фундаментальных подходов среди методов, использующих фильтры, является билатеральный фильтр, который был представлен в 1998 году в [9].

Этот фильтр заменяет значение пикселя средним значением среди похожих и ближайших пикселей. Он имеет два сглаживающих ядра: пространственное и яркостное. Данное свойство позволяет ему сглаживать небольшие колебания цвета в окрестности пикселя, но при этом сохранять жесткие переходы контуров. Билатеральный фильтр выражается формулой [10]:

$$BLF(I) = \frac{1}{\sum_{q \in w_k} W_{BLF_{pq}}(I)} \sum_{q \in w_k} W_{BLF_{pq}}(I) I_q,$$

где функция весов $W_{BLF_{pq}}(I)$ выражается как:

$$W_{BLF_{pq}}(I) = \exp\left(\frac{-\|p-q\|^2}{2\sigma_s^2}\right) \exp\left(\frac{-\left|I_p - I_q\right|^2}{2\sigma_r^2}\right).$$

Несмотря на идейную простоту, алгоритм является довольно сложным вычислительно, используя нелинейные функции и т.д. Это, впоследствии, обусловило появление большой массы улучшений билатерального фильтра в плане производительности, появились быстрые его реализации.

Схожим образом формулируется и направляемый фильтр (guided filter). Этот метод сглаживания изображений подразумевает использование направляющего изображения. Фильтрация происходит с учетом другого изображения, которое может быть как самим обрабатываемым изображением, так и каким-то иным. Метод был предложен в 2013 году в [6].

Существует также анизотропный скользящий фильтр [13] (anisotropic guidance filter). Он является одним из последних на данный момент улучшением другого подхода, а именно скользящего фильтра, представленного в работе [14]. Анизотропия является ключевым свойством, которое было достигнуто в ходе разработки данного подхода.

Изначальный подход можно описать как процесс, состоящий из двух шагов: размытие текстур с помощью гауссовой функции, а затем итеративное восстановление структуры. Исследователи заметили, что гауссов фильтр размывает слабые и небольшие изменения в яркости, но большие структуры сохраняются, будучи в размытом состоянии. Задача состоит в том, чтобы восстановить не до конца утерянные крупные детали изображения. Что касается нового метода, презентованного в [13], то в нем добиваются анизотропии, обобщая исходные гауссовы функции с использованием структурного тензора. Авторы утверждают, что при проведении таких преобразований над функциями достигается наиболее высокое качество сглаживания, чем при использовании изотропного скользящего фильтра.

1.2. Неявные методы сглаживания

Эта группа методов отличается многообразием формулировок подходов и зачастую большой вычислительной сложностью. Задача чаще всего формулируется в виде минимизации суммы [11]:

$$E(S) = E_d(S, I) + \lambda E_v(S),$$

где $E_d(S, I)$ — компонента данных, измеряющая разницу между исходным изображением I и тем, что получено на выходе S, $E_v(S)$ — компонента сглаживания, которая снижает вариабельность цветов в окрестности пикселя, а λ — параметр, балансирующий эти слагаемые.

В методе с оптимизацией L_0 градиента исследователи предлагают разреженную схему подсчета градиентов в оптимизационном фреймворке [12]. Основным нововведением является стратегия, подсчитывающая дискретное число изменений интенсивности цвета в окрестности пикселя, что математически ведет к норме L_0. При использовании метода стремятся сделать выделяющиеся границы тоньше, что улучшает их визуальную различимость. Задача минимизации имеет вид:

$$\min_S \left\{ \sum_p \left(S_p - I_p\right)^2 + \lambda\, C(S) \right\}.$$

При этом функция C(S) имеет вид:

$$C(S) = \#\{p \mid |\partial_x S_p| + |\partial_y S_p| \neq 0\},$$

где $\#\{\cdot\}$ — оператор, возвращающий количество элементов, для которых выполняется условие в скобках. В данном случае он подсчитывает число всех пикселей, для которых градиенты по обеим осям не равны нулю.

Форма функции для минимизации напоминает уже описанную в начале раздела. Задача минимизации этой функции нетривиальна, так как сама функция содержит параметр $C(S)$, являющийся дискретной недифференцируемой функцией. Авторы подхода разработали собственный алгоритм (подробности которого мы в данной работе опустим) для решения данной задачи, который предполагает введение новых переменных в целевую функцию.

Среди последних разработок популярностью пользуются подходы, использующие глубокие сверточные сети, позволяющие достигать зачастую лучших результатов, чем традиционные методы. В данном случае реализуется особый подход при использовании сетей [7]. Авторы полагают, что конструирование математического описания текстур является нетривиальной задачей само по себе. Также они отмечают, что текстуры на изображениях не всегда имеют слабые и небольшие значения градиентов, что, как правило, подразумевается при конструировании методов сглаживания. Для решения этих проблем авторы предлагают применить сверточные сети.

Был сгенерирован собственный набор данных для обучения сети. Замечено, что изображения, используемые в мультипликации, имеют края только такие, которые обозначают структуры, заполненные непрерывным цветом, без текстуры. На подобные изображения накладывались текстуры разных масштабов. Таким образом, избегая ручной разметки изображений на текстуры и структуры, был создан способ обучения сети выполнять сглаживание. Три различных сети использовались при решении задачи: две сети предсказывают по входному изображению маски его текстур и его структуру, после чего эти данные передаются на результирующую сеть для фильтрации. На выходе получился подход, реализующий фильтрацию на уровне, который далеко не всегда достижим локальными методами.

2. Описание исследуемого метода

В этом разделе будет приведено краткое описание исследуемого метода. Впервые метод был предложен в работе [4]. В разделе будет описана основная идея, стоящая за реализацией, что должно облегчить понимание результатов, полученных экспериментально в разделе 3, а также поведения алгоритма на тех или иных исходных данных.

2.1. Теоретическая основа

Как уже было упомянуто в разделе 1, исследуемый метод принадлежит к категории явных, а значит, формулируется согласно (1). При формировании исследуемого подхода рассматриваются два типа границ объектов, которые могут попадать в окрестность точки: это регулярные и нерегулярные границы. Пример, поясняющий эти две концепции, показан на рис. 1.

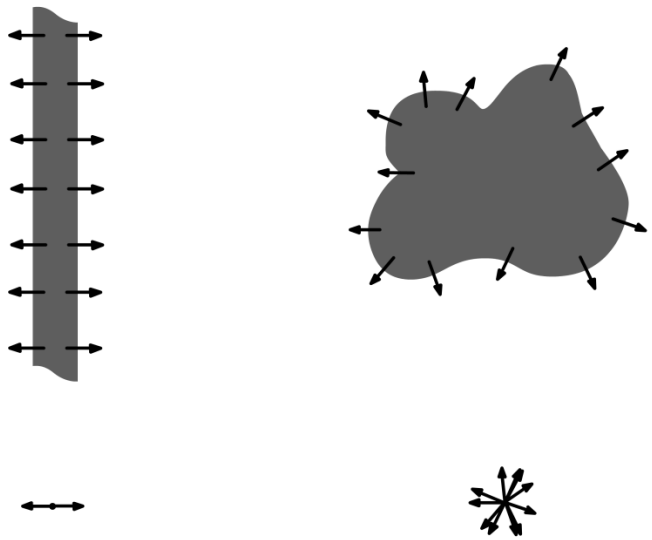


Рис. 1. Два разных вида границ и соответствующие им направления градиентов

Алгоритм реализован таким образом, что он сохраняет границы первого типа и размывает границы второго. Рассмотрим направления градиентов, которые производят два типа границ: градиенты на границах на рис. 1 изображены в двух видах: на самом изображении и сведенные в одну точку. Можно заметить, насколько сильно отличаются направления градиентов относительно друг друга у двух типов границ. При формировании весов в рассматриваемом подходе используется именно это свойство векторов градиента. Помимо направлений градиента, также предполагается, что пиксели с меньшим модулем градиента должны получать больший вес, исходя из предположения о том, что текстуры более однородны, чем границы между объектами.

Подытожим все, что было дано ранее. Веса для рассматриваемого фильтра конструируются таким образом, чтобы при применении он позволял сглаживать незначительные детали и оставлять границы объектов и общую структуру изображения нетронутыми. Для достижения этих свойств используются углы градиента и его длины, которые, по предположению, отличаются для тех областей, которые нужно сгладить и тех, которые нужно сохранить. Используя все вышесказанное, сформулируем теперь исследуемый фильтр математически.

2.2. Реализация

Во-первых, чтобы реализовать исследуемый фильтр, необходимо рассчитать, насколько углы градиентов в точках p и q близки друг к другу. Близость углов рассчитывается как косинус удвоенного угла между векторами. При получении углов градиентов получаются значения от $-\pi/2$ до $\pi/2$. А удвоенная разница между ними распределена от -2π до 2π. Цель удвоения разницы в том, чтобы сблизить углы, разница между которыми близка к π. Это делается исходя из предположения, что такие векторы принадлежат к одной границе. Определим теперь формулу для формирования весов. Используя ту же нотацию, что и в (1):

$$w(p,q) = (cos(\beta) + 1) / \alpha,$$

где α — это модуль градиента, а β — это удвоенная разница между углами градиентов в точках p и q. Также необходимо прибавлять единицу для того, чтобы не получать веса с отрицательным знаком, что может сказаться на расчете взвешенной суммы. Из особенностей реализации стоит отметить, что если в окрестности пикселя не меняется интенсивность цвета, то значения пикселя остаются без изменений.

3. Исследование свойств

В данном разделе изложены результаты экспериментов над изучаемым алгоритмом. Основная цель проводимых экспериментов — выявление возможных приложений и изучение свойств алгоритма, получение представления о качестве его работы.

Для экспериментов были подобраны произвольные изображения, не являющиеся стандартизованными, но обладающие характерными качествами, необходимыми нам для исследования свойств алгоритма. К таким качествам относится, например, наличие обоих типов границ, упомянутых в разделе 1.3. На таких изображениях присутствуют как крупные структуры, так и мелкие текстуры. Это те объекты, за поведением которых мы хотим понаблюдать, чтобы определить в конечном итоге свойства алгоритма и его спектр приложений.

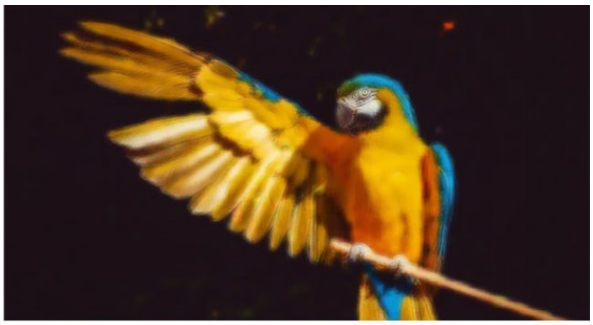
Мы продемонстрируем результаты работы алгоритма в задачах сглаживания, предобработки для задачи выделения контуров, а также в подавлении шума. Эксперименты по сглаживанию включают в себя применение алгоритма к подходящим для исследования изображениям и визуальную оценку качества. Особенность алгоритмов сглаживания в том, что хотя количественные метрики качества и присутствуют, но они не повсеместны — не все авторы приводят соответствующие вычисления. Как правило, такие метрики применяются для оценки качества работы алгоритмов для подавления шума. Сглаживание же не во всём аналогично задачам борьбы с шумом, хотя подобные метрики и применяются, экспертная оценка, как правило, с ними расходится. Мы можем оценить качество метода самого по себе либо визуально, либо косвенно, через оценку качества тех методов обработки изображений, в которых сглаживание можно использовать для предобработки. Такими методами, например, являются методы выделения контуров, где оценивая качество полученных изображений контуров можно оценить качество алгоритма сглаживания, использованного на стадии предобработки.

3.1. Сглаживание

Продемонстрируем для начала результаты применения фильтра и рассмотрим их подробно. На рис. 2 заметно, как достигаются желаемые свойства: сглаживаются мелкие

детали, и в то же время сохраняются границы объектов. Особенно на правой паре увеличенных картинок на рис. 2 в) заметно, как круглый блик на фоне почти исчез после сглаживания, в то время как граница объекта сохранилась.

а) Исходное изображение



б) Обработанное фильтром

в) Фрагменты исходного и обработанного изображений

Рис. 2. Результаты применения метода и их увеличенные фрагменты

Известно, что применение алгоритма в несколько итераций подряд бывает полезно, если необходим высокий уровень абстракции изображения, освобождение от мелких деталей. Рассмотрим поведение описываемого метода при применении его таким образом. На рис. 3 изображены подряд: исходное изображение — а), результат применения фильтра одну итерацию — б) и семь итераций подряд — в). Виден тот уровень абстракции, который достигается с помощью такого способа применения фильтра. Остаются только крупные объекты, сохраняющие свои границы.

а) Исходное изображение

б) Одна итерация

в) Семь итераций

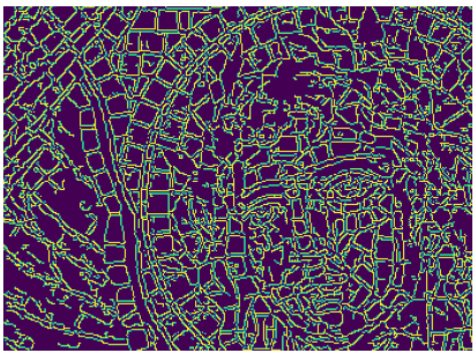
Рис. 3. Применение фильтра одну и семь итераций

3.2. Выделение контуров

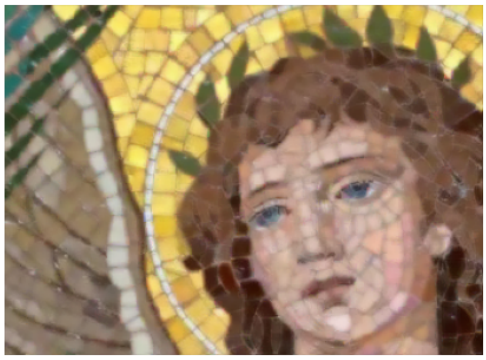
Зачастую алгоритмы сглаживания применяются в задачах извлечения контуров с изображения. При этом часто нас интересуют контуры именно у границ объектов, а не контуры текстур. По этой причине алгоритм сглаживания встроен прямо внутрь известного алгоритма для извлечения контуров — оператора Кэнни, который был представлен в работе [2] в 1986 году. В алгоритм оператора Кэнни встроена фильтрация гауссовым фильтром для удаления мелких деталей, которые могут мешать извлечению контуров. Но гауссов фильтр не способен учесть границы объектов на изображении, а значит, может понадобиться дополнительная предобработка.

Рассмотрим изображение с мозаикой как характерный пример изображения с множеством контуров, которые не являются контурами объектов, и попробуем применить к нему оператор Кэнни в двух случаях: без дополнительной обработки фильтром и с ней. На рис. 4 изображены результаты извлечения контуров с изображения напрямую а), б) и после предварительного сглаживания в), г). Заметна разница — выделились те границы, которые принадлежат объектам, в то время как большая часть текстурных границ подавилась сглаживанием. Подход может найти применение в задачах извлечения контуров, так как позволяет достигать приемлемого уровня абстракции изображения.

а) Исходное изображение



б) Контуры исходного изображения



в) Обработанное изображение

г) Контуры обработанного изображения

Рис. 4. Извлечение контуров с изображения напрямую и после сглаживания

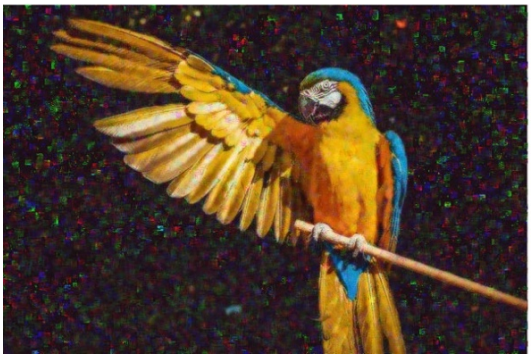
3.3. Подавление шума

Подавление шума также является характерной задачей для алгоритмов сглаживания, но не основной: для подавления шума, как правило, используются специализированные подходы. Мы, тем не менее, считаем нужным оценить поведение рассматриваемого фильтра в задаче подавления шума.

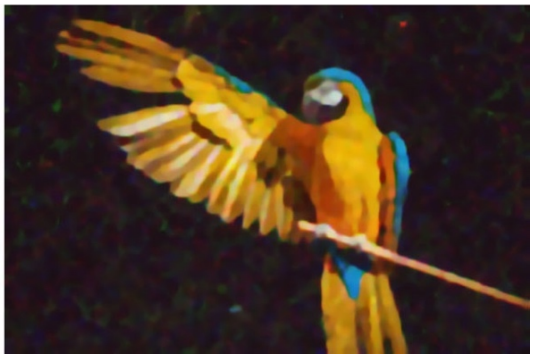
На рис. 5 мы видим результаты применения фильтра на зашумленном изображении. Шум равномерный небольшой концентрации, входное изображение показано рис. 5 а). На

паре картинок ниже снова два варианта применения фильтра: одна и пять итераций. Стоит отметить явный минус: после применения одной итерации шум не сгладился, не слился с фоном, а начал образовывать более крупные структуры, но далее по мере увеличения количества итераций он стал закономерно пропадать вместе с другими деталями. По этим результатам стоит отметить, что борьба с шумом может производиться в некоторой степени, но только как побочный эффект в рамках какой-то иной задачи. Если задача будет поставлена так, что нужно как можно более точно сохранить содержание изображения, как это делается для алгоритмов подавления шума, этот метод неприменим. Он может найти применение там, где требуется освобождение от мелких деталей, включая шум.

а) Исходное изображение



б) Одна итерация обработки



в) Пять итераций обработки

Рис. 5. Применение для устранения шума

Заключение

В данной работе проводится изучение свойств алгоритма сглаживания изображений на основе анализа градиентов. Изучаемый метод формирует веса в явной форме, исходя из свойств окрестности. Он позволяет по-разному обрабатывать два вида границ, встречающихся в окрестности изображения, благодаря анализу направления градиентов и их длин. Был проведен обзор существующих подходов и теоретическое основание метода, после чего были подробно рассмотрены результаты экспериментов. Рассматриваемый алгоритм хорошо показал себя в задачах сглаживания, абстракции и выделения контуров. Мы видим интересные результаты на упомянутых задачах: алгоритм позволяет выполнять сглаживание, сохраняя контуры объектов, при этом являясь простым в понимании и реализации. Отдельно стоит задача сглаживания зашумленных изображений, где алгоритм не показывает удовлетворительных результатов. Будущие исследования в этой об-

ласти предполагают поиск новых приложений, например, в области сегментации изображений, а также исследование свойств алгоритма на других задачах наряду со сравнением его с аналогами в количественных показателях.

Литература

1. Bi S. An L1 Image Transform for Edge-preserving Smoothing and Scene-level Intrinsic Decomposition // ACM Transactions on Graphics. 2015. Vol. 4, no. 34. P. 78–90. DOI: 10.1145/2766946.
2. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986. Vol. PAMI-8, no. 6. P. 679–698. DOI: 10.1109/TPAMI.1986.4767851.
3. Chen Q. Fast Image Processing with Fully-convolutional Networks // 2017 IEEE International Conference on Computer Vision, ICCV (Venice, Italy, October, 22–29, 2017), 2017. P. 2497–2506. DOI: 10.1109/iccv.2017.273.
4. Gudkov V. Image Smoothing Algorithm Based on Gradient Analysis // Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology, USBEREIT (Yekaterinburg, Russia, May, 14–15, 2020). 2020. P. 403–406. DOI: 10.1109/USBEREIT48449.2020.9117646.
5. Fan Q. Image Smoothing via Unsupervised Learning // ACM Transactions on Graphics. 2018. Vol. 37, no. 6. DOI: 10.1145/3272127.3275081.
6. He K. Guided Image Filtering // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013. Vol. 6, no. 35. P. 1397–1409. DOI: 10.1109/TPAMI.2012.213.
7. Lu K. Deep Texture and Structure Aware Filtering Network for Image Smoothing // Computer Vision, ECCV 2018. Lecture Notes in Computer Science. Springer, Cham, 2018. Vol. 11208. P. 229–245. DOI: 10.1007/978-3-030-01225-0_14.
8. Perona P. Scale-space and Edge Detection Using Anisotropic Diffusion // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1990. Vol. 7, no. 12. P. 629–639. DOI: 10.1109/34.56205.
9. Tomasi C. Bilateral Filtering for Gray and Color Images // Proc. of the IEEE International Conference on Computer Vision (Bombay, India, January, 7, 1998), 1998. P. 839–846.
10. Pham C. Adaptive Guided Image Filtering for Sharpness Enhancement and Noise Reduction // PSIVT 2011. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2011. Vol. 7087. P. 323–334. DOI: 10.1007/978-3-642-25367-6_29.
11. Xiaonan F. Learning Explicit Smoothing Kernels for Joint Image // Pacific Graphics. 2019. Vol. 7, no. 38. P. 180–190. DOI: 10.1111/cgf.13827.
12. Xu L. Image Smoothing via L0 Gradient // ACM Transactions on Graphics. 2011. Vol. 6, no. 30. P. 1–12. DOI: 10.1145/2070781.2024208.
13. Yoshimura K. Structure-tensor-based Anisotropic Rolling Filter for Image Smoothing // Proceedings of SPIE. 2019. Vol. 1104904. P. 13–18. DOI: 10.1117/12.2517892.
14. Zhang Q. Rolling Guidance Filter // Computer Vision-ECCV2014. Springer, Cham. 2014. P. 851–830. DOI: 10.1007/978-3-319-10578-9_53.

Гудков Владимир Юльевич, д.ф.-м.н., профессор, кафедра электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Моисеев Илья Юрьевич, студент, кафедра электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse200301

ON THE PROPERTIES OF AN IMAGE SMOOTHING ALGORITHM FOR COLORED IMAGES BASED ON GRADIENT ANALYSIS

© 2020 V.Y. Gudkov, I.Y. Moiseev
South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)
E-mail: diana@sonda.ru, villeman.5@yandex.ru
Received: 26.05.2020

In this paper properties and possible applications of an image smoothing algorithm are explored. The algorithm allows us to remove small textures and to preserve main structures on the image. Algorithm is based on the analysis of two gradient components. These components are the length and the angle of gradient vector. Theory that underlies the algorithm is based on distinction between two types of boundaries, which differ in behavior of gradient vectors. We suppose that closeness of gradient angles in given neighborhood means that points belong to the same boundary. This in turn means that they should have bigger weights. We also take into consideration inverted gradient length as the factor for weight computation. Our goal is to focus on the results of applying the algorithm as a preprocessing step for the tasks like edge detection. This method shows interesting results as a preprocessing step for edge detection tasks. It smooths insignificant details, from which we do not need edges to be shown at the image of the edges. We also explore interesting properties of using algorithm for several iterations and its behavior on noise reduction task.

Keywords: image smoothing, filter, gradient.

FOR CITATION

Gudkov V.Y., Moiseev I.Y. On the Properties of an Image Smoothing Algorithm for Colored Images Based on Gradient Analysis. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 3. P. 5–16. (in Russian) DOI: 10.14529/cmse200301.

References

1. Bi S. An L1 Image Transform for Edge-preserving Smoothing and Scene-level Intrinsic Decomposition. ACM Transactions on Graphics. 2015. Vol. 4, no. 34. P. 78–90. DOI: 10.1145/2766946.
2. Canny J. A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986. Vol. PAMI-8, no. 6. P. 679–698. DOI: 10.1109/TPAMI.1986.4767851.
3. Chen Q. Fast Image Processing with Fully-convolutional Networks. 2017 IEEE International Conference on Computer Vision, ICCV (Venice, Italy, October, 22–29, 2017). 2017. P. 2497–2506. DOI: 10.1109/iccv.2017.273.
4. Gudkov V. Image Smoothing Algorithm Based on Gradient Analysis. Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology, USBEREIT (Yekaterinburg, Russia, May, 14–15, 2020). 2020. P. 403–406. DOI: 10.1109/USBEREIT48449.2020.9117646.
5. Fan Q. Image Smoothing via Unsupervised Learning. ACM Transactions on Graphics 2018. Vol. 37, no. 6. DOI: 10.1145/3272127.3275081.

6. He K. Guided Image Filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013. Vol. 6, no. 35. P. 1397–1409. DOI: 10.1109/TPAMI.2012.213.
7. Lu K. Deep Texture and Structure Aware Filtering Network for Image Smoothing. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2018. Vol. 2, no. 11208. P. 229–245. DOI: 10.1007/978-3-030-01225-0_14.
8. Perona P. Scale-space and Edge Detection Using Anisotropic Diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1990. Vol. 7, no. 12. P. 629–639. DOI: 10.1109/34.56205.
9. Tomasi C. Bilateral Filtering for Gray and Color Images. Proceedings of the IEEE International Conference on Computer Vision (Bombay, India, January, 7, 1998). 1998. P. 839–846.
10. Pham C. Adaptive Guided Image Filtering for Sharpness Enhancement and Noise Reduction. PSIVT 2011. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2011. Vol. 7087. P. 323–334. DOI: 10.1007/978-3-642-25367-6_29.
11. Xiaonan F. Learning Explicit Smoothing Kernels for Joint Image. Pacific Graphics. 2019. Vol. 7, no. 38. P. 180–190. DOI: 10.1111/cgf.13827.
12. Xu L. Image Smoothing via L0 Gradient. ACM Transactions on Graphics. 2011. Vol. 6, no. 30. P. 1–12. DOI: 10.1145/2070781.2024208.
13. Yoshimura K. Structure-tensor-based Anisotropic Rolling Filter for Image Smoothing. Proceedings of SPIE. 2019. no. 1104904. P. 13–18. DOI: 10.1117/12.2517892.
14. Zhang Q. Rolling Guidance Filter. Computer Vision-ECCV2014. Springer, Cham. 2014. P. 851–830. DOI: 10.1007/978-3-319-10578-9_53.

УДК 004.272.25, 004.421, 004.032.24 DOI: 10.14529/cmse200302

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПОИСКА ЛЕЙТМОТИВОВ ВРЕМЕННОГО РЯДА ДЛЯ ГРАФИЧЕСКОГО ПРОЦЕССОРА*

© 2020 М.Л. Цымблер, Я.А. Краева

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

Поступила в редакцию: 26.07.2020

Лейтмотив представляет собой пару подпоследовательностей временного ряда, наиболее похожих друг на друга. Задача поиска лейтмотивов встречается в широком спектре предметных областей: медицина, биология, предсказание погоды и др. В работе предложен новый параллельный алгоритм поиска лейтмотива во временном ряде на платформе графического процессора для случая, когда входные данные могут быть размещены в оперативной памяти. Предлагаемый алгоритм использует в качестве основы алгоритм MK, в котором применяется евклидово расстояние и неравенство треугольника для отбрасывания бесперспективных лейтмотивов без вычисления расстояния. MK позволяет сократить время поиска в разы по сравнению с другими последовательными алгоритмами, однако его производительность значительно снижается на временных рядах, имеющих длину от сотен тысяч элементов. Распараллеливание выполнено с помощью технологии программирования OpenACC. Разработаны матричные структуры данных, позволяющие эффективно распараллелить вычисления на графическом процессоре. Представлены результаты вычислительных экспериментов на реальных и синтетических наборах данных, подтверждающих высокую масштабируемость разработанного алгоритма.

Ключевые слова: временной ряд, поиск лейтмотивов, параллельный алгоритм, NVIDIA GPU, OpenACC.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Цымблер М.Л., Краева Я.А. Параллельный алгоритм поиска лейтмотивов временного ряда для графического процессора // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 3. С. 17–34. DOI: 10.14529/cmse200302.

Введение

Лейтмотив временного ряда представляет собой пару подпоследовательностей этого ряда заданной длины, наиболее похожих друг на друга [20]. В настоящее время поиск лейтмотивов является актуальной задачей в широком спектре приложений обработки временных рядов: биоинформатика [3], обработка речи [2], прогнозирование природных катаклизмов [12], неврология [15] и др.

Поиск лейтмотива методом полного перебора подпоследовательностей временного ряда, очевидно, имеет временную сложность $O(n^2)$, где n — длина ряда. В силу этого в работах [5, 13, 14, 22, 23] был предложен ряд алгоритмов поиска *приближенного лейтмотива*, имеющих меньшую временную сложность $O(n)$ и $O(n \log n)$. Однако для ряда приложений, например, в сейсмологии [25], недопустима потеря точности результирующего лейтмотива, даже за счет выигрыша во времени поиска. Алгоритм MK [15] является одним из самых быстрых последовательных алгоритмов поиска *точного лейтмотива* и сокращает время

*Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2020».

поиска в разы по сравнению с другими алгоритмами, однако его производительность значительно снижается на временных рядах, имеющих длину от сотен тысяч элементов [15].

Одной из тенденций развития современной процессорной техники является увеличение количества вычислительных ядер вместо тактовой частоты [7]. В настоящее время ускорители архитектур Intel MIC (Many Integrated Core) [6] и NVIDIA GPU [18] обеспечивают от сотен до тысяч процессорных ядер и значительно опережают традиционные процессоры по производительности. В соответствии с этим перспективным направлением исследований является разработка параллельных алгоритмов поиска лейтмотивов временного ряда на ускорителях архитектур Intel MIC и NVIDIA GPU.

В настоящей статье предлагается новый параллельный алгоритм поиска лейтмотивов временного ряда на графическом процессоре (GPU) для случая, когда входные данные могут быть размещены в оперативной памяти. Данная статья продолжает исследование авторов, начатое в работе [26], где представлен параллельный алгоритм поиска лейтмотивов на ускорителях Intel MIC с помощью технологии параллельного программирования OpenMP [11]. В настоящем исследовании используются схожие базовые идеи, однако иная архитектура графического ускорителя и используемая в реализации соответствующая технология параллельного программирования OpenACC [8] требуют существенной переработки предложенных ранее технологических решений.

Статья организована следующим образом. В разделе 1 приводится формальная постановка задачи и кратко описан последовательный алгоритм MK. Раздел 2 содержит обзор работ по тематике исследования. В разделе 3 дано описание предлагаемого параллельного алгоритма. В разделе 4 представлены результаты вычислительных экспериментов по исследованию эффективности предложенного алгоритма. Заключение резюмирует результаты, полученные в рамках исследования.

1. Постановка задачи

1.1. Формальные определения и нотация

В данном разделе приводятся обозначения и определения используемых терминов в соответствии с работой [15].

Временной ряд представляет собой хронологически упорядоченную последовательность числовых значений: $T = (t_1, \ldots, t_n)$, $t_i \in \mathbb{R}$. Число n обозначается $|T|$ и называется длиной временного ряда.

Подпоследовательность $T_{i,m}$ временного ряда T представляет собой непрерывное подмножество T, состоящее из m элементов и начинающееся с позиции i: $T_{i,m} = (t_i, \ldots, t_{i+m-1})$, $1 \leqslant i \leqslant n - m + 1,\ m \ll n$.

Множество всех подпоследовательностей ряда T, имеющих длину m, обозначается S_T^m. Мощность указанного множества обозначим как N, $N = |S_T^m| = n - m + 1$.

В качестве *функции расстояния* между подпоследовательностями ряда возьмем неотрицательную симметричную функцию $\mathrm{Dist} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$.

Пара подпоследовательностей $\{T_{i,m}, T_{j,m}\}$ ряда T называется *лейтмотивом (motif)*, если

$$\begin{gathered} \forall a, b, i, j\ \ \mathrm{Dist}(T_{i,m}, T_{j,m}) \leqslant \mathrm{Dist}(T_{a,m}, T_{b,m}), \\ |i - j| \geqslant w,\ |a - b| \geqslant w,\ w > 0, \end{gathered} \tag{1}$$

где w — параметр, определяющий минимальный промежуток, на который должны отстоять

друг от друга подпоследовательности в лейтмотиве. Данный параметр позволяет отбрасывать лейтмотивы, которые состоят из взаимопересекающихся подпоследовательностей и потому не имеют практической ценности [20].

В качестве функции расстояния между двумя подпоследовательностями X и Y нами будет использоваться *евклидово расстояние*, определяемое следующим образом:

$$\mathrm{ED}(X,\ Y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}. \tag{2}$$

Для корректного определения схожести подпоследовательностей временного ряда, имеющих разные амплитуды, в дальнейшем изложении предполагается, что перед вычислением расстояния каждая подпоследовательность подвергается z-нормализации, обеспечивающей среднее арифметическое и стандартное отклонение элементов подпоследовательности, равные нулю и единице, соответственно. *Z-нормализация* подпоследовательности $C \in S_T^m$ представляет собой подпоследовательность $\widehat{C} = (\widehat{c}_1, \ldots, \widehat{c}_m)$, элементы которой вычисляются следующим образом:

$$\widehat{c}_i = \frac{c_i - \mu}{\sigma} :\ \mu = \frac{1}{m}\sum_{i=1}^{m} c_i,\ \sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m} c_i^2 - \mu^2}. \tag{3}$$

1.2. Последовательный алгоритм

Алгоритм MK [15] представляет собой один из самых быстрых последовательных алгоритмов для нахождения точного лейтмотива во временном ряде. Алгоритм сокращает пространство поиска лейтмотивов на основе введения т.н. опорных подпоследовательностей.

Опорная подпоследовательность представляет собой случайно выбранную подпоследовательность исходного ряда. Алгоритм вычисляет расстояние от каждой подпоследовательности ряда до опорной и упорядочивает все подпоследовательности в порядке возрастания вычисленных расстояний. Полученный порядок называется *линейным*. Затем используется следующее свойство: если два объекта конечномерного метрического пространства близки друг другу, то они также должны быть близки в линейном порядке (обратное утверждение неверно) [15]. В соответствии с неравенством треугольника расстояние между подпоследовательностями лейтмотива в линейном порядке является нижней границей расстояния между этими подпоследовательностями в пространстве $\mathbb{R}^m$:

$$\mathrm{ED}(ref,\, T_{i,\,m}) - \mathrm{ED}(ref,\, T_{j,\,m}) \leqslant \mathrm{ED}(T_{i,\,m},\, T_{j,\,m}),\ |i-j| \geqslant w,\ w > 0, \tag{4}$$

где ref — опорная подпоследовательность, $\{T_{i,\,m},\, T_{j,\,m}\}$ — пара подпоследовательностей, отличных от ref. Далее для того, чтобы различать два вышеупомянутых вида расстояний, расстояние между подпоследовательностями в пространстве $\mathbb{R}^m$ мы будем называть *истинным расстоянием*.

Переменная алгоритма bsf *(best-so-far)* представляет собой текущее минимальное истинное расстояние между подпоследовательностями лейтмотива, и обновляется алгоритмом, как только найдена пара подпоследовательностей, истинное расстояние между которыми меньше, чем bsf.

Просматривая подпоследовательности ряда в соответствии с их линейным порядком, алгоритм вычисляет нижние границы. Если нижняя граница превышает bsf, то истинное расстояние также превысит этот порог. Поэтому пара соответствующих подпоследовательностей заведомо не является лейтмотивом и может быть отброшена без вычисления истинного расстояния. Если пара подпоследовательностей не была отброшена, выполняется вычисление истинного расстояния. Если полученное значение меньше bsf, пороговое значение заменяется вычисленным.

Действуя таким образом, алгоритм просматривает все возможные пары подпоследовательностей, которые отстоят друг от друга в линейном порядке на величину $offsef$ $(1 \leqslant offset \leqslant N - 1)$. Просмотр продолжается до тех пор, пока не будет достигнуто такое значение $offset$, для которого не существует пар подпоследовательностей, нижние границы которых больше, чем bsf, после чего алгоритм завершается.

Для получения более узких нижних границ, позволяющих отбрасывать больше бесперспективных пар подпоследовательностей, алгоритм использует более одной опорной подпоследовательности. Опорная подпоследовательность с наибольшим стандартным отклонением используется для сортировки по возрастанию расстояний между этой опорной подпоследовательностью и всеми прочими подпоследовательностями исходного ряда. Если хотя бы одна из нижних границ больше, чем bsf, то соответствующая пара подпоследовательностей отбрасывается. Алгоритм завершается, если все нижние границы всех пар подпоследовательностей, отстоящих друг от друга на величину $offset$, больше, чем bsf.

Число опорных подпоследовательностей берется существенно меньшим, чем количество подпоследовательностей в исходном ряде. Как показывают эксперименты [15], количество опорных подпоследовательностей в диапазоне от 5 до 60 обеспечивает стабильное сокращение времени поиска по сравнению с другими алгоритмами в 2–3 раза независимо от типа данных и длины временного ряда, а также длины искомого лейтмотива.

2. Обзор работ

Поскольку алгоритм поиска лейтмотива во временном ряде методом полного перебора имеет квадратичную временную сложность относительно длины временного ряда, в работах [5, 13, 14, 22, 23] были предложены алгоритмы поиска приближенного лейтмотива во временном ряде, которые имеют линейную или логарифмическую сложность.

В работе [24] Вилсон и др. представили алгоритм FLAME для нахождения точного лейтмотива в ДНК. Однако алгоритм FLAME является приближенным для дискретных временных рядов, значения которых представляют собой вещественные числа.

В работе [15] Муином, Кеогом и др. предложен алгоритм MK, который находит точный лейтмотив во временном ряде. Алгоритм MK основан на следующей идее. Рассмотрим подпоследовательности исходного ряда как конечное множество точек конечномерного метрического пространства. Выберем случайным образом некую точку данного множества и назовем ее опорной. Упорядочим точки исходного множества по возрастанию их расстояния до опорной точки и назовем такой порядок линейным. Для любой пары точек, отличных от опорной, верно следующее утверждение: если точки находятся близко друг к другу в исходном пространстве, то они также будут располагаться близко в линейном порядке (обратное утверждение неверно). Данное свойство вкупе с неравенством треугольника позволяет отбрасывать пары подпоследовательностей, заведомо не являющихся лейтмотивом, без вычисления расстояния. Для большего сокращения пространства поиска алгоритм использует

несколько опорных точек. Эксперименты показывают, что алгоритм MK позволяет в разы ускорить поиск лейтмотива полным перебором [15].

В работе [17] Наранг и др. предложили параллельный алгоритм Par-MK для точного обнаружения лейтмотива на SMP системах для случая, когда данные полностью помещаются в оперативную память. Для распараллеливания Par-MK использует нити стандарта POSIX [19]. Каждая нить обрабатывает часть отсортированного массива, содержащего расстояния до опорной подпоследовательности. Алгоритм также распараллеливает цикл просмотра пар подпоследовательностей, отстоящих друг от друга в линейном порядке, выполняемый по величине этого отступа. Нити, которые обрабатывают разные значения отступа, но одну и ту же часть массива расстояний, выполняются на ядрах, которые разделяют кэш второго уровня.

Для устранения дисбаланса загрузки нитей из-за непредсказуемого и неравномерного количества подпоследовательностей, отбрасываемых различными нитями, авторы предложили две версии своего алгоритма: Par-MK-SLB и Par-MK-DLB (соответственно статическая и динамическая балансировка нагрузки). В вычислительных экспериментах на реальном временном ряде длины $|T| = 1,8 \cdot 10^5$ при длине лейтмотива $m = 200$, лучшая версия алгоритма продемонстрировала сверхлинейное ускорение на 32 ядрах благодаря динамической балансировке нагрузки в сочетании с превосходной производительностью кэша второго уровня. Однако на синтетическом временном ряде длины $|T| = 5 \cdot 10^4$ при длине лейтмотива $m = 1024$ ускорение лучшей версии уменьшилось более чем в 8,5 раза из-за снижения производительности кэша за счет большей длины лейтмотива. В своей дальнейшей работе авторы планировали модернизировать алгоритм для многоядерных ускорителей, но это исследование не было проведено.

Данная работа продолжает исследования авторов настоящей статьи [1, 10, 26, 27], посвященные разработке параллельных алгоритмов решения различных задач интеллектуального анализа временных рядов для современных многоядерных ускорителей. В работе [26] авторами представлен параллельный алгоритм поиска лейтмотивов на ускорителях Intel MIC для случая, когда входные данные могут быть размещены в оперативной памяти. Алгоритм использует набор матрично-векторных структур данных для хранения и индексации временного ряда и лейтмотивов, обеспечивающих эффективную векторизацию вычислений на платформе Intel MIC. Реализация выполнена на основе технологии параллельного программирования OpenMP [11]. Эксперименты показали высокую масштабируемость параллельного алгоритма и его более высокую производительность при исполнении на многоядерном ускорителе, чем на узле, состоящем из двух многоядерных процессоров Intel. Механический перенос данной разработки на графический ускоритель, однако, невозможен, как в силу существенных различий как между архитектурами Intel MIC и GPU, так и ввиду отсутствия реализации технологии параллельного программирования OpenMP для GPU. Технология параллельного программирования OpenACC [8], хотя и является идеологическим аналогом OpenMP для GPU, помимо разницы в синтаксисе, имеет ряд существенных семантических отличий, которые обуславливают переработку имеющегося решения.

3. Параллельный поиск лейтмотивов на GPU

В данном разделе описан подход к распараллеливанию описанного выше алгоритма MK [15] для графического процессора. Ниже в разделе 3.1 изложены особенности данной аппаратной платформы и применяемой технологии параллельного программирования. В

разделах 3.2 и 3.3 приводится описание структур данных и реализации параллельного алгоритма соответственно.

3.1. Аппаратно-программная платформа

Графический процессор (Graphics Processing Unit, GPU) компании NVIDIA [18] представляет собой один из наиболее популярных в настоящее время многоядерных ускорителей. GPU имеет иерархическую архитектуру и состоит из симметричных потоковых мультипроцессоров (Streaming Multiprocessor, SM). Каждый мультипроцессор, в свою очередь, состоит из симметричных ядер, называемых CUDA-ядрами (Compute Unified Device Architecture, CUDA — вычислительная унифицированная архитектура устройства). Современные GPU насчитывают тысячи CUDA-ядер, способных опередить по производительности центральные процессоры на задачах, допускающих массивно-параллельные вычисления в сочетании с векторной обработкой данных.

Параллельное приложение запускается на GPU как набор нитей, где каждая нить исполняется отдельным CUDA-ядром и предусмотрена следующая иерархия нитей. Верхним уровнем иерархии, соответствующим всем нитям, является *сетка нитей (grid)*, которая состоит из одномерного или двумерного массива симметричных блоков нитей. *Блок нитей (thread block)* представляет собой d-мерный ($1 \leqslant d \leqslant 3$) массив нитей. Внутри блока нити логически разделяются на группы по 32 нити — *варпы (warp)*. Нити варпа исполняются в режиме *SIMT (Single Instruction Multiple Threads)*, когда каждая нить выполняет одну и ту же инструкцию над собственной порцией общих данных.

При запуске приложения блоки нитей распределяются для исполнения между потоковыми мультипроцессорами и выполняются далее параллельно без возможности их синхронизации. Нити в пределах блока допускают синхронизацию и имеют доступ к разделяемой памяти, отведенной для данного блока. Для передачи данных между потоками, принадлежащих разным блокам, используется глобальная память ускорителя.

В настоящее время для графических процессоров разработаны технологии параллельного программирования CUDA [4], OpenCL [16] и OpenACC [8], последняя из которых применена в данном исследовании. *OpenACC* представляет собой открытый стандарт параллельного программирования для создания гетерогенных параллельных программ, задействующих как центральный, так и графический процессоры. Стандарт включает библиотеку функций, переменные окружения и директивы компилятора для определения участков исходного кода программы, которые необходимо выполнить на графическом процессоре.

Модель исполнения OpenACC-приложения предусматривает иерархию нитей и соответствующие уровни параллелизма. Одна или более нитей составляют *бригаду (gang)*, исполняемую на одном потоковом мультипроцессоре. В рамках бригады определяется одна или более симметричных групп нитей — *рабочих (worker)*. Внутри рабочего нити исполняются в режиме SIMT, обеспечивая еще один, *векторный (vector)* уровень параллелизма.

Одной из основных директив компилятора в технологии OpenACC является `#pragma acc parallel loop`, которая распараллеливает цикл с фиксированным количеством повторений, равномерно распределяя итерации цикла между нитями бригад для исполнения (при отсутствии между итерациями цикла зависимостей по данным). Указанная директива может быть дополнена одним или несколькими ключевыми словами `gang`, `worker`, `vector`, которые обяжут компилятор применить в данном цикле соответствующие уровни параллелизма.

3.2. Реализация структур данных

Для хранения данных в оперативной памяти графического процессора предлагаемый алгоритм использует матрицы и массивы, что обеспечивает возможность векторизации распараллеливаемых вычислений.

Пусть варп графического процессора состоит из $size_{warp}$ нитей. Если длина искомого лейтмотива m не кратна $size_{warp}$, то подпоследовательность выравнивается посредством дополнения фиктивными нулевыми элементами. Выравнивание данных позволяет избежать простаивания нитей в варпе, снижающего производительность вычислений. Обозначим количество фиктивных элементов через $pad = size_{warp} - (m \bmod size_{warp})$, тогда выровненная подпоследовательность $\tilde{T}_{i,\,m}$ определяется следующим образом:

$$\tilde{T}_{i,\,m} = \begin{cases} t_i, t_{i+1}, \ldots, t_{i+m-1}, \underbrace{0, \ldots, 0}_{pad}, & if\ m \bmod size_{warp} > 0 \\ t_i, t_{i+1}, \ldots, t_{i+m-1}, & otherwise. \end{cases} \tag{5}$$

Все выровненные подпоследовательности ряда хранятся в *матрице подпоследовательностей* $S_T^m \in \mathbb{R}^{N \times (m+pad)}$, которая определяется следующим образом:

$$S_T^m(i, j) = \tilde{t}_{i+j-1}. \tag{6}$$

Обозначим количество опорных подпоследовательностей за r $(0 < r \ll N)$. Тогда *матрица опорных подпоследовательностей* $Ref \in \mathbb{R}^{r \times (m+pad)}$ определяется следующим образом:

$$Ref(i, \cdot) = T_{i_k,\,m} :\ T_{i_k,\,m} \in S_T^m,\ 1 \leqslant k \leqslant r,\ i_k = random(1..N),\ \forall p \neq q\ i_p \neq i_q. \tag{7}$$

Индекс опорных подпоследовательностей $I_{Ref} \in \mathbb{N}^r$ для каждой опорной подпоследовательности хранит позицию ее начала во временном ряде.

Матрица расстояний $D \in \mathbb{R}^{r \times N}$ предназначена для хранения истинного расстояния между каждой опорной подпоследовательностью и каждой подпоследовательностью ряда и определяется следующим образом:

$$D(i, j) = \mathrm{ED}(T_{I_{Ref}(i),\,m}, T_{j,\,m}). \tag{8}$$

Вектор стандартных отклонений представляет собой массив $SD \in \mathbb{R}^r$, который для каждой опорной подпоследовательности содержит стандартное отклонение ее истинного расстояния до каждой подпоследовательности ряда.

Индекс стандартных отклонений представляет собой массив $I_{SD} \in \mathbb{N}^r$, содержащий уникальные числа в диапазоне от 1 до r, где числа соответствуют номерам опорных подпоследовательностей в Ref, упорядоченных по убыванию их стандартного отклонения.

Индекс подпоследовательностей $I_S \in \mathbb{N}^N$ представляет собой массив, который содержит позиции подпоследовательностей в матрице S_T^m, упорядоченные по возрастанию их истинных расстояний до опорной подпоследовательности, имеющей наибольшее стандартное отклонение.

Индекс лейтмотива $I_M \in \mathbb{N}^{N \times 2}$ предназначен для хранения позиций двух подпоследовательностей в матрице S_T^m, которые являются потенциальным лейтмотивом и в индексе подпоследовательностей I_S отстоят друг от друга на величину $offset$.

Матрица нижних границ $LB \in \mathbb{R}^{r \times N}$ содержит значения нижних границ между каждой опорной подпоследовательностью и каждым возможным лейтмотивом и в соответствии с (4) вычисляется следующим образом:

$$LB(i,j) = |D\big(I_{SD}(i), I_M(j,1)\big) - D\big(I_{SD}(i), I_M(j,2)\big)|. \tag{9}$$

Битовая карта представляет собой массив булевых значений $B \in \mathbb{B}^N$, в котором для каждого потенциального лейтмотива хранится результат конъюнкции проверок превышения текущим значением порога bsf каждой из нижних границ. Если элемент битовой карты равен `FALSE`, то соответствующий лейтмотив отбрасывается без вычисления истинного расстояния между парой подпоследовательностей. Битовая карта определяется следующим образом:

$$B(i) = \wedge_{j=1}^{r}(LB(i,j) < bsf). \tag{10}$$

3.3. Реализация алгоритма

Предлагаемая параллельная реализация поиска лейтмотива временного ряда представлена в алг. 1. Входными данными алгоритма являются временной ряд T, длина искомого лейтмотива m, минимальный промежуток между подпоследовательностями лейтмотива w и количество опорных подпоследовательностей r. Алгоритм возвращает найденный лейтмотив в виде кортежа, состоящего из двух подпоследовательностей и величины расстояния между ними.

Алгоритм выполняется следующим образом. Сначала на центральном процессоре формируются основные структуры данных: матрицы выровненных подпоследовательностей S_T^m и опорных подпоследовательностей Ref, а также их индексы I_S и I_{Ref} соответственно, — а затем с помощью директивы OpenACC `#pragma acc data` передает их на графический процессор. Затем выполняются фазы предварительной обработки данных и нахождения лейтмотива, рассматриваемые ниже в разделах 3.3.1 и 3.3.2 соответственно.

3.3.1. Предварительная обработка данных

Фаза предварительной обработки данных предполагает z-нормализацию строк матрицы подпоследовательностей S_T^m и вычисление на ее основе структур данных, описанных выше в разделе 3.2: матрица расстояний D, вектор стандартных отклонений SD и индекс стандартных отклонений I_{SD}, а также вычисление начального значения порога bsf и генерация индексов левых частей предполагаемых лейтмотивов $I_M(\cdot, 1)$.

Z-нормализация (см. алг. 2) реализуется с помощью двух вложенных циклов. Внешний цикл по строкам матрицы подпоследовательностей распараллеливается на уровне бригады с помощью директивы `#pragma acc parallel loop gang`. Внутренние циклы по элементам подпоследовательностей, выполняющие вычисление среднего арифметического и стандартного отклонения в соответствии с (3) и обновление элементов подпоследовательностей нормализованными значениями, распараллеливаются на уровне вектора с помощью директивы `#pragma acc parallel loop vector`.

Вычисление матрицы расстояний (см. алг. 3) предполагает два вложенных цикла, внешний из которых выполняется параллельно бригадами нитей, а внутренний распараллеливается на векторном уровне. Поскольку циклы независимы и количество итераций во внутреннем цикле больше, чем во внешнем, используется стандартный прием, обеспечивающий

Алг. 1 MotifDiscovery(in T, m, w, r; out $motif$)

▷ **Инициализация**

1: $Ref \leftarrow r$ случайно выбранных подпоследовательностей из S_T^m
2: $I_{Ref} \leftarrow r$ индексов элементов в Ref
3: #pragma acc data create(LB, SD, B, I_M) copyin(S_T^m, D, I_{Ref}, I_S, N, m, w, r) copyout($motif$)
4: {

▷ **Фаза предварительной обработки данных**

5: $S_T^m \leftarrow$ Znormalize(S_T^m)
6: $D \leftarrow$ CalculateDistances(S_T^m, I_{Ref})
7: $SD \leftarrow$ CalculateStdDev(D)
8: $I_{SD} \leftarrow$ Sort(SD); $I_S \leftarrow$ Sort($D(I_{SD}(1), \cdot)$)
9: $bsf \leftarrow$ InitBSF(D, I_{Ref}, bsf)
10: $I_M(\cdot, 1) \leftarrow$ GeneratePairs(I_S, $offset$)

▷ **Фаза поиска лейтмотива**

11: **for all** $offset \in 1..N$ **do**
12: $abandon \leftarrow$ `FALSE`
13: $I_M(\cdot, 2) \leftarrow$ GeneratePairs(I_S, $offset$)
14: $LB \leftarrow$ CalculateLowerBounds(D, I_{Ref}, I_M, bsf)
15: $abandon \leftarrow$ Verify(LB, I_M, bsf, B, $abandon$)
16: **if** $abandon$ **then**
17: **break**
18: **else**
19: $bsf \leftarrow$ UpdateBSF(S_T^m, B, I_M, bsf, $motif$)
20: }
21: **return** $motif$

Алг. 2 Znormalize(in out S_T^m)

1: #pragma acc parallel loop gang
2: **for all** $i \in 1..N$ **do**
3: $mean \leftarrow 0$; $std \leftarrow 0$
4: #pragma acc loop vector reduction(+: $mean$, std)
5: **for all** $j \in 1..m$ **do**
6: $mean \leftarrow mean + S_T^m(i, j)$
7: $std \leftarrow std + S_T^m(i, j)^2$
8: $mean \leftarrow \frac{mean}{m}$; $std \leftarrow \frac{std}{m}$; $std \leftarrow \sqrt{(std - mean^2)}$
9: #pragma acc loop vector
10: **for all** $j \in 1..m$ **do**
11: $S_T^m(i, j) \leftarrow \frac{S_T^m(i,j) - mean}{std}$

бóльшую степень параллелизма: свертка указанных циклов в один, выполняемая добавлением в директиву распараллеливания внешнего цикла атрибута `collapse(2)`.

Вычисление вектора стандартных отклонений (см. алг. 4) организуется как два вложенных цикла: внешний — по опорным подпоследовательностям, внутренний — по элементам

Алг. 3 CalculateDistances(in S_T^m, I_{Ref}; out D)

```
1: #pragma acc parallel loop gang collapse(2)
2: for all i ∈ 1..r do
3:     for all j ∈ 1..N do
4:         d ← 0
5:         #pragma acc loop vector reduction(+: d)
6:         for all k ∈ 1..m do
7:             d ← d + (S_T^m(j,k) − S_T^m(I_Ref(i),k))^2
8:         D(i,j) ← √d
```

матрицы расстояний. Поскольку количество опорных подпоследовательностей существенно меньше, чем количество нитей, запущенных на графическом процессоре (см. раздел 1.2), распараллеливанию подвергается только внутренний цикл, на бригадном и векторном уровнях.

Алг. 4 CalculateStdDev(in D; out SD)

```
1: for all i ∈ 1..r do
2:     mean ← 0; std ← 0
3:     #pragma acc parallel loop gang vector reduction(+: mean, std)
4:     for all j ∈ 1..N do
5:         mean ← mean + D(i,j)
6:         std ← std + D(i,j)^2
7:     mean ← mean/(N−1); std ← std/(N−1)
8:     SD(i) ← √(std − mean^2)
```

В инициализации порога bsf минимумом матрицы расстояний D (см. алг. 5) применяется двухуровневое распараллеливание вычислений (бригадное и векторное), а также прием свертки циклов, рассмотренный выше.

Алг. 5 InitBSF(in D, I_{Ref}; out bsf)

```
1: bsf ← +∞
2: #pragma acc parallel loop gang vector collapse(2) reduction(min: bsf)
3: for all i ∈ 1..r do
4:     for all j ∈ 1..N do
5:         if |j − I_Ref(i)| ⩾ w then
6:             bsf ← min(bsf, D(i,j))
```

3.3.2. Поиск лейтмотива

Фаза поиска лейтмотива представляет собой просмотр пар подпоследовательностей, отстоящих друг от друга на $offset$ позиций и реализуется с помощью цикла по указанной переменной. На каждой итерации цикла индекс лейтмотива $I_M(\cdot, 2)$ заполняется индексами подпоследовательностей из правой части возможного лейтмотива, которые смещены на $offset$ позиций относительно индекса $I_M(\cdot, 1)$ в индексе I_S.

Затем в соответствии с неравенством треугольника осуществляется параллельное вы-

численение нижних границ потенциальных лейтмотивов и заполнение матрицы нижних границ LB (см. алг. 6), выполняемые с помощью двухуровневого распараллеливания и свертки циклов.

Алг. 6 CalculateLowerBounds(in D, I_{Ref}, I_M; out LB)

```
1: #pragma acc parallel loop gang vector collapse(2)
2: for all i ∈ 1..r do
3:     for all j ∈ 1..N − offset − 1 do
4:         LB(i, j) ← |D(I_Ref(i), I_M(j, 1)) − D(I_Ref(i), I_M(j, 2))|
```

Далее осуществляется проверка потенциальных лейтмотивов путем вычисления битовой карты и нахождение условия останова поиска лейтмотива *abandon* (см. алг. 7). Если дизъюнкция всех элементов битовой карты дает в результате `TRUE` (все нижние границы для всех пар подпоследовательностей, отстоящих друг от друга на $offset$ позиций, больше, чем текущее значение порога bsf), то результирующий лейтмотив найден, а остальные кандидаты могут быть отброшены. Проверка реализуется посредством двух вложенных циклов: внешний — по потенциальным лейтмотивам и внутренний — по элементам матрицы нижних границ. Внешний цикл распараллеливается на уровнях бригад и вектора, а с помощью конструкции `reduction` обеспечивается нахождение условия останова поиска лейтмотива *abandon*. Поскольку каждая порождаемая внешним циклом нить вычисляет несколько элементов битовой карты, для корректного результата внутренний цикл должен исполняться последовательно, что обеспечивается директивой компилятора `#pragma acc parallel seq`.

Алг. 7 Verify(in LB, I_M, bsf; in out B, *abandon*)

```
1: #pragma acc parallel loop gang vector reduction(OR: abandon)
2: for all i ∈ 1..N − offset − 1 do
3:     B(i) ← TRUE
4:     if |I_M(i, 1) − I_M(i, 2)| ⩾ w then
5:         #pragma acc loop seq
6:         for all j ∈ 1..r do
7:             B(i) ← B(i) and (LB(j, i) < bsf)
8:         abandon ← abandon or B(i)
9: abandon ← not abandon
```

Если описанная выше процедура проверки не выявила лейтмотив, то выполняется обновление порога bsf следующим образом (см. алг. 8). Вычисляется истинное расстояние между подпоследовательностями в каждом потенциальном лейтмотиве, для которого соответствующий элемент битовой карты равен `TRUE`. Порог обновляется вычисленным значением истинного расстояния, если оно меньше текущего значения bsf. Цикл, выполняющий вычисление истинного расстояния, распараллеливается на уровнях бригады и рабочего и дополнен конструкцией `reduction` для свертки операции нахождения минимума bsf среди значений, вычисленных запущенными нитями.

Алг. 8 UpdateBSF(in S_T^m, B, I_M; in out bsf; out $motif$)

```
#pragma acc parallel loop gang worker reduction(min: bsf)
for all i ∈ 1..N − offset − 1 do
    if B(i) and |I_M(i,1) − I_M(i,2)| ⩾ w then
        d ← 0
        #pragma acc loop vector reduction(+: d)
        for all j ∈ 1..m do
            d ← d + (S_T^m(I_M(i,1), j) − S_T^m(I_M(i,2), j))^2
        if bsf > √d then
            bsf ← √d
            motif ← {I_M(i,1); I_M(i,2); bsf}
```

4. Вычислительные эксперименты

Для исследования эффективности разработанного алгоритма нами были проведены вычислительные эксперименты на графическом процессоре NVIDIA GeForce RTX 2080 Ti[1] со следующими характеристиками: количество ядер — 4362 (68 мультипроцессоров), частота ядра — 1,35 ГГц, память — 11 Гб, пиковая производительность — 11 TFLOPS.

В экспериментах исследовались ускорение и параллельная эффективность алгоритма, понимаемые как его способность адекватно адаптироваться к увеличению параллельно работающих блоков мультипроцессоров графического ускорителя при варьируемой длине искомого лейтмотива.

Ускорение и параллельная эффективность параллельного алгоритма, запускаемого на k блоках мультипроцессоров, вычисляются как $s(k) = \frac{t_1}{t_k}$ и $e(k) = \frac{s(k)}{k}$ соответственно, где t_1 и t_k — время работы алгоритма на одном и на k мультипроцессорах соответственно. При измерении времени работы не учитывалось время, затрачиваемое алгоритмом на загрузку данных в память графического процессора и на выдачу результата.

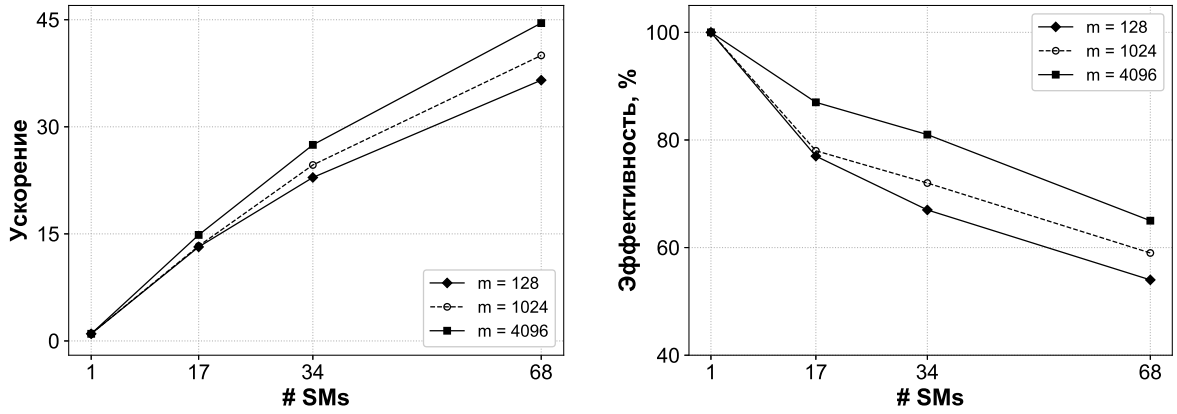
В экспериментах использовались синтетический и реальный временные ряды, состоящие из 10^5 элементов. Генерация синтетического временного ряда осуществлена на основе модели случайных блужданий (Random Walk) [21]. Реальный временной ряд взят из работы [9] и представляет собой сигналы ЭКГ, снятые с дискретизацией 128 Гц. Количество опорных подпоследовательностей в экспериментах взято $r = 10$.

Результаты экспериментов представлены на рисунке. Можно видеть, что разработанный параллельный алгоритм демонстрирует ускорение, близкое к линейному, и параллельную эффективность от 50 до 100 процентов (в зависимости от длины искомого лейтмотива). При этом лучшие показатели ожидаемо наблюдаются при бóльших значениях длины искомого лейтмотива, обеспечивающих алгоритму бóльшую вычислительную нагрузку.

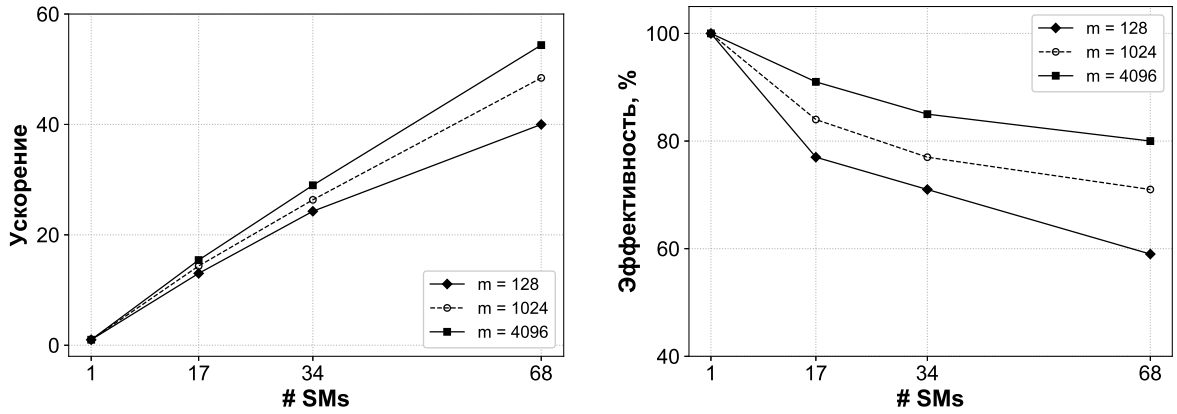
Заключение

В настоящей статье рассмотрена задача ускорения поиска лейтмотива временного ряда на современном графическом процессоре. Лейтмотив представляет собой пару подпоследовательностей временного ряда, наиболее похожих друг на друга. Задача поиска лейтмотивов встречается в широком спектре предметных областей: биоинформатика, обработка речи, прогнозирование природных катаклизмов, неврология и др.

[1]Спецификации видеокарты GeForce RTX 2080 Ti



а) синтетический ряд Random Walk ($n = 10^5$)



б) реальный ряд ЭКГ ($n = 10^5$)

Рис. Ускорение и параллельная эффективность алгоритма

Разработан новый параллельный алгоритм поиска лейтмотива временного ряда на графическом процессоре для случая, когда временной ряд может быть размещен в оперативной памяти. Предложенный алгоритм является параллельной версией последовательного алгоритма MK [15]. Распараллеливание выполнено с помощью технологии программирования OpenACC. Разработаны матричные структуры данных, позволяющие эффективно распараллелить и векторизовать вычисления на графическом процессоре. Представлены результаты вычислительных экспериментов, показывающие высокую масштабируемость алгоритма.

Проведенное исследование может быть продолжено в следующих направлениях: разработка версии с алгоритма использованием технологии программирования CUDA, а также распределенной версии алгоритма для высокопроизводительных кластеров с вычислительными узлами на базе графических процессоров.

Авторы благодарят Ксению Юрьевну Никольскую за помощь в проведении вычислительных экспериментов.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 20-07-00140) и Министерства образования и науки РФ (государственное задание FENU-2020-0022).

Литература

1. Цымблер М.Л. Параллельный алгоритм поиска диссонансов временного ряда для многоядерных ускорителей // Вычислительные методы и программирование: Новые вычислительные технологии. 2019. Т. 20, № 3. С. 211–223. DOI: 10.26089/NumMet.v20r320
2. Balasubramanian A., Wang J., Prabhakaran B. Discovering Multidimensional Motifs in Physiological Signals for Personalized Healthcare // J. Sel. Topics Signal Processing. 2016. Vol. 10, no. 5. P. 832–841. DOI: 10.1109/JSTSP.2016.2543679.
3. Brown A., Yemini E., Grundy L., Jucikas T., Schafer W. A Dictionary of Behavioral Motifs Reveals Clusters of Genes Affecting Caenorhabditis Elegans Locomotion // Proceedings of the National Academy of Sciences of the United States of America. 2012. Vol. 110, no. 2. P. 791–796. DOI: 10.1073/pnas.1211447110.
4. Cheng J., Grossman M., McKercher T. Professional CUDA C Programming. 1st Edition. Wrox, 2014. 528 p.
5. Chiu B.Y., Keogh E.J., Lonardi S. Probabilistic Discovery of Time Series Motifs // Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03 (Washington, D.C., USA, August, 24–27, 2003). ACM, 2003. P. 493–498. DOI: 10.1145/956750.956808.
6. Duran A., Klemm M. The Intel Many Integrated Core Architecture // Proceedings of the 2012 International Conference on High Performance Computing and Simulation, HPCS 2012 (Madrid, Spain, July, 2–6, 2012). 2012. P. 365–366. DOI: 10.1109/HPCSim.2012.6266938.
7. Fang J., Varbanescu A.L., Sips H.J. Sesame: A User-Transparent Optimizing Framework for Many-Core Processors // Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013 (Delft, Netherlands, May, 13–16, 2013). 2013. P. 70–73. URL: 10.1109/CCGrid.2013.79.
8. Farber R. Parallel Programming with OpenACC. 1st Edition. Morgan Kaufmann, 2016. 326 p.
9. Goldberger A., Amaral L., Glass L., Hausdorff J., Ivanov P., et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals // Circulation. 2000. Vol. 101, no. 23. P. 215–220. DOI: 10.1161/01.CIR.101.23.e215.
10. Kraeva Ya., Zymbler M. Scalable Algorithm for Subsequence Similarity Search in Very Large Time Series Data on Cluster of Phi KNL // 20th International Conference on Data Analytics and Management in Data Intensive Domains, DAMDID/RCDL 2018 (Moscow, Russia, October, 9–12, 2018), Revised Selected Papers. Communications in Computer and Information Science. 2019. Vol. 1003. P. 149–164. DOI: 10.1007/978-3-030-23584-0_9
11. Mattson T. S08 - Introduction to OpenMP // Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing (Tampa, FL, USA, November, 11–17, 2006). 2006. P. 209. DOI: 10.1145/1188455.1188673.
12. McGovern A., Rosendahl D.H., Brown R.A., Droegemeier K. Identifying Predictive Multidimensional Time Series Motifs: an Application to Severe Weather Prediction // Data Min. Knowl. Discov. 2011. Vol. 22, no. 1–2. P. 232–258. DOI: 10.1007/s10618-010-0193-7.
13. Meng J., Yuan J., Hans M., Wu Y. Mining Motifs from Human Motion // Proceedings of the Eurographics 2008 - Short Papers (Crete, Greece, April, 14–18, 2008). Eurographics Association, 2008. P. 71–74. DOI: 10.2312/egs.20081024.

14. Minnen D., Isbell C.L., Essa I.A., Starner T. Discovering Multivariate Motifs using Subsequence Density Estimation and Greedy Mixture Learning // Proceedings of the 22nd AAAI Conference on Artificial Intelligence (Vancouver, British Columbia, Canada, July, 22–26, 2007). AAAI Press, 2007. P. 615–620.

15. Mueen A., Keogh E.J., Zhu Q., Cash S., Westover M.B. Exact Discovery of Time Series Motifs // Proceedings of the SIAM International Conference on Data Mining, SDM 2009 (Sparks, Nevada, USA, April, 30 – May, 2, 2009). SIAM, 2009. P. 473–484. DOI: 10.1137/1.9781611972795.41.

16. Munshi A., Gaster B.R., Mattson T.G., Fung J., Ginsburg D. OpenCL Programming Guide. 1st Edition. Addison-Wesley, 2011. p. 646.

17. Narang A., Bhattacherjee S. Parallel Exact Time Series Motif Discovery // Proceedings of the 16th International Euro-Par Conference (Ischia, Italy, August, 31 – September, 3, 2010). Part II. Lecture Notes in Computer Science. Vol. 6272. Springer, 2010. P. 304–315. DOI: 10.1007/978-3-642-15291-7_28.

18. Owens J. GPU Architecture Overview // Proceedings of the International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '07 (San Diego, California, USA, August, 5–9, 2007). ACM, New York, NY, USA. DOI: 10.1145/1281500.1281643.

19. Padua D.A. POSIX Threads (Pthreads) // Encyclopedia of Parallel Computing. Springer, 2011. P. 1592–1593. DOI: 10.1007/978-0-387-09766-4_447.

20. Patel P., Keogh E.J., Lin J., Lonardi S. Mining Motifs in Massive Time Series Databases // Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002 (Maebashi City, Japan, December, 9–12, 2002). IEEE Computer Society, 2002. P. 370–377. DOI: 10.1109/ICDM.2002.1183925.

21. Pearson K. The Problem of the Random Walk // Nature. 1905. Vol. 72, no. 294. DOI: 10.1038/072342a0.

22. Shieh J., Keogh E.J. *i*SAX: Indexing and Mining Terabyte Sized Time Series // Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Las Vegas, Nevada, USA, August, 24–27, 2008). ACM, 2008. P. 623–631. DOI: 10.1145/1401890.1401966.

23. Tanaka Y., Iwamoto K., Uehara K. Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle // Machine Learning. 2005. Vol. 58, no. 2-3. P. 269–300. DOI: 10.1007/s10994-005-5829-2.

24. Wilson D.R., Martinez T.R. Reduction Techniques for Instance-based Learning Algorithms // Machine Learning. 2000. Vol. 38, no. 3. P. 257–286. DOI: 10.1023/A:1007626913721.

25. Yoon C.E., O'Reilly O., Bergen K.J., Beroza G.C. Earthquake Detection through Computationally Efficient Similarity Search // Science Advances. 2015. Vol. 1, no. 11. P. 1–13. DOI: 10.1126/sciadv.1501057.

26. Zymbler M.L., Kraeva Ya.A. Discovery of Time Series Motifs on Intel Many-Core Systems // Lobachevskii Journal of Mathematics. 2019. Vol. 40, no. 12. P. 2124–2132. DOI: 10.1134/S199508021912014X.

27. Zymbler M., Polyakov A., Kipnis M. Time Series Discord Discovery on Intel Many-Core Systems // 13th International Conference, PCT 2019 (Kaliningrad, Russia, April, 2–4, 2019).

Revised Selected Papers. Communications in Computer and Information Science. Springer, 2019. Vol. 1063. P. 168–182. DOI: 10.1007/978-3-030-28163-2_12.

Цымблер Михаил Леонидович, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

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

DOI: 10.14529/cmse200302

PARALLEL ALGORITHM FOR TIME SERIES MOTIF DISCOVERY ON GRAPHIC PROCESSOR

© 2020 M.L. Zymbler, Ya.A. Kraeva

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

E-mail: mzym@susu.ru, kraevaya@susu.ru

Received: 26.07.2020

A time series motif is a pair of subsequences of the series that are most similar to each other. The problem of motif discovery occurs in a wide range of subject areas: medicine, biology, weather prediction, etc. The paper proposes a novel parallel algorithm for time series motif discovery on GPU for the case when the input data fit in the main memory. The proposed algorithm is based on the MK algorithm, which exploits the Euclidean distance and the triangle inequality to prune clearly unpromised pairs of subsequences without computation of the distance. MK decreases the running time up to several orders of magnitude in comparison to the most serial algorithms. However, the performance of MK decreases significantly when the time series length is greater then hundreds of thousands of elements. We designed matrix data structures that ensure the efficient parallel computations on GPU, and paralleled the calculations through the OpenACC programming technology. The results of experimental evaluation on synthetic and real-world datasets confirmed the high scalability of the developed algorithm.

Keywords: time series, motif discovery, parallel algorithm, NVIDIA GPU, OpenACC.

FOR CITATION

Zymbler M.L., Kraeva Ya.A. Parallel Algorithm for Time Series Motif Discovery on Graphic Processor. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. vol. 9, no. 3. pp. 17–34. (in Russian) DOI: 10.14529/cmse200302.

References

1. Zymbler M.L. A Parallel Discord Discovery Algorithm for Time Series on Many-core Accelerators. Numerical methods and programming. 2019. Vol. 20, no. 3. P. 211–223. DOI: 10.26089/NumMet.v20r320

2. Balasubramanian A., Wang J., Prabhakaran B. Discovering Multidimensional Motifs in Physiological Signals for Personalized Healthcare. J. Sel. Topics Signal Processing. 2016.

Vol. 10, no. 5. P. 832–841. DOI: 10.1109/JSTSP.2016.2543679.

3. Brown A., Yemini E., Grundy L., Jucikas T., Schafer W. A Dictionary of Behavioral Motifs Reveals Clusters of Genes Affecting Caenorhabditis Elegans Locomotion. Proceedings of the National Academy of Sciences of the United States of America. 2012. Vol. 110, no. 2. P. 791–796. DOI: 10.1073/pnas.1211447110.

4. Cheng J., Grossman M., McKercher T. Professional CUDA C Programming. 1st Edition. Wrox, 2014. 528 p.

5. Chiu B.Y., Keogh E.J., Lonardi S. Probabilistic Discovery of Time Series Motifs. Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03 (Washington, D.C., USA, August, 24–27, 2003). ACM, 2003. P. 493–498. DOI: 10.1145/956750.956808.

6. Duran A., Klemm M. The Intel Many Integrated Core Architecture. Proceedings of the 2012 International Conference on High Performance Computing and Simulation, HPCS 2012 (Madrid, Spain, July, 2–6, 2012). 2012. P. 365–366. DOI: 10.1109/HPCSim.2012.6266938.

7. Fang J., Varbanescu A.L., Sips H.J. Sesame: A User-Transparent Optimizing Framework for Many-Core Processors. Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013 (Delft, Netherlands, May, 13–16, 2013). 2013. P. 70–73. DOI: 10.1109/CCGrid.2013.79.

8. Farber R. Parallel Programming with OpenACC. 1st Edition. Morgan Kaufmann, 2016. 326 p.

9. Goldberger A., Amaral L., Glass L., Hausdorff J., Ivanov P., et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation. 2000. Vol. 101, no. 23. P. 215–220. DOI: 10.1161/01.CIR.101.23.e215.

10. Kraeva Ya., Zymbler M. Scalable Algorithm for Subsequence Similarity Search in Very Large Time Series Data on Cluster of Phi KNL. 20th International Conference on Data Analytics and Management in Data Intensive Domains, DAMDID/RCDL 2018 (Moscow, Russia, October, 9–12, 2018), Revised Selected Papers. Communications in Computer and Information Science. 2019. Vol. 1003. P. 149–164. DOI: 10.1007/978-3-030-23584-0_9

11. Mattson T. S08 - Introduction to OpenMP. Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing (Tampa, FL, USA, November, 11–17, 2006). 2006. P. 209. DOI: 10.1145/1188455.1188673.

12. McGovern A., Rosendahl D.H., Brown R.A., Droegemeier K. Identifying Predictive Multi-dimensional Time Series Motifs: an Application to Severe Weather Prediction. Data Min. Knowl. Discov. 2011. Vol. 22, no. 1–2. P. 232–258. DOI: 10.1007/s10618-010-0193-7.

13. Meng J., Yuan J., Hans M., Wu Y. Mining Motifs from Human Motion. Proceedings of the Eurographics 2008 - Short Papers (Crete, Greece, April, 14–18, 2008). Eurographics Association, 2008. P. 71–74. DOI: 10.2312/egs.20081024.

14. Minnen D., Isbell C.L., Essa I.A., Starner T. Discovering Multivariate Motifs using Subsequence Density Estimation and Greedy Mixture Learning. Proceedings of the 22nd AAAI Conference on Artificial Intelligence (Vancouver, British Columbia, Canada, July, 22–26, 2007). AAAI Press, 2007. P. 615–620.

15. Mueen A., Keogh E.J., Zhu Q., Cash S., Westover M.B. Exact Discovery of Time Series Motifs. Proceedings of the SIAM International Conference on Data Mining, SDM

2009 (Sparks, Nevada, USA, April, 30 – May, 2, 2009). SIAM, 2009. P. 473–484. DOI: 10.1137/1.9781611972795.41.

16. Munshi A., Gaster B.R., Mattson T.G., Fung J., Ginsburg D. OpenCL Programming Guide. 1st Edition. Addison-Wesley, 2011. p. 646.

17. Narang A., Bhattacherjee S. Parallel Exact Time Series Motif Discovery. Proceedings of the 16th International Euro-Par Conference, (Ischia, Italy, August, 31 – September, 3, 2010). Part II. Lecture Notes in Computer Science. Vol. 6272. Springer, 2010. P. 304–315. DOI: 10.1007/978-3-642-15291-7_28.

18. Owens J. GPU Architecture Overview. Proceedings of the International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '07 (San Diego, California, USA, August, 5–9, 2007). ACM, New York, NY, USA. DOI: 10.1145/1281500.1281643.

19. Padua D.A. POSIX Threads (Pthreads). Encyclopedia of Parallel Computing. Springer, 2011. P. 1592–1593. DOI: 10.1007/978-0-387-09766-4_447.

20. Patel P., Keogh E.J., Lin J., Lonardi S. Mining Motifs in Massive Time Series Databases. Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002 (Maebashi City, Japan, December, 9–12, 2002). IEEE Computer Society, 2002. P. 370–377. DOI: 10.1109/ICDM.2002.1183925.

21. Pearson K. The Problem of the Random Walk. Nature. 1905. Vol. 72, no. 294. DOI: 10.1038/072342a0.

22. Shieh J., Keogh E.J. *i*SAX: Indexing and Mining Terabyte Sized Time Series. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Las Vegas, Nevada, USA, August, 24–27, 2008). ACM, 2008. P. 623–631. DOI: 10.1145/1401890.1401966.

23. Tanaka Y., Iwamoto K., Uehara K. Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle. Machine Learning. 2005. Vol. 58, no. 2-3. P. 269–300. DOI: 10.1007/s10994-005-5829-2.

24. Wilson D.R., Martinez T.R. Reduction Techniques for Instance-based Learning Algorithms. Machine Learning. 2000. Vol. 38, no. 3. P. 257–286. DOI: 10.1023/A:1007626913721.

25. Yoon C.E., O'Reilly O., Bergen K.J., Beroza G.C. Earthquake Detection through Computationally Efficient Similarity Search. Science Advances. 2015. Vol. 1, no. 11. P. 1–13. DOI: 10.1126/sciadv.1501057.

26. Zymbler M.L., Kraeva Ya.A. Discovery of Time Series Motifs on Intel Many-Core Systems. Lobachevskii Journal of Mathematics. 2019. Vol. 40, no. 12. P. 2124–2132. DOI: 10.1134/S199508021912014X.

27. Zymbler M., Polyakov A., Kipnis M. Time Series Discord Discovery on Intel Many-Core Systems. 13th International Conference, PCT 2019 (Kaliningrad, Russia, April, 2–4, 2019). Revised Selected Papers. Communications in Computer and Information Science. Springer, 2019. Vol. 1063. P. 168–182. DOI: 10.1007/978-3-030-28163-2_12.

УДК 004.031.43 **DOI: 10.14529/cmse200303**

ОБЗОР ТЕХНОЛОГИЙ ОРГАНИЗАЦИИ ТУМАННЫХ ВЫЧИСЛЕНИЙ

© 2020 А.А. Кирсанова[1], Г.И. Радченко[1], А.Н. Черных[1,2]
[1]*Южно-Уральский государственный университет*
(454080 Челябинск, пр. им. В.И. Ленина, д. 76),
[2]*Научно-исследовательский центр Энсенады*
(22860 Энсенада, Мексика, Carretera Ensenada - Tijuana No. 3918)
E-mail: alexander.a.kirsanov@susu.ru, gleb.radchenko@susu.ru, chernykh@cicese.mx
Поступила в редакцию: 14.06.2020

Поскольку Интернет вещей (IoT) становится частью нашей повседневной жизни, наблюдается быстрый рост числа подключенных устройств. Устоявшийся подход, основанный на технологиях облачных вычислений, не может обеспечить необходимое качество обслуживания в таких условиях, в частности, в вопросах уменьшения времени задержки при передаче данных. Технология туманных вычислений сегодня рассматриваются как многообещающее решение для обработки большого объема критически важных и чувствительных ко времени данных. В этой статье рассмотрена технология облачных вычислений, а также анализируются предпосылки к эволюционному развитию этого подхода и появлению концепции туманных вычислений. В рамках обзора ключевых особенностей туманных вычислений произведен разбор часто встречающейся путаницы с объединением понятий туманных и краевых вычислений. Приведен обзор технологий организации туманных вычислений: виртуализация, контейнеризация и оркестрация, а также систематический анализ наиболее популярных платформ, обеспечивающих поддержку туманных вычислений. В результате анализа нами предлагается два подхода к классификации платформ туманных вычислений: по принципу открытости/закрытости компонентов, а также трехуровневая классификация на основе предоставляемого функционала платформы (Deploy-, Platform- и Ecosystem as a Service).

Ключевые слова: облачные вычисления, туманные вычисления, краевые вычисления, интернет вещей.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Кирсанова А.А., Радченко Г.И., Черных А.Н. Обзор технологий организации туманных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 3. С. 35–63. DOI: 10.14529/cmse200303.

Введение

В нынешнюю эпоху данные являются основным товаром, а наличие большего количества данных и возможность их эффективного интеллектуального анализа создает большу́ю ценность для предприятий, управляемых на основе данных [26]. По данным Международной корпорации данных (IDC), объем генерируемых цифровых данных в 2010 году превысил 1 зеттабайт [60]. Кроме того, с 2012 года ежедневно генерируется 2,5 эксабайта новых данных [49]. По оценкам Cisco, к 2020 году будет около 50 миллиардов подключенных устройств [16]. Эти подключенные устройства составляют *Интернет вещей* (англ. *Internet of Things — IoT*) и генерируют огромное количество данных в реальном времени. Современные архитектуры мобильных сетей уже сейчас планируются с учетом тех нагрузок, которые возникают при передаче и обработке таких астрономических объемов данных.

В текущих реализациях облачных приложений большая часть данных, требующих хранения, анализа и принятия решений, отправляется в центры обработки данных в облаке [59]. По мере увеличения объема данных, перемещение информации между IoT-устройством и облаком может быть неэффективным или даже невозможным в некоторых случаях из-за ограничений пропускной способности или требований к латентности

вычислительной сети. По мере появления приложений, чувствительных ко времени отклика (таких как мониторинг пациентов, автомобили с автоматическим управлением и другое), удаленное облако не сможет удовлетворить потребность этих приложений в обеспечении сверхнадежной связи с минимальной задержкой [81]. Более того, в некоторых приложениях отправка данных в облако может оказаться невозможной из-за проблем конфиденциальности.

Для решения проблем приложений, требующих высокой пропускной способности вычислительной сети, возможности работы с географически рассредоточенными источниками данных, сверхнизкими задержками и обеспечением локальности обработки данных существует типичная потребность в вычислительной парадигме, которая обеспечивала бы универсальный подход к организации вычислений как в облаке, так и на базе вычислительных узлов ближе к подключенным устройствам. Концепция *туманных вычислений* (англ. *Fog computing*) была предложена индустрией и научным сообществом для устранения разрыва между облаком и устройствами IoT путем предоставления вычислительных возможностей, хранения, организации сетевого взаимодействия и управления данными на узлах сети, расположенных в непосредственной близости от устройств IoT [4, 54]. Исследовательское сообщество предложило ряд подобных вычислительных парадигм для решения упомянутых проблем, таких как *краевые вычисления* (англ. *Edge computing*), *мглистые вычисления* (англ. *Mist computing*), *росистые вычисления* (англ. *Dew computing*) и другие. В этом обзоре мы рассматриваем туманные вычисления и технологии их обеспечения и утверждаем, что туманные вычисления являются более общей формой вычислений, главным образом, из-за их всеобъемлющей области определения и гибкости. Также, мы представляем анализ наиболее популярных платформ, обеспечивающих поддержку туманных вычислений, в части базовых технологий их организации и предоставляемых сервисов. На основе проведенного анализа мы предлагаем два подхода к классификации платформ туманных вычислений: по принципу открытости/закрытости компонентов, а также трехуровневую классификацию на основе предоставляемого функционала платформы (Deploy-, Platform- и Ecosystem as a Service).

Статья организована следующим образом. В разделе 1 рассмотрены облачные вычисления как основа новых вычислительных концепций, предпосылки появления облачных вычислений, их ключевые характеристики и предпосылки к появлению новых вычислительных концепций. Раздел 2 посвящен туманным и краевым вычислениям, их истории появления, а также определению и их ключевым характеристикам. В разделе 3 рассматриваются технологии, обеспечивающие поддержку туманных вычислений: виртуализация и оркестрация. В разделе 4 представлен обзор платформ туманных вычислений: приватных, публичных, с открытым исходным кодом, а также сделано предложение о классификации туманных платформ. В заключении приводится краткая сводка результатов, полученных в рамках данного исследования, и указаны направления дальнейших исследований.

1. Облачные вычисления как основа новых вычислительных концепций

1.1. Предпосылки появления облачных вычислений

Наиболее ранним предком облачных технологий принято считать зародившуюся в 1960-е годы концепцию *коммунальных вычислений* (англ. *utility computing*) [18, 75]. Подход коммунальных вычислений подразумевал предоставление вычислительных ресурсов

по тем же принципам, по которым предоставляются такие коммунальные услуги как водо-, электро- и газоснабжение. Идея была призвана обеспечить пользователей «вычислениями по требованию» для получения максимальной эффективности при гибком ценообразовании [46]. В пакет предоставляемых услуг включались различные типы ресурсов: собственно вычислительные ресурсы, ресурсы хранения данных и другие вычислительные сервисы. Данные идеи описали Джон МакКарти [75] и Дуглас Пархил [66], который описал практически все основные характеристики существующих сегодня облаков, а также впервые употребив сравнение с электрической сетью.

Данная концепция не нашла широкого распространения вплоть до 90-х годов XX века из-за технических сложностей, возникающих при развертывании и использовании данной архитектуры [3, 6, 27, 44, 46, 75]. Основной причиной аналитики называют недостатки организации сетей того времени, не обеспечивающих достаточной пропускной способности для реализации коммунальных вычислений [30].

В середине 90-х годов, совершенствование сетевых технологий и увеличение скорости передачи данных, привело к новому витку исследований коммунальных вычислений, но уже в рамках понятия *грид вычислений* (англ. *Grid computing* от англ. grid — сеть) [20, 27, 44] — по аналогии с electric power grid — электрической сетью. Идея коммунальных вычислений в рамках грид вычислений претерпела преображение [24, 61] и под грид-вычислениями стали понимать объединение ресурсов отдельных высокопроизводительных вычислительных систем в единую сеть.

В [72] указываются следующие ключевые характеристики грид-вычислений.

– *Децентрализованная координация ресурсов*. Грид-система должна объединять и координировать ресурсы и пользователей в разных частях сети и решать вопросы политики безопасности, оплаты и членства.
– *Стандарты и протоколы* с открытым исходным кодом должны использоваться для аутентификации, авторизации, обнаружения ресурсов и доступа к ресурсам грид-системы.
– *Качество предоставления услуг*: ресурсы должны использоваться скоординированным образом для обеспечения необходимого качества обслуживания, времени отклика, пропускной способности и доступности для удовлетворения сложных требований пользователей.

В качестве недостатков такого подхода, можно выделить то, что принципиальная децентрализованность грид-систем, отсутствие единого сетевого адресного пространства и единого администратора базовой вычислительной инфраструктуры усложняет обеспечение отказоустойчивости и динамическое управление ресурсами, доступными конечным пользователям [29, 41, 61]. Данные недостатки обусловили дальнейшее эволюционное развитие и появление облачных вычислений, которые зачастую используют модель грид-вычислений для расширения вычислительных ресурсов [41].

1.2. Ключевые характеристики облачных вычислений

Развитие облачного рынка таким, каким мы его знаем сегодня, стало возможным с появлением Amazon Web Services (AWS) в 2002 году [53]. В 2006 году Amazon запустил Elastic Compute Cloud (EC2) в качестве коммерческого веб-сервиса, который позволял небольшим компаниям и отдельным лицам арендовать часть ИТ-инфраструктуры, на которой можно запускать любые приложения [14]. Именно данным шагом Amazon обеспечил дальнейшую популяризацию облачных вычислений. На сегодняшний день компания Amazon предоставляет полный пакет облачных инфраструктурных услуг, включая

хранение, вычисление и даже возможности человеческого интеллекта через Amazon Mechanical Turk.

В 2011 году Национальный институт стандартов и технологий США (The National Institute of Standards and Technology, NIST) опубликовал определение облачных вычислений, их основные характеристики, а также модели их развертывания и обслуживания. Так, *облачные вычисления* (англ. *Cloud computing*) определены как модель для обеспечения повсеместного, удобного сетевого доступа по требованию к общему пулу настраиваемых вычислительных ресурсов (например, сетей, серверов, хранилищ, приложений и услуг), которые могут быть быстро предоставлены и освобождены с минимальными усилиями по управлению или взаимодействию с поставщиком услуг [50]. В рамках данного документа отмечается, что ключевыми характеристиками облачных вычислений являются:

- **самообслуживание по требованию** — потребитель самостоятельно определяет свои вычислительные потребности: серверное время, скорости доступа и обработки данных, объем хранимых данных, без необходимости прямого взаимодействия с представителем поставщика услуг. Так, в статьях [40, 52, 79] приведены примеры решений, обеспечивающих повышение быстродействия в системах облачных вычислений с обеспечением самообслуживания по требованию;
- **универсальный доступ по сети** — услуги облачных систем доступны потребителям по сети передачи данных вне зависимости от используемого терминального устройства;
- **объединение ресурсов** — поставщик услуг объединяет ресурсы вычислительной системы в единый пул, обеспечивая возможность динамического перераспределения мощностей между большим числом независимых потребителей в условиях постоянного изменения спроса на мощности. При этом, потребители контролируют только основные параметры запрашиваемых услуг (например, объем необходимой памяти, скорость доступа), но фактическое распределение ресурсов, предоставляемых потребителю, осуществляет поставщик. Эффективные механизмы объединения ресурсов позволяют оптимизировать расход ресурсов памяти, хранения и передачи данных в облачных системах от 50% до 65% [78].
- **эластичность** — облачные услуги могут быть предоставлены, расширены, сужены в любой момент времени, без дополнительных издержек на взаимодействие с поставщиком, как правило, в автоматическом режиме [6, 41, 83];
- **учет потребления** — поставщик услуг автоматически исчисляет потребленные ресурсы на определенном уровне абстракции (например, объем хранимых данных, пропускная способность, количество пользователей, количество транзакций) и на основе этих данных оценивает объем предоставленных потребителям услуг. Такой подход, как отмечено в работе [42], позволяет получить эффективные вычислительные мощности с гибкой ценой в зависимости от предъявляемых требований.

С точки зрения потребителя эти характеристики позволяют получить услуги с *высоким уровнем доступности* (англ. *high availability*) и низкими рисками неработоспособности, обеспечить быстрое масштабирование вычислительной системы благодаря эластичности без необходимости создания, обслуживания и модернизации собственной аппаратной инфраструктуры [29, 41, 61].

Период попыток внедрения облачных вычислений для различных целей и требований, привел к выделению следующих моделей развертывания облачных систем: **частное облако**, **публичное облако**, а также **гибридное облако** [15, 76].

Частное облако разворачивается в рамках одной организации, доступно только внутренним пользователям и не предоставляет свои ресурсы пользователям вне этой организации. **Публичное облако** разворачивается сторонними организациями и предоставляет свои ресурсы внешним пользователям на условиях договора на право пользования. **Гибридное облако** является комбинацией из двух выше описанных типов развертывания, что позволяет выстраивать баланс между частными и публичными вычислениями [15].

Частные облака хороши тем, что чаще всего физически разворачиваются как можно ближе к конечному пользователю облака, что снижает время отклика вычислительного узла и повышению быстродействия передачи данных между узлами системы. Однако частное облако настраивается исключительно под вычислительные потребности своего владельца, что одновременно является как плюсом, так и минусом частных облаков. Не каждая организация имеет достаточно ресурсов для содержания собственного частного облака, которое должно удовлетворять как техническим требованиям по доступности и надежности, так и требованиям закона того государства, на территории которого находится как облако, так и сама организация [31, 62].

Пользователи публичных облаков, в свою очередь, часто сталкиваются с проблемой отсутствия прямого контроля над базовой вычислительной инфраструктурой. Это может привести к целому ряду проблем, таких как неконтролируемый доступ третьих лиц к приватным данным, размещенным в публичном облаке; блокировка серверов пользователя, которые могут быть развернуты в одной подсети с узлами, заблокированными в тех или иных государствах; неопределенность в качестве доступных ресурсов, так как они разворачиваются на серверах, используемых совместно с третьими лицами [31]. Также встает вопрос миграции и конвертации данных при необходимости смены провайдера, предоставляющего облако в пользование.

В связи с этими недостатками каждого типа развертывания, часто провайдеры, предоставляющие облака частным компаниям, разворачивают именно гибридные облака [67], которые по требованию могут вести себя, как частные или публичные, что снимает проблемы с задержками передачи данными, безопасностью и вопросами миграции, а также эластичной настройки вычислительных ресурсов под каждую требуемую задачу.

1.3. Предпосылки к появлению новых вычислительных концепций

Несмотря на все существенные достоинства, гарантируемые публичными облачными платформами, в последние 5 лет активно стали появляться задачи, которые не могут быть эффективно решены этими подходами [21]. Так, большое число пользователей мобильных приложений, «умных» систем, таких как «умный дом», «умное предприятие», «умный город» и других IoT-решений, не всегда могут быть удовлетворены качеством услуг, предоставляемых облачными решениями, в частности, из-за увеличения объема пересылаемых данных между пользователем/устройством и облаком [33].

Появление подхода «умных» домов, производств, городов и др., наполненных множеством датчиков, исполнительных механизмов и других систем привело к пересмотру концепции архитектуры систем сбора и анализа данных. Концепция *интернета вещей* требует новых подходов к решениям хранения и быстрой обработки данных, а также возможностей быстрого отклика на изменение состояния конечных устройств [55, 56, 80]. Также, распространение мобильных устройств в качестве основных платформ для клиентских приложений затрудняет передачу и обработку большого количества данных

без возникновения проблем с задержками отклика, в связи с постоянным перемещением мобильных устройств.

С увеличением объема пересылаемых данных между устройствами IoT, клиентами и облаком, появляются проблемы увеличения времени отклика, связанные с ограничениями физической ширины сетевых каналов [46]. С другой стороны, появились чувствительные к времени отклика приложения и устройства, такие как системы жизнеобеспечения, машины-автопилоты, дроны и другие. В этих условиях удаленное централизованное облако стало неспособно удовлетворить требованиям сверхнизких временных задержек [80]. Также передача данных через множество шлюзов и подсетей поднимает вопрос о передаче чувствительных к конфиденциальности данных [36].

В ответ на данные проблемы частные предприятия и академическое сообщество подняли вопрос о необходимости разработки вычислительной парадигмы, удовлетворяющей требованиям новых концепций, таких как IoT [4, 47, 56]. Данная парадигма должна была заполнить разрыв между облаком и конечными устройствами, обеспечив вычисления, хранение и передачу в промежуточных сетевых узлах, наиболее приближенных к конечным устройствам. На данный момент разработано и применяется несколько парадигм, решающих данную проблему, включая туманные (fog) и краевые (edge) вычисления [12]. Каждая из этих парадигм имеет свои особенности, но все они сводятся к общему принципу — уменьшение временных задержек обработки и передачи данных, за счет переноса вычислительных задач ближе к конечному устройству.

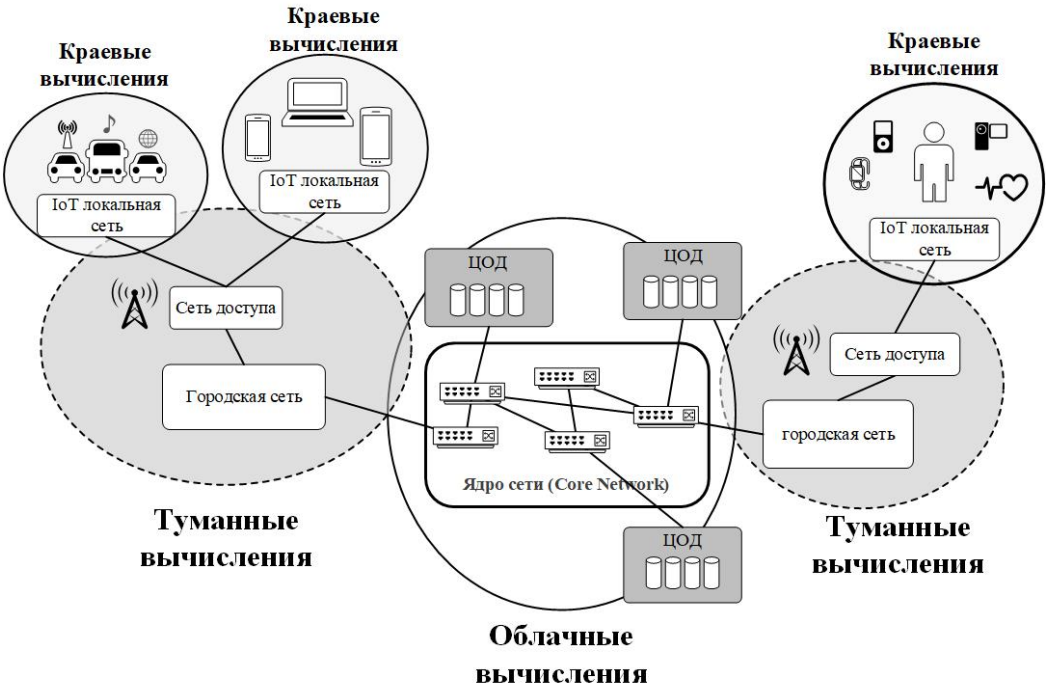


Рис. 1. Схема размещения облачных, туманных и краевых вычислений

На рис. 1 изображена диаграмма относительного распределения вычислительных ресурсов, определяемых концепциями краевых, туманных и облачных вычислений. В соответствии с ней, облачные вычисления представляют собой отдельный центр обработки данных (ЦДО) или же сеть ЦОД, расположенных далеко от пользователя, но обеспечивающих высокие вычислительные возможности. С другой стороны, краевые вычисления расположены непосредственно на краю вычислительной системы и обеспечивают небольшие вычислительные возможности, но в непосредственной близости к потребителю этих ресурсов. Туманные вычисления располагаются между краем сети и об-

лачным ЦОД, обеспечивая существенные вычислительные ресурсы близко к конечному пользователю, которые, с другой стороны, не сопоставимы с общим объемом облачных вычислений, но могут быть настраиваемы и масштабируемы в зависимости от задач конечного пользователя.

В данной статье будут рассматриваться туманные вычисления как отправная точка для появления остальных парадигм [36]. Также будут рассмотрены отличительные особенности краевых вычислений, с которыми часто объединяют туманные.

2. Туманные и краевые вычисления

2.1. История и определение

Первым из вариантов технологии для устранения проблем с временными задержками стала технология *туманных вычислений.* Термин «Туманные вычисления» впервые был предложен компанией CISCO в 2002 году [28] и был описан как «расширение для облачных вычислений, обеспечивающее предоставление вычислительных и сетевых ресурсов, а также ресурсов для хранения информации, расположенных между конечными устройствами и облачными вычислительными центрами» [4]. В 2015 году был основан консорциум OpenFog — группа компаний и академических организаций, таких как Cisco, Dell, Intel и Microsoft Corp, а также Принстонского университета, ориентированная на стандартизацию туманных вычислений (18 декабря 2018 консорциум OpenFog стал частью The Industrial Internet Consortium) [92].

В 2018 году Национальный институт стандартов и технологий США сформулировал официальное определение термина туманные вычисления:

«Туманные вычисления (ТВ) — это многоуровневая модель, обеспечивающая повсеместный доступ к общей совокупности масштабируемых вычислительных ресурсов. Модель ТВ облегчает развертывание распределенных приложений и услуг, учитывающих сетевые задержки, и состоит из *туманных узлов* (англ. *fog nodes*) (физических или виртуальных), располагающихся между умными конечными устройствами и централизованными (облачными) сервисами. *Туманные узлы* являются контекстно-зависимыми и поддерживают единую систему управления данными и организации связи. Они могут быть организованы в кластеры вертикально (для поддержки изоляции), горизонтально (для поддержки федераций сервисов) или в привязке к сетевой близости от конечных умных устройств. Туманные вычисления минимизируют время сетевого отклика поддерживаемых приложений а также обеспечивают конечные устройства локальными вычислительными ресурсами и, при необходимости, сетевым подключением к централизованным сервисам» [36].

Устранение разрыва между облаком и конечными устройствами за счет вычисления, хранения и управления данными не только в облаке, но и на промежуточных узлах [45] расширило область применения туманных вычислений, что позволило применять их в новых задачах: IoT, транспортные средства [34], «умный» город [11], здравоохранение [23], «умная» доставка (в том числе с использование использованием беспилотников) [74], подземная съемка в реальном времени, видеонаблюдение и др. [82]

2.2. Ключевые характеристики туманных вычислений

Из-за позднего отделения понятий туманных и краевых вычислений многие компании вводили собственные характеристики [1] и определения для туманных и краевых вычислений, зачастую объединяя их в одно [45]. В табл. 1 представлены ключевые характеристики, которые выделялись различными авторами для туманных и краевых вычислений.

В 2017 году консорциум OpenFog выпустил эталонную архитектуру туманных вычислений, которая построена на восьми основных принципах: программируемость, иерархия, гибкость, удобство обслуживания (Reliability, availability and serviceability — RAS), доступность, надежность, автономность, открытость и безопасность [54].

Таблица 1

Характеристики туманных вычислений [51]

Кто предложил характеристику	**Характеристика**
Bonomi et al. [68]	Высокая степень виртуализации
	Размещается между конечными устройствами и облаком
	Может быть расположено не только на самом краю сети
Cisco Systems [37]	Расширяет Облако
	Чаще используется в сфере IoT
	Может быть развернуто, где угодно
	Туманные устройства отвечают за обработку, хранение и подключение к сети
Vaquero and Rodero–Merino [86]	Гетерогенное, вездесущее и децентрализованное общение устройств между собой
	Хранение и обработка данных выполняются без использования сторонних устройств и ресурсов
	Выполнение процессов на устройстве в режиме песочницы
	«Аренда» пользовательских устройств для нужд поддержки системы
IBM [3]	Определили Туманные и Краевые вычисления как единую концепцию
	Не зависит от централизованного облака
	Располагается на конечных точках сети
	Размещение некоторых вычислительных ресурсов на краю облака
Обобщение [51]	Может использовать как виртуализацию, так и другие средства развертывания
	Обращение в облако для выполнения нечувствительных к временным задержкам операций и хранения информации
	Любое краевое устройство с достаточными вычислительными мощностями и объемом для хранения данных может выступать Туманным узлом
	Всегда размещается между конечным пользователем и облаком

В [36] выделены следующие ключевые характеристики туманных вычислений.

- **Осведомленность о местоположении и низкая латентность** — туманные узлы осведомлены о логическом расположении друг относительно друга, что позволяет производить расчет временных затрат на связь с другими узлами.
- **Географическая распределенность** — туманные сервисы и приложения способны работать с расположенными в различных географических точках шлюзами, через которые и осуществляется подключение в туман.

- **Поддержка разнородных данных** — поддержка сбора и обработки данных различных форматов, полученных с помощью различных типов сетевых коммуникационных возможностей.
- **Операционная совместимость и федеративность** — компоненты туманных вычислений должны быть способны взаимодействовать друг с другом вне зависимости от различий между собой, а сервисы должны быть распределены по различным доменам для обеспечения доступа.
- **Взаимодействия в реальном времени** — туманные приложения должны иметь возможность работать в режиме реального времени, а не с использованием пакетной обработки запросов.
- **Масштабируемость и динамичность** федеративных, туманно-узловых кластеров — туманные вычисления должны быть адаптивными по своей сути. Должна быть обеспечена поддержка следующих ключевых механизмов адаптации: эластичность вычислений, объединение возможностей ресурсов, подстройка под изменения в нагрузке данных и изменения состояния сети.

2.3. Понятия туманных и краевых вычислений

В некоторых источниках туманные вычисления называют краевыми или граничными вычислениями, основываясь на ключевом описании технологии, что сбор и анализ данных организован не в централизированном облаке, а как можно ближе к конечному устройству, «на краю сети» [4, 21, 33, 36]. Однако в [80] указано, что хотя туманные и краевые вычисления перемещают вычисления и хранение данных на край сети, ближе к конечным узлам, эти парадигмы не идентичны.

В парадигме туманных вычислений туманные узлы размещаются на границе локальной сети, зачастую они разворачиваются на базе роутеров, беспроводных точек доступа (если данные устройства поддерживают требуемые технологии для разворачивания туманного узла) [74]. В отличие от туманных вычислений, краевые вычисления размещаются еще «ближе» к конечным устройствам, уже внутри самой локальной сети на промежуточных точках доступа, а иногда и сами конечные устройства могут выступать краевыми вычислительными узлами: смартфоны, планшеты, другие вычислительные устройства с достаточными вычислительными возможностями и поддержкой развертывания вычислительных узлов [70]. Однако это в то же самое время ограничивает их вычислительные мощности, и поэтому имеются некоторые ограничения в области их применения. На данный момент краевые вычисления применяются для решения таких задач как видеонаблюдение, кеширование видео и контроль трафика [80].

Консорциум OpenFog заявляет, что граничные вычисления часто ошибочно называют туманными вычислениями, и определят, что основным их отличием является то, что туманные вычисления являются иерархическими и обеспечивают вычисления, создание сетей, хранение, управление и ускорение в любом месте — от облака до конечных узлов интернета вещей; в то время как краевые вычисления имеют тенденцию ограничиваться вычислениями на узлах конечных пользователей [48]. Кроме того, в [9] о туманных и краевых вычислениях авторы отмечают, что «туман включает облако, ядро, транспорт, край, клиентов и все остальное» и «туман стремится реализовать непрерывность вычислительных услуг от облака до устройств, а не рассматривать границы сети как изолированные вычислительные платформы».

Таким образом, термин «краевые вычисления» преимущественно используется в телекоммуникационной отрасли и обычно относится к базовым станциям 4G/5G, RAN (Radio Access Network) и ISP (Internet Service Provider) [9, 43]. Тем не менее, данный

термин стал с недавних пор использоваться в предметной области IoT [22, 43, 84] по отношению к локальной сети, где расположены датчики и устройства IoT. Другими словами, «краевые вычисления» расположены в пределах первого от самого IoT устройства транзитного участка сети, например, на точках доступа WiFi или шлюзах.

3. Технологии, обеспечивающие поддержку туманных и краевых вычислений

3.1. Виртуализация

Облака размещаются в ЦОД, где оборудование рассчитано на сверхбольшие нагрузки. Однако не всегда серверное оборудование загружено в полной мере, что не позволяет использовать его эффективно. Ключевой технологией, которая обеспечила поддержку облачных, а затем и туманных вычислений стала технология **виртуализации** [69], которая позволяет использовать ресурсы одной физической машины несколькими логическими виртуальными машинами (ВМ) на уровне слоя аппаратных абстракций (англ. Hardware Abstraction Layer — HAL). Технология виртуализации использует **гипервизор** — программный слой, обеспечивающий работу виртуальных машин на базе аппаратных ресурсов. Машина с гипервизором называется хост-машиной. Виртуальная машина, выполняемая на хост-машине, называется гостевой машиной, на которой в свою очередь могу быть установлены гостевые операционные системы (ОС). Данный вид виртуализации называют **виртуализацией на основе гипервизора**.

Также существует **виртуализация на основе контейнеров** [13], которые представляют собой упакованный, автономный, развертываемый набор прикладных компонентов, которые могут также включать промежуточное программное обеспечение и бизнес-логику в виде бинарных файлов и библиотек для запуска приложений.

В работе [58] представлен сравнительный анализ обоих типов виртуализации, на основе которого можно выделить некоторые преимущества виртуализации на основе контейнеров.

- **Аппаратные ресурсы**. Виртуализация на основе контейнеров снижает затраты на оборудование за счет возможности консолидации. Это позволяет параллельному программному обеспечению воспользоваться преимуществами истинного параллелизма, обеспечиваемого многоядерной аппаратной архитектурой.
- **Масштабируемость**. Система управления контейнерами может эффективно управлять большим количеством контейнеров, позволяя создавать дополнительные контейнеры по мере необходимости.
- **Пространственная изоляция**. Контейнеры поддерживают легкую пространственную изоляцию, предоставляя каждому контейнеру свои собственные ресурсы (например, ядро процессора, память и доступ к сети) и специфические для контейнера пространства имен.
- **Хранение**. Контейнеры имеют малый вес по сравнению с виртуальными машинами. Приложения внутри контейнеров имеют общие двоичные файлы и библиотеки.
- **Производительность**. По сравнению с виртуальными машинами, контейнеры имеют более высокую производительность (сквозную), т.к. они не эмулируют оборудование.
- **Портативность**. Контейнеры поддерживают легкую переносимость из среды разработки в производственную среду, особенно для облачных приложений.

Таким образом, существуют две основные технологии виртуализации, которые применяются в данный момент для обеспечения поддержки туманных вычислений [39].

Причем виртуализация на основе контейнеров получает все большое распространение, благодаря меньшим требованиям к производительности аппаратного обеспечения, чтобы обеспечить развертывание вычислительных узлов на промежуточных устройствах, которые могут не обладать высокими вычислительными мощностями, что особенно актуально для краевых вычислений, т.к. они запускаются даже не на самих IoT устройствах [57], а на промежуточных точках доступа, наиболее близких к устройствам.

3.2. Концепция оркестрации туманных вычислений

Когда контейнеризация стала развиваться как одна из технологий поддержки туманных вычислений, встал вопрос управления вычислительной нагрузкой для обеспечения эффективного использования географически-распределенных ресурсов [38]. Реализация туманных вычислений требует решения проблемы управления вычислительными ресурсами на другом уровне, по сравнению, например, с облачными вычислениями [73].

Первая сложная задача, которая возникает при работе с туманными вычислениями в отличие от облачных, — это управление распределением вычислительной нагрузки (**оркестрация**) между узлами тумана [43, 44] посредством размещения на них туманных сервисов, а также оркестрация этих сервисов, т.е. обеспечение эффективной совместной работы вычислительных сервисов для решения задач, возложенных на туманную среду.

В работе [77] сформулировано, что оркестрация в применении к туманным вычислениям обеспечивает централизованный механизм поиска и обнаружения туманных ресурсов, привязку приложений на основе их требований к физическим ресурсам (развертывание и планирование); управление выполнением рабочей нагрузки с контролем качества обслуживания (QoS).

Рассмотрим ключевые задачи, которые должны решаться системой оркестрации туманных вычислений (Fog Orchestrator) [5, 73].

- **Планирование заданий**. Система оркестрации должна организовывать эффективную совместную работу туманных узлов для выполнения вычислительных заданий. Брокер вычислительных ресурсов должен учитывать специфику рабочего процесса выполнения каждого задания для оптимизации процессов их выполнения.
- **Вычисление и оптимизация маршрутов**. Система оркестрации должна учитывать распределенный характер туманной среды при построении сетевых маршрутов. Она должна обеспечивать поддержку сквозного соединения узлов в случае отсутствия прямого соединения, адаптацию к динамическим сетевым топологиям, максимизацию пропускной способности сети и производительности приложений, обеспечение устойчивости сети.
- **Обнаружение**. Должно обеспечиваться обнаружение физических и виртуальных туманных устройств, а также ресурсов, связанных с ними.
- **Функциональная совместимость**. Оркестратор туманной вычислительной среды должен поддерживать единый набор протоколов, стандартных интерфейсов и онтологий. Это обеспечит возможность различным узлам и приложениям в системе взаимодействовать друг с другом.
- **Минимизация задержек**. Одна из основных задач оркестратора туманных вычислений — уменьшение задержек передачи и обработки данных. Добиться этого позволяют интеллектуальные механизмы оптимизации потоков данных и планирования ресурсов.

– **Отказоустойчивость**. Должно обеспечиваться бесшовное взаимодействие между всеми участниками туманной среды, вне зависимости от возможных сбоев и проблем как на физическом, так и логическом уровне.
– **Прогнозирование и оптимизация**. За счет сбора, хранения и анализа работы узлов системы, система оркестрации может обеспечивать оптимизацию маршрутов передачи данных и взаимодействия устройств между собой для выполнения требований по задержкам и качеству предоставляемых услуг.
– **Безопасность и приватность**. Из-за усложнения топологии туманной сети существенно меняется постановка задачи обеспечения безопасности и конфиденциальности обрабатываемых данных.
– **Аутентификация и контроль доступа**. Решение вопросов безопасности приводит к введению учетных записей, ролей и прав доступа к каждому отдельному узлу или группе узлов для управления, развертывания и использования.

4. Обзор платформ туманных вычислений

При обзоре существующих платформ для развертывания туманных вычислений были рассмотрены коммерческие платформы, а также платформы с открытым исходным кодом. Сложность анализа коммерческих платформ заключается в отсутствии информации об их архитектуре и применяемых технических решениях, которые составляют коммерческую тайну. Однако анализ коммерческих решений показал, что среди коммерческих туманных платформ встречаются как платформы с полным обеспечением поддержки туманных вычислений (собственно вычисления, аналитика и организация транспортного уровня туманной сети), так и платформы, которые обеспечивают лишь транспортный уровень туманной сети и не обеспечивают управления вычислительными узлами и собственно туманными вычислениями на них. Платформы, обеспечивающие лишь транспортный уровень туманных вычислений, не будут рассматриваться в данной работе.

Можно выделить следующие ключевые характеристики частных и публичных коммерческих туманных платформ (см. табл. 2–3).

1. **Поддерживаемые аппаратные платформы** — платформа может работать с любым устройством, поддерживающим виртуализацию или контейнеризацию, или только с ограниченным списком устройств — через драйвера или фирменные устройства. Smartiply Fog, ThingWorx и Cisco IOx работают только с собственным аппаратным обеспечением.
2. **Базовая технология разработки** — на базе какой исполняемой среды создаются и запускаются приложения.
3. **Открытость коммуникационных протоколов и SDK** — есть ли ограничения на приложения, которые могут использоваться в тумане: требуется ли портирование приложений, или в принципе могут исполняться только приложения, написанные с использованием специальных поставляемых SDK, как, например, в случае с ThingWorx, чьи туманные приложения должны быть написаны с использованием фирменного SDK для запуска в тумане.
4. **Технология развертывания** — какая из технологий развертывания туманных узлов используется, если известно.
5. **Возможности интеграции** — имеется ли возможность интеграции с другими платформами, например, корпоративными решениями или публичными облаками.

6. **Подключение внешних источников данных** — возможность платформы подключаться к сторонним базам и хранилищам данных, физически расположенных вне центрального облака для хранения и обработки данных.
7. **Доступность дополнительных сервисов (Machine Learning, Analytics и т.п.)** — возможность подключения и использования дополнительно поставляемых сервисов, которые предоставляют дополнительный функционал по анализу и работе с данными в тумане.
8. **Поддержка Edge** — возможность подключения и использования краевых устройств и краевых вычислений, и дальнейшего сбора и обработка информации от них.

4.1. Частные туманные платформы

Частные туманные платформы обеспечивают создание частных туманных решений на базе вычислительной инфраструктуры, развернутой непосредственно на ресурсах заказчика.

Платформа Cisco IOx была представлена компанией Cisco в 2014 году [2] как развитие сетевой инфраструктуры в связи с ожидаемым ростом IoT. Основной упор в платформе сделан на уменьшение трудозатрат по портированию приложений на туманные узлы, что достигается за счет использования технологий контейнеризации и базирования собственной ОС на основе Linux системы.

Cisco IOx — это среда приложений, которая объединяет в себе Cisco IOS (мини операционная система, устанавливаемая на всю аппаратуру Cisco) и Linux. Для разработки приложений применяются утилиты Linux с открытым исходным кодом. Используется единый протокол взаимодействия туманных приложений во всей сети, организованной с использованием технологий Cisco IoT. Туманные приложения, которые можно запускать на инфраструктуре IOx, поставляются как Cisco, так и партнерами компании. Разработку приложений можно вести на множестве языков программирования общего назначения.

Для разработки и развертывания приложений используется Docker. Поддерживаются разнообразные типы приложений, включая Docker-контейнеры и виртуальные машины (если такая возможность есть у сетевого оборудования). Также возможно использовать собственную исполняемую среду IOx для написания приложений на высокоуровневых языках программирования (например, Python).

Платформа Nebbiolo Technologies нацелена на корпоративный индустриальный рынок, который поддерживает концепцию Индустрии 4.0 [89]. Компания Nebbiolo Technologies тесно сотрудничает с Toshiba Digital Solutions [93] в поставке готовых вычислительных решений для промышленного и IoT сектора.

Платформа состоит из аппаратного обеспечения fogNode, программного стека fogOS и системного администратора fogSM, развертываемого в облаке или локально [32]. Fog System Manager (fogSM) предоставляет облачную платформу централизованного управления, которая позволяет развертывать и настраивать устройства на периферии.

Ключевой особенностью платформы является *fogOS* [32] — программный стек, обеспечивающий связь, управление данными и развертывание приложений на уровне тумана. Основанная на гипервизоре, fogOS предоставляет набор функций в виртуализированной форме. Поддерживается широкий спектр стандартов подключения устройств, а также позволяет размещать приложения и управлять ими в реальном времени.

Платформа ClearBlade представляет собой стек технологий, обеспечивающий быструю разработку и развертывание корпоративных IoT решений, начиная от краевых

устройств, заканчивая облачными сервисами. Она включает в себя программные компоненты, устанавливаемые на весь стек IoT устройств, а также обеспечивает возможность подключения сторонних систем через предоставляемый API для интеграции с устройствами, внутренними бизнес-приложениями и облачными сервисами. Платформа ClearBlade обеспечивает централизованную консоль управления IoT-приложениями, с возможностью развертывания как локально, так и в облаке. Функции управления платформой делегируются краевым узлам (либо на самих конечных устройствах или шлюзах к ним) при помощи системы туманных и краевых вычислений ClearBlade Edge [35].

Таблица 2

Обзор частных туманных платформ

Характеристика	**ClearBlade**	**Smartiply Fog**	**LoopEdge**	**ThingWorx**	**Nebbiolo Technologies**	**Cisco IOx**
Поддерживаемые аппаратные платформы	Универсально	Собственная аппаратура	Универсально	Собственная аппаратура	Универсально	Собственная аппаратура
Базовая технология разработки	JavaScript	Нет данных	Универсально (Docker)	Java VM	Универсально (Docker)	Docker, Linux, IOx
Открытость коммуникационных протоколов и SDK	+	+	+	–	+	+
Технология развертывания	Linux KVM	Нет данных	Docker	Нет данных	Docker	Linux KVM
Возможности интеграции	Oracle, SAP, Microsoft, Salesforce	–	–	Microsoft Azure IoT Hub	–	Microsoft Azure IoT Hub
Подключение внешних источников данных	+	–	+	+	+	+
Доступность дополнительных сервисов	Нет данных	+	–	+	+	+
Поддержка Edge	+	+	+	+	+	+

Платформой поддерживается бессерверный (англ. *Serverless computing*) подход к разработке сервисов на основе языка JavaScript, которые могут быть настроены на реализацию методов машинного обучения и анализа данных. Платформа обеспечивает механизмы экспорта данных и аналитики, собранной системой, в широко применяемые бизнес-системы, приложения и базы данных за счет интеграции с корпоративными платформенными решениями от Oracle, SAP, Microsoft и Salesforce. ClearBlade также предоставляет собственные панели управления, бизнес-приложения и системы управления базами данных для комплексного наблюдения и управления IoT экосистемой.

ClearBlade использует модель OAuth для контроля доступа, где каждый пользователь и устройство получает токен, который должен быть авторизован для получения доступа в систему или ее узлу. Данные шифруются как на самих устройствах, так и пе-

редачи по сети. Передаваемые данные шифруются с помощью библиотек OpenSSL с TLS-шифрованием.

Платформа Smartiply Fog — это туманная вычислительная платформа, которая делает упор на оптимизацию ресурсов и поддержание работы устройств в системе даже без подключения к облаку. Для онлайн сред платформа обеспечивает более высокую надежность благодаря оптимизации ресурсов и вычислений, которые производятся на базе аппаратуры собственного производства [85]. Платформа обеспечивает взаимодействие между устройствами по принципу точка-точка. Таким образом, система узлов может продолжать автономную работу получения, анализа и хранения данных, вплоть до восстановления связи с внешней сетью [88].

Платформа LoopEdge от Litmus Automation позволяет подключать различные устройства в единую систему, собирать и анализировать данные от них. Также Litmus Automation предоставляет отдельную платформу Loop, позволяющую управлять жизненным циклом любого IoT устройства и экспортировать данные в реальном времени во внутренние аналитические и бизнес-приложениями.

Разработчики платформы подчеркивают, что она способна работать с практически любым устройством, причем как с промышленным, так и доступным бытовому потребителю. Например, платформа поддерживает подключение устройств на базе Arduino и Raspberry Pi. Даже если какое-то устройство не поддерживается, подключить его к платформе достаточно легко, за счет устанавливаемых на само устройство исполняемых пакетов, которые можно расширять и создавать с нуля под конкретное устройство. Все инструкции находятся в открытом доступе.

Данная платформа получила широкое распространение среди известных машиностроительных концернов: Nissan, Renault, Mitsubishi Corporation Techno.

Платформа PTC ThingWorx — это IoT платформа, предлагающая подключение достаточно большого числа устройств (доступны драйвера для 150 видов устройств). Однако из-за того, что подключение устройств осуществляется через драйвера, которые требуют установки, прежде чем устройство можно использовать в тумане, эта платформа не является универсальной и имеет ограничения по используемым устройствам.

Приложения для платформы требуется писать с используемых поставляемых SDK. Дальнейший анализ данных и управление бизнес-процессами идет также через предоставляемые инструменты самой платформы. Для выполнения этих задач платформа имеет обширный раздел для разработчиков с инструкциями и учебными пособиями, а также помощью специалистов от самой компании для установки, настройки и расширения платформы. Также «из коробки» имеется возможность подключения к Microsoft Azure IoT Hub.

4.2. Публичные туманные платформы

Публичные туманные платформы сегодня представляют собой решения крупных игроков на рынке облачных вычислений, ориентированные на решения задач обработки данных от IoT-систем, привязанные к возможностям соответствующей облачной платформы. Ключевые характеристики рассматриваемых публичных туманных платформа приведены в табл. 3.

Платформа Azure IoT предоставляет платформу для туманных и краевых вычислений на основе стека технологий от компании Microsoft. Платформа Azure IoT состоит из нескольких крупных подсистем, таких как IoT Central, IoT Edge, которые в свою очередь основывают свою работу на облачной технологии Microsoft Azure. Подключение устройств от партнеров Microsoft возможно без использования драйверов или

программного кода за счет технологии *IoT Plug and Play*. Такой подход возможен для устройств под управлением любой ОС, включая Linux, Android, Azure Sphere OS, Windows IoT, RTOS и другие.

Создание, установка и управление туманными приложениями осуществляется через портал *Azure IoT Hub*. IoT Hub — это управляемая служба, размещенная в облаке, которая выступает в качестве центрального обработчика сообщений для двунаправленной связи между IoT приложением и устройствами, которыми оно управляет. IoT Hub поддерживает передачу данных как от устройства к облаку, так и от облака к устройству. IoT Hub поддерживает несколько шаблонов обмена сообщениями, таких как телеметрия между устройствами и облаками, загрузка файлов с устройств и технологию «запрос-ответ» для управления устройствами из облака.

Для развертывания вычислений ближе к самим устройствам или же на самих устройствах использует *Azure IoT Edge*, которая позволяет разворачивать приложения с собственной бизнес логикой или уже имеющиеся в каталоге готовые приложения на конечных устройствах при помощи технологии контейнеризации.

Платформа Amazon AWS IoT Greengrass позволяет распространить возможности AWS (Amazon Web Services) на периферийные устройства, что позволяет им локально работать с данными, используя при этом облако для управления, анализа и надежного хранения данных. AWS IoT Greengrass позволяет подключенным устройствам выполнять функции AWS Lambda, запускать контейнеры Docker, формировать прогнозы на основе моделей машинного обучения, синхронизировать данные устройств и безопасно взаимодействовать с другими устройствами даже без подключения к Интернету.

AWS IoT Greengrass позволяет создавать решения IoT, которые подключают различные типы устройств к облаку и друг к другу. AWS IoT Greengrass Core можно использовать на устройствах под управлением Linux (в том числе дистрибутивов Ubuntu и Raspbian), которые поддерживают архитектуры Arm или x86. Сервис AWS IoT Greengrass Core обеспечивает локальное исполнение кода AWS Lambda, передачу сообщений, управление данными и безопасность. Устройства с AWS IoT Greengrass Core выступают в качестве порталов сервиса и могут взаимодействовать с другими устройствами, на которых работает FreeRTOS (Real-time operating system for microcontrollers) или установлен пакет SDK AWS IoT для устройств. Размер таких устройств может быть очень разным: от небольших устройств на базе микроконтроллеров до крупных бытовых приборов. Когда устройство с AWS IoT Greengrass Core теряет связь с облаком, устройства в группе AWS IoT Greengrass могут продолжать взаимодействовать друг с другом по локальной сети.

Платформы Google, Yandex и Mail.ru предоставляют собственные облачные и туманные решения для сбора, хранения, обработки, анализа и визуализации данных. Собранные данные с устройств интегрируются в публичную облачную систему для более глубокой обработки и анализа (включая машинное обучение и искусственный интеллект) за счет высоких вычислительных мощностей облака. Данные платформы поддерживают множество протоколов подключения и взаимодействия через предоставляемое API. Имеется большое количество готовых к использованию сервисов, доступные для установки в каталоге самой платформы, которые можно подключать к собственному туманному решению, комбинируя между собой.

Таблица 3

Обзор публичных туманных платформ

Характеристика	AWS Greengrass	Azure IoT	Google	Yandex	Mail.ru
Поддерживаемые аппаратные платформы	Универсально	Универсально	Универсально	Универсально	Универсально
Базовая технология разработки	Универсально (Docker)	Универсально (Docker)	Универсально	Универсально	Универсально
Открытость коммуникационных протоколов и SDK	+	+	+	+	+
Технология развертывания	Docker	Docker	Docker	Docker	Docker
Возможность интеграции	Amazon Elastic Compute 2	Azure, через API	Сервисы Google и партнеров, через API	Универсально через API	Универсально через API
Подключение внешних источников данных	–	–	+	+	–
Доступность дополнительных сервисов (Machine Learning, Analytics и т.п.)	+	+	+	+	+
Поддержка Edge	+	+	+	+	+

4.3. Туманные платформы с открытым исходным кодом

В ходе анализа существующих решений нами был проведен обзор существующих туманных платформ с открытым исходным кодом. В отличие от коммерческих решений, для open source платформ представлены полные описания архитектур, требований к вычислительным ресурсам, а также используемым технологиям, как на аппаратном, так и программном уровнях (см. табл. 4).

Таблица 4

Обзор туманных платформ с открытым исходным кодом

	Цель	Внедрение
FogFrame2.0	Проверить концептуальную модель	–
FogFlow	Более простая и гибкая оркестрация сервисов	+
FogBus	Преодолеть неоднородность на уровне связи ОС и P2P различных узлов тумана	–

Платформа FogFrame2.0 — туманная платформа с открытым исходным кодом [91], нацеленная на развертывание на одноплатных компьютерах (Raspberry Pi). Была представлена для решения следующих задач [63]:

– определить и реализовать функции туманной инфраструктуры, т.е. создание и поддержание туманного ландшафта (множества туманных узлов и устройств) и управление приложениями [64, 65];
– реализовать эвристические алгоритмы для размещения услуг в тумане, а именно алгоритм первичной подгонки и генетический алгоритм;
– ввести механизмы адаптации к динамическим изменения в тумане и для восстановления после перегрузок и сбоев.

Чтобы оценить поведение FogFrame, применялись различные модели поступления запросов приложений (константы, пирамиды и случайные обходы), а также исследовались процессы размещения вычислительных сервисов. Целью исследования в итоге было наблюдение за поведением платформы и ее реакцией на сбои. Платформа динамически реагирует на события во время выполнения, то есть, когда новые устройства появляются или отключаются, когда устройства испытывают сбои или перегрузки, выполняются необходимые передислокации узлов.

Платформа FogFlow — это туманная платформа с открытым исходным кодом [90]. Основной задачей разработчиков данной платформы было обеспечение простого и гибкого способа разработки, развертывания и оркестрации туманных сервисов [8]. Уникальность их подхода заключается в:

– контекстно-зависимой оркестрации сервисов в то время, как другие сервисы оперируют просто событиями (event) или группами событий (topic), происходящими в сети;
– сервисы и приложения FogFlow разработаны с учетом общего представления всех облачных узлов и пограничных узлов, а не с точки зрения каждого отдельного пограничного узла.

FogFlow предлагает использовать программную модель потоков данных, дополненную декларативными подсказками, основанными на широко используемом стандарте NGSI, что дает разработчикам услуг два преимущества:

1) быстрая и простая разработка приложений туманных вычислений, т.к. подобные подсказки скрывают множество настроек и сложностей развертывания от разработчиков сервисов;
2) высокая степень открытости и функциональная совместимость для обмена информацией и интеграции источников данных.

FogFlow является одним из компонентов большой открытой инфраструктуры IWARE [19], обеспечивающей разработку и внедрение различных smart решений [7, 10, 17]. Эта инфраструктура является одним из современных облачных фреймворков наряду с Amazon Web Services [25]. Для внедрения и использования FogFlow доступна широкая библиотека готовых решений от сообщества разработчиков и подробные инструкции по внедрению [87].

Платформа FogBus (при поддержке Melbourn Clouds Lab) объединяет различные аппаратные инструменты через программные компоненты, которые обеспечивают структурированное взаимодействие и независимое от платформы выполнение приложений [71]. FogBus применяет блокчейн для обеспечения целостности данных при передаче конфиденциальных данных. Независимая от платформы архитектура исполнения приложений и взаимодействия между узлами позволяет преодолеть неоднородность в интегрированной среде.

FogBus поддерживает реализацию различных политик управления ресурсами и планирования для выполнения приложений IoT, составленных с использованием моделей параллельного программирования, таких как SPMD (single program, multiple data).

Для оценки характеристик платформы FogBus используется прототип прикладной системы для анализа данных Sleep Apnea. На этом примере иллюстрируется, как приложение (в области здравоохранения), составленное с использованием модели SPMD может быть реализовано с использованием различных настроек FogBus для обработки IoT-данных в интегрированной вычислительной среде.

Данный фреймворк облегчает развертывание приложений IoT, мониторинг ресурсов и управление ими. Системные сервисы FogBus разработаны на кроссплатформенных языках программирования (PHP и Java) и используются с расширяемым протоколом прикладного уровня (HTTP), который помогает FogBus преодолевать неоднородность на уровне связи ОС и P2P различных узлов тумана. Кроме того, платформа FogBus функционирует как модель «Платформа как услуга» (PaaS) для интегрированной среды Fog Cloud, которая не только помогает разработчикам приложений создавать различные типы приложений IoT, но также поддерживает пользователей для настройки служб, и поставщики услуг для управления ресурсами в соответствии с условиями системы.

4.4. Методы классификации туманных платформ

Для формирования единого подхода к классификации туманных платформ, нами были рассмотрены ключевые туманные платформы и их ключевые характеристики. Так, например, **AWS Greengrass** способна работать без доступа к публичному облаку[1], однако в таком режиме работы возможно лишь хранение локальных данных. Центральное управление устройствами, а также централизованный сбор и обработка данных становится невозможен. Для полноценного функционирования платформы требуется доступ до AWS IoT Core, выступающий центральным сервисом для управления и организации работы тумана и который является публичным облаком.

Azure IoT также может работать в частных сетях[2], но при условии, что внутри частной сети будет шлюз, который должен подключаться к центральному узлу управления и сбора данных, и данный узел также является публичным облаку. Отличие от обычной организации работы тумана с публичным облаком заключается в наличии единой точки выхода во внешнюю сеть, а не множества различных шлюзов, которые общаются с публичным облаком.

Другие публичные туманные платформы имеют те же ограничения в отличие от частных туманных платформ и платформ с открытым исходным кодом, центральный узел управления которых может быть развернут на любом сервере в локальной сети или отсутствовать вовсе (в таком случае задачи управления и оркестрации разделяются между промежуточными узлами тумана, как это сделано, например, у FogFlow2.0).

Поэтому все туманные платформы можно классифицировать **по признаку открытости или закрытости развертывания хаба (Hub)** — сервиса, который отвечает за подключение, мониторинг и управление подключенными в туман устройствами. В том или ином виде хаб имеют практически все коммерческие туманные платформы: LoopEdge и Azure IoT так и называют данный сервис — Hub. Платформы ClearBlade и

[1]https://aws.amazon.com/ru/greengrass/faqs/#Local_Resource_Access

[2]https://azure.microsoft.com/en-us/blog/introducing-iot-hub-device-streams-in-public-preview/

FogHorn имеют сервис с тем же функционалом, но называется он — Device Manager. У AWS Greengrass этот сервис называется AWS IoT Core.

Однако управление устройствами имеют только те системы, которые имеют Edge подсистемы. В остальных случаях туман оперирует не устройствами, а развернутыми вычислительными узлами, например, AWS IoT Greengrass Core[1], которые разворачиваются на устройстве, где это возможно, но о самом устройстве ничего неизвестно, взаимодействие идет с виртуализированным узлом на этом устройстве.

Также в предыдущих разделах рассматривались платформы, которые в некоторых случаях могли быть развернуты только на определенном перечне устройств, поставляемых самой платформой. Другие же туманные платформы не имели данного ограничения и могли работать с оборудованием пользователя (если оно удовлетворяет требованиям развертывания платформы). Это показатель классификации туманных платформ **по признаку открытости или закрытости аппаратной инфраструктуры**.

Тот же принцип наблюдается и при сравнении платформ **по признаку открытости или закрытости программной инфраструктуры**: платформа может поддерживать открытые протоколы обмена данными между узлами тумана или же туманные программы поставляются исключительно разработчиками самой платформы и лицензированными партнерами.

Таким образом, любую туманную платформу можно классифицировать по принципу открытости или закрытости ее компонентов (см. рис. 2). Также стоит отметить, что платформы с публичным хабом в большей степени стремятся к открытости своих аппаратных и программных инфраструктур.

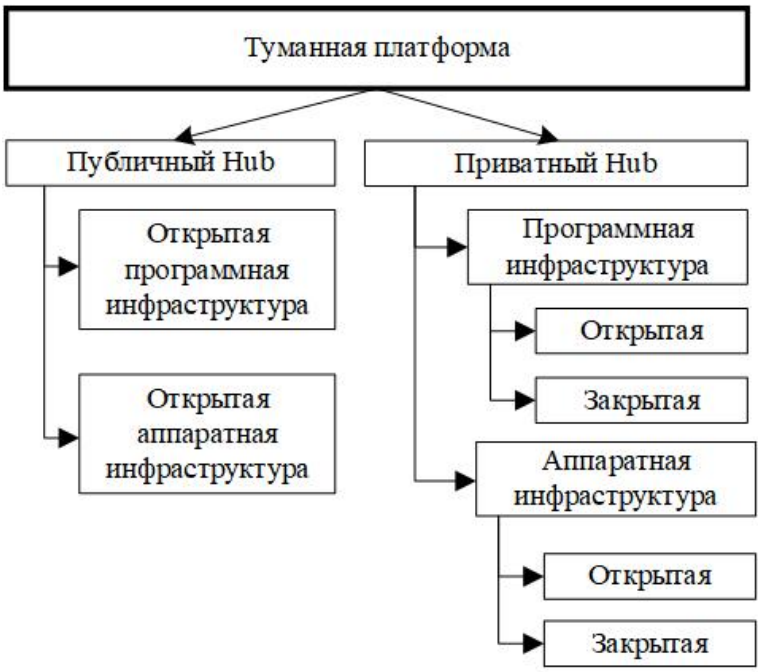


Рис. 2. Классификация туманных платформ по принципу открытости или закрытости ее компонентов

[1]AWS IoT Core — это центральная точка управления туманом. AWS IoT Greengrass Core — это виртуализируемая система, разворачиваемая на устройстве, которое должно выступать вычислительным узлом.

Помимо открытости или закрытости своих компонентов некоторые платформы делали упор на доступность различных предоставляемых возможностей или сервисов, которыми обладает платформа. **Azure IoT Hub**, который является неотъемлемой частью платформы Azure IoT, явно называет себя **PaaS** (Platform as a Service), предоставляя готовые решения для реализации требуемых задач пользователя. Стоит отметить, что ни одна из публичных туманных платформ не позиционирует свои платформы как чисто туманные. Они предоставляют туманные вычисления как некий базовый функционал, который лежит в основе остальных предоставляемых функций и сервисов платформы.

Таким образом, сами платформы позиционируют некоторый функционал как базовый, который должен быть в любой туманной платформе, а пользователь заинтересован уже не только в простом развертывании и базовом управлении туманными узлами, а в решении своих конкретных задач: Индустрия 4.0, медицина, умный город и т.п. Платформы должны максимально предоставить готовые решения для каждой из задач пользователя.

Помимо прочего, некоторые платформы позволили пользователям обмениваться собственными готовыми решениями, созданными в рамках платформы при помощи «магазинов» — ресурсов, где пользователь может опубликовать свое готовое туманное приложение. Это привело к появлению целых туманных экосистем — **EaaS** (Ecosystem as a Service), которые позволяют создать собственное туманное решение из готовых компонентов, доступных на платформе.

Под данное описание также попадают и Open Source решения, которые предоставляют лишь базовый уровень функционала — **DaaS** (Deploy as a Service): разворачивание туманных узлов на имеющихся устройствах, оркестрацию и т.п. С другой стороны, FogFlow имеет более широкий функционал и даже собственную экосистему, в которую включены готовые к установке компоненты как от разработчиков платформы[1], так и от сообщества.

Классификацию «as a service» можно привести как способ классификации на основе предоставляемого функционала платформы (см. рис. 3).

Заключение

Увеличение передаваемых объемов данных и повышенная нагрузка на облако для работы клиентских сервисов стали предпосылкой к появлению концепции туманных вычислений. В данной работе была рассмотрена концепция туманных вычислений, их определение и ключевые характеристики. Также были рассмотрены, классифицированы и обобщены некоторые туманные платформы, которые являются предметами исследования или уже используются бизнесом и частными клиентами. В конце были описаны общие архитектурные характеристики присущие всем рассмотренным платформам.

Туманные вычисления являются более гибким и эффективным видом вычислений по сравнению с облачными за счет решения задач, требующих высокой пропускной способности вычислительной сети, возможности работы с географически рассредоточенными источниками данных, сверхнизкими задержками и обеспечением локальности обработки данных.

[1]https://fogflow.readthedocs.io/en/latest/setup.html

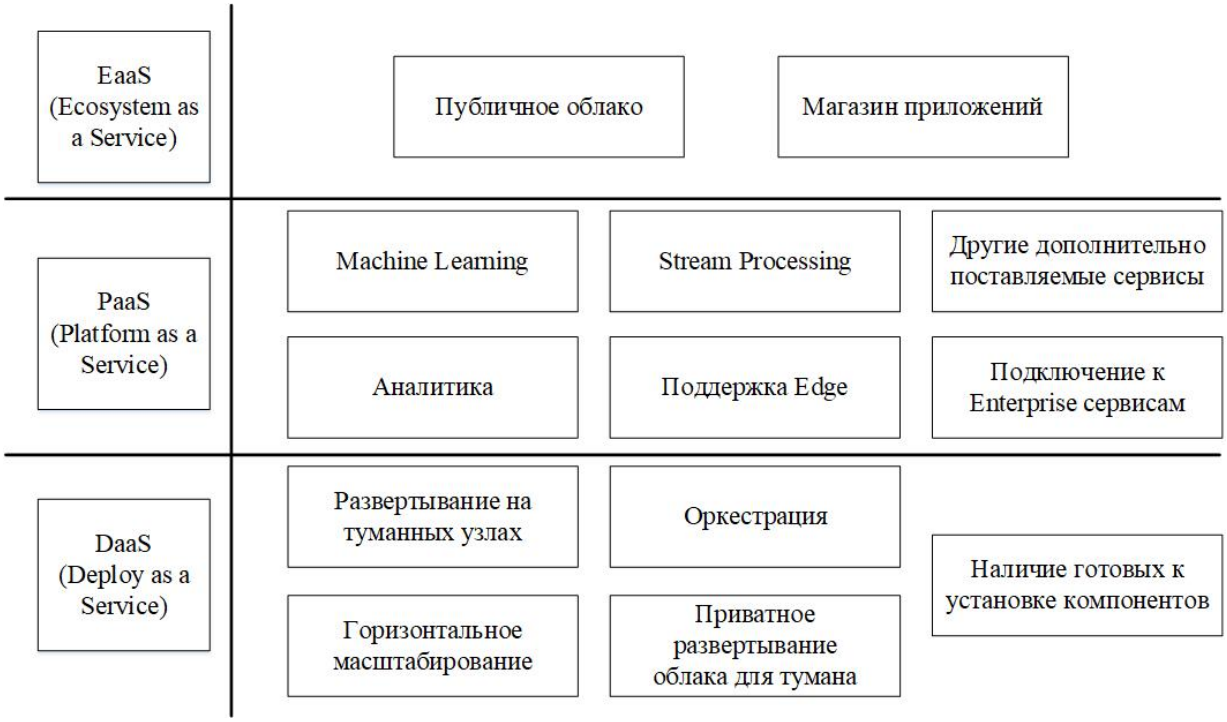


Рис. 3. Классификация туманных платформ на основе предоставляемого функционала платформы

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта No 18-07-01224 а и при финансовой поддержке Министерства науки и высшего образования РФ (государственное задание FENU-2020-0022).

Литература

1. Al-Doghman F., Chaczko Z., Ajayan A.R., et al. A review on Fog Computing Technology // Conference Proceedings of 2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 (Budapest, Hungary, October, 9–12, 2017). Institute of Electrical and Electronics Engineers Inc., 2017. P. 1525–1530. DOI: 10.1109/SMC.2016.7844455.
2. Antonio S. Cisco Delivers Vision of Fog Computing to Accelerate Value from Billions of Connected Devices 2014. P. 1–4.
3. Armbrust M., Fox A., Griffith R., et al. A View of Cloud Computing // Communications of the ACM. 2010. Vol. 53, no. 4. P. 50–58. DOI: 10.1145/1721654.1721672.
4. Bonomi F., Milito R., Zhu J., et al. Fog Computing and Its Role in the Internet of Things // Proceedings of the 1st ACM Mobile Cloud Computing Workshop, MCC'12 (Helsinki, Finland, August, 17, 2012). ACM Press, 2012. P. 13–15. DOI: 10.1145/2342509.2342513.
5. Brito M.S.D., Hoque S., Magedanz T., et al. A Service Orchestration Architecture for Fog-enabled Infrastructures // 2017 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017 (Valencia, Spain, May, 8–11, 2017). Institute of Electrical and Electronics Engineers Inc., 2017. P. 127–132. DOI: 10.1109/FMEC.2017.7946419.
6. Brynjolfsson E., Hofmann P., Jordan J. Cloud Computing and Electricity: Beyond the Utility Model // Communications of the ACM. 2010. Vol. 53, no. 5. P. 32–34. DOI: 10.1145/1735223.1735234.

7. Celesti A., Fazio M., Márquez F.G., et al. How to Develop IoT Cloud E-Health Systems Based on Fiware: A Lesson Learnt // Journal of Sensor and Actuator Networks. 2019. Vol. 8, no. 1. DOI: 10.3390/jsan8010007.
8. Cheng B., Solmaz G., Cirillo F., et al. FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities // IEEE Internet of Things Journal. 2018. Vol. 5, no. 2. P. 696–707. DOI: 10.1109/JIOT.2017.2747214.
9. Chiang M., Ha S., Chih-Lin I., et al. Clarifying Fog Computing and Networking: 10 Questions and Answers // IEEE Communications Magazine. 2017. Vol. 5, no. 4. P. 18–20. DOI: 10.1109/MCOM.2017.7901470.
10. Dantas L., Cavalcante E., Batista T. A Development Environment for FIWARE-Based Internet of Things Applications // Proceedings of the 2019 Workshop on Middleware and Applications for the Internet of Things, Part of Middleware 2019 Conference, M4IoT 2019 (Davis CA, USA, December, 7–11, 2019). Association for Computing Machinery, Inc, 2019. P. 21–26. DOI: 10.1145/3366610.3368100.
11. Dar B.K., Shah M.A., Islam S.U., et al. Delay-Aware Accident Detection and Response System Using Fog Computing // IEEE Access. 2019. Vol. 7. P. 70975–70985. DOI: 10.1109/ACCESS.2019.2910862.
12. Donno M. De, Tange K., Dragoni N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog // IEEE Access. 2019. Vol. 7. P. 150936–150948. DOI: 10.1109/ACCESS.2019.2947652.
13. Eder M. Hypervisor-vs. Container-Based Virtualization // Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM). 2016. Vol. 1.
14. Emeras J., Varrette S., Bouvry P. Amazon Elastic Compute Cloud (EC2) vs. In-house HPC Platform: A Cost Analysis // IEEE International Conference on Cloud Computing, CLOUD (Honolulu, USA, June, 25–30, 2017). IEEE, 2017. P. 284–293. DOI: 10.1109/CLOUD.2016.44.
15. Eugene G. Cloud Computing Models. 2013.
16. Evans D. The Internet of Things: How the Next Evolution of the Internet is Changing Every Thing // CISCO white paper. 2011. Vol. 1. P. 1–11.
17. Fazio M., Celesti A., Marquez F.G., et al. Exploiting the FIWARE Cloud Platform to Develop a Remote Patient Monitoring System // Proceedings of IEEE Symposium on Computers and Communications, 2016. Institute of Electrical and Electronics Engineers Inc., 2016. P. 264–270. DOI: 10.1109/ISCC.2015.7405526.
18. Feeney G.J. Utility computing — A Superior Alternative? // AFIPS Conference Proceedings — 1974 National Computer Conference, AFIPS 1974 (Chicago, Illinois, USA, May, 6–10, 1974). ACM Press, 1974. P. 1003–1004. DOI: 10.1145/1500175.1500370.
19. FIWARE "About us". 2015. URL: https://www.fiware.org/about-us/ (дата обращения: 03.03.2020).
20. Foster I., Kesselman C. The History of the Grid // Advances in Parallel Computing. 2011. Vol. 20. P. 3–30. DOI: 10.3233/978-1-60750-803-8-3.
21. Garcia J., Simo E., Masip-Bruin X., et al. Do We Really Need Cloud? Estimating the Fog Computing Capacities in the City of Barcelona // Proceedings of the 11th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018 (Zurich, Switzerland, December, 17–20, 2019). 2019. P. 290–295. DOI: 10.1109/UCC-Companion.2018.00070.
22. GE Digital What is Edge Computing? | GE Digital. 2018. URL: https://www.ge.com/digital/blog/what-edge-computing (дата обращения: 03.03.2020).

23. Gu L., Zeng D., Guo S., et al. Cost Efficient Resource Management in Fog Computing Supported Medical Cyber-Physical System // IEEE Transactions on Emerging Topics in Computing. 2017. Vol. 5, no. 1. P. 108–119. DOI: 10.1109/TETC.2015.2508382.
24. Guharoy R., Sur S., Rakshit S., et al. A Theoretical and Detail Approach on Grid Computing. A Review on Grid Computing Applications // 2017 8th Industrial Automation and Electromechanical Engineering Conference, IEMECON 2017 (Bangkok, Thailand, August, 16–18, 2017). Institute of Electrical and Electronics Engineers Inc., 2017. P. 142–146. DOI: 10.1109/IEMECON.2017.8079578.
25. Guth J., Breitenbucher U., Falkenthal M., et al. Comparison of IoT Platform Architectures: A Field Study Based on a Reference Architecture // 2016 Cloudification of the Internet of Things, CIoT 2016, 2017. Institute of Electrical and Electronics Engineers Inc., 2017. DOI: 10.1109/CIOT.2016.7872918.
26. Hagiu A., Wright J. When Data Creates Competitive Advantage...And When It Doesn't // Harvard Business Review. 2020. Vol. 98, no. 1. P. 94–101.
27. Hannabuss S. The Big Switch: Rewiring the World, from Edison to Google // Library Review. 2009. Vol. 58, no. 2. P. 136–137.
28. Haouari F., Faraj R., Alja'Am J.M. Fog Computing Potentials, Applications, and Challenges // 2018 International Conference on Computer and Applications, ICCA 2018 (Beirut, Lebanon, July, 25–26, 2018). IEEE, 2018. P. 399–406. DOI: 10.1109/COMAPP.2018.8460182.
29. Hashemi S.M., Bardsiri A.K. Cloud Computing vs. Grid Computing // ARPN Journal of Systems and Software. 2012. Vol. 2, no. 5.
30. Hilbrich M., Frank M. Abstract Fog in the Bottle - Trends of Computing in History and Future // Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018 (Prague, Czech Republic, August, 29–31, 2018). Institute of Electrical and Electronics Engineers Inc., 2018. P. 519–522. DOI: 10.1109/SEAA.2018.00089.
31. Hofmann P., Woods D. Cloud Computing: The Limits of Public Clouds for Business Applications // IEEE Internet Computing. 2010. Vol. 14, no. 6. P. 90–93. DOI: 10.1109/MIC.2010.136.
32. Hong C.H., Varghese B. Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms // ACM Computing Surveys. 2019. Vol. 52, no. 5. P. 1–37. DOI: 10.1145/3326066.
33. Hong H.J. From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices // Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom (Hong Kong, Hong Kong, December, 11–14, 2017). IEEE Computer Society, 2017. P. 331–334. DOI: 10.1109/CloudCom.2017.53.
34. Huang C., Lu R., Choo K.K.R. Vehicular Fog Computing: Architecture, Use Case, and Security and Forensic Challenges // IEEE Communications Magazine. 2017. Vol. 55, no. 11. P. 105–111. DOI: 10.1109/MCOM.2017.1700322.
35. Hughes I., Immerman D., Daly P. ClearBlade Demonstrates Scalability and Edge Analytics With IoT Platform, 2017.
36. Iorga M., Feldman L., Barton R., et al. Fog Computing Conceptual Model. Gaithersburg, MD, 2018.
37. Jalali F., Hinton K., Ayre R., et al. Fog Computing May Help to Save Energy in Cloud Computing // IEEE Journal on Selected Areas in Communications. 2016. Vol. 34, no. 5. P. 1728–1739. DOI: 10.1109/JSAC.2016.2545559.

38. Jiang Y., Huang Z., Tsang D.H.K.K. Challenges and Solutions in Fog Computing Orchestration // IEEE Network. 2018. Vol. 32, no. 3. P. 122–129. DOI: 10.1109/MNET.2017.1700271.
39. Kakakhel S.R.U., Mukkala L., Westerlund T., et al. Virtualization at the Network Edge: A Technology Perspective // 2018 3rd International Conference on Fog and Mobile Edge Computing, FMEC 2018 (Barcelona, Spain, April, 23–26, 2018). Institute of Electrical and Electronics Engineers Inc., 2018. P. 87–92. DOI: 10.1109/FMEC.2018.8364049.
40. Kokkinou A., Cranage D.A. Using Self-Service Technology to Reduce Customer Waiting Times // International Journal of Hospitality Management. 2013. Vol. 33, no. 1. P. 435–445. DOI: 10.1016/j.ijhm.2012.11.003.
41. Kumar R., Charu S. Comparison Between Cloud Computing, Grid Computing, Cluster Computing and Virtualization // International Journal of Modern Computer Science and Applications. 2015. Vol. 8, no. 31. P. 2321–2632. DOI: 10.13140/2.1.1759.7765.
42. Lee J. A View of Cloud Computing // International Journal of Networked and Distributed Computing. 2013. Vol. 1, no. 1. P. 2–8. DOI: 10.2991/ijndc.2013.1.1.2.
43. Li C., Xue Y., Wang J., et al. Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management // ACM Computing Surveys. 2018. Vol. 51, no. 2. DOI: 10.1145/3154815.
44. Liu L., Wang Y., Yang Y., et al. Utility-Based Computing Model for Grid // Proceedings of the 1st International Conference on Semantics, Knowledge and Grid, SKG 2005 (Beijing, China, November, 27–29, 2005). 2005. P. 109–109. DOI: 10.1109/SKG.2005.140.
45. Liu Y., Fieldsend J.E., Min G. A Framework of Fog Computing: Architecture, Challenges, and Optimization // IEEE Access. 2017. Vol. 5. P. 25445–25454. DOI: 10.1109/ACCESS.2017.2766923.
46. Madsen H., Albeanu G., Burtschy B., et al. Reliability in the Utility Computing Era: Towards Reliable Fog Computing // International Conference on Systems, Signals, and Image Processing (Rio de Janeiro, Brazil, June, 3–5, 2013). IEEE Computer Society, 2013. P. 43–46. DOI: 10.1109/IWSSIP.2013.6623445.
47. Mahmood Z., Ramachandran M. Fog Computing: Concepts, Principles and Related Paradigms // Springer International Publishing, 2018. P. 3–21. DOI: 10.1007/978-3-319-94890-4_1.
48. Mahmoudi C., Mourlin F., Battou A. Formal Definition of Edge Computing: An Emphasis on Mobile Cloud and IoT Composition // 2018 3rd International Conference on Fog and Mobile Edge Computing, FMEC 2018 (Barcelona, Spain, April, 23–26, 2018). Institute of Electrical and Electronics Engineers Inc., 2018. P. 34–42. DOI: 10.1109/FMEC.2018.8364042.
49. McAfee A., Brynjolfsson E. Big Data: The Management Revolution // Harvard Business Review. 2012. Vol. 90, no. 10. P. 4.
50. Mell P., Grance T. The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology. 2012. P. 97–101.
51. Naha R.K., Garg S., Georgakopoulos D., et al. Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions // IEEE Access. 2018. Vol. 6. P. 47980–48009. DOI: 10.1109/ACCESS.2018.2866491.
52. Nakagawa M., Hasegawa H., Sato K., et al. Adaptive Self-Reconfigurable Network to Create Cost-Effective Bandwidth-on-Demand Services // Optics InfoBase Conference Papers (Rochester, NY, United States, October, 24–28, 2010). 2010. DOI: 10.1364/nfoec.2010.nwa2.

53. Narula S., Jain A. Prachi Cloud Computing Security: Amazon Web Service // International Conference on Advanced Computing and Communication Technologies, ACCT (Rohtak, Haryana, India, February, 21–22, 2015). Institute of Electrical and Electronics Engineers Inc., 2015. P. 501–505. DOI: 10.1109/ACCT.2015.20.
54. OpenFog Consortium Architecture Working Group OpenFog Reference Architecture for Fog Computing. 2017.
55. Pinchuk A., Sokolov N., Freinkman V. General principles of foggy computing // Last-Mile. 2018. no. 3. P. 38–45. DOI: 10.22184/2070-8963.2018.72.3.38.45.
56. Proferansov D.Y., Safonova I.E. To the Question of Fog Computing and the Internet of Things // Educational Resources and Technology. 2017. Vol. 4, no. 21. P. 30–39.
57. Puliafito C., Mingozzi E., Vallati C., et al. Virtualization and Migration at the Network Edge: An Overview // Proceedings of 2018 IEEE International Conference on Smart Computing, SMARTCOMP 2018 (Taormina, Sicily, Italy, June, 18–20, 2018). Institute of Electrical and Electronics Engineers Inc., 2018. P. 368–374. DOI: 10.1109/SMARTCOMP.2018.00031.
58. Radchenko G.I., Alaasam A.B.A., Tchernykh A.N. Comparative Analysis of Virtualization Methods in Big Data Processing // Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 1. P. 48–79. DOI: 10.14529/jsfi190107.
59. Ravandi B., Papapanagiotou I. A Self-Learning Scheduling in Cloud Software Defined Block Storage // IEEE International Conference on Cloud Computing, CLOUD (Honolulu, Hawaii, USA, June, 25–July, 1, 2017). IEEE Computer Society, 2017. P. 415–422. DOI: 10.1109/CLOUD.2017.60.
60. Reinsel D., Gantz J. Extracting Value from Chaos // IDC Report. 2011. Vol. 1142. P. 1–12.
61. Sadashiv N., Kumar S.M.D. Cluster, Grid and Cloud Computing: A Detailed Comparison // Final Program and Proceedings of the 6th International Conference on Computer Science and Education, ICCSE 2011 (Chennai, India, December, 14–15, 2011). IEEE, 2011. P. 477–482. DOI: 10.1109/ICCSE.2011.6028683.
62. Sehgal N.K., Bhatt P.C.P., Sehgal N.K., et al. Features of Private and Public Clouds Cham: Springer International Publishing, 2018. P. 51–60.
63. Skarlat O., Karagiannis V., Rausch T., et al. A Framework for Optimization, Service Placement, and Runtime Operation in the Fog // Proceedings of the 11th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2018 (Zurich, Switzerland, December, 17–20, 2019). Institute of Electrical and Electronics Engineers Inc., 2019. P. 164–173. DOI: 10.1109/UCC.2018.00025.
64. Skarlat O., Nardelli M., Schulte S., et al. Optimized IoT Service Placement in the Fog // Service Oriented Computing and Applications. 2017. Vol. 11, no. 4. P. 427–443. DOI: 10.1007/s11761-017-0219-8.
65. Skarlat O., Schulte S., Borkowski M., et al. Resource Provisioning for IoT Services in the Fog // Proceedings of 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications, SOCA 2016 (Macau, China, November, 4–6, 2016). Institute of Electrical and Electronics Engineers Inc., 2016. P. 32–39. DOI: 10.1109/SOCA.2016.10.
66. Smirnov Y. Cloud computing. The History and Impact on Libraries' Future // Scientific and Technical Libraries. 2016. no. 6. P. 62–73. DOI: DOI: 10.33186/1027-3689-2016-6-62-73.

67. Sotomayor B., Montero R.S., Llorente I.M., et al. Virtual Infrastructure Management in Private and Hybrid Clouds // IEEE Internet Computing. 2009. Vol. 13, no. 5. P. 14–22. DOI: 10.1109/MIC.2009.119.
68. Taleb T., Samdanis K., Mada B., et al. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration // IEEE Communications Surveys and Tutorials. 2017. Vol. 19, no. 3. P. 1657–1681. DOI: 10.1109/COMST.2017.2705720.
69. Tseng F.H., Tsai M.S., Tseng C.W., et al. A Lightweight Autoscaling Mechanism for Fog Computing in Industrial Applications // IEEE Transactions on Industrial Informatics. 2018. Vol. 14, no. 10. P. 4529–4537. DOI: 10.1109/TII.2018.2799230.
70. Tuli S., Basumatary N., Buyya R. EdgeLens: Deep Learning Based Object Detection in Integrated IoT, Fog and Cloud Computing Environments // Proceedings of the 4th IEEE International Conference on Information Systems and Computer Networks, ISCON 2019 (Mathura, India, November 21–22, 2019). IEEE Press, USA, 2019. P. 496–502. DOI: 10.1109/ISCON47742.2019.9036216.
71. Tuli S., Mahmud R., Tuli S., et al. FogBus: A Blockchain-Based Lightweight Framework for Edge and Fog Computing // Journal of Systems and Software. 2019. Vol. 154. P. 22–36. DOI: 10.1016/j.jss.2019.04.050.
72. Vandenberg A. Grid Computing for All // Charles River Media, 2005. P. 3. DOI: 10.1145/1167350.1167353.
73. Velasquez K., Abreu D.P., Assis M.R.M., et al. Fog Orchestration for the Internet of Everything: State-of-the-Art and Research Challenges // Journal of Internet Services and Applications. 2018. Vol. 9, no. 1. DOI: 10.1186/s13174-018-0086-3.
74. Wadhwa H., Aron R. Fog Computing with the Integration of Internet of Things: Architecture, Applications and Future Directions // Proceedings of the 16th IEEE International Symposium on Parallel and Distributed Processing with Applications, 17th IEEE International Conference on Ubiquitous Computing and Communications, 8th IEEE International Conference on Big Data and Cloud Computing, (Melbourne, Australia, December, 11–13, 2019). IEEE, 2019. P. 987–994. DOI: 10.1109/BDCloud.2018.00144.
75. Webb K. Reviews. Architects of the Information Society: 35 Years of the Laboratory for Computer Science at MIT // Internet Research. 2000. Vol. 10, no. 1. P. 169–174.
76. Weinhardt C., Anandasivam A., Blau B., et al. Cloud Computing — A Classification, Business Models, and Research Directions // Business & Information Systems Engineering. 2009. Vol. 1, no. 5. P. 391–399. DOI: 10.1007/s12599-009-0071-2.
77. Wen Z., Yang R., Garraghan P., et al. Fog Orchestration for Internet of Things Services // IEEE Internet Computing. 2017. Vol. 21, no. 2. P. 16–24. DOI: 10.1109/MIC.2017.36.
78. Wood T., Ramakrishnan K.K., Shenoy P., et al. CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines // IEEE/ACM Transactions on Networking. 2015. Vol. 23, no. 5. P. 1568–1583. DOI: 10.1109/TNET.2014.2343945.
79. Yang J., Pang J., Qi N., et al. On-Demand Self-Adaptivity of Service Availability for Cloud Multi-Tier Applications // Proceedings of 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015, 2015. Institute of Electrical and Electronics Engineers Inc., 2015. P. 1237–1240. DOI: 10.1109/CCGrid.2015.146.
80. Yousefpour A., Fung C., Nguyen T., et al. All One Needs to Know about Fog Computing and Related Edge Computing Paradigms: A Complete Survey // Journal of Systems Architecture. 2019. Vol. 98. P. 289–330. DOI: 10.1016/j.sysarc.2019.02.009.

81. Zhang B., Mor N., Kolb J., et al. The Cloud is Not Enough: Saving IoT from the Cloud // 7th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2015, 2020.
82. Zhang P., Liu J. K., Richard Yu F., et al. A Survey on Access Control in Fog Computing // IEEE Communications Magazine. 2018. Vol. 56, no. 2. P. 144–149. DOI: 10.1109/MCOM.2018.1700333.
83. Zlatanov N. The Data Center Evolution from Mainframe to Cloud // IEEE Computer Society. 2016. DOI: 10.13140/RG.2.1.4103.8489.
84. A Guide to Edge IoT Analytics: Internet of Things Blog. URL: https://www.ibm.com/blogs/internet-of-things/edge-iot-analytics/ (дата обращения: 02.03.2020).
85. Edge Gateway | Smartiply. URL: https://www.smartiply.com/gateway (дата обращения: 02.03.2020).
86. Fog computing brings new business opportunities and disruptions - IoT Agenda. URL: https://internetofthingsagenda.techtarget.com/blog/IoT-Agenda/Fog-computing-brings-new-business-opportunities-and-disruptions (дата обращения: 27.02.2020).
87. FogFlow — FogFlow v2.0 documentation. URL: https://fogflow.readthedocs.io/en/latest/ (дата обращения: 03.03.2020).
88. Mobile Platform | Smartiply. URL: https://www.smartiply.com/mobile (дата обращения: 02.03.2020).
89. Nebbiolo Technologies — Pioneers in Fog Computing. URL: https://www.nebbiolo.tech/ (дата обращения: 02.03.2020).
90. Smartfog/Fogflow: FogFlow is a Standard-Based IoT Fog Computing Framework that Supports Serverless Computing and Edge Computing with Advanced Programming Models. URL: https://github.com/smartfog/fogflow (дата обращения: 03.03.2020).
91. Softls/FogFrame-2.0: FogFrame Framework (with Extensions). URL: https://github.com/softls/FogFrame-2.0 (дата обращения: 02.03.2020).
92. The Industrial Internet Consortium and Openfog Consortium Join Forces | Industrial Internet Consortium. URL: https://www.iiconsortium.org/press-room/01-31-19.htm (дата обращения: 27.02.2020).
93. Toshiba Digital Solutions Corporation and Nebbiolo Technologies Inc. Sign an Industrial IoT Strategic Partnership Agreement. URL: https://www.prnewswire.com/news-releases/toshiba-digital-solutions-corporation-and-nebbiolo-technologies-inc-sign-an-industrial-iot-strategic-partnership-agreement-300632595.html (дата обращения: 02.03.2020).

Кирсанова Александра Александровна, научный сотрудник кафедры электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Радченко Глеб Игоревич, к.ф.-м.н., доцент, директор Высшей школы электроники и компьютерных наук, заведующий кафедрой электронных вычислительных машин, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Черных Андрей Николаевич, к.т.н., доцент, заведующий научно-исследовательской лабораторией проблемно-ориентированных облачных сред, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация), профессор, научно-исследовательский центр Энсенады (Энсенада, Мексика)

DOI: 10.14529/cmse200303

OVERVIEW OF FOG COMPUTING ORGANIZATION TECHNOLOGIES

© 2020 A.A. Kirsanova[1], G.I. Radchenko[1], A.N. Chernykh[1,2]
[1]*South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)*
[2]*Ensenada Research Center*
(Carretera Ensenada - Tijuana No. 3918, Ensenada, 22860 Mexico)
E-mail: alexander.a.kirsanov@susu.ru, gleb.radchenko@susu.ru, chernykh@cicese.mx
Received: 14.06.2020

As the Internet of Things (IoT) becomes a part of our daily life, there is a rapid growth in the number of connected devices. A well-established approach based on cloud computing technologies cannot provide the necessary quality of service in such an environment, particularly in terms of reducing data latency. Today, fog computing technology is seen as a promising solution for processing large amounts of critical and time-sensitive data. This article reviews cloud computing technology and analyzes the prerequisites for the evolution of this approach and the emergence of the concept of fog computing. As part of an overview of the key features of fog computing, we analyze the frequent confusion with the fusion of the concepts of fog and edge computing. The paper provides an overview of fog computing technologies: virtualization, containerization, and orchestration, as well as a systematic analysis of the most popular platforms that support fog computing. As a result of the analysis, we offer two approaches to the classification of fog-computing platforms: on the principle of openness/closure of components, as well as three-level classification based on the provided platform functionality (Deploy-, Platform- and Ecosystem as a Service).

Keywords: cloud computing, fog computing, edge computing, internet of things.

FOR CITATION

Kirsanova A.A., Radchenko G.I., Chernykh A.N. Overview of Fog Computing Organization Technologies. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 3. P. 35–63. (in Russian) DOI: 10.14529/cmse200303.

УДК 004.67 DOI: 10.14529/cmse200304

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ ДЛЯ ИССЛЕДОВАНИЯ ПСИХОЭМОЦИОНАЛЬНОГО СОСТОЯНИЯ СТУДЕНТОВ

© 2020 Е.П. Бобкова[1], С.В. Зыкин[1,2], А.Н. Полуянов[2]
[1]*Омский государственный технический университет*
(644050 Омск, пр. Мира, д. 11),
[2]*Институт математики им. С.Л. Соболева СО РАН*
(630090 Новосибирск, пр. ак. Коптюга, д. 4)
E-mail: lissqa@mail.ru, szykin@mail.ru, andrey.poluyanov@gmail.com
Поступила в редакцию: 30.07.2020

В связи со сложностью объекта исследования анализ данных в медицине является основным инструментом поиска закономерностей и проверки гипотез. Прежде всего, это относится к психологии, в том числе, к анализу поведения субъектов в тех или иных ситуациях. Для выявления высокотревожного состояния студентов, анализа склонности к депрессии или суициду ежегодно в Омском промышленно-экономическом колледже проводится исследование психоэмоционального состояния студентов. Традиционно для этого используются стандарные тесты, основанные на методике «Шкалы тревоги» Спилберегера—Ханина. Целью данной работы является снижение трудоемкости стандартных тестов. Значительные и слабо мотивированные усилия приходится прилагать студентам при заполнении тестов, затем преподавателям при обработке и анализе тестов. Для решения указанной проблемы предлагается сделать тест компактным за счет применения стандартных и оригинальных методов анализа данных с минимизацией потери точности тестирования. Основным новым результатом данной работы является диагностическая шкала, положенная в основу экспресс-оценки психоэмоционального состояния студентов. Расчет диагностической шкалы был выполнен с использованием графических процессоров на суперкомпьютере ИМ СО РАН. Исследования ориентированы на старшие классы общеобразовательных школ и младшие курсы учебных заведений среднего профессионального образования.

Ключевые слова: уровень тревожности, корреляционный анализ, дискриминантный анализ, диагностическая шкала.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Бобкова Е.П., Зыкин С.В., Полуянов А.Н. Применение технологий интеллектуального анализа данных для исследования психоэмоционального состояния студентов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 3. С. 64–76. DOI: 10.14529/cmse200304.

Введение

Методы математического анализа используются для обработки и исследования данных с целью обнаружения зависимостей различной природы в исследуемых объектах. Это позволяет, в том числе, построить дискриминацию объектов и впоследствии найти эффективные способы управления проблемными ситуациями. Значительное применение методов анализа имеет место в психологической науке. Одной из актуальных проблем этого направления [1–5] является выявление психоэмоционального состояния молодежи.

Согласно Ч.Д. Спилбергеру [1], персональное беспокойство подразумевает наличие тенденции восприятия довольно широкого спектра жизненных ситуаций как угрозу и реагирование на каждую из них. Так персональное беспокойство становится активным, когда возникают определенные ситуации, интерпретируемые человеком как опасные для самоуважения. Ситуационное или реактивное беспокойство определено Ю.Л. Ханиным как «состояние, которое характеризуется субъективно переживаемыми эмоциями: напряжением, беспокойством, озабоченностью, нервозностью» [2]. Это является эмоциональной реакцией на ситуацию стресса и может различаться интенсивностью во времени.

Лица, принадлежащие группе высокотревожных, могут видеть опасность в большом разнообразии ситуаций и показывать сильно выраженное состояние беспокойства. Если

психологический тест показывает высокий индекс персонального беспокойства, то это дает право допускать, что он отображает состояние беспокойства в различных ситуациях. Особенно когда ситуации имеют отношение к оценке знания, опыта и общественного статуса личности.

Психологи используют ряд методов, чтобы оценивать персональное беспокойство: «лабиринт», «тест анаграммы», «метод Роршаха» и многое другое. Для определения уровня персонального беспокойства Л.М. Костина рекомендует методику «Шкала тревоги» Спилберегера—Ханина, метод экспертных интервью учителей и родителей студентов [3]. Оценивание тревожности дает возможность сделать рекомендации для изменения поведения субъекта. Например, когда оценка показывает высокий уровень тревожности, то рекомендуется мотивировка снижения субъективного значения ситуации и смещение акцентов на понимание конечных целей и средств их достижения. Наоборот, при низком уровне тревожности необходима дополнительная мотивировка к процессу достижения результата.

В психологии используются методы для установления аналитической зависимости параметров исследуемых объектов [6–8]. Наиболее распространены методы, основанные на параметрической статистике. В основе этих методов используется предположение о нормальном распределении значений параметров, например, t-критерий Стьюдента и F-критерий Фишера. Указанное предположение не всегда является обоснованным. В этих случаях используются непараметрические методы, например, критерий Манна—Уитни и критерий Вилкоксона.

Статья организована следующим образом. В разделе 1 представлены результаты анализа данных, полученные по традиционной методике расчета «Шкалы тревоги» Спилберегера—Ханина. В разделе 2 рассматривается оригинальная методика расчета неравномерных диагностических шкал. Раздел 3 посвящен анализу данных с целью минимизации исходного множества параметров и построению диагностической шкалы на их основе. В заключении приводятся выводы, полученные в результате проведенных исследований.

1. Исходные данные для анализа

В данной работе в качестве обучающей выборки были использованы экспериментальные данные, которые в колледже были классифицированы по «Шкале тревоги» Спилберегера—Ханина. В исследовании приняли участие 149 студентов первого курса. Сокращенный перечень параметров представлен в табл. 1. Всего в базисной таблице присутствует 40 параметров.

Таблица 1

Перечень параметров для эксперимента

№ параметра	Параметр
1	Я спокоен
2	Мне ничто не угрожает
3	Я нахожусь в напряжении
4	Я внутренне скован
5	Я чувствую себя свободно
	...
36	Я бываю доволен
37	Всякие пустяки отвлекают и волнуют меня
38	Бывает, что я чувствую себя неудачником
39	Я уравновешенный человек
40	Меня охватывает беспокойство, когда я думаю о своих делах и заботах

Для поиска новых зависимостей было решено в анализе использовать двенадцать дополнительных параметров, приведенных в табл. 2. Значения параметров 50–52 были получены за счет дополнительного тестирования. Параметрам 41, 43, 45, 48, 49 были поставлены в соответствие целочисленные значения: чем хуже показатель, тем больше значение. Далее на параметры будем ссылаться по их номерам. Каждый параметр исходного теста имеет четыре значения: 1 — «нет, это не так»; 2 — «пожалуй, так»; 3 — «верно»; 4 — «совершенно верно». Наличие всего четырех дискретных значений параметров снижает качество анализа традиционными методами. Непрерывная шкала оценок могла бы дать более качественный результат, в том числе в тесте Спилберегера—Ханина. Есть несколько вариантов электронной версии этого теста в сети Internet [9–11]. Эти приложения позволяют провести тестирование без регистрации на сайтах и обрабатывают данные в онлайн-режиме. У приложений отсутствует способность сбора статистики по группам пользователей.

Таблица 2

Дополнительный перечень параметров

№ параметра	**Параметр**
41	Тип семьи
42	Количество детей в семье
43	Условия проживания в городе Омске
44	Средний балл аттестата при поступлении
45	Место проживания до поступления в колледж
46	Средний балл оценок за первый семестр обучения в колледже
47	Средний балл оценок за второй семестр обучения в колледже
48	Тип общеобразовательного учреждения до поступления
49	Состав группы
50	Склонность к аддиктивному поведению
51	Склонность к делинквентному поведению
52	Склонность к суицидальному поведению

Интерпретация результатов по методике определения тревожности Спилбергера—Ханина позволила разделить испытуемых на 3 группы по уровню выраженности личностной тревожности и на 3 группы по уровню выраженности ситуативной тревожности. Результаты распределения представлены в табл. 3. Группа 1 — испытуемые с низким уровнем тревожности, группа 2 — со средним уровнем тревожности, группа 3 — с высоким уровнем тревожности. Далее в эксперименте использовался результат интегрированной оценки тревожности Спилбергера—Ханина.

Таблица 3

Распределение результатов исследованияпо уровням и видам тревожности

Уровень тревожности	**Ситуативная тревожность**	**Личностная тревожность**
Высокий	1	41
Средний	14	97
Низкий	134	11

2. Метод расчета диагностической шкалы

Методика предварительной обработки параметров описана в экспериментальной части работы, что необходимо для поиска информативных параметров. Целевым этапом эксперимента является построение диагностической шкалы [12, 13] по следущей методике.

Для расчета шкалы применяется решающая функция [14] в виде линейной комбинации N значимых параметров:

$$F(x) = a_1x_1+a_2x_2+...+ a_Nx_N,$$

где $x = (x_1,x_2,...,x_N)$ — вектор значений выделенных параметров (координат в пространстве параметров), $a = (a_1,a_2,...,a_N)$ — веса выделенных параметров (коэффициенты).

Для значений $F(x)$ при расчете определяются границы (оценочная шкала):

$$g_0,\ g_1,\ ...,\ g_K,$$

где K — количество групп объектов O_1, O_2, ..., O_K. При условии, что $g_0 < g_1 < ... < g_K$, для определения принадлежности произвольного объекта o, представленного вектором значений параметров x', к группе O_j выполняется проверка следующего неравенства:

$$g_{j-1} < F(x') < g_j.$$

Введем понятие функционала риска E — суммарного количества ошибок при отнесении объекта к группе. Тогда задача построения оценочной шкалы может быть записана в следующем виде:

$$E \to min,\ g_0 < g_1 < ... < g_K,\ -1 \le a_i \le 1,\ i = 1,2,...,N.$$

Рассмотрим в общем виде схему реализации алгоритма построения шкалы [13]:

1) задаем начальное значение вектора весов $a^0 = (a^0_1,a^0_2,...,a^0_N)$;
2) методом покоординатного спуска с заданным шагом перебираем возможные значения вектора весов $a = (a_1,a_2,...,a_N)$, выполняя на каждом шаге пункты 3–6;
3) рассчет значений решающей функции $F_{ij} = F(x_{ij})$ по всем объектам для текущего набора весов;
4) сортировка F_{ij} по возрастанию;
5) $g_0 = min(F_{ij})-\varepsilon$, $g_K = max(F_{ij})+\varepsilon$, где ε — малая величина;
6) подбор g_1, ..., g_{K-1} с учетом выполнения неравенства $g_0 < g_1 < ... < g_K$ и минимизации функционала риска.

Для ускорения работы алгоритма используется технология CUDA [13], предназначенная для разработки приложений, исполняемых на графических процессорах (GPU). Программная реализация алгоритма выполнена на языке C++ с использованием технологии CUDA. Код приложения разбивается на последовательную и параллельные части. Последовательная часть выполняется на CPU, а параллельная часть, оформленная в виде функции ядра (kernel function), выполняется GPU.

В ходе анализа последовательного варианта алгоритма построения диагностической шкалы выявлено, что максимальное время затрачивается на выполнение шага 6 — 96% от общего времени выполнения программы, следовательно, согласно закону Амдаля, максимально возможный прирост производительности при распараллеливании вычислений на данном шаге — 25-кратный.

Функция ядра для шага 6 алгоритма (выполняемая на GPU) на входе получает массив со значениями F_{ij} для каждого объекта выборки. Каждая нить GPU (аналог потока на CPU) рассчитывает значение функционала риска E для определенного набора границ g_0, ..., g_K (набор зависит от номера нити). Полученные результаты записываются в массив в памяти GPU, массив передается из памяти GPU в память CPU, далее CPU определяет минимальное значение функционала риска для текущего набора весов и выполняется следующий шаг алгоритма. При использовании параллельного алгоритма получено в среднем десятикратное ускорение вычислений по сравнению с последовательным алгоритмом.

3. Результаты анализа данных

Основным показателем будем считать количество ошибок, поскольку он лежит в основе большинства методов анализа данных. Для организации экспресс-тестирования необходимо выделить наиболее значимые параметры в предложенной модели и выявить характер их взаимодействия между собой и с другими параметрами. Оригинальный метод ранжирования параметров применялся Н.Г. Загоруйко [15, 16]. Параметры обучающей выборки предварительно ранжируются по значимости: вычисляется Евклидова мера для всех данных, затем аналогичная мера для данных без одного параметра. В итоге, чем больше убывание меры, тем существеннее влияние параметра, хотя характер этого влияния пока не известен. Результаты расчета в 52-мерном пространстве приведены в табл. 4, в которой указаны параметры с наибольшим убыванием меры.

Таблица 4

Результаты расчетов по Евклидовой мере

Параметр	**Значение меры**
49	8,221
48	8,355
25	8,373
29	8,387
31	8,437
10	8,471
7	8,478
45	8,484
32	8,485

Альтернативой Евклидовой мере при определении значимости параметра является использование методов дискриминантного анализа. В его основе лежат методы решения задач «различения (дискриминации) объектов наблюдения по определенным признакам» [7, 17]. На данном этапе был использован одномерный анализ в предположении, что области значений отдельного параметра могут пересекаться для отдельных групп субъектов. Другими словами, часть ответов на один и тот же вопрос могут совпадать для различных психоэмоциональных состояний студентов. Задачей реализованного нами алгоритма является минимизация этой области. Итоговый список значимых параметров представлен в табл. 5.

Таблица 5

Итоговый список параметров при дискриминации

Параметр	**Суммарное количество ошибок**
29	13
31	16
23	21
32	21
25	22
28	30
37	31
40	31
35	33
38	35
11	36
10	37

На следующем этапе необходимо выяснить наличие и характер зависимости между параметрами. Зависимости могут ухудшить результаты моделирования за счет доминирования таких параметров. Простейшую зависимость, близкую к линейной, можно выяснить за счет вычисления корреляции [18]. В [7] рассматриваются две основные задачи корреляционного анализа: а) определение формы связи в виде математической формулы; б) измерение меры связи между признаками для установления степени влияния данного параметра на результат за счет решения корреляционного уравнения.

В данной работе влияние параметров на результат исследования определяется за счет построения диагностической шкалы, для которой целесообразно отсутствие коррелированных параметров. В табл. 6 представлены коэффициенты корреляции для значимых параметров.

Согласно табл. 6, значимый коэффициент корреляции (0,74) между параметрами 31 и 29. Относительно этих параметров необходимо дополнительное исследование их влияния на шкалу, если они и далее будут находиться в списке значимых параметров.

Заметим, что отсутствие корреляционной зависимости не гарантирует отсутствие иной зависимости между параметрами. Эта зависимость может иметь нелинейный характер. Например, значения параметра x принадлежат множеству {1, 2,..., 11}, а параметр $y = (x - 6)^2$. Между параметрами x и y квадратичная зависимость, однако корреляция между ними равна нулю.

Традиционно для построения многомерной дискриминантной функции используется кусочно-линейная аппроксимация [19, 20]. Нами разработан новый метод построения нелинейной дискриминантной функции. Основная идея метода заключается в построении функции в $(n+1)$-мерном пространстве, где n количество параметров. Функция аппроксимирует положительные значения с весами для первой группы объектов и отрицательные значения с весами для другой группы объектов. Граница дискриминации образуется пересечением аппроксимации с пространством параметров. Для выбора весов используется метод штрафных функций с целью получения минимального количества ошибок дискри-

Таблица 6

Корреляция параметров

Параметры	**10**	**11**	**23**	**25**	**28**	**29**	**31**	**32**	**35**	**37**	**38**	**40**
10	1											
11	0,49	1										
23	–0,34	–0,3	1									
25	–0,29	–0,2	0,65	1								
28	–0,4	–0,4	0,47	0,48	1							
29	–0,29	–0,3	0,66	0,65	0,62	1						
31	–0,28	–0,3	0,66	0,65	0,5	0,74	1					
32	–0,44	–0,5	0,44	0,47	0,45	0,52	0,48	1				
35	–0,34	–0,4	0,46	0,42	0,4	0,39	0,37	0,47	1			
37	–0,19	–0,3	0,52	0,5	0,47	0,6	0,51	0,35	0,47	1		
38	–0,38	–0,5	0,44	0,45	0,46	0,46	0,46	0,57	0,5	0,4	1	
40	–0,22	–0,2	0,32	0,35	0,42	0,51	0,42	0,28	0,33	0,41	0,31	1

минации на обучающей выборке. Пока метод реализован для двух групп объектов, однако он может быть обобщен на большее количество групп.

Значения отобранных и упорядоченных по рангу параметров компонуются в группы сначала по два, затем по три и т.д. Для каждой группы строится своя нелинейная дискриминантная функция, минимизирующая количество ошибок. Выделенные функцией области можно интерпретировать как области саморегулирующихся параметров для некоторой группы объектов и отсутствие такой саморегуляции для другой группы объектов. Такой подход позволяет выявить ранее неизвестные зависимости. Как и в случае с корреляцией, связанные параметры можно удалить из дальнейшего рассмотрения, оставив только более значимые. В случае с анализом на психоэмоциональную устойчивость разделения на области имеют место, но вид этих областей говорит не о саморегуляции, а демонстрирует связанность по смыслу тех или иных групп вопросов исходного теста. На рис. 1 показаны области дискриминации, полученные в результате расчетов, для параметров 28 и 32 для групп средней и высокой тревожности с наименьшим количеством ошибок.

Анализ других парных взаимодействий параметров дает еще большее количество ошибок, что говорит об отсутствии каких-либо зависимостей среди значимых (ранее выделенных) параметров. Расчеты, проведенные для большего количества параметров, не показали какого-либо существенного снижения ошибок.

В данном эксперименте дополнительно была выявлена слабая различимость среднего и низкого уровня тревожности: количество ошибок практически совпадает с количеством студентов с низким уровнем тревожности. То есть все они поглощаются группой со средним уровнем. Указанный результат подтверждается экспериментом с использованием пакета SPSS Statistics [21].

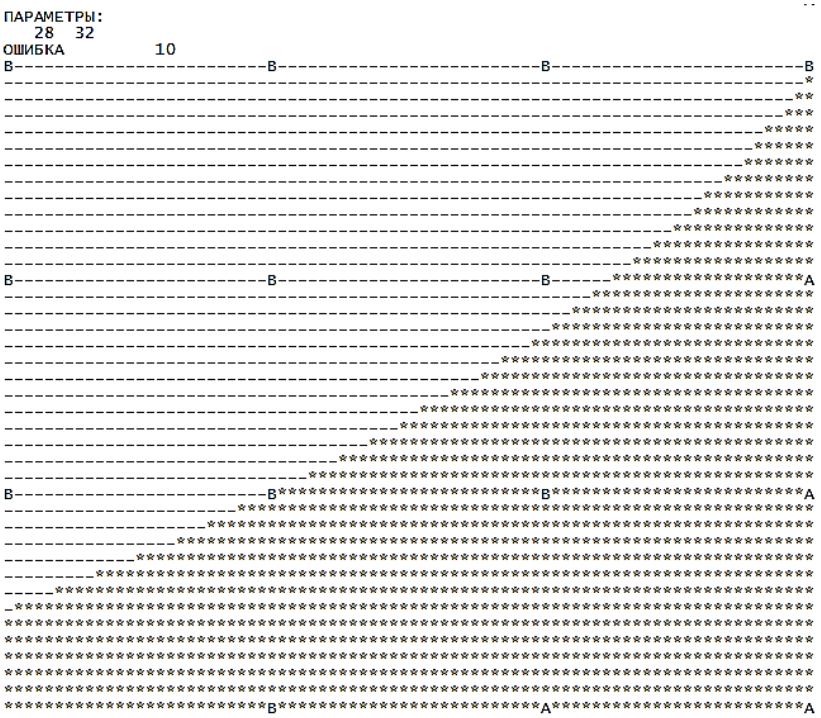


Рис. 1. Области дискриминации параметров 28 и 32

На рис. 2 видно, что центроиды первой и второй группы на графике довольно близко расположены друг к другу. А объекты первой группы (низкая тревожность) перемешаны с частью объектов второй группы (средняя тревожность), что согласуется с предыдущим результатом. Суммарное количество ошибок диагностирования, полученное с использованием пакета SPSS Statistics, равно 14.

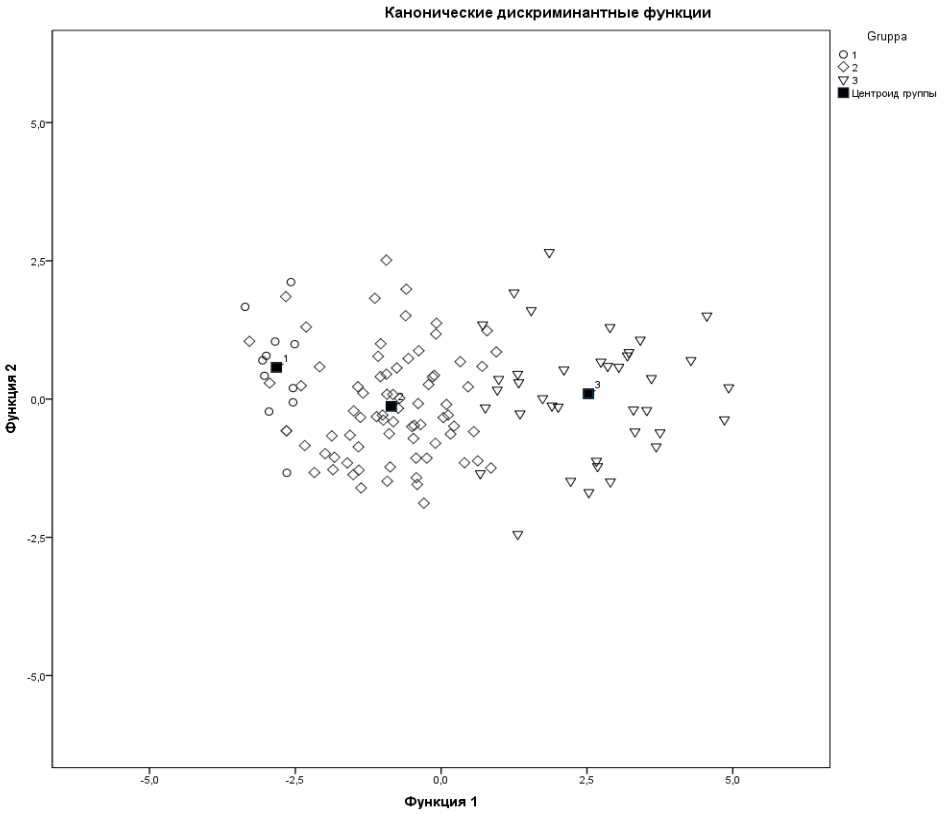


Рис. 2. Результаты анализа с использованием пакета SPSS Statistics

Заключительным этапом эксперимента явилось построение диагностической шкалы [12, 14], адаптированной для расчетов на графических процессорах [13]. Идея и реализация метода описаны в разделе 2 работы. Итоговый список параметров, прошедших предварительный отбор, представлен в табл. 7. В итоговом результате вычислительного эксперимента (см. рис. 3) осталось только 4 значимых параметров для диагностики: 29, 31, 32, 38. При этом достигнут уровень ошибок — 4.

Таблица 7

Сводный список параметров для построения шкалы

№ параметра	Параметр
10	Я испытываю чувство внутреннего удовлетворения
11	Я уверен в себе
23	Я легко расстраиваюсь
25	Я сильно переживаю неприятности и долго не могу о них забыть
28	Меня тревожат возможные трудности
29	Я слишком переживаю из-за пустяков
31	Я все принимаю близко к сердцу
32	Мне не хватает уверенности в себе
37	Всякие пустяки отвлекают и волнуют меня
38	Бывает, что я чувствую себя неудачником
40	Меня охватывает беспокойство, когда я думаю о своих делах и заботах
44	Средний балл аттестата при поступлении
47	Средний балл за 2 семестр в колледже

```
Version 4
<<<<<<<<<check result>>>>>>>>>
----------------------------------------------------------------
ВЕСА ПАРАМЕТРОВ:
  №1  10  0
  №2  11  0
  №3  23  0
  №4  25  0
  №5  28  0
  №6  29  0.1
  №7  31  0.1
  №8  32  0.1
  №9  37  0
  №10  38  0.1
  №11  40  0
  №12  44  0
  №13  47  0

ГРАНИЦЫ ГРУПП:
0.3  (Низкая)  0.4  (Средняя)  0.9  (Высокая)  1.6

КОЛИЧЕСТВО ОШИБОК: 4
----------------------------------------------------------------
FINAL_RESULT. RASCHET OKONCHEN.
Time clocked by CUDA events: millseconds
1.13183999061584
Time clocked by clock() CPU milliseconds
0
```

Рис. 3. Результаты расчета шкалы психоэмоционального состояния

Пусть x_1 — значение параметра 29, x_2 — 31, x_3 — 32, x_4 — 38, подставим значения этих параметров в выражение:

$$y = 0{,}1{*}x_1 + 0{,}1{*}x_2 + 0{,}1{*}x_3 + 0{,}1{*}x_4.$$

Если $0{,}3 \le y \le 0{,}4$, то тестируемый принадлежит к группе с низким уровнем тревожности, $0{,}4 \le y \le 0{,}9$ — средняя тревожность, если $0{,}9 \le y \le 1{,}6$ — высокая тревожность. При попадании на границу интервала необходимо провести полное тестирование по исходной шкале. Ранее было получено, что параметры 29 и 31 имеют значительный уровень корреляции. Расчет шкалы без параметра 29 дал 6 ошибок, без параметра 31 — 5 ошибок, что дает основание не учитывать корреляцию.

Полученные результаты позволяют рекомендовать применение рассмотренного метода для построения диагностических шкал. Окончательный выбор применяемого метода остается за экспертом, аргументом в пользу выбора того или иного метода является количество ошибок на обучающей выборке и количество параметров, используемых для построения шкалы.

Заключение

Диагностическая шкала позволила свести до четырех количество ошибочного диагностирования на обучающих выборках, в то время как в пакете SPSS Statistics количество ошибок на тех же данных — 14 при использовании полного набора параметров. В рассмотренном материале не использовались какие-либо статистические гипотезы, поскольку это не влияет на конечный результат.

Применение рассмотренной «Шкалы экспресс-оценки» психоэмоционального состояния позволило ускорить сбор и обработку исходных данных, поскольку при ответе на четыре вопроса у студентов меньше мотивации отложить тестирование либо исказить его результаты. По предложенной методике был протестирован 501 студент, среди котрых выявлено 15 высокотревожных студентов.

Рассмотренный материал предполагает дальнейшее развитие методики тестирования. Прежде всего, необходимо отказаться от дискретных значений параметров, привлечь дополнительную информацию из других тестов для повышения достоверности ответов, разработать адаптивное тестирование [22], собрать и обработать статистику по прецедентам.

Работа выполнена при поддержке программы фундаментальных научных исследований СО РАН № I.5.1., проект № 0314-2019-0020.

Литература

1. Spielberger, C.D. Assessment of state and trait anxiety: Conceptual and methodological issues // Southern Psychologist. 1985. Vol. 2, no 4. P. 6–16.
2. Ханин Ю.Л. Краткое руководство к применению шкалы реактивной личностной тревожности Ч.Д. Спилбергера. Ленинград: ЛНИИФК, 1976. 18 с.
3. Костина Л.М. Методы диагностики тревожности. СПб.: Изд-во «Речь», 2005. 198 с.
4. Прихожан А.М. Тревожность у детей и подростков: психологическая природа и возрастная динамика. Воронеж: Изд-во НПО «МОДЭК», 2000. 304 с.
5. Карелин А.П. Большая энциклопедия психологических тестов. М.: Изд-во «Проспект», 2015. 420 с.
6. Титкова Л.С. Математические методы в психологии. Владивосток: Изд-во Дальневосточного университета, 2014. 140 с.
7. Сидоренко Е.В. Методы математической обработки в психологии. СПб.: Изд-во «Речь», 2015. 350 с.
8. Герасевич В.А., Аветисов А.Р. Современное программное обеспечение для статистической обработки биомедицинских исследований // Белорусский медицинский журнал. 2015. № 1. С. 12–25.
9. Тест Спилбергера—Ханина. URL: https://psytests.org/psystate/spielberger-run.html (дата обращения: 27.07.2020).
10. Тест исследование тревожности (опросник Спилберга). URL: https://onlinetestpad.com/ru/test/714-test-issledovanie-trevozhnosti-oprosnik-spilberga (дата обращения: 27.07.2020).
11. Тест на уровень тревожности Спилбергера—Ханина. URL: https://psychojournal.ru/tests_online/127-test-na-uroven-trevozhnosti-spilbergera-hanina.html (дата обращения: 27.07.2020).
12. Зыкин С.В., Редреев П.Г., Чернышев А.К. Формирование представлений данных для построения медицинских диагностических шкал // Омский научный вестник. 2011. № 2(100). С. 190–193.
13. Полуянов А.Н. Расчет диагностической шкалы на графических процессорах // Омский научный вестник. 2012. № 3(113). С. 277–279.
14. Лбов Г.С., Неделько В.М., Неделько С.В. Метод адаптивного поиска логической решающей функции // Сибирский журнал индустриальной математики. 2009. Т. XII, № 3(39). С. 66–74.
15. Загоруйко Н.Г. Прикладные методы анализа данных и знаний. Новосибирск: Изд-во Института математики СО РАН, 1999. 270 с.
16. Загоруйко Н.Г., Кутненко О.А., Борисова И.А., Дюбанов В.В., Леванов Д.А., Зырянов О.А. Выбор информативных признаков для диагностики заболеваний по генетическим данным // Вавиловский журнал генетики и селекции. 2014. Т. 18, № 4–2. С. 898–903.
17. Дюк В.А. Компьютерная психодиагностика. СПб.: Изд-во «Братство», 1994. 364 с.
18. Буров А.В. Корреляционный анализ взаимосвязей между индивидуально-психологическими особенностями детей и показателями адаптации к школьному обучению // Научно-методический электронный журнал «Концепт». 2014. № 7. С. 101–105. URL: http://e-koncept.ru/2014/14193.htm (дата обращения: 27.07.2020).
19. Glen J.J. Mathematical Programming Models for Piecewise-Linear Discriminant Analysis // The Journal of the Operational Research Society. 2005. Vol. 56, no. 3. P. 331–341. DOI: 10.1057/palgrave.jors.2601818.

20. Chang D.S., Kuo Y.C. An approach for the two-group discriminant analysis: An application of DEA // Mathematical and Computer Modelling. 2008. Vol. 47, no. 9–10. P. 970–981. DOI: 10.1016/j.mcm.2007.05.010.
21. Дубнов П.Ю. Обработка статистической информации с помощью SPSS. М.: Изд-во «АСТ», 2004. 221 с.
22. Томашев М.В., Авдеев А.С., Краснова М.В. Адаптивное тестирование как средство управления качеством образования // Информатика и образование. 2018. № 9. С. 27–33. DOI: 10.32517/0234-0453-2018-33-9-27-33.

Бобкова Елена Петровна, магистрант кафедры ПМиФИ Омского государственного технического университета, преподаватель Омского промышленно-экономического колледжа (Омск, Российская Федерация)

Зыкин Сергей Владимирович, д.т.н., профессор, заведующий лабораторией МППИ Института математики им. С.Л. Соболева СО РАН (Новосибирск, Российская Федерация), профессор кафедры ПМиФИ Омского государственного технического университета (Омск, Российская Федерация)

Полуянов Андрей Николаевич, к.т.н., старший научный сотрудник лаборатории МППИ Института математики им. С.Л. Соболева СО РАН (Новосибирск, Российская Федерация)

DOI: 10.14529/cmse200304

APPLICATION OF INTELLIGENT DATA ANALYSIS TECHNOLOGIES FOR STUDENTS PSYCHO-EMOTIONAL STATE STUDY

© 2020 E.P. Bobkova[1], S.V. Zykin[1,2], A.N. Poluyanov[2]

[1]*Omsk State Technical University (pr. Mira 11, Omsk, 644050 Russia),*

[2]*Sobolev Institute of Mathematics SB RAS*

(pr. Acad. Koptyug 4, Novosibirsk, 630090 Russia)

E-mail: lissqa@mail.ru, szykin@mail.ru, andrey.poluyanov@gmail.com

Received: 30.07.2020

Due to the complexity of the research object, data analysis in medicine is the main tool for finding patterns and testing hypotheses. First of all, this applies to psychology, including the analysis of the behavior of subjects in certain situations. In order to identify the high-anxiety state of students, to analyze the tendency to depression or suicide, a study of the psycho-emotional state of students is conducted annually at the Omsk Industrial and Economic College. Traditionally, standard tests based on the technique of "Anxiety Scale" of Spielberger–Hanin's are used for this. The purpose of this work is to reduce the complexity of the standard tests. Significant and poorly motivated efforts have to be made by students in completing the tests, and then by teachers in processing and analyzing the tests. To solve this problem, it is proposed to make the test compact by applying standard and original data analysis methods while minimizing the loss of test accuracy. The main result of this work is a diagnostic scale, which forms the basis for the rapid assessment of the psycho-emotional state of students. The diagnostic scale was calculated using graphics processors on a supercomputer of IM SB RAS. Target audience: senior classes of secondary schools and junior courses of educational institutions of secondary vocational education.

Keywords: anxiety level, correlation analysis, discriminant analysis, diagnostic scale.

FOR CITATION

Bobkova E.P., Zykin S.V., Poluyanov A.N. Application of Intelligent Data Analysis Technologies for Students Psycho-emotional State Study. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 3. P. 64–76. (in Russian) DOI: 10.14529/cmse200304.

References

1. Spielberger C.D. Assessment of State and Trait Anxiety: Conceptual and Methodological Issues. Southern Psychologist. 1985. Vol. 2, no 4. P. 6–16.
2. Khanin Yu.L. A Brief Guide to the Use of the Scale of Reactive Personal Anxiety by Ch.D. Spielberger. Leningrad, LNIIFK, 1976. 18 p. (in Russian)
3. Kostina L.M. Methods for Diagnosing Anxiety. St. Petersburg, Rech, 2005. 198 p. (in Russian)
4. Prikhozhan A.M. Anxiety in Children and Adolescents: the Psychological Nature And Age Dynamics. Voronezh, NPO "MODEK", 2000. 304 p. (in Russian)
5. Karelin A.P. Great Encyclopedia of Psychological Tests. Moscow, Prospect, 2015. 420 p. (in Russian)
6. Titkova L.S. Mathematical Methods in Psychology. Vladivostok, Publishing of the Far Eastern State University, 2014. 140 p. (in Russian)
7. Sidorenko Ye.V. Methods of Mathematical Processing in Psychology. St. Petersburg, Rech, 2015. 350 p. (in Russian)
8. Gerasevich V.A., Avetisov A.R. Modern Software for Statistical Processing of Biomedical Research. Belarusian Medical Journal. 2015. no. 1. P. 12–25. (in Russian)
9. Spielberger–Khanin Test. Available at: https://psytests.org/psystate/spielberger-run.html (accessed: 27.07.2020). (in Russian)
10. Test Anxiety Study (Spielberg questionnaire). Available at: https://onlinetestpad.com/ru/test/714-test-issledovanie-trevozhnosti-oprosnik-spilberga (accessed: 27.07.2020). (in Russian)
11. Test on The Level of Anxiety Spielberger–Hanina. Available at: https://psychojournal.ru/tests_online/127-test-na-uroven-trevozhnosti-spilbergera-hanina.html (accessed: 27.07.2020). (in Russian)
12. Zykin S.V., Redreyev P.G., Chernyshev A.K. Formation of Data Representations for the Construction of Medical Diagnostic Scales. Omsk Scientific Bulletin. 2011. no. 2(100). P. 190–193. (in Russian)
13. Poluyanov A.N. Calculation of the Diagnostic Scale on Graphics Processors. Omsk Scientific Bulletin. 2012. no. 3(113). P. 277–279. (in Russian)
14. Lbov G.S., Nedel'ko V.M., Nedel'ko S.V. The Method of Adaptive Search for a Logical Solving Function. Journal of Applied and Industrial Mathematics. 2009. Vol. 12, no. 3(39). P. 66–74. (in Russian)
15. Zagoruyko N.G. Applied Methods of Data and Knowledge Analysis. Novosibirsk, Sobolev Institute of Mathematics SB RAS, 1999. 270 p. (in Russian)
16. Zagoruyko N.G., Kutnenko O.A., Borisova I.A., Dyubanov V.V., Levanov D.A., Zyryanov O.A. The Choice of Informative Features for the Diagnosis of Diseases According to Genetic Data. Russian Journal of Genetics: Applied Research. 2014. Vol. 18, no. 4–2. P. 898–903. (in Russian)
17. Dyuk V.A. Computer Psychodiagnostics. St. Petersburg, Bratstvo, 1994. 364 p. (in Russian)
18. Burov A.V. Correlation Analysis of the Relationship Between the Individual Psychological Characteristics of Children and Indicators of Adaptation to School Education. Scientific-

Methodical Electronic Journal “Koncept”. 2014. no. 7. P. 101–105. Available at: http://e-koncept.ru/2014/14193.htm (accessed: 27.07.2020). (in Russian)

19. Glen J.J. Mathematical Programming Models for Piecewise-Linear Discriminant Analysis. The Journal of the Operational Research Society. 2005. Vol. 56, no. 3. P. 331–341. DOI: 10.1057/palgrave.jors.2601818.
20. Chang D.S., Kuo Y.C. An approach for the two-group discriminant analysis: An application of DEA. Mathematical and Computer Modelling. 2008. Vol. 47, no. 9–10. P. 970–981. DOI: 10.1016/j.mcm.2007.05.010.
21. Dubnov P.Yu. Statistical information processing using SPSS. Moscow, ACT NT Press, 2004. 221 p. (in Russian)
22. Tomashev M.V., Avdeyev A.S., Krasnova M.V. Adaptive Testing as a Tool for Managing Quality of Education. Informatics and Education. 2018. no. 9. P. 27–33. (in Russian) DOI: 10.32517/0234-0453-2018-33-9-27-33.

УДК 338.2 **DOI: 10.14529/cmse200305**

ПРОГНОЗИРОВАНИЕ БАНКРОТСТВ ПРЕДПРИЯТИЙ С ПОМОЩЬЮ ЭКСТРЕМАЛЬНОГО ГРАДИЕНТНОГО БУСТИНГА

© 2020 В.В. Мокеев, Р.В. Войтецкий
Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: mokeyev@mail.ru, romanvoitetckii@gmail.com
Поступила в редакцию: 27.07.2020

Использование моделей прогнозирования банкротства предприятий для управления инвестиционными рисками лежит в основе управленческой деятельности финансовых учреждений. Важным фактором, позволяющим финансовым учреждениям определять объем капитала для покрытия кредитных потерь, является точность прогноза. В большинстве исследований для построения моделей банкротства предприятий используются традиционные методы статистики (например, дискриминантный анализ и логистическая регрессия). Однако точность построенных моделей обычно является достаточно низкой. Это обусловлено несбалансированностью классов обучающих выборок (доля фирм-банкротов составляет несколько процентов от общего числа фирм), которые используются при построении моделей. В настоящее время широкое распространение получают такие методы машинного обучения как метод случайного леса и метод градиентного бустинга. В данном исследовании основной акцент делается на использовании экстремального градиентного бустинга для прогнозирования банкротства. Экстремальный градиентный бустинг, используя LASSO или Ridge регуляризацию, штрафует сложные модели, что помогает избежать переобучения. Также в ходе обучения экстремальный градиентный бустинг заполняет пропущенные значения в наборе данных в зависимости от величины потерь. В статье для повышения эффективности экстремального градиентного бустинга предлагается использовать технологию SMOTE для улучшения сбалансированности классов. Метрики качества решений, полученных улучшенным экстремальным градиентным бустингом, сравниваются с решениями полученными другими методами.

Ключевые слова: экстремальный градиентный бустинг, банкротство, предприятие, synthetic minority oversampling technique.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мокеев В.В., Войтецкий Р.В. Прогнозирование банкротств предприятий с помощью экстремального градиентного бустинга // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 3. С. 77–90. DOI: 10.14529/cmse200305.

Введение

Инвестиционные риски являются основной проблемой для финансовых учреждений, что заставляет их проверять и контролировать финансовую платежеспособность предприятия. Модели банкротств используются финансовыми учреждениями для оценки кредитного убытка, который они могут понести, если их контрагенты не возместят свои долги. Точность прогноза является ключевым фактором, поскольку она позволяет финансовым учреждениям определять объем капитала, необходимый для покрытия кредитных потерь. В настоящее время существует достаточно большое число исследований, направленных на повышение точности моделей банкротств предприятий. Практическая значимость прогноза банкротства связана со значительным интересом со стороны кредиторов в точном прогнозировании будущего существования компаний.

В большинстве исследований прогноз банкротства рассматривается как проблема бинарной классификации. Целевая переменная моделей обычно принимает два значения: 0 (платежеспособное предприятие) и 1 (предприятие банкрот). Традиционно для оценки банкротства используются дискриминантный анализ [1–5] и регрессионный ана-

лиз [5–7]. Альтман использовал линейный дискриминантный анализ (Linear Discriminant Analysis, LDA) для того, чтобы различать предприятия банкротов от не банкротов [1]. LDA применяет линейную комбинацию показателей для определения рейтинга предприятия. Эта оценка затем применяется для разделения предприятий на банкротов и не банкротов. В работе [2] LDA используется для прогнозирования риска банкротства российских предприятий. Применимость LDA и квадратичного дискриминантного анализа исследуется в работах [4, 5]. В работе [5] модели банкротства предприятий строятся с использованием логистической регрессии, но, в отличие от модели Альтмана [1], модель Олсона [5] определяет вероятность банкротства. Несмотря на относительную легкость построения моделей с помощью дискриминантного анализа и логистической регрессии, использование этих моделей показывает плохую способность к обобщению и низкую точность прогнозирования [8]. Поэтому в дальнейшем многие исследователи используют методы машинного обучения, такие как нейронные сети [9–12], метод опорных векторов [13, 14], бустинговые методы [15, 16].

Одной из характерных особенностей задачи прогнозирования банкротств является то, что доля фирм-банкротов составляет несколько процентов от общего числа фирм. Поэтому исследователи сталкиваются с несбалансированными наборами данных, в которых число обанкротившихся компаний явно превосходит число необанкротившихся компаний. В ряде работ предпринимаются попытки найти методы улучшения несбалансированных наборов данных. Большинство из них используют механизмы сбалансирования выборочных распределений. В работе [17] используется несколько методов, чтобы улучшить два сильно несбалансированных набора данных. Эти методы позволяют методам прогнозирования банкротства достигать лучших результатов, чем прогнозы по исходным несбалансированным наборам данных. В работе [18] используют обучающую выборку, которая содержит 620 обанкротившихся образцов и 7398 необанкротившихся фирм. Для повышения точности прогнозирования банкротств предлагается гибридный метод недостаточной выборки, который сочетает в себе метод ближайших соседей и метод опорных векторов. Результаты исследований показывают, что предлагаемый гибридный метод повышает точность классификаторов, таких как логистическая регрессия, дискриминантный анализ, дерево решений и машины опорных векторов. В работе [19] оцениваются четыре метода улучшения сбалансированности наборов данных: случайная избыточная выборка, случайная недостаточная выборка, метод искусственного увеличения меньшинства (SMOTE) и EasyEnsemble. Результаты исследований показывают, что SMOTE превосходит другие методы улучшения сбалансированности наборов данных в плане повышения точности прогнозирования.

Цель данного исследования заключается в исследовании эффективности метода экстремального бустинга в сочетании с методом улучшения сбалансированности обучающей выборки для решения задач прогнозирования банкротств предприятий. Предлагается сравнить полученные результаты с решениями, полученными другими методами.

Статья организована следующим образом. В разделе 1 описывается экстремальный градиентный бустинг, а также технология SMOTE для улучшения сбалансированности наборов данных. В разделе 2 описываются наборы данных, содержащие финансовые показатели предприятий и метку, указывающую на статус банкротства предприятий, а также результаты исследования эффективности моделей прогнозирования банкротств. В заключении приводится краткая сводка результатов, полученных в работе, и указаны направления дальнейших исследований.

1. Теоретическая основа

1.1. Экстремальный градиентный бустинг

Экстремальный градиентный бустинг (Extreme Gradient Boosting, XGB) представляет развитие метода градиентного бустинга [20, 21]. При обучении с учителем набор данных $D = \{(x_i, y_i): x_i \in R^n, y_i \in R\}$, состоит из n объектов с m признаками и n метками. Необходимо построить модель, которая как можно более точно сможет предсказывать метки для каждого нового объекта. Бустинговая модель F использует N аддитивных функций $f_i(x)$ для прогнозирования меток

$$\tilde{y}_i = F(x_i) = \sum_{j=1}^{N} f_j(x_i). \tag{1}$$

Для обучения набора функций мы минимизируем функцию потерь

$$L(F) = \sum_{i=1}^{n} l(y_i, \tilde{y}_i) + \sum_{j=1}^{K} R(f_j), \tag{2}$$

где $R(f_j)$ является коэффициентом регуляризации, который ограничивает сложность модели. Функция потерь $L(F)$ содержит K функций в качестве параметров, поэтому ее очень трудно оптимизировать напрямую. Вместо этого мы оптимизируем модель аддитивно. Пусть $\tilde{y}_i^t$ будет предсказанием i-го образца на t-й итерации. Мы добавим f_t, чтобы минимизировать

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \tilde{y}_i^{(t-1)} + f_t(x_i)) + R(f_t). \tag{3}$$

Это означает, что f_t добавляется так, чтобы максимально улучшить нашу модель для каждой итерации. Мы используем приближение второго порядка, которое использует градиент этой промежуточной функции потерь $L^{(t)}$. По этой причине мы называем его алгоритмом градиентного бустинга.

Основные отличия экстремального градиентного бустинга от градиентного бустинга связаны с регуляризацией и работой с пропущенными данными. Экстремальный градиентный бустинг штрафует сложные модели, применяя Ridge-регуляризацию или регуляризацию LASSO, что позволяет снизить риски переобучения. Также в ходе обучения экстремальный градиентный бустинг использует алгоритм заполнения пропущенных значений в зависимости от величины потерь.

Точность решений получаемых с помощью XGB зависит от гиперпараметров метода. Важнейшими из них являются максимальная глубина дерева, скорость обучения, число деревьев. Увеличение максимальной глубины дерева делает модель более сложной и повышает вероятность переобучения. Более высокая скорость обучения приводит к тому, что каждое дерево вносит более серьезные поправки в решение. Это приводит к усложнению модели и повышает вероятность переобучения. Увеличение числа деревьев также повышает сложность модели, но при этом появляется возможность повысить точность получаемых решений.

1.2. Экстремальный градиентный бустинг и метод искусственного увеличения экземпляров миноритарного класса

Для повышения качества прогнозирования банкротств предлагается использовать при построении моделей методом экстремального градиентного бустинга метод улучшения сбалансированности обучающей выборки. В качестве последнего предлагается использовать метод Synthetic Minority Oversampling Technique (SMOTE), который основан на идее генерации некоторого количества искусственных образцов, «похожих» на имеющиеся в классе банкротов, но не дублирующих их. Для создания нового образца находят разность $d = X_b - X_a$, где X_a, X_b — это векторы признаков «соседних» образцов a и b класса предприятий банкротов. Их находят, используя алгоритм ближайшего соседа. В нашем случае необходимо и достаточно для генерирования нового образца получить набор из k ближайших соседей, из которого будут выбраны объекты a и b.

Далее вектор разности нового образца $\tilde{d}$ получается путем умножения каждого элемента вектора d на случайное число в интервале (0, 1). Вектор признаков нового образца вычисляется путем сложения $X_c = X_a + \tilde{d}$. Метод SMOTE позволяет задавать количество новых образцов, которое необходимо искусственно сгенерировать.

Для обеспечения контроля качества обучения моделей предлагается следующая схема построения моделей. Весь набор данных предварительно делится на обучающий и тестовый набор. Для обучения модели используется процедура кросс-валидации, в рамках которой набор делится на K блоков (folds). Процесс обучения выполняется K раз на разных обучающих выборках, состоящих из $K–1$ блоков. Для улучшения сбалансированности классов обучающих выборок они расширяются путем генерации искусственных образцов, «похожих» на имеющиеся в классе банкротов, но не дублирующих их. Для генерации используется метод SMOTE. Для контроля качества обучения используется оставшийся валидационный блок a. Таким образом, мы получаем K моделей и K валидационных блоков, которые образуют валидационный набор. Валидационный набор используется для оценки качества обучения. Для оценки обобщающей способности модели используется тестовый набор. Поскольку в рамках процедуры кросс-валидации было построено K моделей, мы получаем на тестовом наборе K прогнозов, которые затем усредняются.

Таким образом, метод SMOTE органично встраивается в схему обучения моделей методом экстремального градиентного бустинга и может представлять улучшенную версию экстремального градиентного бустинга.

1.3. Метрика качества

Традиционно для оценки качества моделей классификации используется метрика *accuracy*, которая численно равна доли правильно классифицированных объектов. Однако для несбалансированных классов такая метрика является не очень удобной. Для оценки качества модели на каждом из классов по отдельности используются метрики *precision* (точность) и *recall* (полнота). *Precision* можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а *recall* показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. *Precision* и *recall* не зависят, в отличие от *accuracy*, от соотношения классов и потому применимы в условиях несбалансированных наборов.

Обычно при выборе гипер-параметров метода используется одна метрика, улучшение которой мы и ожидаем увидеть на тестовом наборе. В качестве такой метрики

предлагается использовать метрику *F1 score*, которая представляет среднее гармоническое *precision* и *recall*. Для вычисления метрик precision, *recall*, *F1* и *accuracy* требуется преобразование вероятностного результата в бинарную метку, т.е. необходимо выбрать какой-либо порог, при котором результат становится 0 или 1. Естественным и близким является порог, равный 0,5, но он не всегда оказывается оптимальным, особенно при отсутствии баланса классов. Одним из способов оценки модели в целом, не привязываясь к конкретному порогу, является метрика AUC, которая численно равна площади под кривой ошибок.

2. Экспериментальное исследование

2.1. Набор данных

В данной статье данные были взяты с сайта UCI machine learning repository[1]. Изначально они были извлечены с ресурса Emerging Markets Information Services[2], который представляет собой базу данных, содержащую информацию о развивающихся рынках по всему миру. Обанкротившиеся компании были проанализированы в период 2000–2012 годов, в то время как все еще действующие компании были оценены с 2007 по 2013 год. Всего предоставлено пять наборов данных, которые содержат финансовые показатели предприятий и метку, которая указывает на статус банкротства через 1, 2, 3, 4 года и 5 лет. Метаданные наборов данных представлены в табл. 1.

Таблица 1

Основные характеристики набора данных

Характеристика	Метка статуса банкротств				
	1 год	2 года	3 года	4 года	5 лет
Число предприятий	5910	9792	10503	10173	7027
Доля предприятий банкротов (%)	6,93	5,26	4,71	3,93	3,85
Доля пропущенных значений (%)	1,21	1,37	1,44	1,83	1,27

Таким образом, мы имеем 5 наборов, которые не являются равноценными. Три набора с метками статуса банкротств через 2, 3 и 4 года содержат близкое число предприятий примерно около 10000. Набор данных с метками статуса банкротств через 1 год имеет на 40% меньше предприятий, а набор данных с метками статуса банкротства через 5 лет включает на 30% меньше предприятий, чем наборы данных с метками статуса банкротств через 2, 3 и 4 года. В наборах каждое предприятие описывается 64 финансовыми показателями, которые измеряют рентабельность, активы предприятий и их денежные потоки. Полное описание показателей можно найти в работе [15], в которой исследуются эффективность методов машинного обучения при построении моделей прогнозирования банкротств. Однако исследования выполненные в работе обладают рядом недостатков.

Во-первых, для процедуры кросс-валидации используется весь набор, и качество моделей определяется на валидационном наборе. Такая оценка характеризует только качество обучения модели, особенно при использовании таких методов как метод случайного леса (Random Forest) и экстремальный градиентный бустинг (Extreme Gradient Boosting) [15]. Поскольку важно знать, насколько хорошо модель обобщает результат на новых, ранее неизвестных данных, кроме валидационного набора требуется отложенный

[1] https://www.re3data.org/repository/r3d100010960

[2] https://www.emis.com/

тестовый набор, который в обучении не принимает участие. Во-вторых, в качестве метрики качества используется метрика AUC, которой не достаточно для полной оценки качества прогнозирования банкротства предприятий. Поэтому мы будем вместе с AUC использовать метрику $F1$ для каждого класса.

2.2. Предварительная обработка данных

При построении моделей используются наборы данных, описанные выше, которые делятся следующим образом. Весь набор делится на две части: обучающий (70%) и тестовый (30%) наборы. Выделение достаточно большого тестового набора позволяет повысить достоверность оценки обобщающей способности модели.

Одна из первых задач предварительной обработки данных заключается в заполнении пропущенные значения. Существует несколько стратегий заполнения пропущенных данных. Наиболее популярными стратегиями являются стратегии, базирующиеся на присвоении пропущенным значениям средних (mean) или медианных (median) значений признаков. При использовании деревьев решений эффективной является стратегия маркировки пропущенных значений, например, присвоение пропущенным значениям больших отрицательных чисел (constant).

Таблица 2 представляет результаты сравнения различных стратегий заполнения пропущенных значений для набора данных с меткой статуса банкротства 5 лет. Метрика качества моделей, полученных методом XGB. Столбцы таблиц с заголовками «N» содержат значения метрик $F1$ и AUC для предприятий, не являющихся банкротами, а столбцы, озаглавленные «B», — метрик $F1$ и AUC для предприятий-банкротов.

Таблица 2

Сравнение качества решений для различных стратегий заполнения пропущенных значений

Стратегия	$F1$		AUC	$F1$		AUC
	N	B		N	B	
Mean	0,9901	0,6595	0,943	0,9887	0,6515	0,962
Median	0,9890	0,6067	0,937	0,9883	0,6308	0,964
Constant	0,9907	0,6855	0,941	0,9904	0,7153	0,969

Анализ результатов, представленных в таблицах, показывает стратегия constant, которая заполняет пропущенные значения постоянными числами, равными -99999, является наиболее эффективной. Исследования на других наборах данных также показывает небольшое преимущество стратегии constant. Исследования, проведенные на других наборах данных, подтверждают предпочтение стратегии constant перед другими стратегиями.

Вторая задача, которая решается на этапе подготовки данных, заключается в выборе наиболее информативных признаков из общего числа. Наличие в данных неинформативных признаков приводит к переобучению модели, снижению точности получаемых решений и росту времени обучения модели. В табл. 3 представлено сравнение качества моделей, полученных при различных количествах признаков, которые были отсортированы по коэффициенту значимости.

Таблица 3

Сравнение качества решений для различных комбинаций признаков

Число признаков	*F1*		*AUC*	*F1*		*AUC*
	N	*B*		*N*	*B*	
64	0,9907	0,6855	0,941	0,990	0,7153	0,969
50	0,9909	0,6926	0,9491	0,99	0,6953	0,970
40	0,9909	0,6947	0,9521	0,991	0,7338	0,970
30	0,9909	0,6947	0,9459	0,991	0,7230	0,941

В топ-10 наиболее важных признаков входят следующие признаки:

1) отношение прибыли от операционной деятельности к финансовым расходам;
2) коэффициент текущей ликвидности;
3) отношение операционных расходов к суммарным обязательствам;
4) логарифм от суммарных активов;
5) отношение суммы денежных средств, краткосрочных инвестиций, дебиторской задолженности минус краткосрочные обязательства к разности операционных расходов и амортизации;
6) отношение суммарных издержек к суммарным продажам;
7) отношение выручки от продаж к суммарным активам;
8) отношение дебиторской задолженность умноженной на 360 к выручке от продаж;
9) отношение разности текущих активов и запасов к долгосрочным обязательствам;
10) отношение суммы валовой прибыли, иных активов и транзакции, финансовых затрат к суммарным активам.

Эти показатели измеряют не только рентабельность предприятий, но и активы предприятий, и их денежные потоки. Анализируя результаты выполненных исследований можно сказать, что эти показателя появляются в списке важных показателей для каждого набора данных. Помимо этих 10 показателей были отобраны еще 35 показателей, которые положительно влияют на качество моделей.

2.3. Сравнение качества моделей, построенных различными методами

Для построения моделей используются следующие методы:

- Linear Discriminant Analysis (LDA);
- Logistic Regression (LR);
- Random Forest (RF);
- Extreme gradient boosting (XGB);
- Extreme gradient boosting and Smote (XGB-Sm).

При использовании методов LR, RF, XGB, XGB-Sm осуществлялся поиск наилучших гипер-параметров путем перебора их значений в заданных диапазонах. В ходе исследования было проведено 10 экспериментов. В ходе эксперимента каждый набор делится на обучающий (70%) и тестовый (30%) наборы. При построении модели на обучающем наборе используется процедура кросс-валидации на 10-ти блоках. В результате качество модели оценивается на валидационном и тестовом наборах. В процессе оценки качества модели с использованием метрик *F1* и *accuracy*, выбор порога, при котором вероятностный результат преобразовывается в бинарную метку, осуществляется из условия максимизации метрики *F1* для класса предприятий банкротов. Метрики качества, полученные в ходе 10 экспериментов, усредняются. В табл. 4–8 представлена метрика качества моделей, полученных различными методами: среднее значение метрики

F1 для действующих предприятий (N) и предприятий банкротов (B), и среднее значение метрики AUC и *accuracy*.

Таблица 4

Сравнение качества моделей с горизонтом прогнозирования 1 год

Метод	Валидационный набор				Тестовый набор			
	F1		AUC	Acc	*F1*		AUC	Acc
	N	B			N	B		
LDA	0,9662	0,5075	0,855	0,9367	0,9666	0,5075	0,855	0,9374
LR	0,9713	0,4851	0,709	0,9456	0,9709	0,319	0,716	0,9447
RF	0,9638	0,5767	0,919	0,9333	0,9669	0,6197	0,935	0,9391
XGB	0,9813	0,6971	0,953	0,9647	0,9830	0,7136	0,968	0,9679
XGB-Sm	0,9814	0,7252	0,957	0,9652	0,9829	0,7534	0,968	0,9690

В табл. 4 представлены результаты, полученные на наборе данных с метками статуса банкротства через 1 год. Доля предприятий банкротов в валидационном и тестовом наборах составляют, соответственно, 0,0696 и 0,0688. Таким образом, если в качестве решения использовать простое решение (все предприятия являются действующими), то значения метрики *accuracy* будут равны 0,9304 и 0,9312, соответственно. Будем считать эти значения пороговыми, т.е. прогнозы, метрика *accuracy* которых лежит ниже пороговых значений, будем считать плохими. Как видно из таблицы, у всех моделей значение метрики качества *accuracy* превышает их пороговые значения. Качество моделей, полученных методами LDA и RF, превышает пороговые значения незначительно, в то время как метрика AUC имеет достаточно высокие значения. Использование метода логистической регрессии позволяет получить прогнозы, метрика *accuracy* которых превышает пороговые значения почти на 1,5%, а метрика AUC намного ниже аналогичной метрики в случае применения методов LDA и RF.

При использовании логистической регрессии построенные модели позволяют точно классифицировать 106 из 288 и 39 из 122 предприятий банкротов для валидационного и тестового наборов, соответственно. Также правильно классифицируются 3806 из 3849 и 1636 из 1651 действующих предприятий для валидационного и тестового наборов данных, соответственно. При использовании моделей, построенных с помощью LDA, правильно классифицируются 135 из 288 и 57 из 122 предприятий банкротов для валидационного и тестового наборов, соответственно. Из действующих предприятий валидационного и тестового наборов правильно классифицируются 3750 из 3849 и 1605 из 1651, соответственно.

Модели, построенные с помощью RF, позволяют правильно классифицировать 188 из 288 и 88 из 122 предприятий банкротов валидационного и тестового наборов, соответственно. Из действующих предприятий валидационного и тестового наборов правильно классифицируются 3673 из 3849 и 1577 из 1651, соответственно.

Наибольшую точность демонстрируют модели, полученные с помощью XGB и XGB-Sm. Улучшение сбалансированности обучающих наборов данных повышают качество прогноза, так метрика качества *F1* для предприятий банкротов увеличивается на 2,8% и 3,9% для валидационного и тестового наборов данных, соответственно. Метрика *F1* для действующих предприятий практически не изменяется. Оптимальный коэффициент баланса составляет 0,09.

Таблица 5

Сравнение качества моделей с горизонтом прогнозирования 2 года

Метод	Валидационный набор				Тестовый набор			
	F1		*AUC*	*Accuracy*	*F1*		*AUC*	*Accuracy*
	N	*B*			*N*	*B*		
LDA	0,9753	0,5038	0,825	0,9529	0,9732	0,4526	0,825	0,9489
LR	0,9742	0,4329	0,690	0,9507	0,9743	0,3777	0,658	0,9506
RF	0,9821	0,6274	0,883	0,9610	0,9819	0,5887	0,886	0,9632
XGB	0,9839	0,6201	0,934	0,9691	0,9835	0,5903	0,934	0,9683
XGB-Sm	0,9835	0,6861	0,942	0,9710	0,9823	0,6374	0,944	0,9694

В табл. 5 представлена метрика качества моделей, полученных на наборах с метками статуса банкротства предприятий через 2 года. Пороговые значения метрики *accuracy* для валидационного и тестового наборов составляют 94,61% и 95,03%, соответственно. Как видно из таблицы, на тестовом наборе модели LDA имеют значение *accuracy* ниже порогового значения, а значение метрики *accuracy* моделей LR незначительного превышает соответствующее пороговое значение.

Модели, построенные методами XGB и XGB-Sm, показывают наиболее высокие значения метрик качества. Улучшение сбалансированности обучающих наборов данных повышает качество прогноза, например, метрика качества *F1* для предприятий банкротов увеличивается на 6,6% и 4,7% для валидационного и тестового наборов, соответственно. Хотя при этом метрика *F1* для действующих предприятий немного снижается (0,04% и 0,01%). Оптимальный коэффициент баланса классов составляет 0,09.

Таблица 6

Сравнение качества моделей с горизонтом прогнозирования 3 года

Метод	Валидационный набор				Тестовый набор			
	F1		*AUC*	*Accuracy*	*F1*		*AUC*	*Accuracy*
	N	*B*			*N*	*B*		
LDA	0,9526	0,3421	0,790	0,9116	0,9543	0,3182	0,790	0,9143
LR	0,9677	0,2720	0,661	0,9381	0,9650	0,2327	0,663	0,9330
RF	0,9795	0,5207	0,885	0,9607	0,9752	0,4186	0,856	0,9524
XGB	0,9842	0,5591	0,931	0,9695	0,9828	0,4274	0,912	0,9667
XGB-Sm	0,9852	0,6534	0,938	0,9716	0,9823	0,5597	0,919	0,9660

В табл. 6 представлена метрика качества моделей, полученных на наборах с метками статуса банкротства предприятий через 3 года. Пороговые значения метрики *accuracy* для валидационного и тестового наборов составляют 95,25% и 95,36%, соответственно. Как видно из таблицы, метрика *accuracy* моделей, построенных методами LDA и LR, на тестовом наборе лежит ниже пороговых значений. Модели, построенные с использованием метода RF, при классификации предприятий тестового набора показывают результаты, в которых доля правильно классифицированных предприятий ниже порогового значения, что также говорит о невысоком качестве модели.

Модели, построенные методами XGB и XGB-Sm, показывают наиболее высокие значения метрики качества. Улучшение сбалансированности набора данных повышают качество прогноза, Метрика качества *F1* для предприятий банкротов увеличивается на 9,5% и 13,1% на валидационном и тестовом наборах данных, соответственно. Хотя при

этом метрика *F1* для действующих предприятий увеличивается на 0,1% на валидационной наборе и снижается на 0,05% на тестовом наборе. Оптимальный коэффициент баланса классов составляет 0,06.

Таблица 7

Сравнение качества моделей с горизонтом прогнозирования 4 года

Метод	Валидационный набор				Тестовый набор			
	F1		*AUC*	*Accuracy*	*F1*		*AUC*	*Accuracy*
	N	*B*			*N*	*B*		
LDA	0,9663	0,3289	0,721	0,9358	0,9693	0,2800	0,721	0,9410
LR	0,9741	0,2452	0,601	0,9499	0,9809	0,2083	0,606	0,9626
RF	0,9822	0,5243	0,865	0,9656	0,9855	0,5275	0,850	0,9718
XGB	0,9868	0,5667	0,914	0,9744	0,9882	0,5333	0,937	0,9771
XGB-Sm	0,9874	0,6547	0,921	0,9757	0,9882	0,6316	0,939	0,9780

В табл. 7 представлена метрика качества моделей, полученных на наборах с метками статуса банкротства предприятий через 4 года. Пороговые значения метрики *accuracy* для валидационного и тестового наборов данных составляют 95,87% и 96,53%, соответственно. Как видно из таблицы, метрика качества *accuracy* моделей, построенных методами LDA и LR, на валидационном и тестовом наборах имеют значение ниже пороговых значений.

Модели, построенные методами XGB и XGB-Sm, показывают наиболее высокие значения метрики качества. Улучшение сбалансированности набора данных повышают качество прогноза, так метрика качества *F1* для предприятий банкротов увеличивается на 8,8% и 9,8% для валидационного и тестового наборов, соответственно. Хотя при этом метрика *F1* для действующих предприятий увеличивается на 0,06% на валидационном наборе и не меняется на тестовом наборе. Оптимальный коэффициент баланса классов составляет 0,06.

Таблица 8

Сравнение качества моделей с горизонтом прогнозирования 5 лет

Метод	Валидационный набор				Тестовый набор			
	F1		*AUC*	*Accuracy*	*F1*		*AUC*	*Accuracy*
	N	*B*			*N*	*B*		
LDA	0,9775	0,4928	0,842	0,9569	0,9812	0,5870	0,924	0,9640
LR	0,9814	0,5294	0,882	0,9642	0,9820	0,5629	0,893	0,9654
RF	0,9815	0,4699	0,908	0,9642	0,9808	0,5185	0,942	0,9630
XGB	0,9909	0,6926	0,937	0,9823	0,9909	0,6947	0,942	0,9806
XGB-Sm	0,9910	0,7075	0,943	0,9827	0,9914	0,7552	0,969	0,9844

В табл. 8 представлена метрика качества моделей, полученных на наборах с метками статуса банкротства предприятий через 5 лет. Пороговые значения метрики *accuracy* для валидационного и тестового наборов, составляют 96,28% и 95,83%, соответственно. Как видно из таблицы, метод LR на валидационном наборе имеют значение *accuracy* ниже пороговых значений, хотя на тестовом наборе ситуация уже меняется.

Модели, построенные методами XGB и XGB-Sm, показывают наиболее высокие значения метрики качества. Улучшение сбалансированности обучающих наборов данных

повышают качество прогноза, но незначительно. Оптимальный коэффициент баланса классов составляет 0,07.

Таким образом, модели, построенные методами LDA и LR, дают самую низкую точность прогноза, которая часто оказвается ниже пороговых значений. Модели, построенные методом RF, хотя и дают более высокую прогноза, но в некоторых случаях не позволяет получить модели с хорошим качеством. Модели, построенные XGB-Sm, демонстриуют наиболее высокие значения метрики качеством как на валидационном, так и на тестовом наборах. Следует отметить, что в случае если число искусственных образцов, генерируемых методом SMOTE, превышает исходное число образцов, качество моделей, построенных экстремальных градиентным бустингом, существенно падает.

Заключение

В статье рассмотрена проблема прогнозирования банкротства на основе финансовых факторов. Для решения поставленной задачи классификации используется модель, построенная с помощью экстремального градиентного бустинга. Результаты, полученные экстремальным градиентным бустингом, были значительно лучше, чем результаты, полученные такими известными методами как линейный дискриминантный анализ, логистическая регрессия, которые широко применяются для прогнозирования финансового состояния компаний. В работе предложено расширение экстремального градиентного бустинга, которое улучшает дисбаланс классов обучающих наборов с помощью метода SMOTE. Применение такого подхода привело к улучшению качества прогнозирования.

Будущие исследования планируется вести в направлении учета темпов роста финансовых показателей при построении моделей, чтобы оценить влияние времени на вероятность банкротства предприятий.

Литература

1. Altman E.I. Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy // The Journal of Finance. 1968. Vol. 23, no. 4. P. 589–609. DOI: 10.1111/j.1540-6261.1968.tb00843.x.
2. Lugovskaya L. Predicting Default of Russian Smes on the Basis of Financial and Nonfinancial Variables // Journal of Financial Services Marketing. 2010. Vol. 14, no. 4. P. 301–313. DOI: 10.1057/fsm.2009.28.
3. Deakin E. A Discriminant Analysis of Predictors of Business Failure // Journal of Accounting Research. 1972. Vol. 10, no. 1. P. 167–179. DOI: 10.2307/2490225.
4. Antunesa F., Ribeiroa B., Pereirab F. Probabilistic Modeling and Visualization for Bankruptcy Prediction // Applied Soft Computing. 2017. Vol. 60. P. 831–843. DOI: 10.1016/j.asoc.2017.06.043.
5. Ohlson J.A. Financial Ratios and the Probabilistic Prediction of Bankruptcy // Journal Of Accounting Research. 1980. Vol. 18, no. 1. P. 109–131. DOI: 10.2307/2490395.
6. Martin D. Early Warning of Bank Failure: A Logit Regression Approach // Journal of Banking & Finance. 1977. Vol. 1, no. 3. P. 249–276.
7. Wiginton J.C. A Note on the Comparison of Logit and Discriminant Models of Consumer Credit Behavior // Journal of Financial and Quantitative Analysis. 1980. Vol. 15. P. 757–770. DOI: 10.2307/2330408.
8. Begley J., Ming J., Watts S. Bankruptcy Classification Errors in the 1980s: an Empirical Analysis of Altman's and Ohlson's Models // Review of Accounting Studies. 1996. Vol. 1, no. 4. P. 267–284. DOI: 10.1007/bf00570833.

9. Wilson R.L, Sharda R., Bankruptcy Prediction Using Neural Networks // Decision Support Systems. 1994. Vol. 11, no. 5. P. 545–557. DOI: 10.1016/0167-9236(94)90024-8.
10. Tam K.Y., Kiang M.Y. Managerial Applications of Neural Networks: the Case of Bank Failure Predictions // Management Science. 1992. Vol. 38, no. 7. P. 926–947. DOI: 10.1287/mnsc.38.7.926.
11. Altman E.I., Marco G., Varetto F. Corporate Distress Diagnosis: Comparisons Using Linear Discriminant Analysis and Neural Networks (the Italian Experience) // Journal of Banking & Finance. 1994. Vol. 18, no. 3. P. 505–529. DOI: 10.1016/0378-4266(94)90007-8.
12. Ciampi F., Vallini C., Gordini N. Using Artificial Neural Networks Analysis for Small Enterprise Default Prediction Modeling: Statistical Evidence from Italian Firms // Oxford Business & Economics Conference Proceedings, Association for Business and Economics Research, ABER. 2009. Vol. 1. P. 126.
13. Wei L., Li J., Chen Z. Credit Risk Evaluation Using Support Vector Machine with Mixture of Kernel // Proceedings of the 7th International Conference on Computational Science. Lecture Notes in Computational Science and Engineering. 2007. Vol. 4488. P. 431–438. DOI: 10.1007/978-3-540-72586-2_62.
14. Härdle W.K., Lee Y.J., Schäfer D. The Default Risk of Firms Examined with Smooth Support Vector Machines // Discussion Papers, German Institute for Economic Research. 2007. no. 757. P. 1–30. DOI: 10.2139/ssrn.2894311.
15. Zieba M., Tomczak S.K., Tomczak J.M. Ensemble Boosted Trees with Synthetic Features Generation in Application to Bankruptcy Prediction // Expert Systems with Applications. 2016. Vol. 58. P. 93–101. DOI: 10.1016/j.eswa.2016.04.001.
16. Xia Y., Liu C.,. Li Y, Liu N. A Boosted Decision Tree Approach Using Bayesian Hyper-Parameter Optimization for Credit Scoring // Expert Systems With Applications. 2017. Vol. 78. P. 225–241. DOI: doi.org/10.1016/j.eswa.2017.02.017.
17. Zhou L. Performance of Corporate Bankruptcy Prediction Models on Imbalanced Dataset: The Effect of Sampling Methods // Knowledge-Based Systems. 2013. Vol. 41. P. 16–25. DOI: 10.1016/j.knosys.2012.12.007.
18. Kim T., Ahn H. A Hybrid Under-Sampling Approach for Better Bankruptcy Prediction // Journal of Intelligent Information Systems. 2015. Vol. 21, no. 2. P. 173–190. DOI: doi.org/10.13088/jiis.2015.21.2.173.
19. Veganzonesa D., Séverina E. An Investigation of Bankruptcy Prediction in Imbalanced Datasets // Decision Support Systems. 2018. no. 112. P. 111–124. DOI: 10.1016/j.dss.2018.06.011.
20. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. 2016. P. 785–794. DOI: 10.1145/2939672.2939785.
21. Friedman J.H. Stochastic Gradient Boosting // Computational Statistics and Data Analysis. 2002. no. 38. P. 367–378. DOI: 10.1016/j.eswa.2016.04.001

Мокеев Владимир Викторович, д.т.н., старший научный сотрудник, профессор кафедры информационных технологии в экономике, Южно-Уральского государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Войтецкий Роман Владимирович, студент кафедры информационных технологии в экономике, Южно-Уральского государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse200305

FORECASTING ENTERPRISES BANKRUPTCY BY EXTREME GRADIENT BOOSTING

© 2020 V.V. Mokeyev, R.V. Voitetckiy
South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)
E-mail: mokeyev@mail.ru, romanvoitetckii@gmail.com
Received: 27.07.2020

The application of models for forecasting bankruptcy of enterprises for controlling investment is the basis for monitoring activities of financial institutions. A crucial factor in allowing financial institutions to determine the amount of capital to cover credit losses is the accuracy of the forecast. Most studies use traditional statistical methods (for example, linear discriminant analysis and logistic regression) to build models of enterprise bankruptcy forecasting, but the accuracy of these models is usually quite low. The reason for that is the imbalanced nature of training data sets (the share of bankrupt firms is a small percent of the total number of firms). Nowadays, such machine learning methods as the random forest and the gradient boosting are becoming widespread. This study focuses on the use of extreme gradient boosting to predict bankruptcy. Extreme gradient boosting, using LASSO or Ridge regularization, penalizes complex models to avoid overfitting. Also, during training, extreme gradient boosting fills in the missing values of the data set, depending on the value of the loss. In this article, we proposed SMOTE technique to enhance the minority class of the training data sets, which helps to improve the performance of extreme gradient boosting. The experiment results of improved extreme gradient boosting are compared to the outcomes obtained by other methods.

Keywords: extreme gradient boosting, bankruptcy, enterprises, synthetic minority oversampling technique.

FOR CITATION

Mokeyev V.V., Voitetckiy R.V. Forecasting Enterprises Bankruptcy by Extreme Gradient Boosting. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2020. Vol. 9, no. 3. P. 77–90. (in Russian) DOI: 10.14529/cmse200305.

References

1. Altman E.I. Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. The Journal of Finance. 1968. Vol. 23, no. 4. P. 589–609. DOI: 10.1111/j.1540-6261.1968.tb00843.x.
2. Lugovskaya L. Predicting Default of Russian Smes on the Basis of Financial and Nonfinancial Variables. Journal of Financial Services Marketing. 2010. Vol. 14, no. 4. P. 301–313. DOI: 10.1057/fsm.2009.28.
3. Deakin E. A Discriminant Analysis of Predictors of Business Failure. Journal of Accounting Research. 1972. Vol. 10, no. 1. P. 167–179. DOI: 10.2307/2490225.
4. Antunesa F., Ribeiroa B., Pereirab F. Probabilistic Modeling and Visualization for Bankruptcy Prediction. Applied Soft Computing. 2017. Vol. 60. P. 831–843. DOI: 10.1016/j.asoc.2017.06.043.
5. Ohlson J.A. Financial Ratios and the Probabilistic Prediction of Bankruptcy. Journal Of Accounting Research. 1980. Vol. 18, no. 1. P. 109–131. DOI: 10.2307/2490395.
6. Martin D. Early Warning of Bank Failure: a Logit Regression Approach. Journal of Banking & Finance. 1977. Vol. 1, no. 3. P. 249–276.

7. Wiginton J.C. A Note on the Comparison of Logit and Discriminant Models of Consumer Credit Behavior. Journal of Financial and Quantitative Analysis. 1980. Vol. 15. P. 757–770. DOI: 10.2307/2330408.
8. Begley J., Ming J., Watts S. Bankruptcy Classification Errors in the 1980s: an Empirical Analysis of Altman's and Ohlson's Models. Review of Accounting Studies. 1996. Vol. 1, no. 4. P. 267–284. DOI: 10.1007/bf00570833.
9. Wilson R.L, Sharda R. Bankruptcy Prediction Using Neural Networks. Decision Support Systems. 1994. Vol. 11, no. 5. P. 545–557. DOI: 10.1016/0167-9236(94)90024-8.
10. Tam K.Y., Kiang M.Y. Managerial Applications of Neural Networks: the Case of Bank Failure Predictions. Management science. 1992. Vol. 38, no. 7. P. 926–947. DOI: 10.1287/mnsc.38.7.926.
11. Altman E.I., Marco G., Varetto F. Corporate Distress Diagnosis: Comparisons Using Linear Discriminant Analysis and Neural Networks (the Italian Experience). Journal of Banking & Finance. 1994. Vol. 18, no. 3. P. 505–529. DOI: 10.1016/0378-4266(94)90007-8.
12. Ciampi F., Vallini C., Gordini N. Using Artificial Neural Networks Analysis for Small Enterprise Default Prediction Modeling: Statistical Evidence from Italian Firms. Oxford Business & Economics Conference Proceedings, Association for Business and Economics Research, ABER. 2009. Vol. 1. P. 126.
13. Wei L., Li J., Chen Z. Credit Risk Evaluation Using Support Vector Machine with Mixture of Kernel. Proceedings of the 7th International Conference on Computational Science. Lecture Notes in Computational Science and Engineering. 2007. Vol. 4488. P. 431–438. DOI: 10.1007/978-3-540-72586-2_62.
14. Härdle W.K., Lee Y.J., Schäfer D. The Default Risk of Firms Examined with Smooth Support Vector Machines. Discussion Papers, German Institute for Economic Research. 2007. Vol. 757. P. 1–30. DOI: 10.2139/ssrn.2894311.
15. Zieba M., Tomczak S.K., Tomczak J.M. Ensemble Boosted Trees with Synthetic Features Generation in Application to Bankruptcy Prediction. Expert Systems with Applications. 2016. Vol. 58. P. 93–101. DOI: 10.1016/j.eswa.2016.04.001.
16. Xia Y., Liu C.,. Li Y, Liu N. A Boosted Decision Tree Approach Using Bayesian Hyper-Parameter Optimization for Credit Scoring. Expert Systems With Applications. 2017. Vol. 78. P. 225–241. DOI: 10.1016/j.eswa.2017.02.017.
17. Zhou L. Performance of Corporate Bankruptcy Prediction Models on Imbalanced Dataset: The Effect of Sampling Methods. Knowledge-Based Systems. 2013. Vol. 41. P. 16–25. DOI: 10.1016/j.knosys.2012.12.007.
18. Kim T., Ahn H. A Hybrid Under-Sampling Approach for Better Bankruptcy Prediction. Journal of Intelligent Information Systems. 2015. Vol. 21, no. 2. P. 173–190. DOI: 10.13088/jiis.2015.21.2.173.
19. Veganzonesa D., Séverina E. An Investigation of Bankruptcy Prediction in Imbalanced Datasets. Decision Support Systems. 2018. no. 112. P. 111–124. DOI: 10.1016/j.dss.2018.06.011.
20. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. 2016. P. 785–794. DOI: 10.1145/2939672.2939785.
21. Friedman J.H. Stochastic Gradient Boosting. Computational Statistics and Data Analysis. 2002. no. 38. P. 367–378. DOI: 10.1016/j.eswa.2016.04.001.

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: 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки), 05.13.17 – Теоретические основы информатики (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 `http://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,

   ответственному секретарю 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»
Том 9, № 3
2020

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 31.08.2020. Дата выхода в свет 07.09.2020. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 10,69. Тираж 500 экз. Заказ 247/295. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.