2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

doi 10.26089/NumMet.v22r423

УДК 004.272.25;
004.421;
004.032.24

Поиск типичных подпоследовательностей временного ряда на графическом процессоре

М. Л. Цымблер

Южно-Уральский государственный университет (национальный исследовательский университет), Челябинск, Российская Федерация

ORCID: https://orcid.org/0000-0001-7491-8656, e-mail: mzym@susu.ru

А. И. Гоглачев

Южно-Уральский государственный университет (национальный исследовательский университет), Челябинск, Российская Федерация

ORCID: https://orcid.org/0000-0003-2122-7303, e-mail: goglachevai@susu.ru

Аннотация: Поиск типичных подпоследовательностей временно́го ряда является одной из актуальных задач интеллектуального анализа временны́х рядов. Данная задача предполагает нахождение набора подпоследовательностей временно́го ряда, которые адекватно отражают течение процесса или явления, задаваемого этим рядом. Поиск типичных подпоследовательностей дает возможность резюмировать и визуализировать большие временны́е ряды в широком спектре приложений: мониторинг технического состояния сложных машин и механизмов, интеллектуальное управление системами жизнеобеспечения, мониторинг показателей функциональной диагностики организма человека и др. Предложенная недавно концепция сниппета формализует типичную подпоследовательность временно́го ряда следующим образом. Сниппет представляет собой подпоследовательность, на которую похожи многие другие подпоследовательности данного ряда в смысле специализированной меры схожести, основанной на евклидовом расстоянии. Поиск типичных подпоследовательностей с помощью сниппетов показывает адекватные результаты для временны́х рядов из широкого спектра предметных областей, однако соответствующий алгоритм имеет высокую вычислительную сложность. В настоящей работе предложен новый параллельный алгоритм поиска сниппетов во временно́м ряде на графическом ускорителе. Распараллеливание выполнено с помощью технологии программирования CUDA. Разработаны структуры данных, позволяющие эффективно распараллелить вычисления на графическом процессоре. Представлены результаты вычислительных экспериментов, подтверждающих высокую производительность разработанного алгоритма.

Ключевые слова: временно́й ряд, поиск типичных подпоследовательностей, матричный профиль, мера MPdist, параллельный алгоритм, графический процессор.

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

Для цитирования: Цымблер М.Л., Гоглачев А.И. Поиск типичных подпоследовательностей временного ряда на графическом процессоре // Вычислительные методы и программирование. 2021. **22**, № 4. 344–359. doi 10.26089/NumMet.v22r423.

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

Discovery of typical subsequences of time series on graphics processor

Mikhail L. Zymbler
South Ural State University (National Research University), Chelyabinsk, Russia
ORCID: https://orcid.org/0000-0001-7491-8656, e-mail: mzym@susu.ru
Andrey I. Goglachev
South Ural State University (National Research University), Chelyabinsk, Russia
ORCID: https://orcid.org/0000-0003-2122-7303, e-mail: goglachevai@susu.ru

Abstract: Discovery of typical subsequences in a time series is one of the topical problems of time series mining. In this problem, we are to find a set of subsequences that adequately represents the specified time series. The solution of such a problem makes it possible to summarize and visualize a large time series in a wide range of applications: monitoring of the technical condition of complex machines and mechanisms, intelligent management of life support systems, monitoring of indicators of functional diagnostics of the human body, etc. The recently proposed snippet concept formalizes a typical time series subsequence as follows. A snippet of a time series is a subsequence that many other subsequences of the given series are similar to, with respect to a specialized similarity measure based on the Euclidean distance. Despite the snippets discovery algorithm shows adequate results for time series from a wide range of subject domains, it has a high computational complexity. In this article, we propose a novel parallel algorithm for snippets discovery on GPU. Parallelization is performed through the CUDA programming technology. We developed data structures that allow for efficient parallelization of GPU calculations. The experimental results show the high performance of the proposed algorithm.

Keywords: time series, typical subsequences, matrix profile, MPdist, parallel algorithm, GPU.

Acknowledgements: This work was financially supported by the Russian Foundation for Basic Research (grant No. 20–07–00140) and Ministry of Science and Higher Education of the Russian Federation (Government Order FENU–2020–0022).

For citation: M. L. Zymbler, A. I. Goglachev, "Discovery of typical subsequences of time series on graphics processor," Numerical Methods and Programming. **22** (4), 344–359 (2021). doi 10.26089/NumMet.v22r423.

---

1. Введение. Поиск типичных подпоследовательностей временно́го ряда является одной из актуальных задач интеллектуального анализа временны́х рядов [1]. Данная задача в неформальной постановке предполагает нахождение набора подпоследовательностей временно́го ряда, которые адекватно отражают течение процесса или явления, задаваемого этим рядом. Поиск типичных подпоследовательностей дает возможность резюмировать (кратко описать) и визуализировать большие временны́е ряды в широком спектре приложений: мониторинг технического состояния сложных машин и механизмов [2], интеллектуальное управление системами жизнеобеспечения [3, 4], мониторинг показателей функциональной диагностики организма человека [5], моделирование климата [6], финансовое прогнозирование [7] и др.

Научным сообществом были предложены различные подходы к формализации понятия типичной подпоследовательности временно́го ряда: лейтмотив (motif) [8], шейплет (shapelet) [9], ослабленный период (relaxed period) и средняя тенденция (average trend) ряда [10] и др. Однако указанные подходы не в полной мере решают поставленную задачу, поскольку не предоставляют возможности указать долю временно́го ряда, которой соответствует найденная типичная подпоследовательность. Данный недостаток преодолен в недавней работе [1], авторы которой предложили концепцию сниппета (snippet). Сниппет представляет собой подпоследовательность заданной длины, на которую похожи многие другие подпоследовательности данного ряда в смысле специально определяемой меры схожести [11], основанной на евклидовом расстоянии. Эксперименты авторов свидетельствуют, что поиск типичных подпоследовательностей с помощью сниппетов показывает адекватные результаты для временны́х рядов из широкого спектра предметных областей [1]. Однако предложенный авторами алгоритм имеет высокую вычислительную

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

сложность, что дает низкую производительность при обработке временны́х рядов, насчитывающих от сотни тысяч элементов.

В настоящей статье предложен новый параллельный алгоритм поиска сниппетов во временно́м ряде на графическом ускорителе. Статья организована следующим образом. В разделе 2 дан краткий обзор работ по тематике исследования. Раздел 3 содержит формальные определения и постановку задачи. В разделе 4 кратко описан оригинальный последовательный алгоритм поиска сниппетов (типичных подпоследовательностей) временно́го ряда. В разделе 5 представлен новый параллельный алгоритм поиска типичных подпоследовательностей временно́го ряда на графическом процессоре. В разделе 6 описаны вычислительные эксперименты по исследованию эффективности предложенного алгоритма. Заключение резюмирует полученные результаты и описывает направления будущих исследований.

2. Обзор связанных работ. Теме поиска типичных подпоследовательностей временно́го ряда посвящены следующие основные работы. В работе [8] Муин (Mueen) и др. описали концепцию лейтмотивов, которая может использоваться для формализации задачи поиска типичных подпоследовательностей временно́го ряда. *Лейтмотив (motif)* представляет собой пару подпоследовательностей временно́го ряда, которые наименее удалены друг от друга в смысле некоторой функции расстояния. Авторами предложен эффективный алгоритм поиска лейтмотивов на основе евклидова расстояния, который использует отбрасывание бесперспективных кандидатов в лейтмотивы с помощью аксиомы треугольника. Позднее в работах [12] и [13] были предложены параллельные версии указанного алгоритма для многоядерного ускорителя и графического процессора соответственно. Использование концепции лейтмотива позволяет найти схожие подпоследовательности ряда, которые можно считать повторяющимися и типичными. Однако лейтмотивы не вполне подходят для резюмирования, поскольку отсутствует возможность указать долю временно́го ряда, которой соответствует лейтмотив.

В предметных областях, не связанных с временны́ми рядами, задачи резюмирования и визуализации множеств объектов могут быть решены с помощью методов кластеризации. *Кластеризация (clustering)* предполагает разбиение заданного множества объектов на подмножества (кластеры) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров, в смысле выбранной меры схожести. Однако кластеризация множества подпоследовательностей временно́го ряда, имеющих одну длину (которые в этом случае рассматриваются как точки многомерного метрического пространства), несостоятельна и приводит к лишенным смысла результатам, как показали Кеог (Keogh) и др. в работе [14].

В работе [9] Йе (Ye) и др. предложили концепцию шейплетов, которая может использоваться для формализации понятия типичных подпоследовательностей временно́го ряда. Данная концепция предполагает, что подпоследовательности ряда предварительно подвергнуты классификации, и *шейплет (shapelet)* представляет собой подпоследовательность ряда, которая минимально удалена от большинства подпоследовательностей заданного класса и максимально удалена от подпоследовательностей, принадлежащих другим классам, в смысле заданной функции расстояния. Применение шейплетов в качестве типичных подпоследовательностей затруднено, поскольку данный подход требует обучения с учителем на основе экспертных знаний в целевой предметной области.

В работе [10] Индык (Indyk) и др. предложили концепции ослабленного периода и средней тенденции ряда, определяемые следующим образом. Пусть для заданного временно́го ряда фиксированы длина подпоследовательности и мера схожести. Подпоследовательность временно́го ряда является *ослабленным периодом (relaxed period)*, если синтетический ряд, который имеет ту же длину, что исходный ряд, и составлен посредством многократной конкатенации указанной подпоследовательности, имеет наибольшую схожесть с исходным рядом. *Средней тенденцией (average trend)* временно́го ряда называется подпоследовательность, для которой достигается наибольшее значение суммы квадратов значений ее схожести со всеми прочими подпоследовательностями ряда. Однако использование данных концепций предполагает, что временно́й ряд предварительно разделен на периоды с четко определенной длительностью и начальным индексом, что существенно сужает спектр предметных областей для применения данного подхода.

Простая случайная выборка (simple random sampling) предполагает отбор подпоследовательностей ряда в случайном порядке без разделения их на группы. Данный подход применим к поиску типичных подпоследовательностей временно́го ряда в существенно узком спектре предметных областей и применяется в качестве линии отсчета для сравнения прочих подходов к решению указанной задачи [1].

В работе [1] Кеогом (Keogh) и др. для формализации задачи поиска типичных подпоследовательностей временно́го ряда предложена концепция сниппетов. *Сниппет (snippet)* представляет собой под-

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

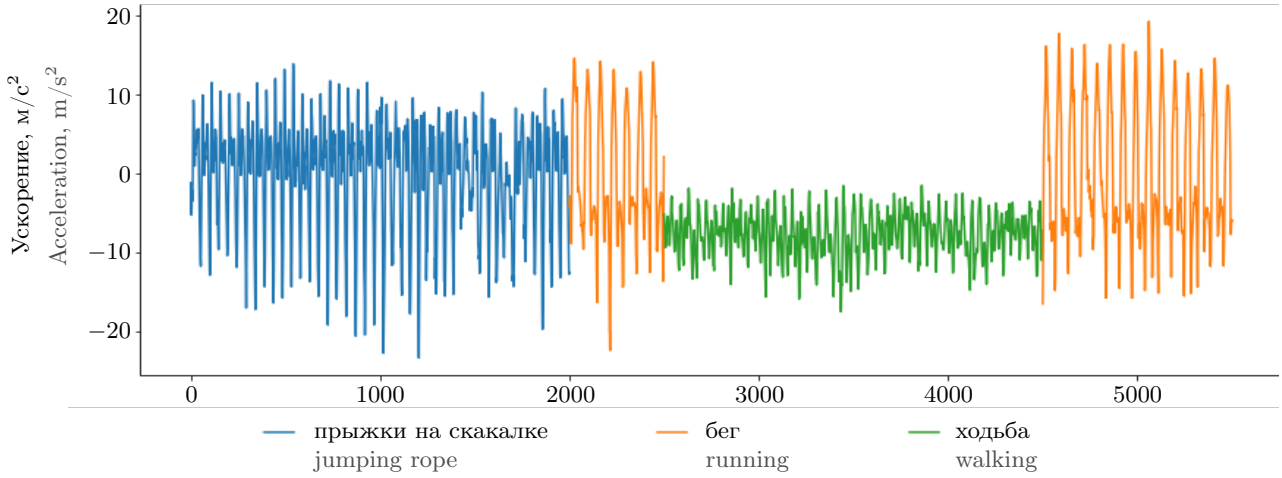

Рис. 1. Резюмирование временно́го ряда PAMAP [15] с помощью сниппетов

Fig. 1. Summarizing PAMAP Time Series [15] Using Snippets

последовательность заданной длины, на которую похожи многие другие подпоследовательности данного ряда в смысле специально определяемой меры схожести. Набор сниппетов имеет существенно меньшую мощность, чем множество подпоследовательностей ряда, имеющих заданную длину, и потому используется для резюмирования исходного временно́го ряда. По сравнению с вышеупомянутыми подходами к решению задачи поиска типичных подпоследовательностей временно́го ряда сниппеты поддерживают для результатов поиска количественно измеряемые свойства достоверности и покрытия. *Достоверность (fidelity)* сниппета означает, что между данным сниппетом и каждой из подпоследовательностей, которые он резюмирует, обеспечивается схожесть, наибольшая из возможных. *Покрытие (coverage)* сниппета означает, что можно указать долю подпоследовательностей ряда, которые резюмирует данный сниппет. Эксперименты авторов показывают, что поиск типичных подпоследовательностей с помощью сниппетов дает адекватные результаты для временны́х рядов из широкого спектра предметных областей [1]. На рис. 1 показан фрагмент временно́го ряда, резюмированного с помощью сниппетов. Временно́й ряд PAMAP [15] представляет собой показания закрепленного на человеке виброакселерометра, для которых известны виды физической активности, выполнявшиеся этим человеком (ходьба, бег, прыжки, глажка белья, положение стоя и др.). Можно видеть, что в приведенном фрагменте хорошо различимы промежутки, соответствующие трем видам активности (подпоследовательности ряда, наиболее похожие на соответствующие сниппеты). Однако алгоритм SnippetFinder, предложенный авторами, имеет высокую сложность $O(n^2 \cdot (n-m)/m)$, где n — длина временно́го ряда, m — длина сниппета [1], что дает низкую производительность при обработке временны́х рядов, насчитывающих от сотни тысяч элементов.

Проведенный обзор показывает, что актуальной является задача распараллеливания алгоритма SnippetFinder [1] для современных высокопроизводительных платформ. В данном исследовании предлагается новый параллельный алгоритм поиска сниппетов временно́го ряда для графического процессора.

3. Постановка задачи.

3.1. Основные определения и нотация. В данном разделе приводятся обозначения и определения используемых терминов в соответствии с работой [1].

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

$$T = (t_1, \ldots, t_n), \quad t_i \in \mathbb{R}.$$

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

Подпоследовательность (subsequence) $T_{i,m}$ временно́го ряда T представляет собой непрерывное подмножество T из m элементов, начиная с позиции i:

$$T_{i,m} = (t_i, \ldots, t_{i+m-1}), \quad 1 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n-m+1.$$

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

Временной ряд T может быть логически разбит на сегменты — непересекающиеся подпоследовательности заданной длины m. Здесь и далее без существенного ограничения общности мы можем считать, что n кратно m, поскольку $m \ll n$. Множество сегментов ряда, имеющих длину $m \ll n$, обозначим как S_T^m, элементы этого множества как $S_1, \ldots, S_{n/m}$:

$$S_T^m = (S_1, \ldots, S_{n/m}), \quad S_i = T_{m\cdot(i-1)+1,\, m}.$$

Концепция *сниппетов (snippet)* предложена Кеогом и др. в работе [1] и уточняет понятие типичных подпоследовательностей временно́го ряда следующим образом. Каждый сниппет представляет собой один из сегментов временно́го ряда. Со сниппетом ассоциируются его ближайшие соседи — подпоследовательности ряда, имеющие ту же длину, что и сниппет, которые более похожи на данный сниппет, чем на другие сегменты. Для вычисления схожести подпоследовательностей используется специализированная мера схожести, основанная на евклидовом расстоянии. Сниппеты упорядочиваются по убыванию мощности множества своих ближайших соседей. Формальное определение сниппетов выглядит следующим образом. Обозначим множество сниппетов ряда T, имеющих длину m, как C_T^m, а элементы этого множества как $C_1, \ldots, C_K$:

$$C_T^m = (C_1, \ldots, C_K), \quad C_i \in S_T^m.$$

Число K $(1 \leqslant K \leqslant n/m)$ представляет собой параметр, задаваемый прикладным программистом, и отражает соответствующее количество наиболее типичных сниппетов. С каждым сниппетом ассоциированы следующие атрибуты: индекс сниппета, ближайшие соседи и значимость данного сниппета.

Индекс сниппета $C_i \in C_T^m$ обозначается как $C_i.index$ и представляет собой номер j сегмента S_j, которому соответствует подпоследовательность ряда $T_{m\cdot(j-1)+1,\, m}$.

Множество ближайших соседей сниппета $C_i \in C_T^m$ обозначается как $C_i.Neighbors$ и содержит подпоследовательности ряда, которые более похожи на данный сниппет, чем на другие сегменты ряда, в смысле меры схожести MPdist [11]:

$$C_i.Neighbours = \{T_{j,m} \mid S_{C_i.index} = \arg \min_{1\leqslant r\leqslant n/m} \mathrm{MPdist}(T_{j,m},\, S_r),\ 1 \leqslant j \leqslant n-m+1\}.$$

Значимость сниппета $C_i \in C_T^m$ обозначается как $C_i.frac$ и представляет собой долю множества ближайших соседей сниппета в общем количестве подпоследовательностей ряда, имеющих длину m:

$$C_i.frac = \frac{|\,C_i.Neighbours\,|}{n-m+1}.$$

Сниппеты упорядочиваются по убыванию их значимости:

$$\forall C_i, C_j \in C_T^m :\ i < j \iff C_i.frac \geqslant C_j.frac.$$

3.2. Мера схожести MPdist. Мера MPdist [11], используемая для вычисления схожести подпоследовательностей при нахождении сниппетов, неформально определяется следующим образом. Два временны́х ряда равной длины m тем более похожи друг на друга в смысле меры MPdist, чем больше в каждом из них имеется подпоследовательностей заданной длины ℓ $(3 \leqslant \ell \leqslant m)$, близких друг к другу в смысле нормализованного евклидова расстояния. Мера MPdist устойчива к выбросам, шумам и пропущенным значениям во временно́м ряде, а также инвариантна к амплитуде, сдвигу и фазе временно́го ряда [11]. Отметим, что MPdist как симметричная неотрицательная функция является мерой, но не метрикой, поскольку для нее выполнены аксиомы тождества и симметрии, но не выполняется аксиома треугольника [11]. Формальное определение меры MPdist выглядит следующим образом.

Пусть имеются два временны́х ряда A и B равной длины $(|A| = |B| = m)$ и задан параметр ℓ $(3 \leqslant \ell \leqslant m)$, который мы будем называть значимой длиной подпоследовательности. Типичным диапазоном для выбора значимой длины подпоследовательности служит отрезок $[\lceil 0.3m \rceil, \lceil 0.8m \rceil]$ [11]. В дальнейшем изложении, упоминая параметр m в контексте вычисления меры MPdist, мы подразумеваем, что также задан параметр ℓ.

Далее в определении используется понятие матричного профиля временно́го ряда, предложенное Кеогом и др. в работе [16]. *Матричным профилем (matrix profile)* временны́х рядов A и B (с учетом значимой длины подпоследовательности ℓ) будем называть временно́й ряд, обозначаемый как P_{AB} и име-

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

ющий длину $m-\ell+1$, элементами которого являются нормализованные евклидовы расстояния от каждой подпоследовательности ряда A, имеющей длину ℓ, до ближайшей к ней в смысле указанного расстояния подпоследовательности ряда B:

$$P_{AB} = (d_1, \ldots, d_{m-\ell+1}), d_i = \mathrm{ED}_{\mathrm{norm}}(A_{i,\ell}, B_{j,\ell}), B_{j,\ell} = \arg \min_{1 \leqslant q \leqslant m-\ell+1} \mathrm{ED}_{\mathrm{norm}}(A_{i,\ell}, B_{q,\ell}). \tag{1}$$

Нормализованное евклидово расстояние между двумя подпоследовательностями представляет собой евклидово расстояние между этими подпоследовательностями, подвергнутыми z-нормализации:

$$\mathrm{ED}_{\mathrm{norm}}(X, Y) = \mathrm{ED}(\hat{X}, \hat{Y}) = \sqrt{\sum_{i=1}^{\ell} (\hat{x}_i - \hat{y}_i)^2}. \tag{2}$$

Z-нормализованная подпоследовательность $\hat{X}$ вычисляется из X следующим образом:

$$\hat{X} = (x_1, \ldots, x_\ell) : \ \hat{x}_i = \frac{x_i - \mu_x}{\sigma_x}, \quad \mu_x = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i, \quad \sigma_x = \sqrt{\frac{1}{\ell} \sum_{i=1}^{\ell} x_i^2 - \mu_x^2}. \tag{3}$$

Аналогичным образом определяется матричный профиль рассматриваемых рядов, взятых в порядке B и A, и обозначается как P_{BA}:

$$P_{BA} = (d_1, \ldots, d_{m-\ell+1}), \quad d_i = \mathrm{ED}_{\mathrm{norm}}(B_{i,\ell}, A_{j,\ell}), \quad A_{j,\ell} = \arg \min_{1 \leqslant q \leqslant m-\ell+1} \mathrm{ED}_{\mathrm{norm}}(B_{i,\ell}, A_{q,\ell}). \tag{4}$$

Выполним конкатенацию матричных профилей P_{AB} и P_{BA} и обозначим полученный временно́й ряд длины $2(m-\ell+1)$ как P_{ABBA} (здесь и далее для обозначения конкатенации будем использовать символ $\odot$):

$$P_{ABBA} = P_{AB} \odot P_{BA}. \tag{5}$$

Далее, обозначим за $\mathit{Sorted}\, P_{ABBA}$ ряд P_{ABBA}, в котором элементы упорядочены по возрастанию. Для вычисления схожести между рядами A и B в смысле меры MPdist используется k-е значение ряда $\mathit{Sorted}\, P_{ABBA}$, где k является параметром. Типичным значением указанного параметра является 5% от $2m$, суммарной длины рядов A и B [11]. Однако, если величина значимой длины подпоследовательности ℓ существенно близка к длине ряда m, то конкатенация матричных профилей P_{ABBA} имеет длину менее 5% от $2m$. В этом случае значением меры MPdist полагается максимальное значение элемента в конкатенации матричных профилей P_{ABBA}. Нижеследующая формула формализует приведенные рассуждения:

$$\mathrm{MPdist}(A, B, \ell) = \begin{cases} \mathit{Sorted}\, P_{ABBA}(k), & |P_{ABBA}| > k, \\ \mathit{Sorted}\, P_{ABBA}(2(m-\ell+1)), & |P_{ABBA}| \leqslant k, \end{cases} \\ k = \lceil 0.05 \cdot 2m \rceil = \lceil 0.1m \rceil. \tag{6}$$

4. Последовательный поиск сниппетов. Для дальнейшего изложения последовательного алгоритма поиска сниппетов SnippetFinder [1] будем использовать следующие определения и обозначения. *MPdist-профилем* временно́го ряда T длины n и временно́го ряда Q меньшей длины m ($m \ll n$) назовем вектор из $n-m+1$ элементов, каждый из которых представляет собой величину схожести соответствующей подпоследовательности ряда T, имеющей длину m, с рядом Q, также имеющим длину m, в смысле меры MPdist, и обозначим его как MPD:

$$\mathit{MPD}\,(Q, T, \ell) = (d_1, \ldots, d_{n-m+1}), \quad d_i = \mathrm{MPdist}(Q, T_{i,m}, \ell). \tag{7}$$

Обозначим MPdist-профиль ряда T и его сегмента S_i как D_i, тогда *набор MPdist-профилей* ряда T и всех его сегментов обозначим как D и определим следующим образом:

$$D = (D_1, \ldots, D_{n/m}), \quad D_i = \mathit{MPD}\,(S_i, T, \ell). \tag{8}$$

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

Поиск сниппетов связан с построением *кривой репрезентативности*, которая состоит из $n-m$ точек и обозначается как M. Параметром построения данной кривой является D_{subset}, заданное непустое подмножество набора MPdist-профилей. Кривая M представляет собой набор точек, в котором i-я точка показывает схожесть по мере MPdist между i-й подпоследовательностью ряда, имеющей длину m, и наиболее похожим на нее сегментом ряда, взятым из заданного подмножества сегментов:

$$M(D_{\text{subset}}) = (M_1, \ldots, M_{n-m+1}), \quad M_i = \min_{D_j \in D_{\text{subset}}} \{d_i \mid d_i \in D_j\}, \quad D_{\text{subset}} \subset D. \tag{9}$$

Площадь под кривой репрезентативности M обозначается как *ProfileArea* и рассматривается как целевая функция поиска сниппетов:

$$\mathit{ProfileArea}\,(D_{\text{subset}}) = \sum_{i=1}^{n-m} M_i(D_{\text{subset}}). \tag{10}$$

Площадь под кривой репрезентативности обладает следующим интуитивно понятным свойством: при выборе в качестве сниппетов всех сегментов ряда и, соответственно, построении кривой репрезентативности по всем сегментам ряда, $\mathit{ProfileArea} = 0$.

Поиск сниппетов подразумевает такой выбор сокращенного множества сегментов ряда в качестве сниппетов, при котором значение *ProfileArea* существенно приблизится к нулю. Поиск сниппетов реализуется как итеративный подбор множества сегментов ряда следующим образом. На первом шаге в качестве сниппета алгоритмом будет выбран сегмент, для которого значение *ProfileArea* минимально:

$$\mathit{step} = 1: \quad C_1.\mathit{index} = \arg \min_{1 \leqslant j \leqslant n/m} \mathit{ProfileArea}\,(\{D_j\}). \tag{11}$$

На каждом следующем шаге MPdist-профиль сегмента, выбранного в качестве сниппета на предыдущем шаге, используется для построения кривой репрезентативности:

$$\mathit{step} = 2: \quad C_2.\mathit{index} = \arg \min_{1 \leqslant j \leqslant n/m} \mathit{ProfileArea}\,(\{D_{C_1.\mathit{index}},\, D_j\}). \tag{12}$$

Все последующие шаги выполняются аналогично второму шагу алгоритма до тех пор, пока не будет найдено K сниппетов, где число сниппетов является параметром алгоритма:

$$\mathit{step} = i, \quad 3 \leqslant i \leqslant K: \quad C_i.\mathit{index} = \arg \min_{1 \leqslant j \leqslant n/m} \mathit{ProfileArea}\,(\{D_{C_1.\mathit{index}}, \ldots, D_{C_{i-1}.\mathit{index}},\, D_j\}). \tag{13}$$

После того, как сниппеты найдены, значимость каждого из них вычисляется следующим образом:

$$C_i.\mathit{frac} = \frac{|M(\{D_{C_1.\mathit{index}}, \ldots, D_{C_i.\mathit{index}}\}) \cap D_{C_i.\mathit{index}}|}{n-m+1}, \quad 1 \leqslant i \leqslant K. \tag{14}$$

Алгоритм 1 показывает псевдокод последовательного поиска сниппетов описанным выше способом. Алгоритм SnippetFinder начинает работу с вычисления набора MPdist-профилей всех сегментов ряда (строка 2). Вспомогательные алгоритмы GetAllProfiles и MPdistProfile для вычисления набора MPdist-профилей и вычисления MPdist-профиля отдельного сегмента приведены в алгоритмах 2 и 3 соответственно. Далее производится поиск сниппетов по формулам (11)–(13) (строки 3–11). Алгоритм завершает работу вычислением значимости найденных сниппетов по формуле (14) (строки 12–14).

5. Параллельный поиск сниппетов на графическом процессоре. Далее описан параллельный алгоритм поиска сниппетов для графических процессоров, который основан на описанном выше алгоритме SnippetFinder и получил название *PSF (Parallel Snippet Finder)*. В разделе 5.1 описана аппаратная платформа, в разделе 5.2 — структуры данных, принципы распараллеливания и реализации алгоритма.

5.1. Аппаратная платформа. В настоящее время *графический процессор (Graphics Processing Unit, GPU)* компании NVIDIA является одной из наиболее популярных платформ высокопроизводительных вычислений. GPU состоит из симметричных *потоковых мультипроцессоров (Streaming Multiprocessor, SM)*, каждый из которых, в свою очередь, содержит до тысячи симметричных *CUDA-ядер (Compute Unified Device Architecture, CUDA — вычислительная унифицированная архитектура устройства)* [17].

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

Параллельное приложение запускается на GPU как набор нитей, где каждая нить исполняется отдельным CUDA-ядром и предусмотрена следующая иерархия нитей. Верхним уровнем иерархии является *сетка нитей (grid)*, которая состоит из одномерного или двумерного массива симметричных блоков нитей. *Блок нитей (thread block)* представляет собой массив нитей, размерность которого может варьироваться от 1 до 3. Внутри блока нити разбиваются на *варпы (warp)* — группы по 32 нити. Нити варпа исполняются в режиме *SIMT (Single Instruction Multiple Threads)*: каждая нить выполняет одну и ту же инструкцию над назначенной ей частью разделяемых 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 блока. Для обмена данными между нитями разных блоков используется глобальная память GPU.

Параллельное приложение реализуется как набор вычислительных CUDA-ядер. *Вычислительное CUDA ядро (kernel)* представляет собой функцию языка программирования *C*, которая исполняется на GPU. Запуск вычислительных ядер осуществляется с центрального процессора (называемого *хостом*). Помимо параметров подпрограммы для вычислительного ядра также указываются параметры его запуска на графическом процессоре: используемое количество блоков, нитей и объем разделяемой памяти.

Алгоритм 1. SnippetFinder (in T, m, K; out C_T^m)
Algorithm 1. SnippetFinder (in T, m, K; out C_T^m)

1: $C_T^m \leftarrow \emptyset$; $M \leftarrow \overline{+\infty}$
2: $D \leftarrow \text{GetAllProfiles}\,(T, m)$
3: **while** $|C_T^m| \neq K$ **do**
4: $\quad minArea \leftarrow +\infty$
5: $\quad$ **for** $i \leftarrow 1$ **to** n/m **do**
6: $\quad\quad ProfileArea \leftarrow \sum_{j=1}^{n-m} \min(D_i(j), M_j)$
7: $\quad\quad$ **if** $ProfileArea < minArea$ **then**
8: $\quad\quad\quad minArea \leftarrow ProfileArea$; $idx \leftarrow i$
9: $\quad M \leftarrow (\min(D_{idx}(1), M_1), \ldots, \min(D_{idx}(n-m), M_{n-m})$
10: $\quad C \leftarrow T_{m\cdot(idx-1)+1,\,m}$; $C.index \leftarrow idx$
11: $\quad C_T^m \leftarrow C_T^m \cup C$
12: **for** $i \leftarrow 1$ **to** K **do**
13: $\quad f \leftarrow |\{neighbor \in D_{C_i.index} \mid neighbor = M_i\}|$
14: $\quad C_i.frac \leftarrow f/(n-m+1)$
15: **return** C_T^m

Алгоритм 2. GetAllProfiles (in T, m; out D)
Algorithm 2. GetAllProfiles (in T, m; out D)

1: $D \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ **to** n/m **do**
3: $\quad D_i \leftarrow \text{MPdistProfile}\,(T, T_{m\cdot(i-1)+1,\,m})$
4: $\quad D \leftarrow D \cup D_i$
5: **return** D

Алгоритм 3. MPdistProfile (in T, Q; out MPD)
Algorithm 3. MPdistProfile (in T, Q; out MPD)

1: $MPD \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ **to** $n-\ell$ **do**
3: $\quad d_i \leftarrow \text{MPdist}(T_{i,\,m}, Q, \ell)$
4: $\quad MPD \leftarrow MPD \cup d_i$
5: **return** MPD

5.2. Структуры данных, принципы распараллеливания и реализация. Вычислительная схема алгоритма PSF имеет следующие отличия от оригинального алгоритма SnippetFinder. Вычисление набора MPdist-профилей (алгоритм 1, строка 2) выполняется более эффективно, чем в оригинальном вспомогательном алгоритме GetAllProfiles (алгоритм 2). Вместо одного последовательного шага, на котором выполняется вычисление MPdist-профиля между сегментом и каждой подпоследовательностью исходного ряда, мы можем выполнить последовательность из следующих шагов, каждый из которых может быть выполнен параллельно. Опишем указанную последовательность шагов для фиксированного сегмента ряда $S \in S_T^m$.

Структуры данных алгоритма PSF представлены на рис. 2. Ключевой для распараллеливания структурой данных является матрица ED_{norm}-расстояний между каждой подпоследовательностью длины ℓ сегмента S и каждой подпоследовательностью длины ℓ исходного ряда. Обозначим указанную матрицу за ED_{matr}:

$$ED_{\text{matr}} \in \mathbb{R}^{(m-\ell+1)\times(n-\ell+1)}: \quad ED_{\text{matr}}(i,j) = \text{ED}_{\text{norm}}(S_{i,\,\ell}, T_{j,\,\ell}).$$

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

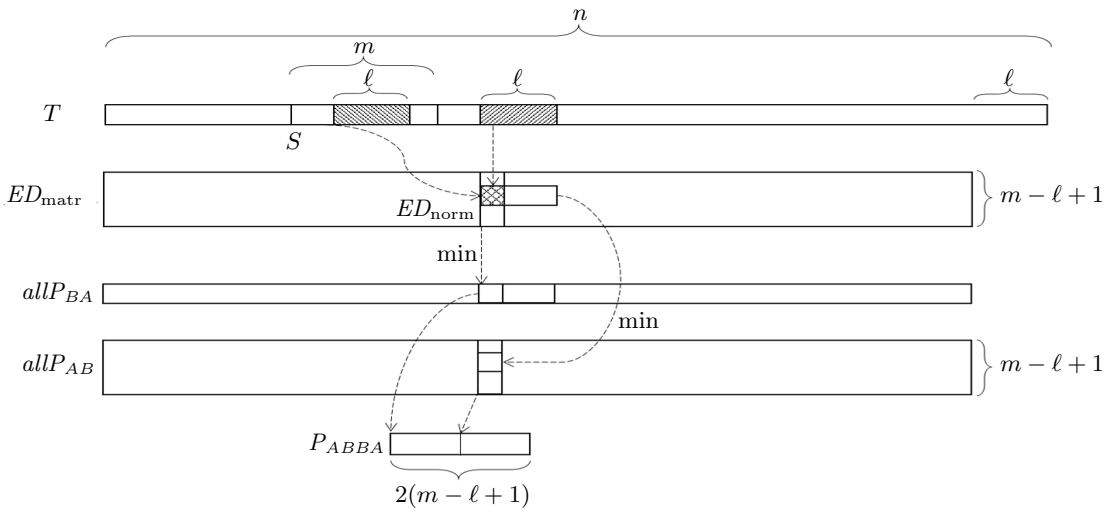

Рис. 2. Структуры данных алгоритма PSF

Fig. 2. PSF data structures

На втором шаге в каждом столбце матрицы ED_{matr}, полученной на первом шаге, находится минимум. Обозначим вектор таких минимумов за $allP_{BA}$:

$$allP_{BA} \in \mathbb{R}^{n-\ell+1}: \quad allP_{BA}(j) = \min_{1 \leqslant i \leqslant m-\ell+1} ED_{\text{matr}}(i,j). \tag{15}$$

На третьем шаге в каждой строке ED_{matr} выполняется поиск минимумов в скользящем окне длины ℓ. Обозначим матрицу таких минимумов за $allP_{AB}$:

$$allP_{AB} \in \mathbb{R}^{(m-\ell+1)\times(n-\ell+1)}: \quad allP_{AB}(i,j) = \min_{j \leqslant c \leqslant j+m-\ell+1} ED_{\text{matr}}(i,c).$$

На четвертом шаге для каждой подпоследовательности ряда, имеющей длину ℓ, и сегмента S выполняется построение матричного профиля. Для построения одного матричного профиля выполняется сцепление соответствующих данной подпоследовательности столбца матрицы $allP_{AB}$ и подпоследовательности длины $m-\ell+1$, входящей в вектор $allP_{BA}$. Результат сцепления обозначим как вектор P_{ABBA}:

$$P_{ABBA} \in \mathbb{R}^{2(m-\ell+1)}: \; P_{ABBA}(T_{j,\ell}) = allP_{AB}(1,j) \odot \ldots \odot allP_{AB}(m-\ell+1,j) \odot \\ \odot \; allP_{BA}(j) \odot \ldots \odot allP_{BA}(m-\ell+1), \tag{16}$$

где $1 \leqslant j \leqslant n-\ell+1$. Для финального вычисления меры схожести MPdist между сегментом и подпоследовательностью необходимо выполнить сортировку P_{ABBA} и взять k-е значение упорядоченного массива, как определено в формуле (6).

Вспомогательный алгоритм 4 ParallelGetAllProfiles реализует описанные выше шаги параллельно. Циклический вызов алгоритма EDmatrSCAMP (строка 3) обеспечивает параллельное вычисление матрицы ED_{norm}-расстояний между заданным сегментом и каждой подпоследовательностью ряда. Параллелизм вычислений реализован на основе следующей техники, предложенной в работе [16]. Сначала вычисляется матрица центрированных сумм произведений значений ряда, обозначаемая как $\overline{QT}$:

$$\overline{QT}_{i,\,j} = \overline{QT}_{i-1,\,j-1} + df_i \cdot dg_j + df_j \cdot dg_i,$$

где df и dg – слагаемые центрированной суммы произведений:

$$df_0 = 0,\; df_i = \tfrac{1}{2}(t_{i+m-1} - t_{i-1}), \\ dg_0 = 0,\; dg_i = (t_{i+m-1} - \mu_i) + (t_{i-1} - \mu_{i-1}),\; \mu_i = \frac{1}{m}\sum_{j=i}^{i+m} t_j. \tag{17}$$

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

Алгоритм 4. ParallelGetAllProfiles (in T, m; out D)
Algorithm 4. ParallelGetAllProfiles (in T, m; out D)

1: $D \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ **to** n/m **do**
3: $\quad ED_{\text{matr}} \leftarrow \text{EDmatrSCAMP}\,(T, S_i, \ell)$
4: $\quad$ **for** $j \leftarrow 1$ **to** $n - \ell$ **do** ▷ PARALLEL
5: $\quad\quad allP_{BA}(j) \leftarrow \min_{1 \leqslant r \leqslant m-\ell+1} ED_{\text{matr}}(r, j)$
6: **for** $r \leftarrow 1$ **to** $m - \ell$ **do** ▷ PARALLEL
7: $\quad$ **for** $q \leftarrow 1$ **to** $n - m + 1$ **do**
8: $\quad\quad allP_{AB}(r, q) \leftarrow \min_{q \leqslant p \leqslant q+\ell} ED_{\text{matr}}(r, p)$
9: $D_i \leftarrow \text{ParallelProfile}\,(allP_{AB}, allP_{BA})$
10: $D \leftarrow D \cup D_i$
11: **return** D

Далее полученные результаты используются для вычисления корреляции по Пирсону между подпоследовательностями ряда, обозначаемой как $P_{i,\,j}$:

$$P_{i,j} = \overline{QT}_{i,j} \cdot \frac{1}{\| T_{i,m} - \mu_i \|} \cdot \frac{1}{\| T_{j,m} - \mu_j \|},$$

где $T_{i,m} - \mu_i = (t_i - \mu_i, \ldots, t_{i+m-1} - \mu_i)$, $T_{j,m} - \mu_j = (t_j - \mu_j, \ldots, t_{j+m-1} - \mu_j)$ и $\| \cdot \|$ означает Евклидову норму вектора.

Наконец, значение корреляции по Пирсону между двумя подпоследовательностями ряда можно преобразовать в z-нормализованное евклидово расстояние между ними следующим образом:

$$\text{ED}_{\text{norm}}(T_{i,m}, T_{j,m}) = \sqrt{2m(1 - P_{i,j})}.$$

Описанная техника использует меньше вычислительных ресурсов по сравнению с прямолинейным выполнением z-нормализации подпоследовательностей и вычислением евклидова расстояния между ними.

Строки 4–5 алгоритма реализуют в соответствии с формулой (15) параллельное вычисление вектора $allP_{BA}$, содержащего минимумы по столбцам ED_{matr}. Соответствующее вычислительное ядро организуется следующим образом. Формируется сетка нитей, состоящая из $n - m + 1$ блоков по $m - \ell + 1$ нитей в каждом блоке. Из глобальной памяти в разделяемую память каждого блока копируется один столбец матрицы ED_{matr}. Далее каждый блок выполняет поиск минимума посредством операции свертки.

Строки 6–8 реализуют в соответствии с формулой (15) параллельное вычисление матрицы $allP_{AB}$, содержащей построчные минимумы ED_{matr} в скользящих окнах длины ℓ. Соответствующее вычислительное ядро организуется следующим образом. Формируется сетка нитей, состоящая из одного блока с $m - \ell + 1$ нитями. Каждая нить блока выполняет вычисление минимума в скользящем окне длины ℓ для одной строки матрицы. Результирующая матрица данного шага записывается в глобальную память.

Далее вычисления продолжаются с помощью вспомогательного параллельного алгоритма ParallelMPdistProfile (см. алгоритм 5). В соответствии с формулой (16) выполняется параллельная конкатенация каждого столбца матрицы $allP_{AB}$ и всех подпоследовательностей длины $m-\ell$, входящих в вектор $allP_{BA}$. Соответствующее вычислительное ядро организуется следующим образом. Формируется сетка нитей, состоящая из $n-m+1$ блоков по $2(m-\ell+1)$ нитей в каждом блоке. Каждый блок выполняет параллельное формирование матричного профиля P_{ABBA} для одного сегмента S. Половина нитей данного блока копирует из глобальной памяти в разделяемую память этого блока данные $allP_{AB}$, другая половина нитей копирует из глобальной памяти в разделяемую память этого блока данные $allP_{BA}$ для каждого столбца матрицы расстояний. Далее каждый блок производит сортировку P_{ABBA} и записывает его k-е значение (см. формулу (6)) в глобальную память, формируя таким образом MPdist-профиль сегмента.

Распараллеливанию также подвергаются вычисление площади под кривой репрезентативности (алгоритм 1, строки 5–11) и вычисление значимости найденных сниппетов (алгоритм 1, строки 12–14). Для вычисления кривой используется сетка нитей, состоящая из n/m блоков по $n - m + 1$ нитей в каждом блоке. Каждый блок параллельно вычисляет минимальные значения кривой репрезентативности. Далее осуществляется суммирование значений кривой репрезентативности посредством операции свертки. После

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

Алгоритм 5. ParallelProfile (in $allP_{AB}$, $allP_{BA}$; out P)
Algorithm 5. ParallelProfile (in $allP_{AB}$, $allP_{BA}$; out P)

```
1: P ← ∅; k ← ⌈0.1 · (m − ℓ)⌉
2: for i ← 1 to n − ℓ do                                   ▷ PARALLEL
3:     P_ABBA ← allP_AB(i, m − ℓ) ⊙ allP_BA(i)
4:     SortedP_ABBA ← Sort(P_ABBA)
5:     P_i ← SortedP_ABBA(k)
6:     P ← P ∪ P_i
7: return P
```

того как найдено необходимое количество сниппетов, формируется сетка нитей, состоящая из K блоков по $n-m+1$ нитей в каждом блоке. Указанная сетка путем сравнения значений MPdist-профилей сниппетов и кривой репрезентативности вычисляет значимость каждого сниппета.

Описанный выше подход реализован на языке программирования $C++$ с применением технологии CUDA. Исходные коды реализации свободно доступны в сети Интернет по адресу https://github.com/Andru7428/CUDASnippetFinder.

6. Вычислительные эксперименты. Для исследования эффективности разработанного параллельного алгоритма PSF нами были проведены вычислительные эксперименты. В экспериментах исследовалась производительность алгоритма при обработке временны́х рядов, резюмированных в табл. 1 для указанной длины сегмента m и длины значимой подпоследовательности $\ell = \lceil m/2 \rceil$. Временны́е ряды GreatBarbet, WildVTrainedBird, WalkRun и TiltABP взяты из набора MixedBag [1], использованного для исследования эффективности оригинального последовательного алгоритма. В данном наборе каждый временно́й ряд имеет два предопределенных сниппета, которые соответствуют двум предопределенным активностям, и имеет место однократная смена указанных активностей. Временно́й ряд PAMAP3 представляет собой показания носимого акселерометра во время трех различных видов физической активности человека (ходьба, глажка белья, положение стоя) и является фрагментом стандартизированного временно́го ряда PAMAP [15] .

В экспериментах сравнивалась производительность предложенного параллельного алгоритма PSF, оригинального последовательного алгоритма SnippetFinder и параллельного алгоритма NaivePSF. При этом количество сниппетов соответствовало числу активностей, отражаемых временны́м рядом (параметр $K = 2$ для временны́х рядов, взятых из набора MixedBag, и $K = 3$ для временно́го ряда PAMAP3).

Алгоритм *NaivePSF* представляет собой упрощенную версию PSF, в которой по формулам (1)–(5) вычисляются матричные профили между сегментами и всеми подпоследовательностями ряда при помощи отдельных вызовов фреймворка SCAMP [18]. Далее последовательно на центральном процессоре по формулам (6)–(8) вычисляются MPdist-профили всех сегментов и производится поиск сниппетов. Данный

Таблица 1. Временны́е ряды, задействованные в экспериментах
Table 1. Time series used in experiments

Название Name	Длина ряда n Time series length n	Длина сегмента m Segment length m	Описание Description
GreatBarbet	2 801	150	Физиологические показатели жизнедеятельности птиц Physiological indicators of the vital functions of birds
WildVTrainedBird	20 002	900	
PAMAP3	20 002	600	Показания носимого акселерометра во время различных видов физической активности человека Wearable accelerometer readings during various types of physical activity of a person
WalkRun	100 000	240	
TiltABP	40 000	630	Показания кровяного давления человека во время быстрых наклонов Human blood pressure readings during rapid tilts

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

подход приводит к избыточным вычислениям. В результате работы фреймворка SCAMP производятся повторные вычисления евклидовых расстояний между подпоследовательностями (1) и поиск минимумов матриц расстояний (4).

Аппаратная платформа экспериментов резюмирована в табл. 2. В экспериментах последовательный алгоритм SnippetFinder выполняется на центральном процессоре (на одном ядре). При выполнении параллельных алгоритмов PSF и NaivePSF предварительные вычисления средних значений подпоследовательностей (17) выполняются на центральном процессоре, остальные вычисления (1)–(14) выполняются на графическом процессоре.

В экспериментах алгоритмы PSF и NaivePSF выдают результаты, идентичные результатам работы оригинального алгоритма SnippetFinder. На рис. 3 приведены примеры результатов работы алгоритма PSF для временны́х рядов, задействованных в экспериментах.

Результаты экспериментов представлены на рис. 4. Можно видеть, что параллельный алгоритм PSF уступает оригинальному последовательному алгоритму только в случае GreatBarbet, самого короткого временно́го ряда из рассмотренных. Это ожидаемый результат, поскольку в случае временно́го ряда относительно небольшой длины (примерно до десятка тысяч элементов) накладные расходы на передачу данных на GPU и инициализацию вычислительных ядер будут больше времени, затраченного на собственно вычисления. В остальных случаях PSF опережает оригинальный последовательный алгоритм минимум на порядок.

В случае временно́го ряда относительно небольшой длины (примерно до десятка тысяч элементов) PSF показывает практически одинаковое быстродействие, что и NaivePSF, поскольку накладные расходы данных алгоритмов одинаковы, а для короткого временно́го ряда избыточные вычисления в наивной версии параллельного алгоритма практически отсутствуют. Для временны́х рядов длины более десятка тысяч элементов PSF быстрее, чем NaivePSF, в 2–4 раза. Преимущество алгоритма PSF тем больше, чем больше длина исследуемого временно́го ряда, поскольку накладные расходы на вычисление матричных профилей между сегментами и подпоследовательностями ряда становятся более существенными.

7. Заключение. В статье рассмотрена проблема поиска типичных подпоследовательностей временно́го ряда, предполагающая нахождение набора подпоследовательностей ряда, которые достоверно отражают течение процесса или явления, задаваемого данным рядом. На практике найденные типичные подпоследовательности используются для резюмирования и визуализации больших временны́х рядов, а также для разработки моделей машинного обучения и искусственных нейронных сетей, решающих задачи анализа и прогнозирования больших временны́х рядов в различных предметных областях. В исследовании применяется концепция сниппетов временно́го ряда, предложенная Кеогом (Keogh) и др. в работе [1]. Сниппет представляет собой подпоследовательность заданной длины, на которую похожи многие другие подпоследовательности данного ряда в смысле меры схожести MPdist [11]. Набор сниппетов имеет существенно меньшую мощность, чем множество подпоследовательностей ряда, имеющих заданную длину, и потому используется для резюмирования исходного временно́го ряда. По сравнению с альтернативными подходами к решению задачи поиска типичных подпоследовательностей временно́го ряда

Таблица 2. Аппаратная платформа экспериментов
Table 2. Hardware platform of experiments

Характеристика Characteristic	Центральный процессор Central processing unit	Графический процессор Graphics processing unit
Модель Model	Intel Xeon Gold 6254	NVIDIA Tesla V100 SXM2
Количество ядер Number of Cores	18	5120
Тактовая частота ядра, ГГц Core clock, GHz	4.0	1.3
Оперативная память, ГБ RAM size, GB	64	32
Пиковая производительность, TFlops Peak performance, TFlops	1.2	15.7

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

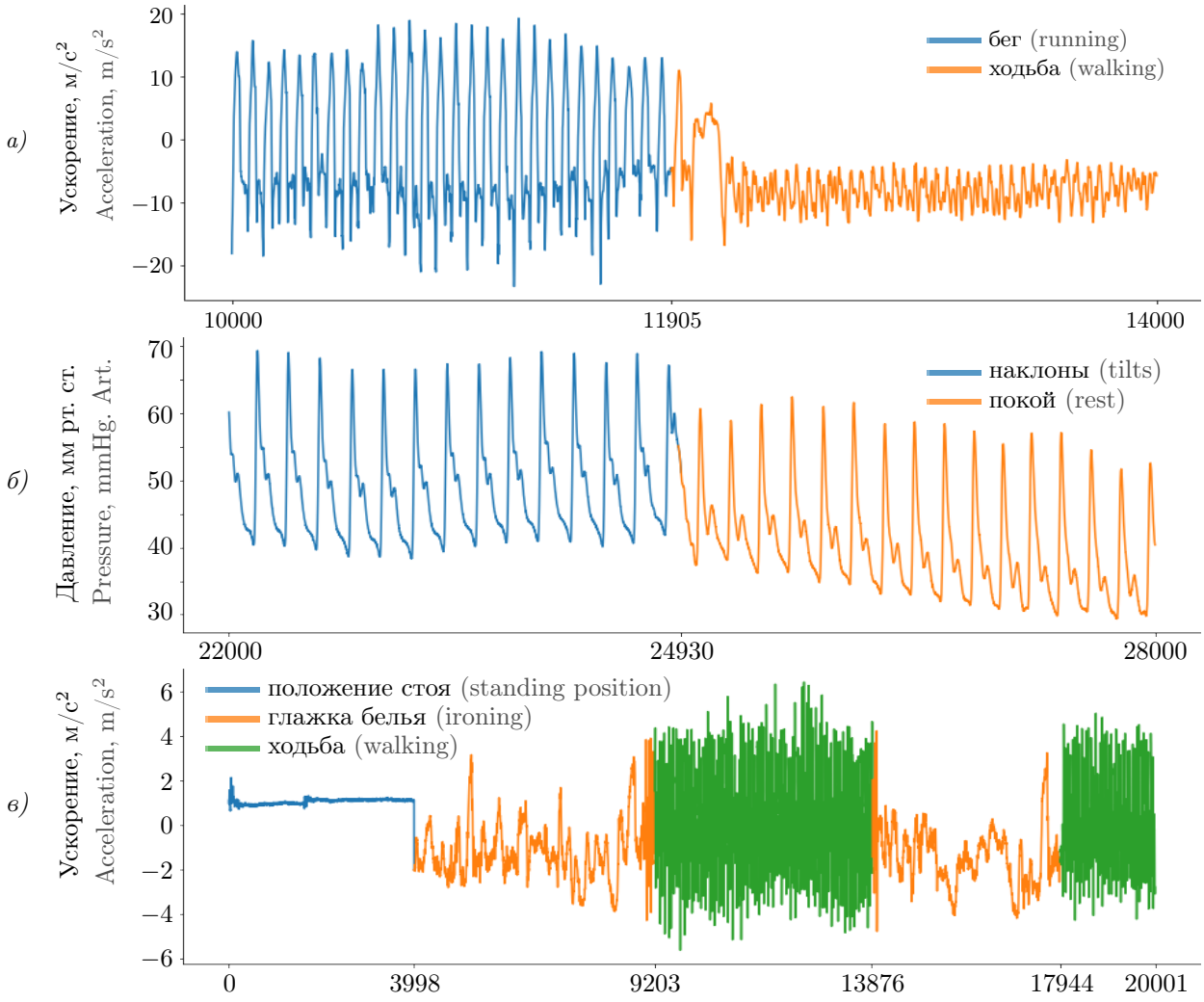

Рис. 3. Примеры результатов работы алгоритма PSF. Временны́е ряды:
а) WalkRun; *б)* TiltABP; *в)* PAMAP3

Fig. 3. Examples of the results of the PSF algorithm. Time series:
а) WalkRun; *б)* TiltABP; *в)* PAMAP3

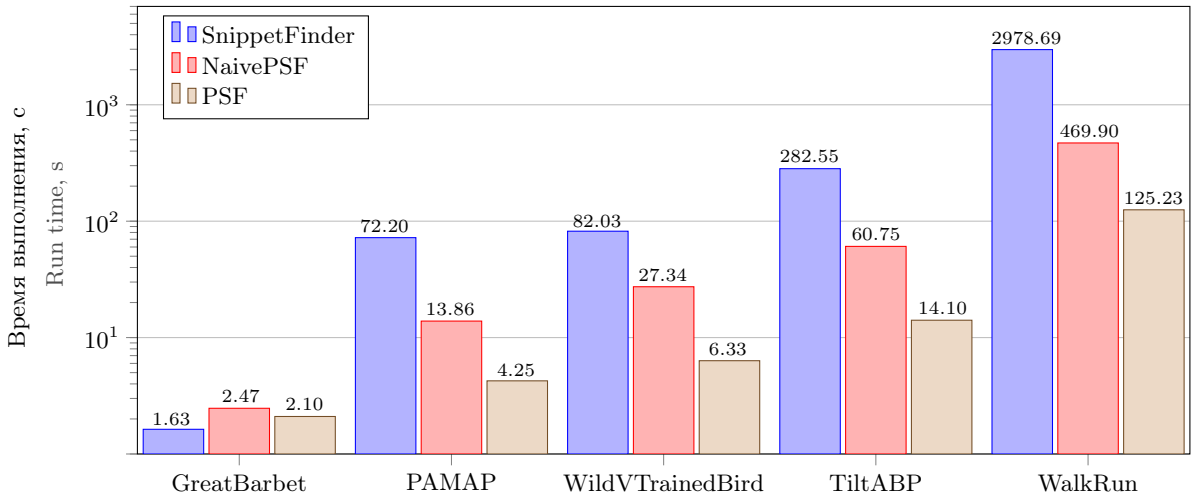

Рис. 4. Производительность алгоритма PSF в сравнении с аналогами

Fig. 4. Performance of the PSF algorithm in comparison with analogues

2021, **22** (4), 344–359. doi 10.26089/NumMet.v22r423

сниппеты обеспечивают более адекватные резюмирование и визуализацию временны́х рядов из широкого спектра предметных областей [1]. Алгоритм SnippetFinder, предложенный авторами концепции сниппетов, однако, имеет высокую вычислительную сложность.

Авторами настоящего исследования предложен новый параллельный алгоритм поиска сниппетов временно́го ряда для графического процессора, названный PSF (Parallel SnippetFinder). В алгоритме PSF вычисление схожести всех подпоследовательностей ряда со сниппетом в смысле меры MPdist выполняется более эффективно, чем в оригинальном алгоритме: указанные вычисления разбиты на несколько шагов, каждый из которых выполняется параллельно. Представлена схема реализации параллельного алгоритма с помощью технологии CUDA. Проведены вычислительные эксперименты с реальными временны́ми рядами из различных предметных областей. В случае временно́го ряда относительно небольшой длины (примерно до десятка тысяч элементов) PSF показывает слабое быстродействие и может незначительно уступать оригинальному последовательному алгоритму, поскольку накладные расходы на передачу данных на графический процессор и инициализацию вычислительных ядер будут больше времени, затраченного на собственно вычисления. Для временны́х рядов с длиной от десятка тысяч элементов PSF опережает оригинальный последовательный алгоритм минимум на порядок. Преимущество алгоритма PSF тем больше, чем больше длина исследуемого временно́го ряда, поскольку временны́е затраты на вычисления становятся более существенными.

В качестве возможного направления будущих исследований мы рассматриваем разработку параллельного алгоритма поиска сниппетов большого временно́го ряда (который не может быть целиком размещен в оперативной памяти) на высокопроизводительном кластере с вычислительными узлами на базе графических процессоров.

Список литературы

1. *Imani S., Madrid F., Ding W., et al.* Matrix profile XIII: time series snippets: a new primitive for time series data mining // Proc. of the 9th IEEE International Conference on Big Knowledge (ICBK). Singapore, 2018. doi 10.1109/ICBK.2018.00058.
2. *Kumar S., Tiwari P., Zymbler M.* Internet of things is a revolutionary approach for future technology enhancement: a review // Journal of Big Data. 2019. **6**. doi 10.1186/s40537-019-0268-2.
3. *Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др.* Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. **10**, № 3. 16–36. doi 10.14529/cmse210302.
4. *Иванов С.А., Никольская К.Ю., Радченко Г.И. и др.* Концепция построения цифрового двойника города // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. **9**, № 4. 5–23. doi 10.14529/cmse200401.
5. *Епишев В.В., Исаев А.П., Миниахметов Р.М. и др.* Система интеллектуального анализа данных физиологических исследований в спорте высших достижений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. **2**, № 1. 44–54. doi 10.14529/cmse130105.
6. *Абдуллаев С.М., Ленская О.Ю., Гаязова А.О. и др.* Алгоритмы краткосрочного прогноза с использованием радиолокационных данных: оценка трасляции и композиционный дисплей жизненного цикла // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. **3**, № 1. 17–32. doi 10.14529/cmse140102.
7. *Дышаев М.М., Соколинская И.М.* Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. **2**, № 4. 103–108. doi 10.14529/cmse130408.
8. *Mueen A., Keogh E., Zhu Q., et al.* Exact discovery of time series motifs // Proc. of the 2009 SIAM International Conference on Data Mining (SDM). Sparks, USA, 2009. doi 10.1137/1.9781611972795.41.
9. *Ye L., Keogh E.J.* Time series shapelets: a new primitive for data mining // Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Paris, France, 2009. doi 10.1145/1557019.1557122.
10. *Indyk P., Koudas N., Muthukrishnan S.* Identifying representative trends in massive time series data sets using sketches // Proc. of the 26th International Conference on Very Large Data Bases (VLDB). Cairo, Egypt, 2000. https://www.vldb.org/conf/2000/P363.pdf. Cited December 18, 2021.
11. *Gharghabi S., Imani S., Bagnall A., et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments // Data Mining and Knowledge Discovery. 2020. **34**. 1104–1135. doi 10.1007/s10618-020-00695-8.

12. *Zymbler M.L., Kraeva Ya.A.* Discovery of time series motifs on Intel many-core systems // Lobachevskii Journal of Mathematics. 2019. **40**, № 12. 2124–2132. doi 10.1134/S199508021912014X.
13. *Цымблер М.Л., Краева Я.А.* Параллельный алгоритм поиска лейтмотивов временного ряда для графического процессора // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. **9**, № 3. 17–34. doi 10.14529/cmse200302.
14. *Keogh E., Lin J.* Clustering of time-series subsequences is meaningless: implications for previous and future research // Knowledge and Information Systems. 2005. **8**. 154–177. doi 10.1007/s10115-004-0172-7.
15. *Reiss A., Stricker D.* Introducing a new benchmarked dataset for activity monitoring // Proc. of the 16th International Symposium on Wearable Computers (ISWC). Newcastle, United Kingdom, 2012. doi 10.1109/ISWC.2012.13.
16. *Yeh C.-C.M., Zhu Y., Ulanova L., et al.* Matrix profile I: all pairs similarity joins for time Series: a unifying view that includes motifs, discords and shapelets // Proc. of the IEEE 16th International Conference on Data Mining (ICDM). Barcelona, Spain, 2016. doi 10.1109/ICDM.2016.0179.
17. *Kirk D.* NVIDIA CUDA software and GPU parallel computing architecture // Proc. of the 6th International Symposium on Memory Management (ISMM). Montreal, Canada, 2007. doi 10.1145/1296907.1296909.
18. *Zimmerman Z., Kamgar K., Senobari N.S., et al.* Matrix profile XIV: scaling time series motif discovery with GPUs to break a quintillion pairwise comparisons a day and beyond // Proc. of the ACM Symposium on Cloud Computing (SoCC). Santa Cruz, USA, 2019. doi 10.1145/3357223.3362721.

Поступила в редакцию
24 ноября 2021 г.

Принята к публикации
17 декабря 2021 г.

Информация об авторах

Михаил Леонидович Цымблер — д.ф.-м.н., доцент, заместитель директора Научно-образовательного центра “Искусственный интеллект и квантовые технологии”, Южно-Уральский государственный университет (национальный исследовательский университет), пр-т им. В. И. Ленина, д. 76, 454080, Челябинск, Российская Федерация.

Андрей Игоревич Гоглачев — программист отдела интеллектуального анализа данных и виртуализации Лаборатории суперкомпьютерного моделирования, Южно-Уральский государственный университет (национальный исследовательский университет), пр-т им. В. И. Ленина, д. 76, 454080, Челябинск, Российская Федерация.

References

1. S. Imani, F. Madrid, W. Ding, et al., “Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining,” in *Proc. 9th IEEE Int. Conf. on Big Knowledge (ICBK). Singapore, November 17–18, 2018*, doi 10.1109/ICBK.2018.00058.
2. S. Kumar, P. Tiwari, and M. Zymbler, “Internet of Things is a Revolutionary Approach for Future Technology Enhancement: A Review,” J. Big Data **6** (2019). doi 10.1186/s40537-019-0268-2.
3. M. L. Zymbler, Ya. A. Kraeva, E. A. Latypova, et al., “Cleaning Sensor Data in Intelligent Heating Control System,” Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **10** (3), 16–36 (2021). doi 10.14529/cmse210302.
4. S. A. Ivanov, K. Yu. Nikolskaya, G. I. Radchenko, et al., “Digital Twin of a City: Concept Overview,” Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **9** (4), 5–23 (2020). doi 10.14529/cmse200401.
5. V. V. Epishev, A. P. Isaev, R. M. Miniakhmetov, et al., “Physiological Data Mining System for Elite Sports,” Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **2** (1), 44–54 (2013). doi 10.14529/cmse130105.
6. S. M. Abdullaev, O. U. Lenskaya, A. O. Gayazova, et al., “Short-Range Forecasting Algorithms Using Radar Data: Translation Estimate and Life-Cycle Composite Display,” Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **3** (1), 17–32 (2014). doi 10.14529/cmse140102.
7. M. M. Dyshaev and I. M. Sokolinskaya, “Representation of Trading Signals Based on Kaufman Adaptive Moving Average as a System of Linear Inequalities,” Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **2** (4), 103–108 (2013). doi 10.14529/cmse130408.

8. A. Mueen, E. J. Keogh, Q. Zhu, et al., "Exact Discovery of Time Series Motifs," in *Proc. 2009 SIAM Int. Conf. on Data Mining (SDM), Sparks, USA, April 30–May 2, 2009*, doi 10.1137/1.9781611972795.41.
9. L. Ye and E. J. Keogh, "Time Series Shapelets: a New Primitive for Data Mining," in *Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Paris, France, June 28–July 1, 2009*, doi 10.1145/1557019.1557122.
10. P. Indyk, N. Koudas, and S. Muthukrishnan, "Identifying Representative Trends in Massive Time Series Data Sets Using Sketches," in *Proc. 26th Int. Conf. on Very Large Data Bases (VLDB), Cairo, Egypt, September 10–14, 2000*, https://www.vldb.org/conf/2000/P363.pdf. Cited December 18, 2021.
11. S. Gharghabi, S. Imani, A. Bagnall, et al., "An Ultra-fast Time Series Distance Measure to Allow Data Mining in More Complex Real-world Deployments," Data Min. Knowl. Discov. **34**, 1104–1135 (2020). doi 10.1007/s10618-020-00695-8.
12. M. L. Zymbler and Ya. A. Kraeva, "Discovery of Time Series Motifs on Intel Many-Core Systems," Lobachevskii J. Math. **40** (12), 2124–2132 (2019). doi 10.1134/S199508021912014X.
13. M. L. Zymbler and Ya. A. Kraeva, "Parallel Algorithm for Time Series Motif Discovery on Graphic Processor," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **9** (3), 17–34 (2020). doi 10.14529/cmse200302.
14. E. Keogh and J. Lin, "Clustering of Time-Series Subsequences is Meaningless: Implications for Previous and Future Research," Knowl. Inf. Syst. **8**, 154–177 (2005). doi 10.1007/s10115-004-0172-7.
15. A. Reiss and D. Stricker, "Introducing a New Benchmarked Dataset for Activity Monitoring," in *Proc. 16th Int. Symposium on Wearable Computers (ISWC), Newcastle, United Kingdom, June 18–22, 2012*, doi 10.1109/ISWC.2012.13.
16. C.-C. M. Yeh, Y. Zhu, L. Ulanova, et al., "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets," in *Proc. IEEE 16th Int. Conf. on Data Mining (ICDM), Barcelona, Spain, December 12–15, 2016*, doi 10.1109/ICDM.2016.0179.
17. D. Kirk, "NVIDIA CUDA Software and GPU Parallel Computing Architecture," in *Proc. 6th Int. Symposium on Memory Management (ISMM'07), Montreal, Canada, October 21–22, 2007*, doi 10.1145/1296907.1296909.
18. Z. Zimmerman, K. Kamgar, N. S. Senobari, et al., "Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond," in *Proc. ACM Symposium on Cloud Computing (SoCC'19), Santa Cruz, USA, November 20–23, 2019*, doi 10.1145/3357223.3362721.

Received
November 24, 2021

Accepted for publication
December 17, 2021

Information about the authors

Mikhail L. Zymbler — Dr. Sci., Associate Professor, Deputy Director of the Scientific and Educational Center "Artificial Intelligence and Quantum Technologies", South Ural State University (National Research University), Lenin prospekt 76, 454080, Chelyabinsk, Russia.

Andrey I. Goglachev — Programmer of the Data Mining and Virtualization Department of the Supercomputer Simulation Laboratory, South Ural State University (National Research University), Lenin prospekt 76, 454080, Chelyabinsk, Russia.