УДК 004.272.25, 004.421, 004.032.24 DOI: 10.14529/cmse220305

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ВОССТАНОВЛЕНИЯ СЕНСОРНЫХ ДАННЫХ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ ДЛЯ МНОГОЯДЕРНОГО ПРОЦЕССОРА

© 2022 М.Л. Цымблер[1], А.Н. Полуянов[2], Я.А. Краева[1]

[1]*Южно-Уральский государственный университет*
(454080, Челябинск, пр. им. В.И. Ленина, д. 76),
[2]*Институт математики им. С.Л. Соболева СО РАН (644043 , Омск, ул. Певцова, д. 13)*
E-mail: mzym@susu.ru, andrey.poluyanov@gmail.com, kraevaya@susu.ru

Поступила в редакцию: 30.07.2022

В настоящее время во многих предметных областях обработка сенсорных данных в режиме реального времени связана с необходимостью синтеза значения соответствующего временного ряда, которое было пропущено ввиду технического сбоя или человеческого фактора. В данной статье предлагается параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени для многоядерного процессора. Алгоритм использует набор опорных временных рядов, которые имеют семантическую связь с исходным рядом. Алгоритм применяет следующую эвристику: если в опорных рядах имеют место повторяющиеся (схожие) подпоследовательности, то в ряде, содержащем пропущенное значение, повторяющиеся подпоследовательности возникают в тех же временны́х интервалах. Образцами поиска для каждого опорного ряда полагаются подпоследовательности заданной длины, оканчивающиеся в момент пропуска значения в исходном ряде. Схожесть подпоследовательностей с образцом определяется на основе меры DTW (Dynamic Time Warping), имеющей квадратичную вычислительную сложность относительно длины подпоследовательности. Применяется техника нижних границ схожести, позволяющая отбрасывать подпоследовательности, заведомо непохожие на образец, без вычисления DTW. Нижние границы имеют меньшую, чем у DTW сложность, и вычисляются параллельно. Восстановленное значение вычисляется как среднее арифметическое последних элементов найденных интервалов. В вычислительных экспериментах предложенный алгоритм демонстрирует высокую точность восстановления в сравнении с аналогами и быстродействие, приемлемое для применения алгоритма в режиме реального времени.

Ключевые слова: временной ряд, восстановление пропущенных значений, параллельный алгоритм, многоядерный процессор, DTW, отбрасывание по нижним границам.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Цымблер М.Л., Полуянов А.Н., Краева Я.А. Параллельный алгоритм восстановления сенсорных данных в режиме реального времени для многоядерного процессора // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 69–90. DOI: 10.14529/cmse220305.

Введение

В настоящее время обработка сенсорных данных в режиме реального времени возникает в широком спектре приложений, например, интеллектуальное управление зданиями [1], цифровые двойники [2], мониторинг показателей функциональной диагностики организма человека [3], моделирование климата [4] и др. В указанных приложениях, как правило, недопустимы пропущенные значения, которые возникат ввиду отказов сенсоров или человеческого фактора, и они должны быть незамедлительно заменены на правдоподобные синтезированные значения. В соответствии с этим является актуальной задача разработки алгоритмов восстановления пропущенных значений в потоковых сенсорных данных, которые обеспечивают высокие точность и быстродействие восстановления.

В данной работе представлен параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени для многоядерного процессора. Алгоритм использует поиск подпоследовательностей ряда, похожих на заданную подпоследовательность-образец в смысле меры схожести DTW (Dynamic Time Warping, динамическая трансформация времени) [5]. На сегодня DTW признается научным сообществом одной из лучших мер схожести временных рядов для многих предметных областей [6]. Однако мера DTW имеет квадратичную вычислительную сложность относительно длины ряда и определяется рекуррентными формулами, что требует нетривиального подхода к распараллеливанию ее вычисления. Работа продолжает исследование авторов, начатое в статье [7]. По сравнению с предшествующей работой существенно увеличена производительность алгоритма за счет внедрения техники, позволившей отбрасывать без вычисления DTW большее количество подпоследовательностей, заведомо непохожих на образец поиска, а также проведены более масштабные вычислительные эксперименты.

Статья организована следующим образом. Раздел 1 содержит обзор работ, наиболее близких к выполненному исследованию. В разделе 2 приводится формальная постановка задачи. В разделе 3 представлен предлагаемый параллельный алгоритм. В разделе 4 описаны вычислительные эксперименты по исследованию эффективности предложенного алгоритма. Заключение резюмирует полученные результаты исследования и обозначает направление будущих работ.

1. Обзор работ

Создание эффективных методов и алгоритмов восстановления пропущенных данных является одной из наиболее актуальных проблем обработки временных рядов [8]. В данном разделе кратко рассмотрены только те работы, которые наиболее близки проведенному исследованию и затрагивают следующие аспекты: применение для восстановления ряда принципа ближайших соседей, меры схожести DTW, опорных рядов (коррелирующих с исходным рядом), а также использование параллельных вычислений.

В работе [9] Батиста (Batista) и др. предложили метод восстановления значений временного ряда kNNI (k-Nearest Neighbor Imputation) на основе ближайших соседей. Пусть имеется объект с множеством атрибутов, один из которых (обозначим его как A) имеет пропущенные значения. Метод предписывает сначала найти k «соседей» — объектов, имеющих схожие значения в атрибутах, отличных от A (без конкретизации меры схожести). Затем отсутствующее значение синтезируется на основе значений в атрибуте A у соседей. В работе [10] Троянская (Troyanskaya) и др. предложили использование взвешенных ближайших соседей, когда вес соседа пропорционален схожести с образцом поиска в смысле евклидовой метрики.

В работе [11] Хсю (Hsu) и др. предложили восстановление отсутствующих значений, основанный на использовании принципа k ближайших соседей и меры DTW. Алгоритм используется для восстановления отсутствующих значений во временном ряде, который представляет собой значения уровня экспрессии генов, полученные с помощью ДНК-микрочипов в серии экспериментов.

В работе [12] Фан (Phan) и др. применили DTW для восстановления пропущенных значений в многомерном временном ряде. Восстановление выполняется отдельно для каждого ряда-координаты следующим образом. Подпоследовательность ряда, которая начинается непосредственно после промежутка пропущенных значений и имеет ту же длину, что и

указанный промежуток, объявляется образцом поиска. Далее в части ряда, следующей после образца, выполняется поиск подпоследовательности, которая по длине равна образцу и является самой похожей на образец в смысле меры DTW. Подпоследовательность, располагающаяся непосредственно перед найденной и имеющая ту же длину, используется для поэлементного заполнения пропущенного промежутка.

Те же авторы в работе [13] представили алгоритм DTWBI (DTW-Based Imputation) для восстановления пропущенных значений в одномерном временном ряде. Восстанавливаемый временной ряд подвергается преобразованию DDTW (Derivative DTW) [14]. Далее подпоследовательность ряда, находящаяся непосредственно перед промежутком из пропущенных значений и имеющая ту же длину, что и указанный промежуток, объявляется образцом поиска. Далее в части ряда, предшествующей образцу, выполняется поиск подпоследовательности, самой похожей на образец в смысле меры DTW. Подпоследовательность, расположенная после найденной и имеющая ту же длину, используется для поэлементного заполнения пропущенного промежутка.

В работе [15] Велленцон (Wellenzohn) и др. предложили алгоритм TKCM (Top-k Case Matching), использующий для восстановления пропущенных значений ряда набор опорных временных рядов (reference time series), которые имеют семантическую связь с исходным рядом. Примерами исходного и опорных временных рядов могут служить показания температурных датчиков, установленных в географически близких локациях [15]. Алгоритм применяет эвристику, согласно которой похожие ситуации в опорных рядах возникают в те же временные промежутки, что и в ряде, подлежащем восстановлению. В каждом из опорных рядов TKCM объявляет образцом поиска подпоследовательность, завершающуюся в момент возникновения пропуска в исходном ряде. Далее в каждом из опорных рядов алгоритм выполняет поиск k подпоследовательностей, которые являются ближайшими соседями выбранных образцов и не могут перекрывать друг друга. При поиске используется схема динамического программирования на основе евклидовой метрики, которая минимизирует целевую функцию суммарного отличия подпоследовательностей в опорных рядах от соответствующих образцов. Восстанавливаемое значение получается как среднее арифметическое точек исходного ряда, которые соответствуют завершающим точкам временных интервалов найденных ближайших соседей. Алгоритм обеспечивает хорошую точность восстановления, когда отсутствуют блоки значений. Однако это достигается за счет прямо пропорциональной зависимости быстродействия алгоритма от его основных параметров: длина образца, количество опорных временных рядов, число ближайших соседей.

В обзоре и экспериментальном сравнении двенадцати современных алгоритмов восстановления пропущенных значений временных рядов [8] Хаяти (Khayati) и др. отмечают, что в настоящее время, по-видимому, отсутствуют успешные попытки использовать аппаратное обеспечение для ускорения алгоритмов восстановления отсутствующих значений в больших временных рядах.

В данном исследовании предлагается параллельный алгоритм восстановления пропущенных значений временного ряда для многоядерного процессора. Подобно алгоритму TKCM, наш алгоритм использует поиск ближайших соседей в опорных временных рядах. Однако при поиске вместо схемы динамического программирования и евклидовой метрики используется мера схожести DTW, которая имеет квадратичную вычислительную сложность, но в общем случае лучше отражает схожесть формы образца поиска и подпоследовательности ряда [6]. При этом применяется каскад нижних границ схожести [16]: под-

последовательности, заведомо непохожие на запрос, отбрасываются без вычисления DTW, что существенно сокращает объем вычислений [6]. В реализации используется идея предварительного параллельного вычисления нижних границ, предложенная нами ранее в работе [17]. Указанные разработки обуславливают возможность применения нашего алгоритма в режиме реального времени.

2. Формальные определения и обозначения

Потоковый временной ряд (streaming time series) представляет собой набор вещественных значений, поступающих последовательно одно за другим в режиме реального времени:

$$T = (\ldots, t_{n-2}, t_{n-1}, t_n), \quad t_i \in \mathbb{R}. \tag{1}$$

Мы полагаем, что n-элементное окно ряда T может быть размещено в оперативной памяти и n имеет порядок десятков-сотен тысяч элементов.

Значение элемента ряда t_n в текущий момент времени пропущено, что обозначается как t_n=`NULL`. Решаемая нами задача состоит в том, чтобы вместо пропущенного значения в режиме реального времени вычислить синтетическое значение $\tilde{t}_n$, которое как можно меньше отличалось бы от t_n.

Предлагаемый в данной работе алгоритм синтеза значения $\tilde{t}_n$ основан на поиске в ряде T подпоследовательности C, которая является самой похожей на заданный образец поиска Q в смысле меры схожести DTW (Dynamic Time Warping) [5]. Ниже приводятся соответствующие формальные определения с использованием нотации из работы [16].

Подпоследовательностью (subsequence) временного ряда T, имеющей длину m, называют непрерывное подмножество T из m элементов, начиная с позиции i:

$$T_{i,m} = (t_i, \ldots, t_{i+m-1}), \quad 1 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n - m + 1. \tag{2}$$

Множество всех подпоследовательностей ряда T, имеющих длину m, обозначается как S_T^m.

Пусть имеются две подпоследовательности ряда T: $C = (c_1, \ldots, c_m)$ и $Q = (q_1, \ldots, q_m)$. Тогда DTW-расстояние между C и Q определяется следующим образом [5]:

$$\begin{gathered} \mathrm{DTW}(C, Q) = D(m, m), \\ D(i, j) = (c_i - q_j)^2 + \min \begin{cases} D(i-1, j) \\ D(i, j-1) \\ D(i-1, j-1) \end{cases}, \\ D(0, 0) = 0, \quad D(i, 0) = D(0, j) = +\infty, \quad 1 \leqslant i \leqslant m, \quad 1 \leqslant j \leqslant m. \end{gathered} \tag{3}$$

В формуле (3) $D \in \mathbb{R}^{m \times m}$ является *матрицей трансформации*, которая задает выравнивание между подпоследовательностями C и Q друг относительно друга. В данной матрице строится *путь трансформации*, представляющий собой набор смежных элементов матрицы, который начинается и заканчивается в диагонально противоположных крайних элементах и устанавливает соответствие между C и Q, минимизируя общее расстояние между ними.

Подпоследовательность $C \in S_T^m$ называется *самой похожей (best match subsequence)* на заданный *запрос (query, образец поиска)* Q, если

$$\mathrm{DTW}(Q, C) \leqslant \mathrm{DTW}(Q, T_{i,m}), \quad 1 \leqslant i \leqslant n - m + 1. \tag{4}$$

Прямолинейный поиск самой похожей подпоследовательности (вычисление DTW-расстояния для всех подпоследовательностей ряда и нахождение среди них глобального минимума) не подходит для режима реального времени, поскольку DTW имеет вычислительную сложность $O(m^2)$ [6]. В силу этого нами используется *техника нижних границ (lower bounding)* [6], которая позволяет отбрасывать подпоследовательности, заведомо непохожие на запрос, без вычисления DTW, и существенно сократить объем вычислений.

Нижняя граница (lower bound) представляет собой симметричную неотрицательную функцию $\mathrm{LB} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ с вычислительной сложность меньше $O(m^2)$, которая является строго доказанным порогом DTW-расстояния между подпоследовательностью ряда и запросом:

$$\forall\, T_{i,\,m} \in S_T^m,\, Q \quad \mathrm{LB}(Q, T_{i,\,m}) \leqslant \mathrm{DTW}(Q, T_{i,\,m}). \tag{5}$$

Техника нижних границ заключается в следующем [6]. Обозначим локальный минимум DTW-расстояния от подпоследовательностей ряда до запроса как *bsf (best-so-far)* и инициализируем указанную переменную значением $+\infty$. Затем будем выполнять движение скользящего окна длины m в ряде T слева направо с шагом 1, вычисляя при этом нижнюю границу текущей подпоследовательности. Если результат превышает bsf, то в силу (5) значение DTW-расстояния между текущей подпоследовательностью и запросом также превысит bsf. Следовательно, данная подпоследовательность заведомо непохожа на запрос и может быть отброшена без вычисления DTW. Если вычисленное DTW-расстояние меньше локального минимума, то оно обновляет значение bsf. Формальная запись описанной выше техники выглядит следующим образом:

$$\begin{cases} bsf_{(0)} & = +\infty \\ bsf_{(i)} & = \min\Big(bsf_{(i-1)}, \begin{cases} +\infty & , \mathrm{LB}(Q, T_{i,\,m}) > bsf_{(i-1)} \\ \mathrm{DTW}(Q, T_{i,\,m}) & , otherwise \end{cases}\Big), \end{cases} \tag{6}$$

где нижний индекс в скобках означает номер шага i $(0 \leqslant i < n - m + 1)$.

Для применения техники нижних границ требуется, чтобы подпоследовательность и запрос были подвергнуты z-нормализации [16]. Z-нормализацией ряда $T = (t_1, \ldots, t_m)$ является ряд $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_m)$, элементы которого вычисляются следующим образом:

$$\hat{t}_i = \frac{t_i - \mu}{\sigma}, \quad \mu = \frac{1}{m}\sum_{i=1}^{m} t_i, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m} t_i^2 - \mu^2. \tag{7}$$

Для повышения эффективности отбрасывания заведомо непохожих подпоследовательностей нижние границы применяют каскадом [16] (одна за другой в порядке возрастания их вычислительной сложности). В нашем исследовании применяются три наиболее известные нижние границы $\mathrm{LB_{Kim}FL}$ [18], $\mathrm{LB_{Keogh}EQ}$ [19] и $\mathrm{LB_{Keogh}EC}$ [16].

Нижняя граница схожести $\mathrm{LB_{Kim}FL}$ представляет собой квадрат Евклидова расстояния между первой и последней парами точек запроса и подпоследовательности:

$$\mathrm{LB_{Kim}FL}(Q, C) = (\hat{q}_1 - \hat{c}_1)^2 + (\hat{q}_m - \hat{c}_m)^2. \tag{8}$$

Нижняя граница $\mathrm{LB_{Keogh}EC}$ показывает схожесть между *оболочкой (envelope) E* запроса и подпоследовательностью и вычисляется следующим образом:

$$\mathrm{LB_{Keogh}EC}(Q,C) = \sum_{i=1}^{m} \begin{cases} (\hat{c}_i - u_i)^2 & , \ \hat{c}_i > u_i \\ (\hat{c}_i - \ell_i)^2 & , \ \hat{c}_i < \ell_i \\ 0 & , \ otherwise \end{cases} . \tag{9}$$

В формуле (9) последовательности $U = (u_1, \ldots, u_m)$ и $L = (\ell_1, \ldots, \ell_m)$ обозначают *верхнюю* и *нижнюю границы оболочки* запроса Q соответственно, которые вычисляются следующим образом:

$$\begin{aligned} u_i &= \max(\hat{q}_{i-r}, \ldots, \hat{q}_{i+r}), \\ \ell_i &= \min(\hat{q}_{i-r}, \ldots, \hat{q}_{i+r}), \end{aligned} \tag{10}$$

где параметр r $(1 \leqslant r \leqslant m)$ представляет собой *ограничение полосы Сако—Чиба (Sakoe–Chiba band)* [6], которое показывает, что путь трансформации не может отклоняться более чем на r ячеек от диагонали матрицы трансформации.

Нижняя граница $\mathrm{LB_{Keogh}EQ}$ представляет собой Евклидово расстояние между запросом Q и оболочкой подпоследовательности C, то есть по сравнению с $\mathrm{LB_{Keogh}EC}$ роли запроса и подпоследовательности меняются местами:

$$\mathrm{LB_{Keogh}EQ}(Q,C) = \mathrm{LB_{Keogh}EC}(C,Q). \tag{11}$$

Каскад может быть дополнен другими нижними границами, например, $\mathrm{LB_{Yi}}$, $\mathrm{LB_{PAA}}$ [6]. Техника нижних границ позволяет отбросить до 99% вычислений DTW [16].

Завершая данный раздел, отметим, что описанные выше построения могут быть тривиально обобщены на случай поиска k подпоследовательностей ряда, наиболее похожих на образец [16], где k — наперед заданный параметр.

3. Параллельный алгоритм восстановления пропусков

В данном разделе представлен новый параллельный алгоритм для многоядерного процессора, который выполняет восстановление пропущенного значения потокового временного ряда в режиме реального времени. Ниже в разделе 3.1 описана общая схема восстановления, используемая алгоритмом. В разделе 3.2 представлены структуры данных алгоритма. Раздел 3.3 содержит описание принципов реализации.

3.1. Общая схема восстановления

Алгоритм применяет опорные временные ряды и следующую эвристику, примененные ранее в алгоритме TKCM [15]: если в опорных рядах имеют место повторяющиеся (похожие) подпоследовательности, то в ряде, содержащем пропущенное значение, повторяющиеся подпоследовательности возникают в тех же временны́х интервалах.

Пусть $R^1, \ldots, R^d$ $(d > 1)$ — опорные потоковые временные ряды для ряда T. Это означает, что в каждом опорном ряде отсутствуют `NULL`-значения, и его элементы получены в те же моменты времени, что и элементы ряда T с соответствующими порядковыми номерами. Пусть в данном списке ряды упорядочены по убыванию корреляции с T, например, по убыванию абсолютного значения коэффициента корреляции Пирсона. Как и в случае исходного ряда, мы предполагаем, что совокупность n-элементных окон опорных рядов размещена в

оперативной памяти. Алгоритм выполняет следующие шаги: поиск по образцу, скоринг и реконструкция.

На шаге *поиска по образцу* сначала для каждого опорного ряда R^i определяется образец, состоящий из m последних по времени элементов ряда (где $1 \leqslant m \ll n$ — параметр алгоритма): $Q^i = R^i_{n-m+1,\,m}$. Далее для каждого опорного ряда R^i в его окне с первого по $(n-2m)$-й элемент выполняется поиск подпоследовательности, которая имеет длину m и является самой похожей на образец Q^i в смысле меры DTW. Попутно при выполнении поиска вычисляется DTW-расстояние от запроса до каждой подпоследовательности опорного ряда, за исключением подпоследовательностей, заведомо не похожих на запрос, для которых сохраняется значение $+\infty$.

На шаге *скоринга*, используя вычисленные на предыдущем шаге DTW-расстояния от запросов до подпоследовательностей опорных рядов, определяется множество $kNNset$ подпоследовательностей исходного ряда, применяемых для реконструкции. Данное множество состоит из k не пересекающихся друг с другом подпоследовательностей длины m (где $1 \leqslant k < \lceil n/m \rceil$ — параметр алгоритма), которые являются наиболее значимыми для восстановления пропущенного значения. Значимость подпоследовательности определяется нами как функция от DTW-расстояний соответствующих подпоследовательностей в опорных рядах, подсчитанных на шаге поиска по образцу.

На шаге *реконструкции* восстановленное значение вычисляется как среднее арифметическое последних элементов подпоследовательностей, найденных на шаге скоринга.

3.2. Структуры данных

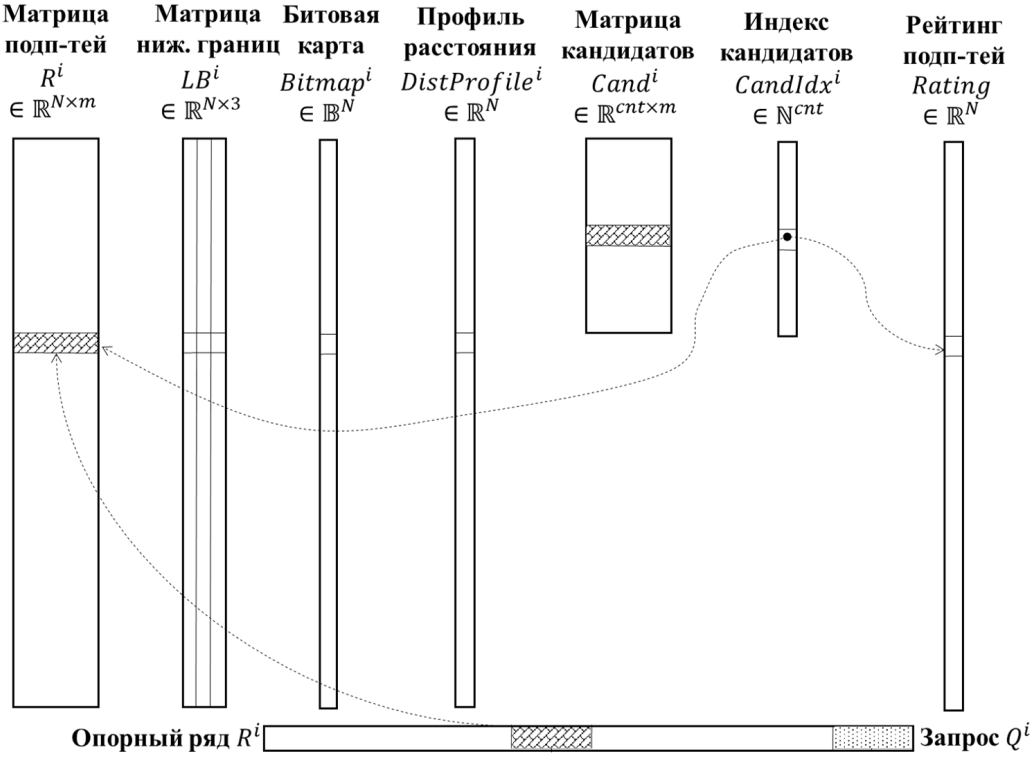

Рис. 1. Структуры данных алгоритма

Структуры данных нашего алгоритма представлены на рис. 1. Каждый опорный временной ряд хранится в виде матрицы подпоследовательностей $R^i \in \mathbb{R}^{N\times m}$, где число

$N = n - m$ представляет собой количество подпоследовательностей, имеющих длину m, исключая запрос. Для каждого опорного ряда определяются следующие структуры данных: матрица нижних границ, битовая карта, профиль расстояния, матрица и индекс кандидатов. Помимо указанных структур данных, для выполнения реконструкции хранится рейтинг интервалов исходного временного ряда.

Матрица нижних границ $LB^i \in \mathbb{R}^{N \times lb_{num}}$ хранит значения нижних границ, вычисляемых по формулам (8)–(11), где lb_{num} означает количество нижних границ в каскаде (в нашем исследовании $lb_{num} = 3$). Отметим, что в оригинальном алгоритме [16] нижние границы вычисляются при движении скользящего окна, и при этом каждая следующая нижняя граница каскада вычисляется только в том случае, если текущая подпоследовательность не была отброшена как заведомо непохожая на запрос. Очевидно, что такие вычисления не могут быть распараллелены в силу наличия между ними зависимостей по данным. В предлагаемом нами подходе все нижние границы для всех подпоследовательностей ряда вычисляются одномоментно до начала движения скользящего окна. Несмотря на избыточность таких вычислений, они выполняются однократно и при этом могут быть распараллелены ввиду отсутствия зависимостей по данным между нижними границами, что в итоге дает выигрыш в производительности. Платой за получаемое быстродействие, очевидно, является необходимость хранения описанной матрицы в памяти.

Битовая карта представляет собой вектор $Bitmap^i \in \mathbb{B}^N$, в котором для каждой подпоследовательности опорного временного ряда хранится конъюнкция результатов сравнения bsf и каждой из нижних границ (8)–(11) для запроса Q^i:

$$Bitmap^i(j) = \bigwedge_{s=1}^{lb_{num}} \big(LB^i(j,s) < bsf\big). \tag{12}$$

Таким образом, если какая-либо нижняя граница каскада превысит значение bsf, то битовая карта содержит `FALSE` и показывает, что соответствующая подпоследовательность заведомо не похожа на запрос.

Профиль расстояния представляет собой вектор $DistProfile^i \in \mathbb{R}^N$, в котором хранятся DTW-расстояния между подпоследовательностями опорного ряда R^i и запросом Q^i. Для подпоследовательностей, заведомо не похожих на запрос, в профиле расстояния хранится значение $+\infty$.

Матрица кандидатов $Cand^i \in \mathbb{R}^{cnt^i \times m}$ хранит подпоследовательности, которые не были отброшены как заведомо непохожие на образец Q^i, где количество таких подпоследовательностей cnt^i определяется после применения каскада нижних границ. Данная матрица используется для параллельного вычисления DTW-расстояний между каждой из ее строк и запросом.

Индекс кандидатов $CandIndex^i \in \mathbb{N}^{cnt^i}$ хранит номера соответствующих строк $Cand^i$ в опорном ряде R^i.

Рейтинг подпоследовательностей $Rating \in \mathbb{R}^N$ представляет собой вектор, который хранит значимость подпоследовательностей длины m в потоковом n-элементном окне для последующего поиска k наиболее значимых из них и выполнения реконструкции отсутствующего значения t_n.

3.3. Реализация

В данном разделе описаны принципы параллельной реализации шагов поиска и скоринга (шаг реконструкции является алгоритмически тривиальным). Параллелизм для многоядерного процессора реализован на основе технологии программирования OpenMP [20]. В модели OpenMP приложение рассматривается как процесс — набор инструкций, исполняемых последовательно. Процесс может запустить (*fork*) заданное количество параллельно исполняемых нитей. Нити разделяют память процесса и в то же время для хранения собственных данных имеют приватную память. Среда исполнения приложения автоматически назначает нитям различные ядра процессора. Формирование параллельных нитей осуществляется вставкой директивы компилятора `#pragma` в исходный код. Одним из наиболее типичных случаев применения данной директивы является распараллеливание цикла **for** с помощью `#pragma omp parallel for`, когда каждая из нитей обрабатывает собственный диапазон значений счетчика цикла.

Алг. 1 SEARCH(IN: R, Q; OUT: $DistProfile$)

1: $LB \leftarrow$ CALCLBS(R, Q)
2: $idx \leftarrow \arg\min_{i \in 1..N} \max_{j \in 1..lb_{num}} LB(i,j)$; $bsf \leftarrow \mathrm{DTW}\big(\hat{Q}, R(idx, \cdot)\big)$
3: $Bitmap \leftarrow$ MAKEBITMAP(LB, bsf)
4: $DistProfile \leftarrow \overline{+\infty}$
5: **while** `TRUE` **do**
▷ Отбрасывание бесперспективных кандидатов с помощью нижних границ
6: #pragma omp parallel for
7: **for** $i \in 1..N$ **do**
8: $Bitmap(i) \leftarrow Bitmap(i)$ **and** $\bigwedge_{j=1}^{lb_{num}} \big(LB(i,j) < bsf\big)$
▷ Заполнение матрицы кандидатов и индекса кандидатов
9: $cnt \leftarrow 0$
10: **for** $i \in 1..N$ **do**
11: **if** $Bitmap(i) =$ `TRUE` **then**
12: $cnt \leftarrow cnt + 1$; $Cand(cnt) \leftarrow R(i, \cdot)$; $CandIndex(cnt) \leftarrow i$
13: **if** $cnt = 0$ **then**
14: **break**
▷ Вычисление DTW-расстояний для кандидатов и улучшение bsf
15: $s \leftarrow 1$
16: **while** `TRUE` **do**
17: $left \leftarrow p \cdot (s-1) + 1$; $right \leftarrow \min(cnt, p \cdot s)$
18: #pragma omp parallel for reduction(min: $dist$)
19: **for** $i \in left..right$ **do**
20: $dist \leftarrow \mathrm{DTW}(\hat{Q}, Cand(i, \cdot))$
21: $DistProfile(CandIndex(i)) \leftarrow dist$
22: $Bitmap(CandIndex(i)) \leftarrow$ `FALSE`
23: $bsf \leftarrow \min(bsf, dist)$; $s \leftarrow s + 1$
24: **if** $dist < bsf$ **or** $right = cnt$ **then**
25: **break**
26: **return** $DistProfile$

Поиск реализуется с помощью двух вложенных циклов: внешний цикл по опорным временным рядам и внутренний цикл по подпоследовательностям опорного временного ряда. Поскольку, как правило, количество подпоследовательностей в опорном ряде существенно больше количества опорных рядов, а последнее, в свою очередь, существенно меньше, чем количество нитей приложения, мы выполняем распараллеливание внутреннего цикла, чтобы обеспечить значимую нагрузку нитей. Далее для упрощения записи алгоритма мы опускаем внешний цикл по опорным временным рядам и не пишем в структурах данных соответствующий верхний индекс. За p обозначено количество нитей, на которых исполняется параллельное приложение. Псевдокод шага поиска представлен в Алг. 1. Алгоритм выполняется следующим образом.

Алг. 2 CalcLBs(IN: R, Q; OUT: LB)

```
1: #pragma omp parallel for
2: for i ∈ 1..N do
3:     R(i,·) ←zNormalize(R(i,·))
4:     LB(i,1) ← LB_KimFL(Q̂, R(i,·))
5:     LB(i,2) ← LB_KeoghEC(Q̂, R(i,·))
6:     LB(i,3) ← LB_KeoghEQ(Q̂, R(i,·))
7: return LB
```

Сначала параллельно нормализуются подпоследовательности и вычисляются нижние границы (см. строку 1 в Алг. 1 и Алг. 2). Затем выполняется инициализация порога bsf (см. строку 2). Начальное значение bsf выбирается нами следующим образом на основе эвристики, описанной в работе [16]. В указанной статье авторами проведены вычислительные эксперименты над 50 временными рядами из архива UCR [21], в которых для каждой нижней границы LB, упомянутой в разделе 2, вычислялось значение функции

$$\mathrm{Tightness}_{\mathrm{LB}}(A,B) = \frac{\mathrm{LB}(A,B)}{\mathrm{DTW}(A,B)} \tag{13}$$

для 100 тыс. случайных подпоследовательностей A и B, имеющих длину 256. Эксперименты показали [16], что, как правило, выполняется следующее неравенство:

$$\begin{gathered}\mathrm{Tightness}_{\mathrm{LB_{Kim}FL}}(A,B) < \mathrm{Tightness}_{\mathrm{LB_{Keogh}EQ}}(A,B) \leqslant \\ \leqslant \max\big(\mathrm{Tightness}_{\mathrm{LB_{Keogh}EQ}}(A,B), \mathrm{Tightness}_{\mathrm{LB_{Keogh}EC}}(A,B)\big) \leqslant 1.\end{gathered} \tag{14}$$

Исходя из соотношения (14) и определения нижней границы (5), в качестве начального значения для порога bsf мы выбираем DTW-расстояние от запроса до такой подпоследовательности ряда, на которой достигается минимум среди максимумов значений нижних границ:

$$bsf_{(0)} = \mathrm{DTW}(Q,C), \quad C = \arg \min_{T_{i,\,m} \in S_T^m} \max_{j \in 1..lb_{num}} \mathrm{LB}_j(Q, T_{i,m}). \tag{15}$$

В отличие от значения $bsf_{(0)} = +\infty$ в (6), это позволяет сократить количество рассматриваемых подпоследовательностей уже на ранней стадии.

После этого выполняется формирование битовой карты подпоследовательностей (см. строку 3 в Алг. 1 и Алг. 3). Сначала в соответствии с начальным значением порога bsf отбрасываются подпоследовательности, заведомо не похожие на запрос (см. строки 2–3

Алг. 3 МакеBitmap(IN: LB, bsf; OUT: $Bitmap$)

```
1:  match ← 0
2:  for i ∈ 1..N do
3:      Bitmap(i) ← ⋀_{j=1}^{lb_num} (LB(i, j) < bsf)
4:      if Bitmap(i) = TRUE then
5:          if match > 0 then
6:              right ← i
7:          else
8:              left ← i
9:          match ← match + 1
10:     if (Bitmap(i) = TRUE and match = m) or (Bitmap(i) = FALSE and match > 1) then
11:         idx ← arg min_{i∈left..right} max_{j∈1..lb_num} LB(i, j)
12:         for i ∈ {left..idx − 1} ∪ {idx + 1..right} do
13:             Bitmap(i) ← FALSE
14:         match ← 0
15: return Bitmap
```

в Алг. 3). Затем среди оставшихся кандидатов выполняется отбрасывание тривиальных совпадений (см. строки 4–14 в Алг. 3). Две подпоследовательности $T_{i,m}, T_{j,m} \in S_T^m$ являются *тривиальными совпадениями (trivial matches)* друг друга, если $|i - j| < m$ [22]. В каждой группе подпоследовательностей, представляющих тривиальные совпадения друг друга, мы оставляем одну подпоследовательность в соответствии с описанной выше эвристикой (см. формулу (15)), а остальные подпоследовательности группы отбрасываются.

Затем алгоритм поиска инициализирует профиль расстояния значениями $+\infty$ и итеративно уменьшает значение bsf, не вычисляя DTW-расстояния до подпоследовательностей, заведомо не похожих на запрос (строки 5–25). При этом сначала параллельно вычисляется битовая карта (строки 6–8), а затем формируются матрица кандидатов и индекс кандидатов (строки 10–12). Если подпоследовательности-кандидаты не найдены, то алгоритм останавливается (строки 13–14). Ситуация, когда после первого сканирования все подпоследовательности отброшены как заведомо непохожие на запрос, означает, что выбрано слишком жесткое ограничение полосы Сако—Чиба, и параметр r необходимо уменьшить.

Далее выполняется параллельное вычисление DTW-расстояния между каждой строкой матрицы кандидатов и запросом (строки 15–25). Вычисления производятся итеративно для групп по p строк, чтобы обеспечить баланс загрузки нитей. На каждой итерации мы автоматически получаем минимум DTW-расстояния и обновляем bsf и профиль расстояния, применяя параллельную свертку (конструкция `reduction` в директиве `#pragma`, строка 18). В элемент битовой карты, соответствующий обработанной подпоследовательности, записывается значение `FALSE`, чтобы обеспечить однократную обработку каждого кандидата. Описанная деятельность продолжается, пока не будет улучшено (уменьшено) значение bsf либо не будут просмотрены все кандидаты без улучшения значения bsf. Если значение bsf улучшено, алгоритм переходит к началу, вычисляя битовую карту и пытаясь отбросить как бесперспективные оставшиеся подпоследовательности.

По окончании работы алгоритма в элементах профиля расстояния записано значение DTW-расстояния между запросом и соответствующей подпоследовательностью либо зна-

чение $+\infty$, которое означает, что данная подпоследовательность заведомо не похожа на запрос, причем в этом случае значение расстояния не вычислялось.

Алг. 4 SCORING(IN: $DistProfile^1, \ldots, DistProfile^d$, k; OUT: $kNNset$)

1: $Rating \leftarrow \overline{0}$; $s \leftarrow 0$; $kNNset \leftarrow \varnothing$
2: #pragma omp parallel for
3: **for** $i \in 1..N$ **do**
4: **for** $j \in 1..d$ **do**
5: $Rating(i) \leftarrow Rating(i) + \frac{1}{DistProfile^j(i)+\varepsilon} \cdot \frac{d-j+1}{d}$
6: **while** TRUE **do**
7: $s \leftarrow s + 1$
8: $maxRating \leftarrow \max\limits_{i \in 1..N} Rating(i)$; $maxIdx \leftarrow \arg\max\limits_{i \in 1..N} Rating(i)$
9: **if** $|maxIdx - prevIdx| > m$ **then**
10: $kNNset \leftarrow kNNset \cup \{T_{maxIdx,\,m}\}$; $prevIdx \leftarrow maxIdx$
11: **else**
12: $Rating(maxIdx) \leftarrow -\infty$
13: **if** $|kNNset| = k$ **or** $s = N$ **then**
14: **break**
15: **return** $kNNset$

Алг. 4 показывает псевдокод шага скоринга, на котором выполняется отбор подпоследовательностей исходного ряда для восстановления пропущенного значения. Сначала на основе профилей расстояния всех опорных рядов, вычисленных на предыдущем шаге, параллельно вычисляется рейтинг интервалов (строки 1–5). Вычисления выполняются с помощью двух вложенных циклов: внешний — по интервалам, внутренний — по опорным рядам; внешний цикл распараллеливается. После этого в векторе $Rating$ мы находим top-k наибольших элементов так, чтобы их индексы различались не менее чем на m (строки 6–14).

Рейтинг подпоследовательности $T_{i,\,m}$ определяется следующим образом:

$$Rating(T_{i,\,m}) = \sum_{s=1}^{d} \frac{w(s,i)}{DistProfile^s(i) + \varepsilon}, \tag{16}$$

где $DistProfile^s(i) = \mathrm{DTW}(Q^s, R^s_{i,\,m})$ — это DTW-расстояние между запросом s-го опорного ряда и его i-й подпоследовательностью, ε — машинный эпсилон (верхняя граница относительной ошибки из-за округления в арифметике с плавающей точкой), $w : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ — весовая функция.

Вес подпоследовательности в формуле (16) позволяет учитывать корреляцию между восстанавливаемым и опорным временными рядами, а также индекс подпоследовательности. Например, весовая функция позволяет выбирать для восстановления, соответственно, подпоследовательности из более коррелированных опорных временных рядов и из наиболее недавних моментов времени. В нашем исследовании мы берем весовую функцию $w(s,i) = \frac{d-s+1}{d}$, которая имеет следующую семантику: больший вес имеет подпоследовательность более коррелированного временного ряда, индекс подпоследовательности не учитывается.

В итоге реконструкция пропущенного значения t_n выполняется следующим образом:

$$\tilde{t}_n = \frac{1}{k} \sum_{T_{i,\,m} \in kNNset} t_m. \tag{17}$$

4. Вычислительные эксперименты

Для оценки эффективности разработанного алгоритма нами были проведены вычислительные эксперименты, в которых исследовались точность восстановления и производительность алгоритма.

Для оценки *точности восстановления* пропущенных значений нами используется мера среднеквадратичной ошибки *RMSE (Root Mean Square Error)*, определяемая следующим образом:

$$RMSE = \sqrt{\frac{1}{h} \sum_{i=1}^{h} (t_i - \tilde{t}_i)^2}, \tag{18}$$

где t_i — фактическое значение элемента ряда, считающегося пропущенным, $\tilde{t}_i$ — восстановленное значение, h — количество пропущенных элементов временного ряда.

Производительность алгоритма понимается как среднее время, затрачиваемое на восстановление одного пропущенного элемента временного ряда. В экспериментах запуск программы осуществлялся 10 раз и в качестве итогового времени вычисления использовалось медианное значение. Эксперименты были проведены на оборудовании Лаборатории суперкомпьютерного моделирования ЮУрГУ [23]: процессор Intel Xeon E5-2687W v2 (8 ядер @3.40 GHz).

Для исследований нами использовался фреймворк ORBITS [24], который реализует одноименный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени и обеспечивает проведение экспериментов со следующими алгоритмами аналогичного назначения сторонних авторов: OGDImpute [25], SPIRIT [26], SAGE [27], TKCM [15].

Таблица 1. Наборы данных

Набор данных	Количество рядов	Длина	Предметная область
BAFU	10	50 000	Сброс воды в реках Швейцарии
Chlorine	50	1 000	Распространение хлора в системе распределения питьевой воды
Climate	10	5 000	Погода в различных локациях Северной Америки
MADRID	10	25 000	Трафик на дорогах Мадрида
MAREL	10	50 000	Характеристики морской воды в Ла-Манше

В экспериментах нами использовались наборы данных, представленные в табл. 1. Chlorine [28] представляет собой набор синтетических временных рядов, который создан путем моделирования системы распределения питьевой воды и описывает концентрацию хлора. Распространение хлора в системе вызывает фазовые сдвиги в рядах набора данных. BAFU [29] представляет собой набор временных рядов с данными о сбросе воды в 10 реках Швейцарии. Частота измерений высока для отслеживания быстро меняющегося давления воды в сезон дождей. Набор содержит периодические временные ряды, некоторые

из которых смещены во времени. Набор Climate [30] представляет ежемесячные агрегированные климатические данные, собранные с метеостанций в различных локациях Северной Америки в 1990–2002 гг. Временные ряды нерегулярны и содержат спорадические всплески. Набор MADRID [31] содержит данные автоматических регистраторов дорожного движения, установленных на городских дорогах и автострадах Мадрида. Набор MAREL [32] содержит данные о различных химических и биологических характеристиках морской воды в проливе Ла-Манш.

В экспериментах для каждого набора данных 10% последних точек одного из рядов, идущие подряд, подлежали восстановлению. При этом осуществлялось накопление ошибки: для восстановления каждого элемента использовались ранее восстановленные, а не исходные значения всех предыдущих элементов. При восстановлении рассматривалось два случая относительно количества опорных рядов: минимальное (число, необходимое для запуска эксперимента в фреймворке ORBITS) и максимальное (общее число рядов в наборе за исключением восстанавливаемого ряда). В экспериментах с алгоритмами-аналогами устанавливались такие параметры их запуска, которые рекомендованы разработчиками фреймворка ORBITS для обеспечения наивысшей точности восстановления [24]. Для разработанного алгоритма во всех экспериментах использовались следующие параметры: длина запроса $m = 50$, ограничение полосы Сако—Чиба $r = \lceil 0.25m \rceil$, количество соседей $k = 3$.

Таблица 2. Результаты экспериментов

Набор данных	К-во опор. рядов, d	Критерий сравнения	SAGE	OGD-Impute	ORBITS		SPIRIT	TKCM	Наш алгоритм
					$k = 2$	$k = 3$			
BAFU	3	**Точность**, RMSE	14.82	1.18	10.03	10.09	0.66	0.18	**0.16**
		Производительность, мс	0.49	1.23	0.09	0.20	*0.003*	99.00	31.86
	9	**Точность**, RMSE	2.77	1.18	9.92	10.12	**0.07**	0.18	0.19
		Производительность, мс	0.50	3.09	0.18	0.25	*0.003*	99.00	93.8
Chlorine	4	**Точность**, RMSE	1.05	0.59	0.70	0.75	0.43	0.41	**0.03**
		Производительность, мс	0.05	0.40	0.14	0.19	*0.003*	2.00	0.80
	49	**Точность**, RMSE	0.29	0.59	0.21	0.08	0.10	0.41	**0.02**
		Производительность, мс	0.03	2.58	0.12	1.16	*0.02*	2.00	11.90
Climate	3	**Точность**, RMSE	3 888.78	33.29	366.78	366.74	4.32	10.19	**1.17**
		Производительность, мс	0.07	0.28	0.09	0.13	*0.007*	9.94	3.02
	9	**Точность**, RMSE	1 587.31	33.29	28.62	6.49	6.13	10.19	**1.51**
		Производительность, мс	0.08	0.71	0.07	0.04	*0.004*	9.97	13.7
MADRID	3	**Точность**, RMSE	903.48	1.09e+12	396.40	845.52	1 090.35	536.14	**212.97**
		Производительность, мс	0.29	0.71	0.10	0.18	*0.003*	47.79	14.67
	9	**Точность**, RMSE	739.84	1.09e+12	249.74	645.32	381.20	536.14	**202.42**
		Производительность, мс	0.32	1.79	0.05	0.32	*0.004*	48.09	42.22
MAREL	3	**Точность**, RMSE	37.86	2.78	10.82	11.18	11.29	**2.75**	4.60
		Производительность, мс	0.56	1.17	0.21	0.30	*0.003*	101.08	33.16
	9	**Точность**, RMSE	81.29	2.78	6.05	4.69	4.63	**2.75**	4.71
		Производительность, мс	0.48	2.95	0.02	0.02	*0.004*	95.90	92.80

Итоги экспериментов представлены в табл. 2, где курсивом выделены результаты наиболее быстрого алгоритма, полужирным — наиболее точного. Можно видеть, что предложенный параллельный алгоритм показывает, как правило, наиболее высокую точность восстановления, уступая лишь алгоритмам TKCM и SPIRIT соответственно на наборах данных MAREL и BAFU (в случае максимального количества опорных рядов). Мы можем заключить, что применение эвристики о схожем поведении восстанавливаемого и опорных временных рядов вкупе с мерой DTW, хорошо определяющей схожесть по форме подпоследовательности ряда, позволило предложенному нами алгоритму достичь высокой точности

восстановления. Однако наиболее производительным является последовательный алгоритм, SPIRIT, что объясняется следующим образом. Применение параллельных вычислений и отбрасывания бесперспективных кандидатов с помощью техники нижних границ позволило нашему алгоритму существенно ускорить время восстановления, однако заложенные в его природе накладные расходы на вычисление меры DTW, как правило, все же не могут быть сведены к нулю.

Отметим также, что в экспериментах предложенный алгоритм показывает быстродействие восстановления в диапазоне 0.8–93 мс в лучшем и худшем случаях соответственно. Данный результат допускает применение разработанного алгоритма в режиме реального времени. Например, в статье [33] указывается, что в системах автоматизации управления для сети передачи данных, обслуживающей датчики измерения температуры, влажности, давления цикл передачи данных составляет до 100 мс. В каталоге температурных датчиков компании Emerson [34] 2021 г., одного из ведущих мировых производителей измерительных систем, указывается, что беспроводные температурные датчики имеют период обновления данных не менее 1 с.

Заключение

В работе затронута проблема восстановления пропущенных значений в потоковом временном ряде в режиме реального времени, которая встречается в настоящее время в приложениях Интернета вещей, персональной медицины, цифровая индустрии и др. Пропуски в данных временного ряда, возникающие в таких приложениях ввиду технических сбоев или человеческого фактора, как правило, неприемлемы и подлежат замене на правдоподобные значения, синтезированные на лету.

В статье предложен новый параллельный алгоритм для многоядерного процессора, который выполняет восстановление пропущенного значения потокового временного ряда в режиме реального времени. Параллелизм вычислений реализован с помощью технологии программирования OpenMP. Алгоритм работает в предположении, что имеется набор опорных временных рядов, которые имеют семантическую связь с исходным рядом. Алгоритм применяет следующую эвристику: если в опорных рядах имеют место повторяющиеся (схожие) подпоследовательности, то в ряде, содержащем пропущенное значение, повторяющиеся подпоследовательности возникают в тех же временны́х интервалах.

Предложенный алгоритм выполняет следующие шаги: поиск по образцу, скоринг и реконструкция. Для каждого опорного ряда образцами поиска полагаются подпоследовательности заданной длины, которые оканчиваются в момент пропуска значения в исходном ряде. Далее для каждого опорного ряда выполняется поиск подпоследовательности, которая является самой похожей на образец в смысле меры DTW (Dynamic Time Warping) [5]. Мера DTW имеет квадратичную вычислительную сложность относительно длины подпоследовательности и определяется рекуррентными формулами, но в общем случае отражает схожесть формы образца поиска и подпоследовательности ряда лучше, чем евклидова метрика. Для ускорения данного шага используется техника нижних границ (lower bounding) [6], которые представляют собой меры схожести с вычислительной сложностью меньше, чем у меры DTW. Данная техника позволяет отбрасывать без вычисления DTW подпоследовательности, заведомо непохожие на образец поиска. Для каждого опорного ряда выполняется параллельное вычисление нижних границ, а также параллельно вычисляются DTW-

расстояния от образца до каждой подпоследовательности опорного ряда. Расстояния до отброшенных подпоследовательностей полагаются равными $+\infty$.

На шаге скоринга алгоритм, используя вычисленные ранее DTW-расстояния, определяет множество подпоследовательностей исходного ряда, применяемых для восстановления пропущенного значения. Данное множество состоит из k не пересекающихся друг с другом подпоследовательностей, которые являются наиболее значимыми для восстановления пропущенного значения. Значимость подпоследовательности определяется нами как функция от DTW-расстояний соответствующих подпоследовательностей в опорных рядах, подсчитанных на предыдущем шаге. Вычисление значимости всех подпоследовательностей исходного ряда выполняется параллельно. На финальном шаге реконструкции восстановленное значение вычисляется как среднее арифметическое последних элементов подпоследовательностей, найденных на шаге скоринга.

В вычислительных экспериментах с реальными и синтетическими данными предложенный алгоритм продемонстрировал точность восстановления, как правило, более высокую, чем у аналогов, и быстродействие восстановления, приемлемое для применения алгоритма в режиме реального времени.

В качестве возможного направления продолжения исследований мы рассматриваем разработку версии алгоритма для графического процессора.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 20-07-00140) и Министерства науки и высшего образования РФ (государственные задания FENU-2020-0022, FWNF-2022-0016).

Литература

1. Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др. Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 16–36. DOI: 10.14529/cmse210302.
2. Иванов С.А., Никольская К.Ю., Радченко Г.И. и др. Концепция построения цифрового двойника города // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 4. С. 5–23. DOI: 10.14529/cmse200401.
3. 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. Т. 2, № 1. С. 44–54. DOI: 10.14529/cmse130105.
4. 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. Т. 3, № 1. С. 17–32. DOI: 10.14529/cmse140102.
5. Berndt D.J., Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series // Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03 / ed. by U.M. Fayyad, R. Uthurusamy. 1994. P. 359–370.

6. Ding H., Trajcevski G., Scheuermann P., *et al.* Querying and mining of time series data: experimental comparison of representations and distance measures // Proc. VLDB Endow. 2008. Vol. 1, no. 2. P. 1542–1552. DOI: 10.14778/1454159.1454226.
7. Цымблер М.Л., Полуянов А.Н. Параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени // Параллельные вычислительные технологии – XVI международная конференция, ПаВТ'2022, Дубна, 29-31 марта 2022. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ. 2022. С. 128–140. DOI: 10.14529/pct2022.
8. Khayati M., Lerner A., Tymchenko Z., Cudré-Mauroux P. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series // Proc. VLDB Endow. 2020. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.
9. Batista G.E.A.P.A., Monard M.C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning // Appl. Artif. Intell. 2003. Vol. 17, no. 5-6. P. 519–533. DOI: 10.1080/713827181.
10. Troyanskaya O.G., Cantor M.N., Sherlock G., *et al.* Missing value estimation methods for DNA microarrays // Bioinform. 2001. Vol. 17, no. 6. P. 520–525. DOI: 10.1093/bioinformatics/17.6.520.
11. Hsu H., Yang A.C., Lu M. KNN-DTW Based Missing Value Imputation for Microarray Time Series Data // J. Comput. 2011. Vol. 6, no. 3. P. 418–425. DOI: 10.4304/jcp.6.3.418-425.
12. Phan T., Poisson É.C., Bigand A., Lefebvre A. DTW-Approach for uncorrelated multivariate time series imputation // 27th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2017, Tokyo, Japan, September 25-28, 2017 / ed. by N. Ueda, S. Watanabe, T. Matsui, *et al.* 2017. P. 1–6. DOI: 10.1109/MLSP.2017.8168165.
13. Phan T., Caillault É.P., Lefebvre A., Bigand A. Dynamic time warping-based imputation for univariate time series data // Pattern Recognit. Lett. 2020. Vol. 139. P. 139–147. DOI: 10.1016/j.patrec.2017.08.019.
14. Keogh E.J., Pazzani M.J. Derivative Dynamic Time Warping // Proceedings of the 1st SIAM International Conference on Data Mining, SDM 2001, Chicago, IL, USA, April 5-7, 2001 / ed. by V. Kumar, R.L. Grossman. 2001. P. 1–11. DOI: 10.1137/1.9781611972719.1.
15. Wellenzohn K., Böhlen M.H., Dignös A., *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series // Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017 / ed. by V. Markl, S. Orlando, B. Mitschang, *et al.* 2017. P. 330–341. DOI: 10.5441/002/edbt.2017.30.
16. Rakthanmanon T., Campana B.J.L., Mueen A., *et al.* Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping // ACM Trans. Knowl. Discov. Data. 2013. Vol. 7, no. 3. 10:1–10:31. DOI: 10.1145/2500489.
17. Краева Я.А., Цымблер М.Л. Совместное использование технологий MPI и OpenMP для параллельного поиска похожих подпоследовательностей в сверхбольших временных рядах на вычислительном кластере с узлами на базе многоядерных процессоров Intel Xeon Phi Knights Landing // Вычислительные методы и программирование. 2019. Т. 20, № 1. С. 29–44. DOI: 10.26089/NumMet.v20r104.

18. Kim S., Park S., Chu W.W. An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases // Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany / ed. by D. Georgakopoulos, A. Buchmann. 2001. P. 607–614. DOI: 10.1109/ICDE.2001.914875.

19. Fu A.W., Keogh E.J., Lau L.Y.H., Ratanamahatana C. Scaling and Time Warping in Time Series Querying // Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 649–660. URL: http://www.vldb.org/archives/website/2005/program/paper/thu/p649-fu.pdf.

20. Supinski B.R. de, Scogland T.R.W., Duran A., *et al.* The Ongoing Evolution of OpenMP // Proc. IEEE. 2018. Vol. 106, no. 11. P. 2004–2019. DOI: 10.1109/JPROC.2018.2853600.

21. Dau H.A., Keogh E., Kamgar K. и др. The UCR Time Series Classification Archive. 2018. URL: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (дата обращения: 12.04.2022).

22. Mueen A., Keogh E.J., Zhu Q., *et al.* Exact Discovery of Time Series Motifs // Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA. SIAM, 2009. P. 473–484. DOI: 10.1137/1.9781611972795.41.

23. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC resources of South Ural State University // 16th International Conference on Parallel Computational Technologies, PCT 2022, Dubna, Russia, March 29-31, 2022, Revised Selected Papers. Communications in Computer and Information Science. Vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

24. Khayati M., Arous I., Tymchenko Z., Cudré-Mauroux P. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams // Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294–306. DOI: 10.5555/3430915.3442429.

25. Anava O., Hazan E., Zeevi A. Online Time Series Prediction with Missing Data // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015. 2015. P. 2191–2199. URL: http://proceedings.mlr.press/v37/anava15.html.

26. Papadimitriou S., Sun J., Faloutsos C. Streaming Pattern Discovery in Multiple Time-Series // Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 697–708. DOI: 10.5555/1083592.1083674.

27. Balzano L., Chi Y., Lu Y.M. Streaming PCA and Subspace Tracking: The Missing Data Case // Proc. IEEE. 2018. Vol. 106, no. 8. P. 1293–1310. DOI: 10.1109/JPROC.2018.2847041.

28. Chlorine Dataset. URL: https://www.cs.cmu.edu/afs/cs/project/spirit-1/www/ (дата обращения: 03.09.2021).

29. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. URL: https://www.hydrodaten.admin.ch/en (дата обращения: 03.09.2021).

30. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F. Elder IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

31. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques // Transportation Research Part C: Emerging Technologies. 2018. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

32. Lefebvre A. MAREL Carnot data and metadata from Coriolis Data Centre. SEANOE. 2015. DOI: 10.17882/39754.

33. Лопухов И. Сети Real-Time Ethernet: от теории к практической реализации // СТА: Современные технологии автоматизациии. 2010. Т. 10, № 3. С. 8–15.

34. Каталог 2021. Датчики температуры Emerson. URL: https://www.c-o-k.ru/library/catalogs/emerson/110477.pdf (дата обращения: 03.09.2021).

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

Полуянов Андрей Николаевич, к.т.н., старший научный сотрудник, Институт математики им. С.Л. Соболева СО РАН (Омск, Российская Федерация)

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

DOI: 10.14529/cmse220305

PARALLEL ALGORITHM FOR REAL-TIME SENSOR DATA RECOVERY FOR A MANY-CORE PROCESSOR

© 2022 M.L. Zymbler[1], A.N. Poluyanov[2], Ya.A. Kraeva[1]

[1]*South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia),*

[2]*S.L. Sobolev Institute of Mathematics SB RAS (Pevtsova str. 13, Omsk, 644043 Russia)*

E-mail: mzym@susu.ru, andrey.poluyanov@gmail.com, kraevaya@susu.ru

Received: 30.07.2022

Currently, in many subject areas, the processing of sensor data in real time assumes imputation of values missed due to a technical failure or a human factor. This article proposes a parallel algorithm for imputation the missing values of a streaming time series in real time for a many-core processor. The algorithm employs a set of reference time series that have a semantic relationship with the original time series. The algorithm exploits the following heuristics: if there are repeated (similar) subsequences in the reference time series, then in the time series containing the missing value, repeated subsequences occur in the same time intervals. For each reference time series, a query is defined as a subsequence of a given length ending at the moment when the value in the original time series was missed. The similarity of the subsequences with the query is determined based on the DTW (Dynamic Time Warping) measure that is of quadratic computational complexity relative to the subsequence length. The algorithm employs the lower bounding technique to discard subsequences that are obviously dissimilar to the query, without calculating DTW. The lower bounds have less complexity than DTW and are calculated in parallel. The imputed value is calculated as the arithmetic mean of the last elements of the found intervals. In computational experiments, the proposed algorithm demonstrates high imputation accuracy in comparison with analogs and performance acceptable for real-time applications.

Keywords: time series, imputation of missing values, parallel algorithm, many-core CPU, DTW, lower bounding.

FOR CITATION

Zymbler M.L., Poluyanov A.N., Kraeva Ya.A. Parallel Algorithm for Real-time Sensor Data Recovery for a Many-core Processor. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 3. P. 69–90. (in Russian) DOI: 10.14529/cmse220305.

References

1. Zymbler M.L., Kraeva Y.A., Latypova E.A., *et al.* Cleaning Sensor Data in Intelligent Heating Control System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 16–36. (in Russian) DOI: 10.14529/cmse210302.
2. Ivanov S.A., Nikolskaya K.Y., Radchenko G.I., *et al.* Digital Twin of a City: Concept Overview. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2020. Vol. 9, no. 4. P. 5–23. (in Russian) DOI: 10.14529/cmse200401.
3. Epishev V.V., Isaev A.P., Miniakhmetov R.M., *et al.* Physiological Data Mining System For Elite Sports. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2013. Vol. 2, no. 1. P. 44–54. (in Russian) DOI: 10.14529/cmse130105.
4. Abdoulaev S.M., Lenskaia O.U., Gayazova A.O., *et al.* Short-Range Forecasting Algorithms Using Radar Data: Translation Estimate And Life-Cycle Composite Display. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2014. Vol. 3, no. 1. P. 17–32. (in Russian) DOI: 10.14529/cmse140102.
5. Berndt D.J., Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series. Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03 / ed. by U.M. Fayyad, R. Uthurusamy. 1994. P. 359–370.
6. Ding H., Trajcevski G., Scheuermann P., *et al.* Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. VLDB Endow. 2008. Vol. 1, no. 2. P. 1542–1552. DOI: 10.14778/1454159.1454226.
7. Zymbler M.L., Poluyanov A.N. Parallel algorithm for imputation of missing values in a streaming time series in real time. Parallel Computational Technologies – 16th International Conference, PCT 2022, Dubna, Russia, March 29-31, 2022. Short papers and posters. Chelyabinsk: SUSU Publishing Center. 2022. P. 128–140. (in Russian) DOI: 10.14529/pct2022.
8. Khayati M., Lerner A., Tymchenko Z., Cudré-Mauroux P. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series. Proc. VLDB Endow. 2020. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.

9. Batista G.E.A.P.A., Monard M.C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning. Appl. Artif. Intell. 2003. Vol. 17, no. 5-6. P. 519–533. DOI: 10.1080/713827181.

10. Troyanskaya O.G., Cantor M.N., Sherlock G., *et al.* Missing value estimation methods for DNA microarrays. Bioinform. 2001. Vol. 17, no. 6. P. 520–525. DOI: 10.1093/bioinformatics/17.6.520.

11. Hsu H., Yang A.C., Lu M. KNN-DTW Based Missing Value Imputation for Microarray Time Series Data. J. Comput. 2011. Vol. 6, no. 3. P. 418–425. DOI: 10.4304/jcp.6.3.418-425.

12. Phan T., Poisson É.C., Bigand A., Lefebvre A. DTW-Approach for uncorrelated multivariate time series imputation. 27th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2017, Tokyo, Japan, September 25-28, 2017 / ed. by N. Ueda, S. Watanabe, T. Matsui, *et al.* 2017. P. 1–6. DOI: 10.1109/MLSP.2017.8168165.

13. Phan T., Caillault É.P., Lefebvre A., Bigand A. Dynamic time warping-based imputation for univariate time series data. Pattern Recognit. Lett. 2020. Vol. 139. P. 139–147. DOI: 10.1016/j.patrec.2017.08.019.

14. Keogh E.J., Pazzani M.J. Derivative Dynamic Time Warping. Proceedings of the 1st SIAM International Conference on Data Mining, SDM 2001, Chicago, IL, USA, April 5-7, 2001 / ed. by V. Kumar, R.L. Grossman. 2001. P. 1–11. DOI: 10.1137/1.9781611972719.1.

15. Wellenzohn K., Böhlen M.H., Dignös A., *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series. Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017 / ed. by V. Markl, S. Orlando, B. Mitschang, *et al.* 2017. P. 330–341. DOI: 10.5441/002/edbt.2017.30.

16. Rakthanmanon T., Campana B.J.L., Mueen A., *et al.* Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping. ACM Trans. Knowl. Discov. Data. 2013. Vol. 7, no. 3. 10:1–10:31. DOI: 10.1145/2500489.

17. Kraeva Y.A., Zymbler M.L. The use of MPI and OpenMP technologies for subsequence similarity search in very long time series on a computer cluster system with nodes based on the Intel Xeon Phi Knights Landing many-core processor. Numerical Methods and Programming. 2019. Vol. 20, no. 1. P. 29–44. (in Russian) DOI: 10.26089/NumMet.v20r104.

18. Kim S., Park S., Chu W.W. An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany / ed. by D. Georgakopoulos, A. Buchmann. 2001. P. 607–614. DOI: 10.1109/ICDE.2001.914875.

19. Fu A.W., Keogh E.J., Lau L.Y.H., Ratanamahatana C. Scaling and Time Warping in Time Series Querying. Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 649–660. URL: http://www.vldb.org/archives/website/2005/program/paper/thu/p649-fu.pdf.

20. Supinski B.R. de, Scogland T.R.W., Duran A., *et al.* The Ongoing Evolution of OpenMP. Proc. IEEE. 2018. Vol. 106, no. 11. P. 2004–2019. DOI: 10.1109/JPROC.2018.2853600.

21. Dau H.A., Keogh E., Kamgar K., *et al.* The UCR Time Series Classification Archive. 2018. URL: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (accessed: 12.04.2022).

22. Mueen A., Keogh E.J., Zhu Q., *et al.* Exact Discovery of Time Series Motifs. Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA. SIAM, 2009. P. 473–484. DOI: 10.1137/1.9781611972795.41.

23. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC resources of South Ural State University. 16th International Conference on Parallel Computational Technologies, PCT 2022, Dubna, Russia, March 29-31, 2022, Revised Selected Papers. Communications in Computer and Information Science. Vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

24. Khayati M., Arous I., Tymchenko Z., Cudré-Mauroux P. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams. Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294–306. DOI: 10.5555/3430915.3442429.

25. Anava O., Hazan E., Zeevi A. Online Time Series Prediction with Missing Data. Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015. 2015. P. 2191–2199. URL: http://proceedings.mlr.press/v37/anava15.html.

26. Papadimitriou S., Sun J., Faloutsos C. Streaming Pattern Discovery in Multiple Time-Series. Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 697–708. DOI: 10.5555/1083592.1083674.

27. Balzano L., Chi Y., Lu Y.M. Streaming PCA and Subspace Tracking: The Missing Data Case. Proc. IEEE. 2018. Vol. 106, no. 8. P. 1293–1310. DOI: 10.1109/JPROC.2018.2847041.

28. Chlorine Dataset. URL: https://www.cs.cmu.edu/afs/cs/project/spirit-1/www/ (accessed: 03.09.2021).

29. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. URL: https://www.hydrodaten.admin.ch/en (accessed: 03.09.2021).

30. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F. Elder IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

31. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques. Transportation Research Part C: Emerging Technologies. 2018. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

32. Lefebvre A. MAREL Carnot data and metadata from Coriolis Data Centre. SEANOE. 2015. DOI: 10.17882/39754.

33. Lopukhov I. Real-Time Ethernet network: from theory to practical implementation. MAT: Modern automation technologies. 2010. Vol. 10, no. 3. P. 8–15.

34. Catalogue 2021. Emerson temperature sensors. URL: https://www.c-o-k.ru/library/catalogs/emerson/110477.pdf (accessed: 03.09.2021).