МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет (национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

Разработка параллельного алгоритма поиска самой похожей подпоследовательности временного ряда для многоядерных процессоров Intel Xeon Phi (Knights Landing)

Я.А. Краева

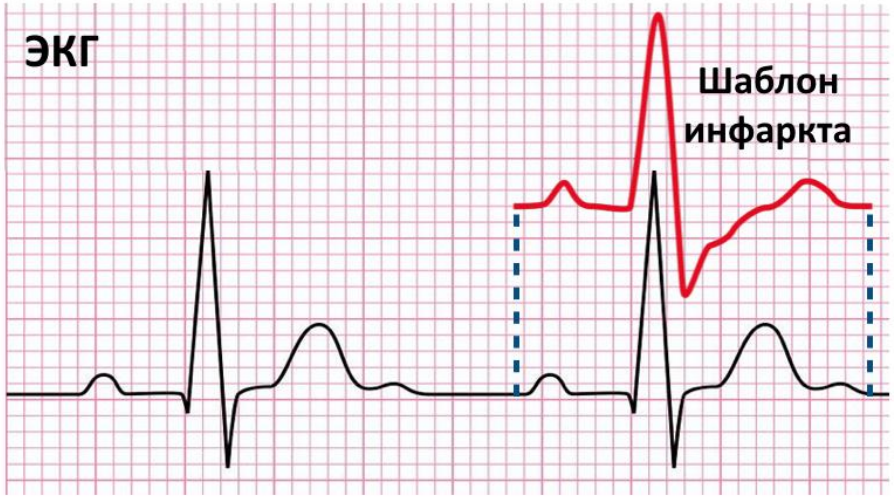
Рецензент:
Зам. директора по ИТ Челябинского обл. мед. инф.-аналитического центра
А.С. Староверов

Научный руководитель:
к.ф.-м.н., доцент
М.Л. Цымблер

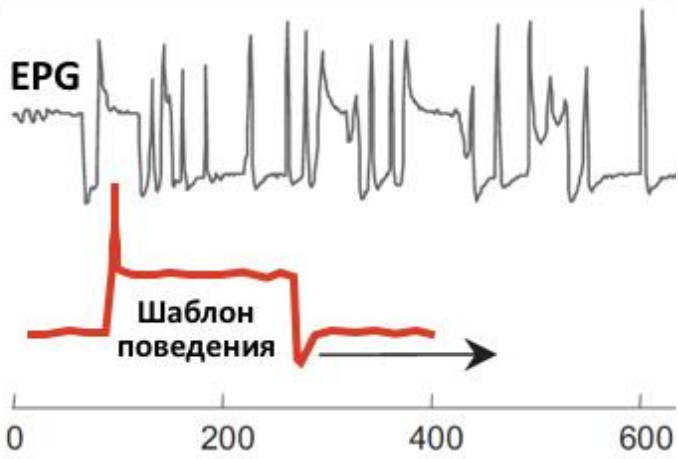
Челябинск-2018

Актуальность

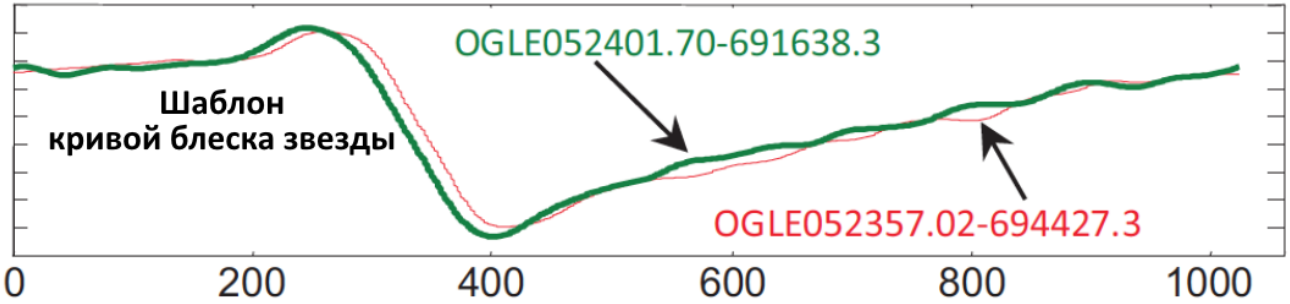
Медицина: обнаружение предынфарктного состояния



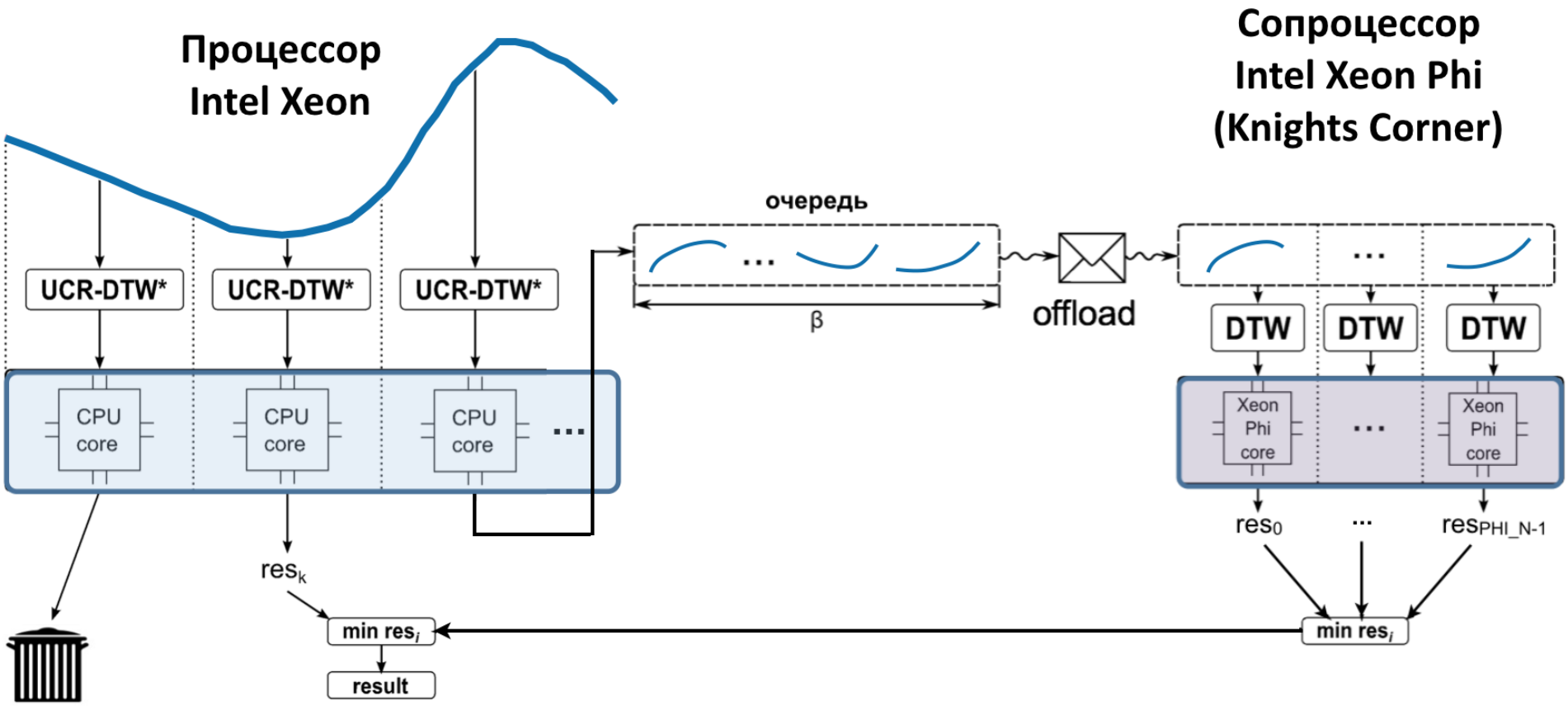
Энтомология: изучение поведения насекомых-вредителей



Астрономия: классификация кривых блеска для исследования характеристик сравнительно близких звезд



Предыдущие исследования



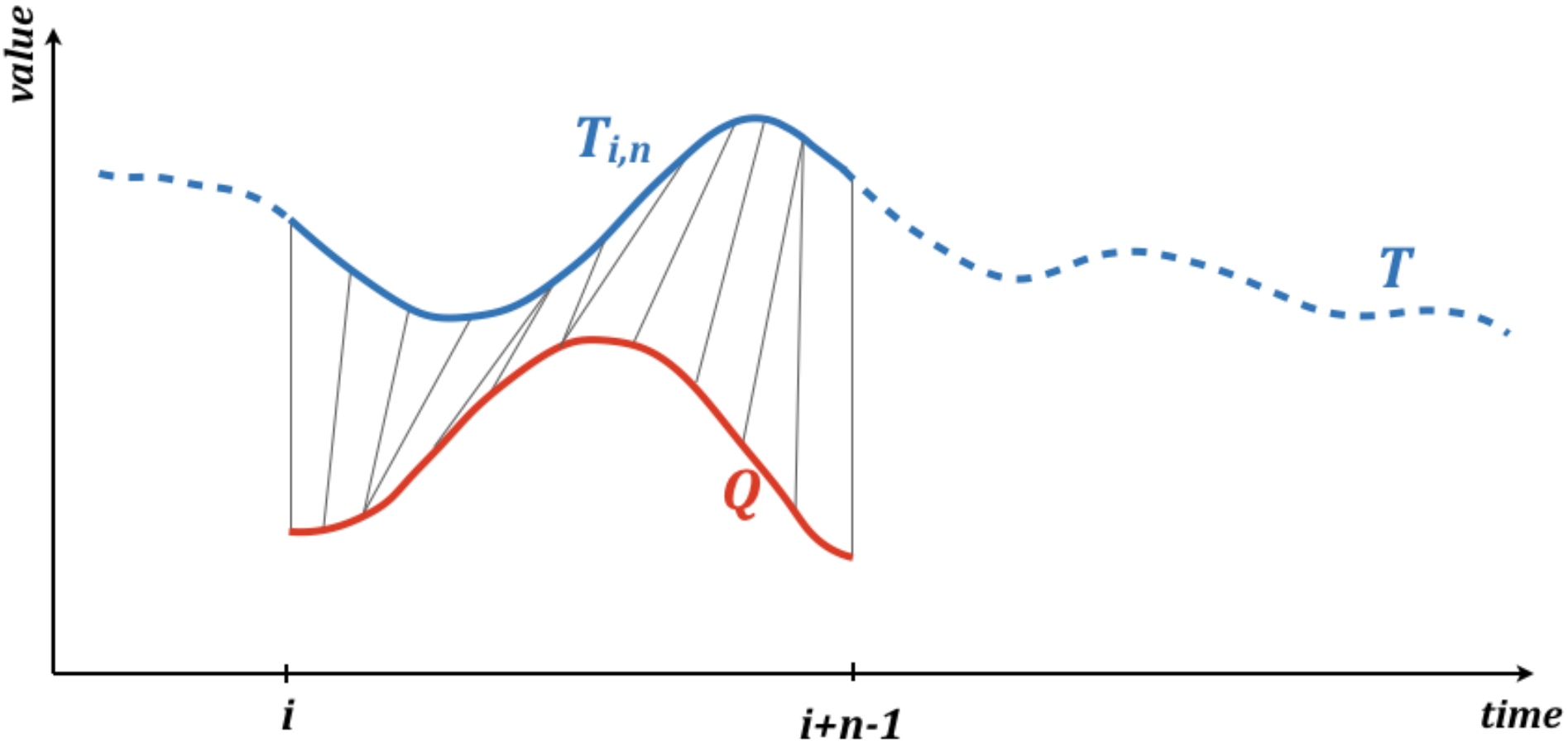
Цель и задачи исследования

Цель: разработать параллельный алгоритм поиска самой похожей подпоследовательности временного ряда для многоядерных процессоров Intel Xeon Phi (Knights Landing).

Задачи:

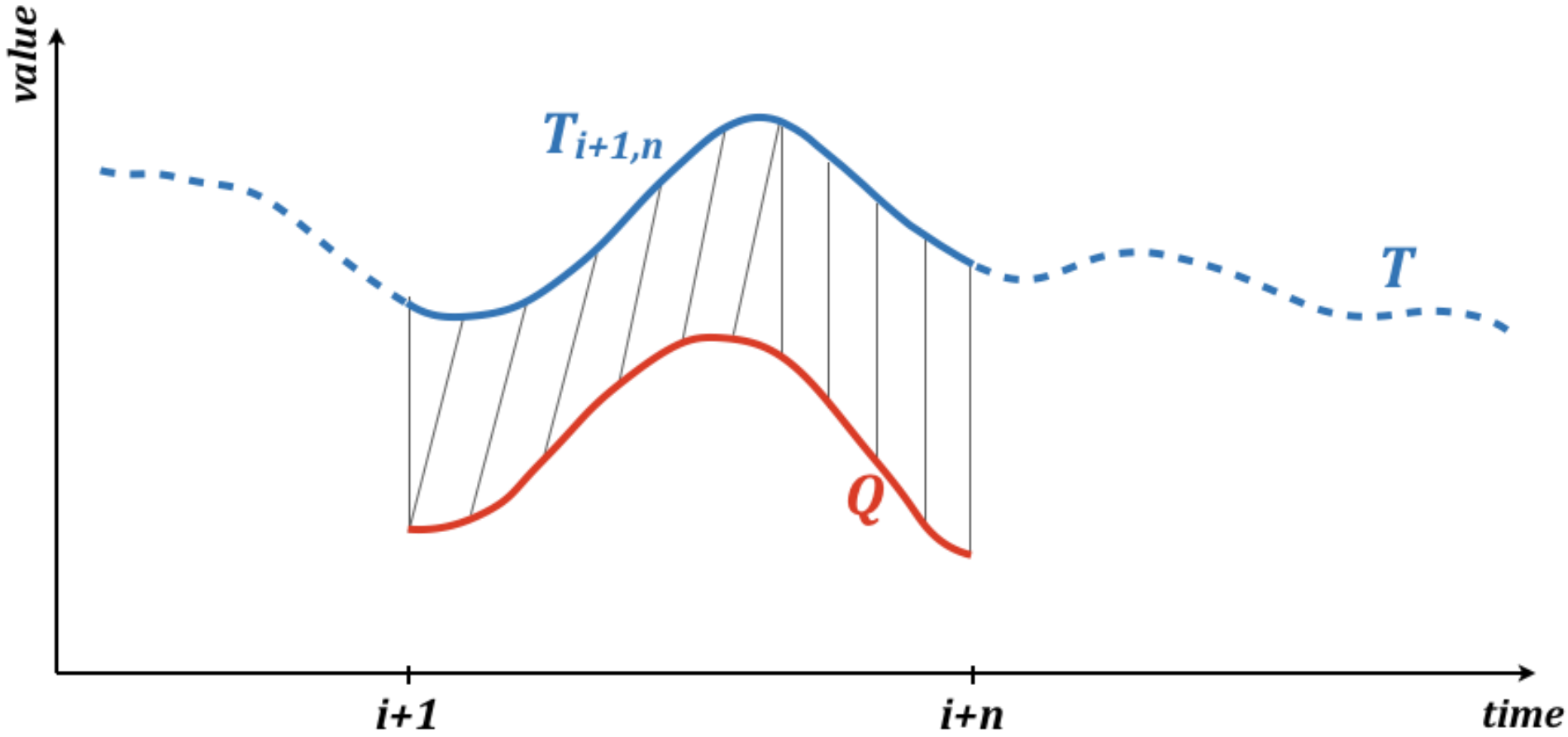
1. Провести обзор параллельных алгоритмов поиска похожих подпоследовательностей временных рядов, изучить аппаратную архитектуру и программную модель системы Intel Xeon Phi (Knights Landing).
2. Спроектировать и реализовать параллельный алгоритм поиска самой похожей подпоследовательности временного ряда для многоядерных процессоров Intel Xeon Phi (Knights Landing).
3. Провести вычислительные эксперименты по анализу эффективности разработанного алгоритма.

Поиск самой похожей подпоследовательности



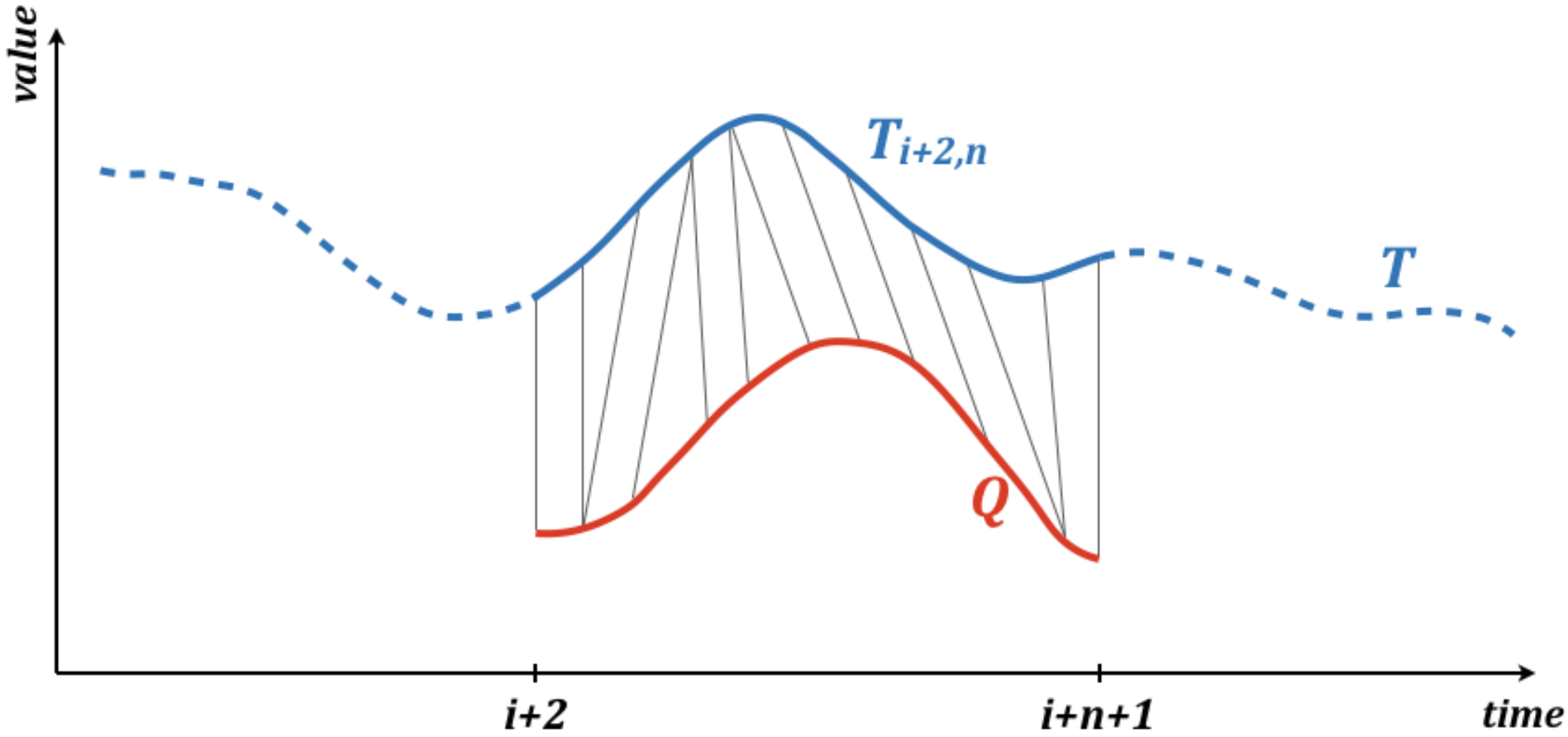
$$\exists i \, \forall k \; D(Q, T_{i,n}) < D(Q, T_{k,n}); 1 < i, k < m - n + 1$$

Поиск самой похожей подпоследовательности



$$\exists i \; \forall k \; D(Q, T_{i,n}) < D(Q, T_{k,n}); 1 < i, k < m - n + 1$$

Поиск самой похожей подпоследовательности



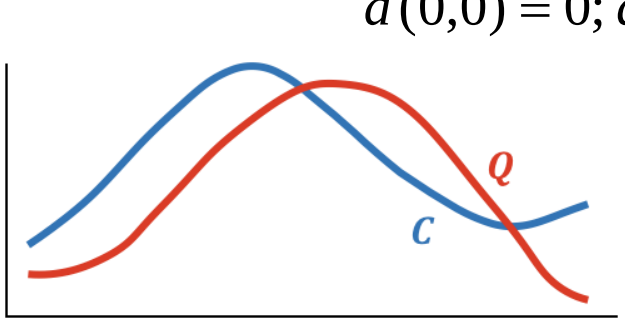
$$\exists i \; \forall k \; D(Q, T_{i,n}) < D(Q, T_{k,n}); 1 < i, k < m - n + 1$$

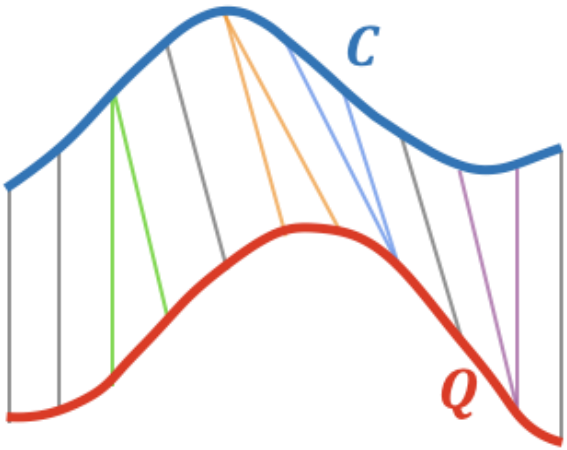
Мера схожести DTW

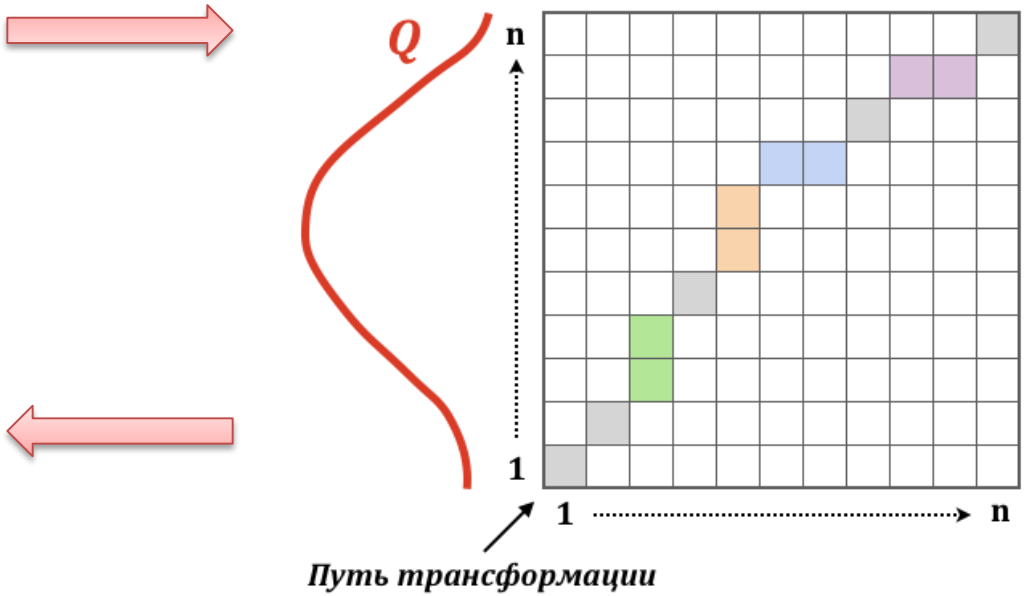
$$DTW(Q,C) = d(n,n),$$

$$d(i,j) = (q_i - c_j)^2 + \min\begin{cases} d(i-1,j) \\ d(i,j-1) \\ d(i-1,j-1), \end{cases}$$

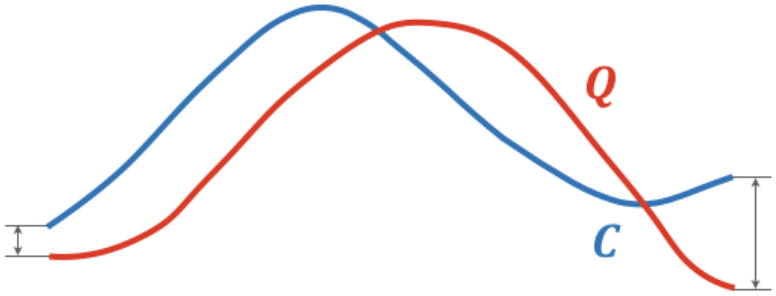
$$d(0,0) = 0; d(i,0) = d(0,j) = \infty; 1 \le i, j \le n.$$





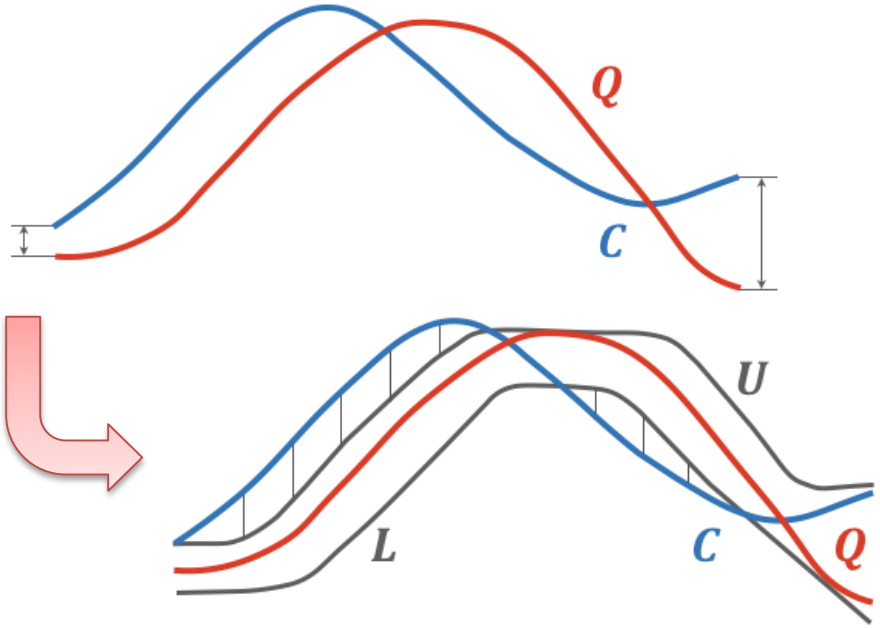


Оценки схожести снизу



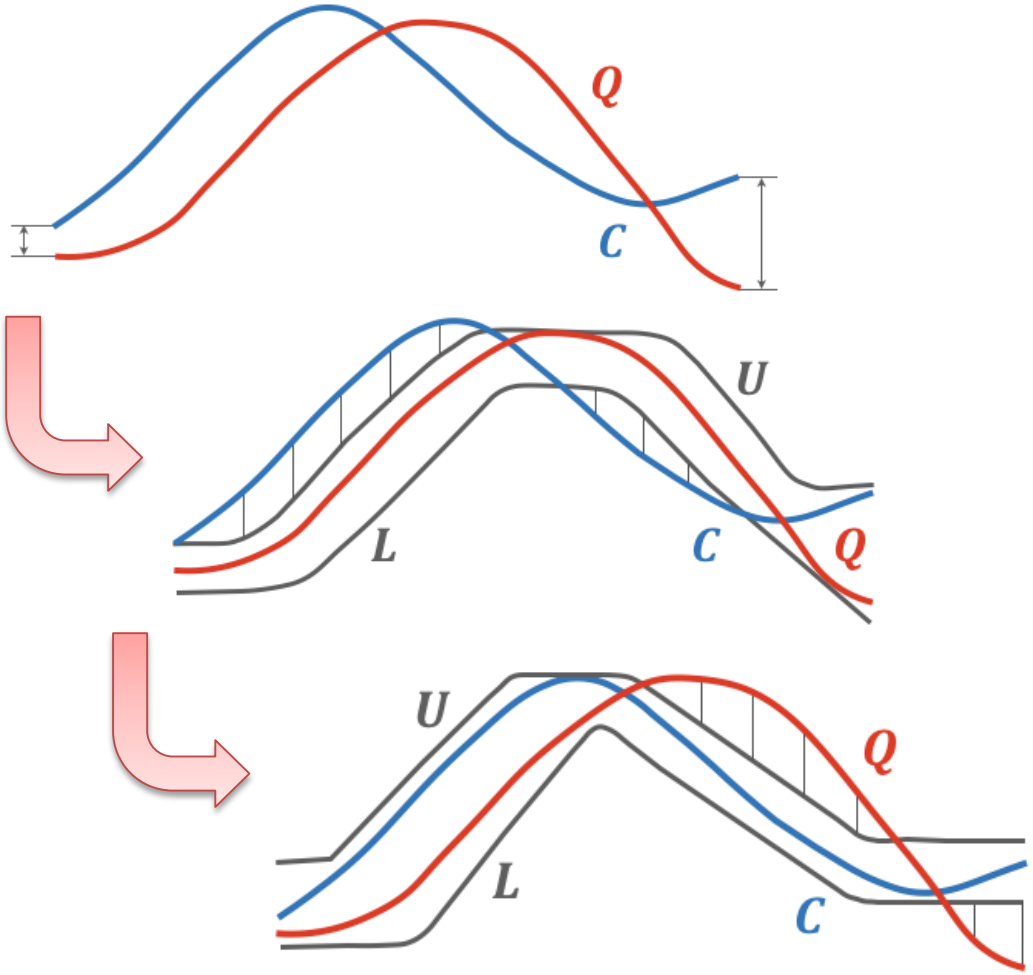
Оценка схожести снизу	Сложность
$LB_{Kim}(Q,C) = (q_1 - c_1)^2 + (q_n - c_n)^2$	$O(1)$

Оценки схожести снизу



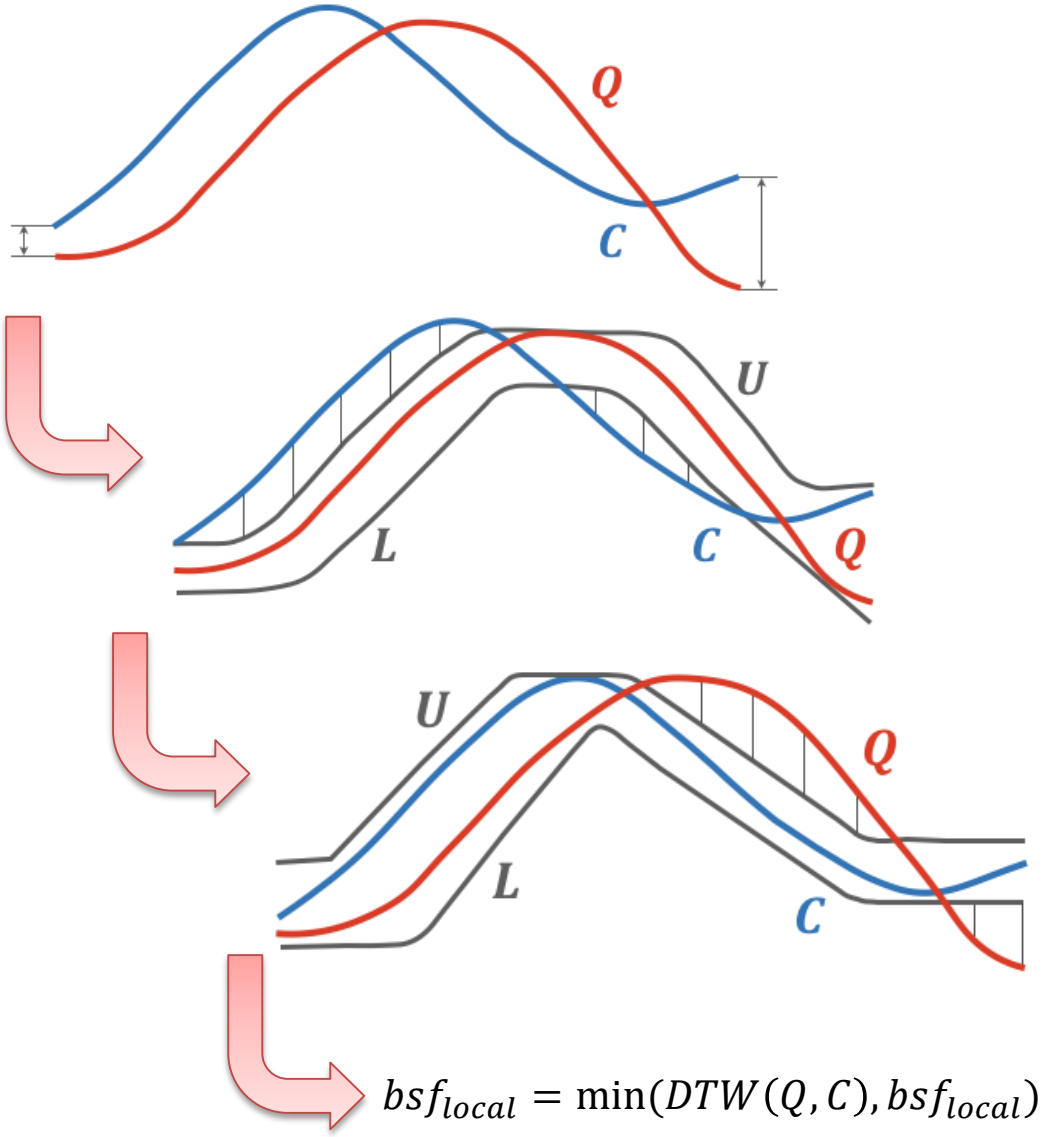
Оценка схожести снизу	Сложность
$LB_{Kim}(Q,C) = (q_1 - c_1)^2 + (q_n - c_n)^2$	$O(1)$
$LB_{Keogh}(Q,C) = \sum_{i=1}^{n} \begin{cases} (c_i - U_i)^2 \; if \; c_i > U_i \\ (c_i - L_i)^2 \; if \; c_i < L_i \\ 0 \; otherwise \end{cases}$	$O(n)$

Оценки схожести снизу



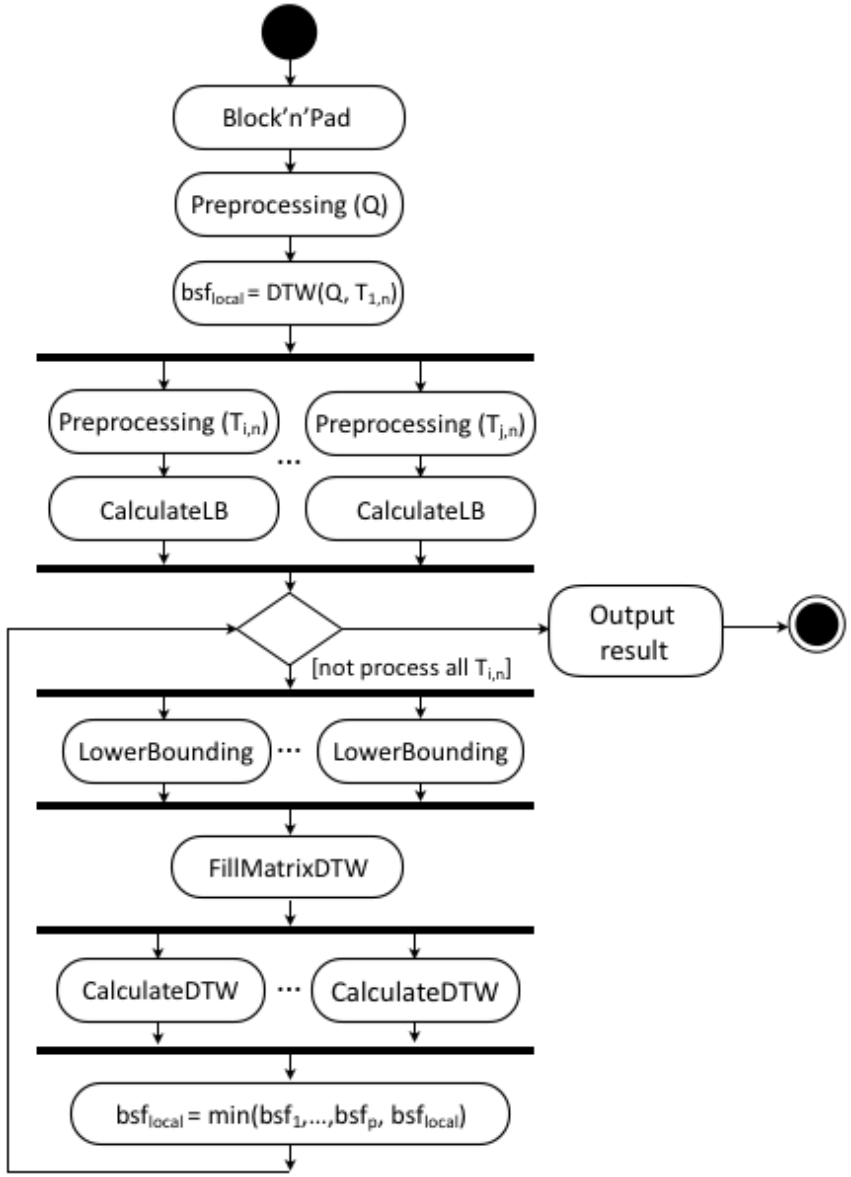
Оценка схожести снизу	Сложность
$LB_{Kim}(Q,C) = (q_1 - c_1)^2 + (q_n - c_n)^2$	$O(1)$
$LB_{Keogh}(Q,C) = \sum_{i=1}^{n} \begin{cases} (c_i - U_i)^2 \; if \; c_i > U_i \\ (c_i - L_i)^2 \; if \; c_i < L_i \\ 0 \; otherwise \end{cases}$	$O(n)$
$LB_{Keogh}(C,Q) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 \; if \; q_i > U_i \\ (q_i - L_i)^2 \; if \; q_i < L_i \\ 0 \; otherwise \end{cases}$	$O(n)$

Оценки схожести снизу



Оценка схожести снизу	Сложность
$LB_{Kim}(Q,C) = (q_1 - c_1)^2 + (q_n - c_n)^2$	$O(1)$
$LB_{Keogh}(Q,C) = \sum_{i=1}^{n} \begin{cases} (c_i - U_i)^2 \ if \ c_i > U_i \\ (c_i - L_i)^2 \ if \ c_i < L_i \\ 0 \ otherwise \end{cases}$	$O(n)$
$LB_{Keogh}(C,Q) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 \ if \ q_i > U_i \\ (q_i - L_i)^2 \ if \ q_i < L_i \\ 0 \ otherwise \end{cases}$	$O(n)$
$DTW(Q,C)$	$O(n^2)$

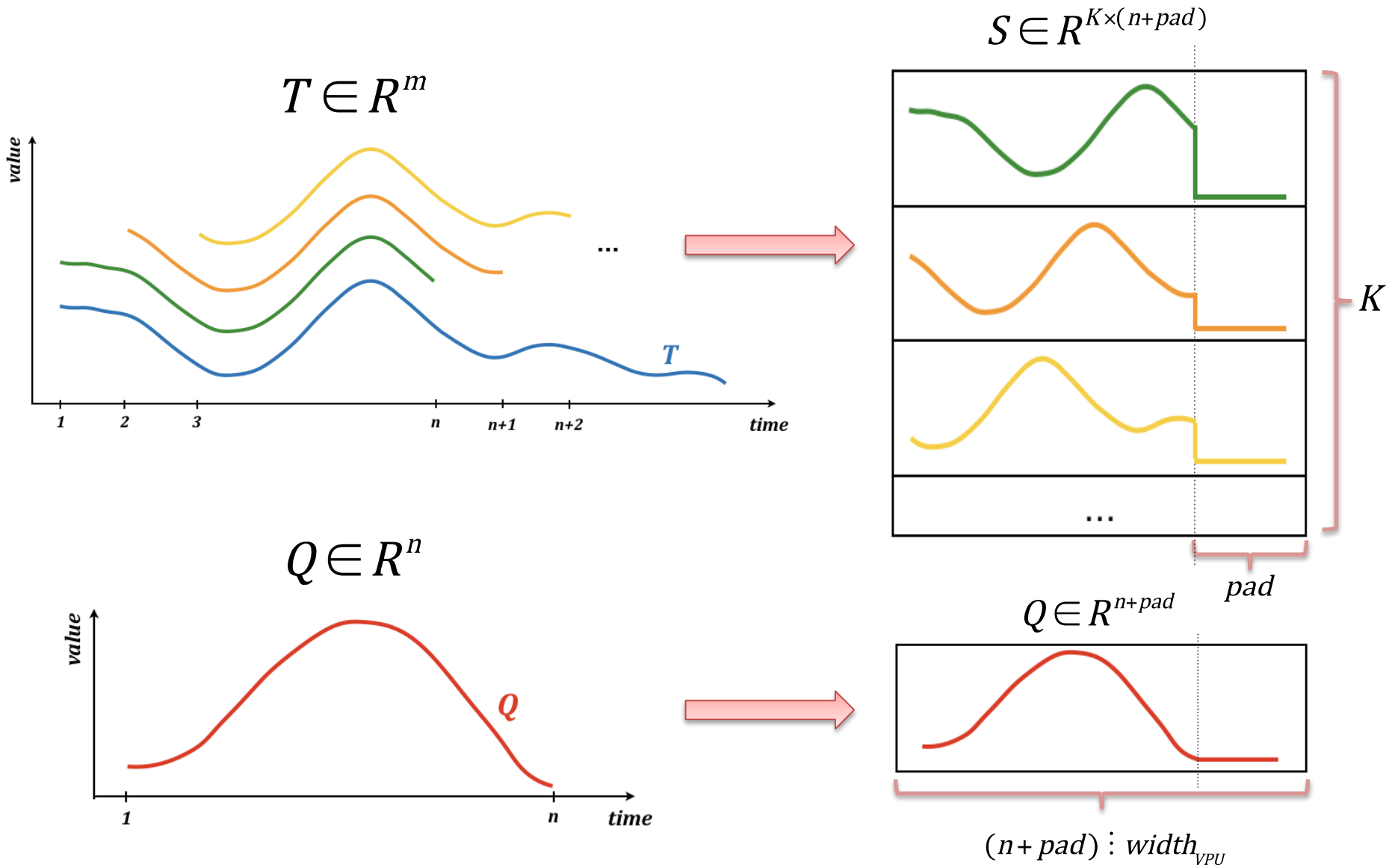
Новый параллельный алгоритм



Основные идеи:

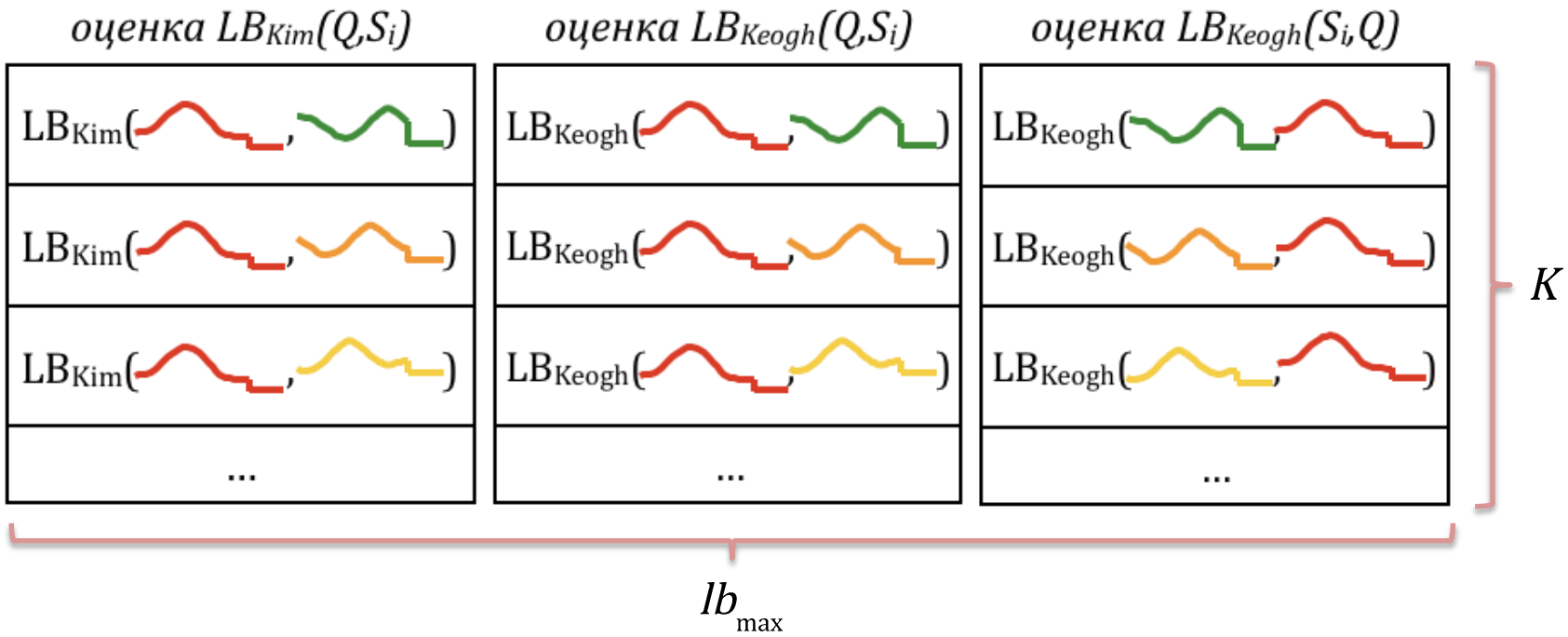
1. Выравнивание данных
2. Однократное вычисление нижних оценок
3. Сбалансированная нагрузка нитей

Матрица подпоследовательностей



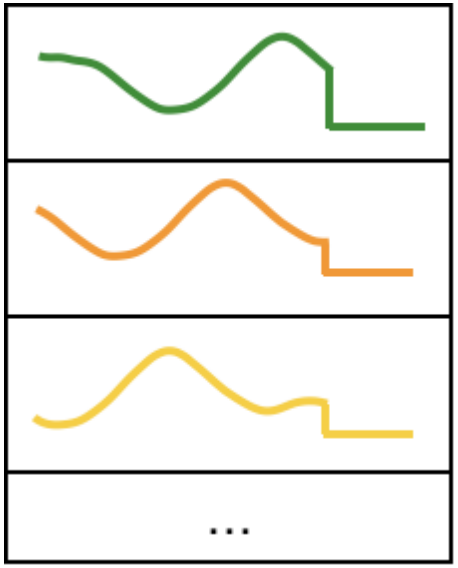
Матрица нижних оценок схожести

$$LB \in R^{K \times lb_{\max}}$$



Матрица битовой карты

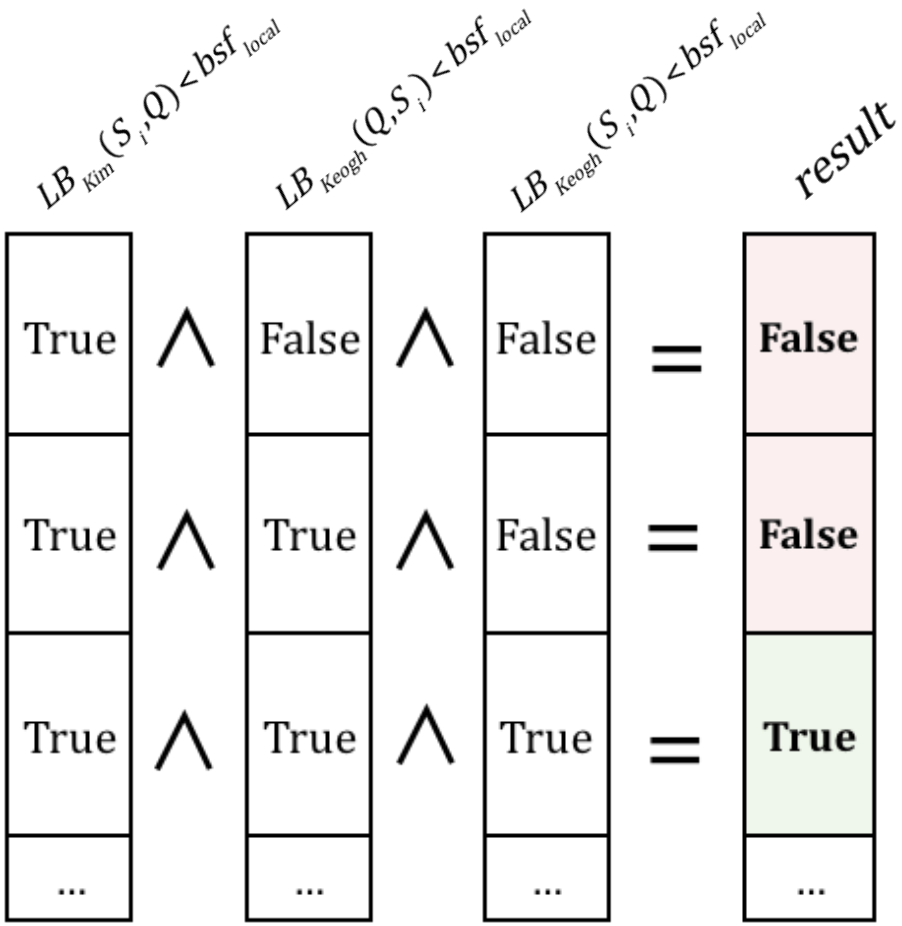
$$S \in R^{K \times (n+pad)}$$

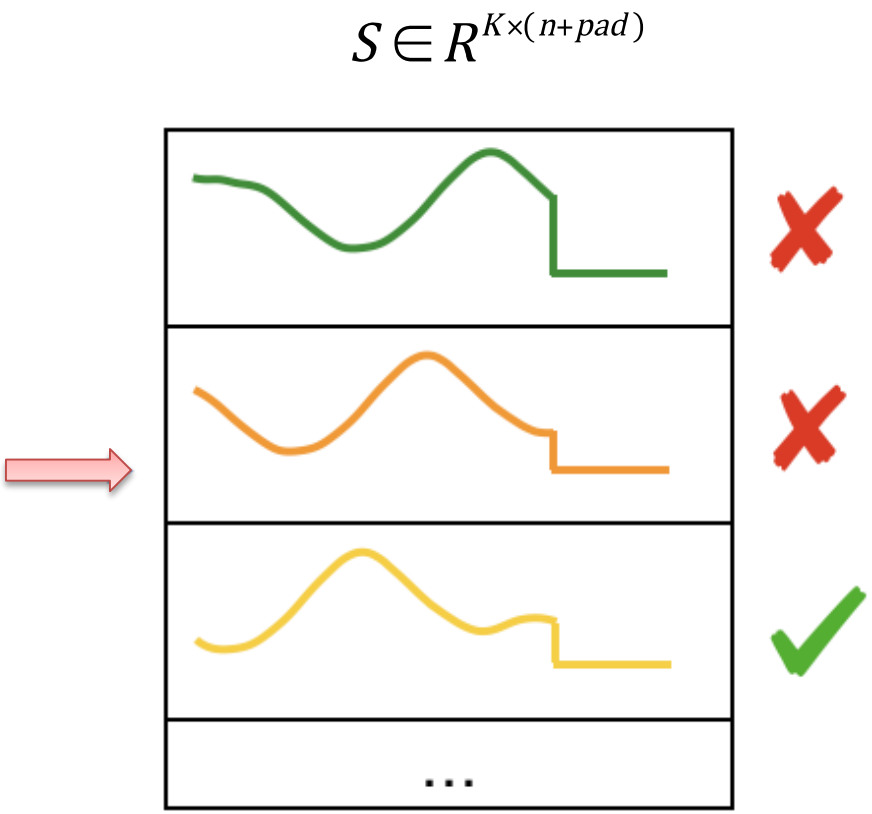


$$BM \in R^{K \times lb_{\max}}$$

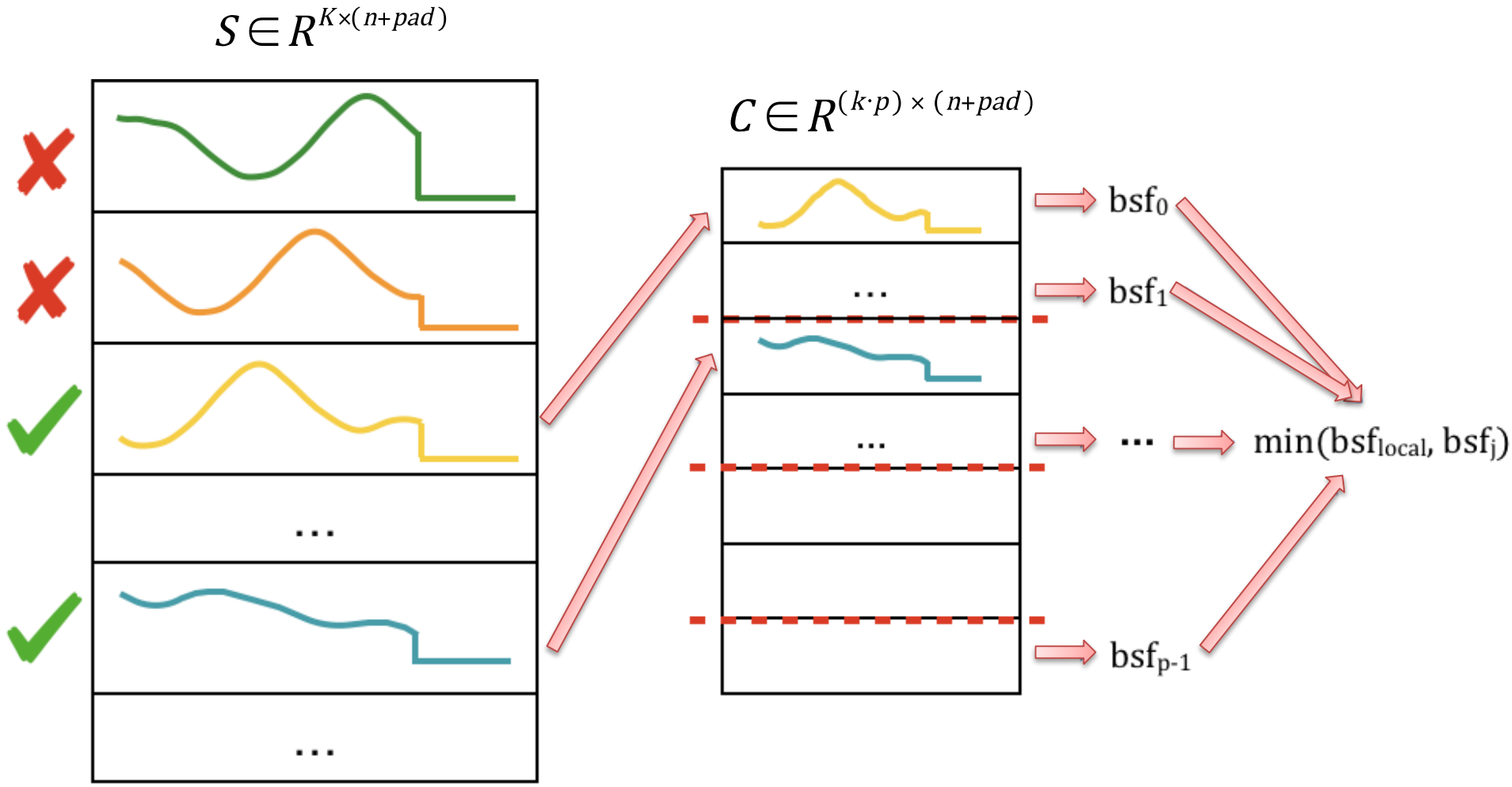
$LB_{Kim}(S_i, Q) < bsf_{local}$	$LB_{Keogh}(Q, S_i) < bsf_{local}$	$LB_{Keogh}(S_i, Q) < bsf_{local}$
True	False	False
True	True	False
True	True	True
...	...	...

Отбрасывание кандидатов





Матрица кандидатов



Реализация

- **Язык программирования:** C
- **Используемый стандарт для распараллеливания:** OpenMP 4.0
- **Пакет компиляторов:** Intel C++ Composer XE 2015 (Intel C++ Compiler v. 15.0)
- **Количество строк кода:** 2600
- **Текст программы размещен в репозитории:** bitbucket.org/YanaKraeva/phibestmatch

Эксперименты

Аппаратная платформа

Процессор: **Intel Xeon Phi SE10X**

Количество физических ядер: **61**

Количество нитей на ядро: **4**

Количество логических ядер: **244**

Тактовая частота: **1.1 ГГц**

Производительность: **1.076 TFLOPS**

Ширина векторных регистров: **512 бит**

Наборы данных

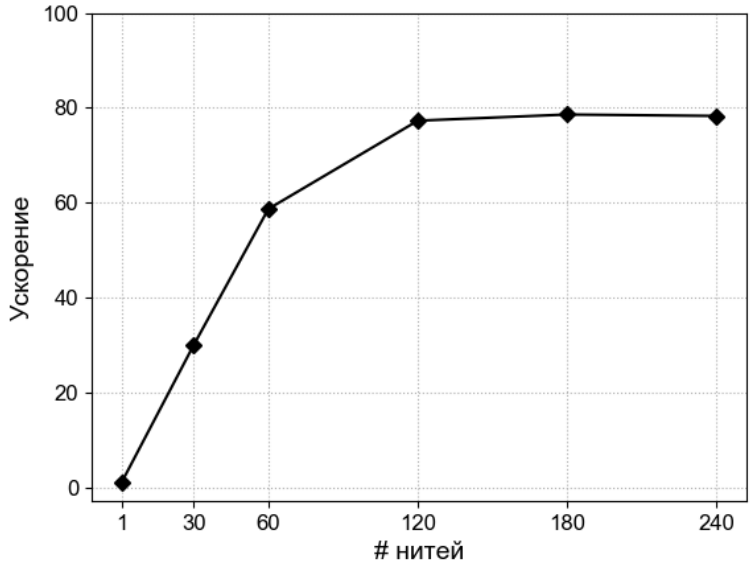
Набор	Длина T	Длина Q	Семантика
Random walk	10^6	128	Синтетические данные, сгенерированные математической моделью случайного блуждания [Rakthanmanon et al. 2012]
EPG	$2.5*10^5$	360	Реальные данные из области энтомологии [Sart et al. 2010]

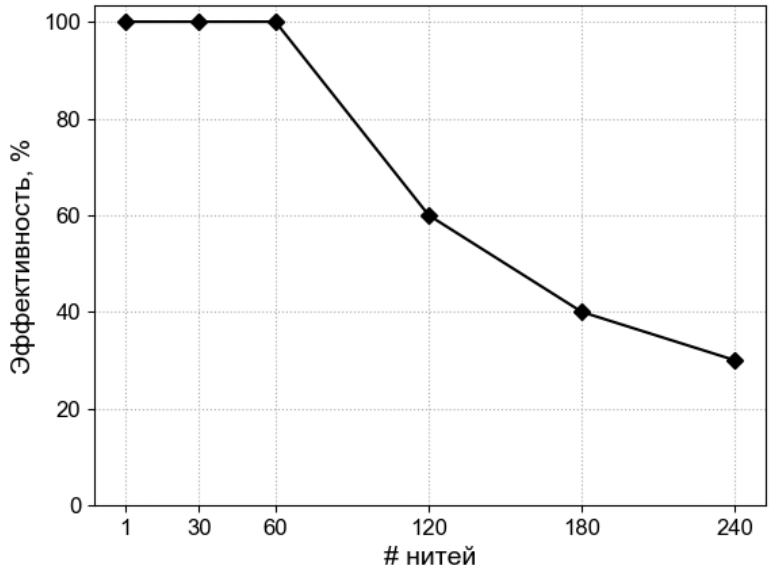
Исследуемые показатели масштабируемости

Ускорение $a(k) = \frac{t_1}{t_k}$ *Эффективность* $e(k) = \frac{a(k)}{k}$

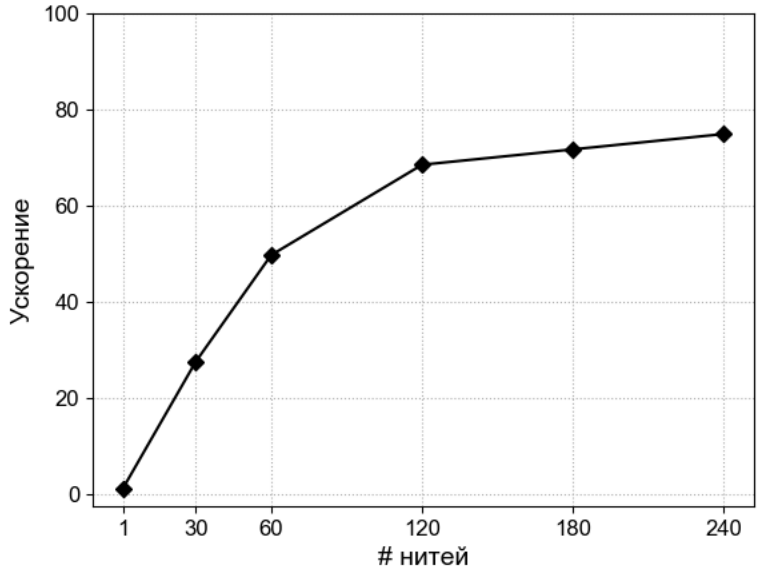
Эксперименты: ускорение и эффективность

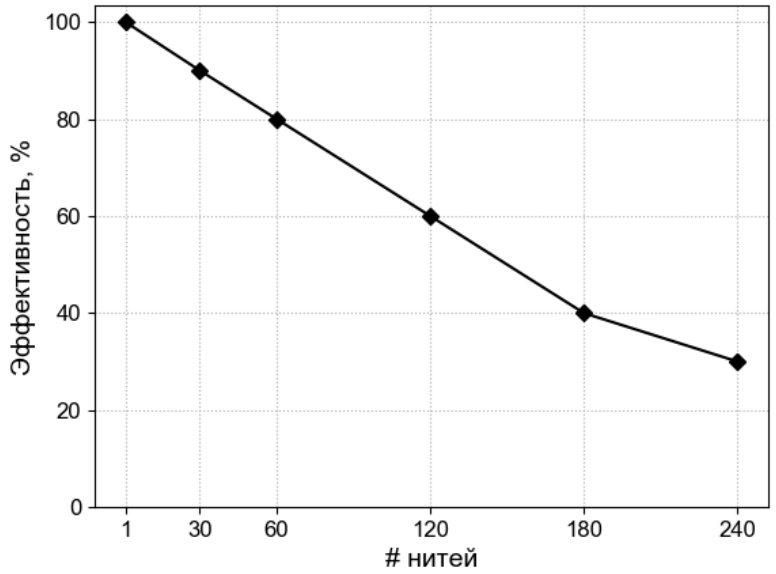
Random walk





EPG





Публикация, апробация

- Подана статья на XX International Conference "Data Analytics and Management in Data Intensive Domains 2018" (9-12 October 2018, Moscow).
- Сделан доклад на 71 студенческой научной конференции ЮУрГУ (Челябинск, 14 мая 2018 г.).
- Получен диплом 2 степени на XIII Уральской выставке НТТМ «Евразийские ворота России – Шаг в будущее» (Челябинск, 2-4 апреля 2018 г.).

Заключение

1. Выполнен обзор параллельных алгоритмов поиска похожих подпоследовательностей временных рядов.
2. Разработан параллельный алгоритм поиска самой похожей подпоследовательности временного ряда для многоядерных процессоров Intel Xeon Phi (Knights Landing).
3. Проведены вычислительные эксперименты на реальных и синтетических данных, показавшие хорошую масштабируемость разработанного алгоритма.

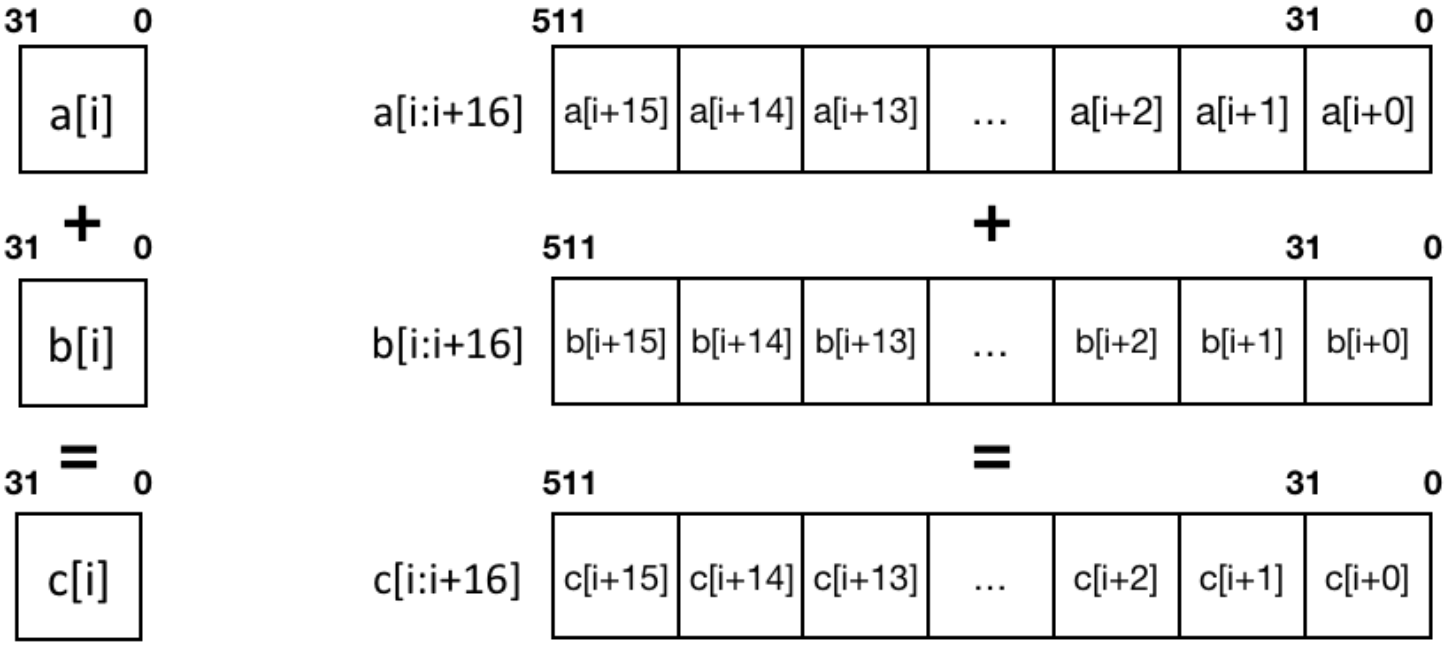
Векторизация

Скалярная операция

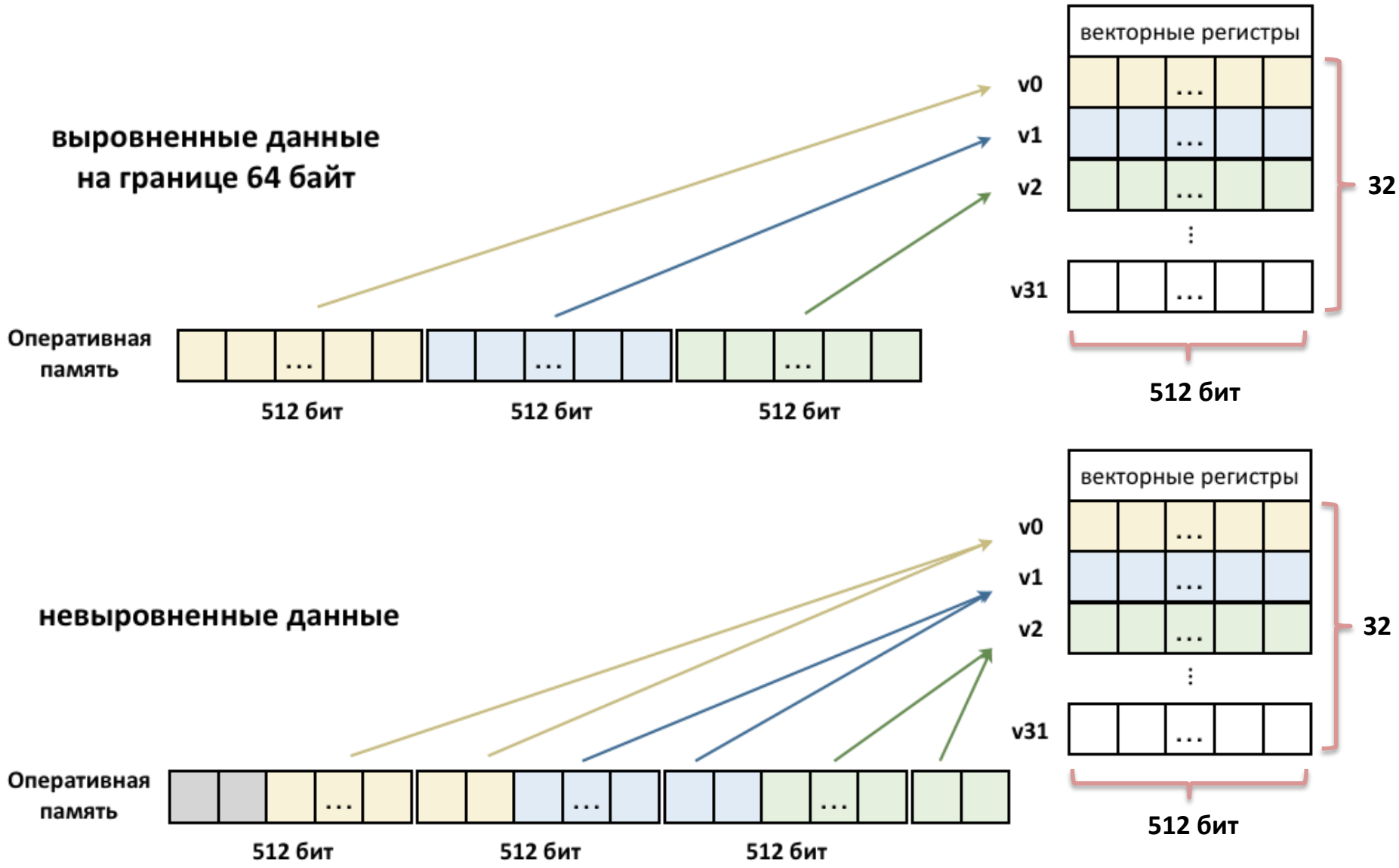
```
float a[N], b[N], c[N];
for (i = 0; i < N; i++)
      c[i] = a[i] + b[i];
```

Векторная операция

```
float a[N] __attribute__ ((aligned(64)));
float b[N] __attribute__ ((aligned(64)));
float c[N] __attribute__ ((aligned(64)));
for (i = 0; i < N; i=+16)
      c[i:i+16] = a[i:i+16] + b[i:i+16];
```

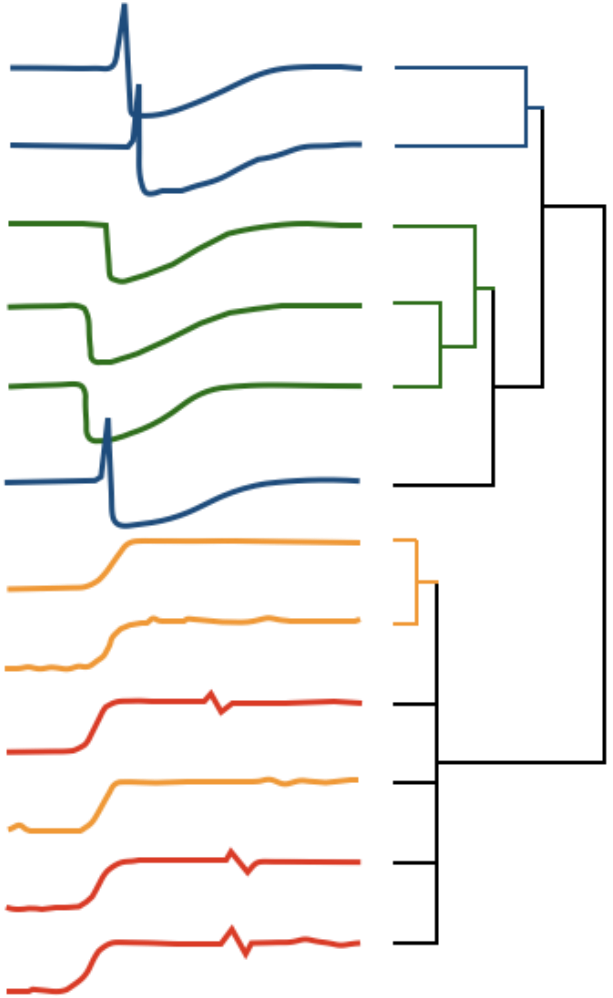


Выравнивание данных



Преимущество DTW над Евклидом

Евклидова метрика



DTW