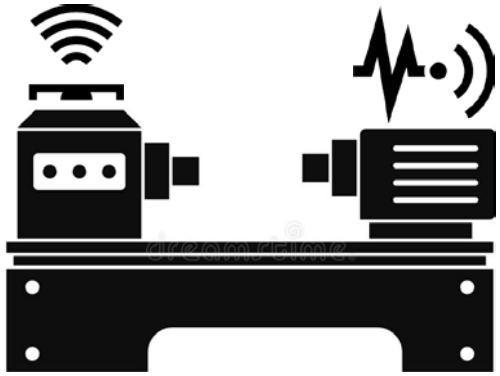
Международная научная конференция
Параллельные вычислительные технологии (ПаВТ'2022)
Дубна, 29–31 марта 2022 г.

Параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени

М.Л. Цымблер[1], А.Н. Полуянов[2],
[1]Южно-Уральский государственный университет (Челябинск),
[2]Институт математики им. С.Л. Соболева СО РАН (Омск)

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

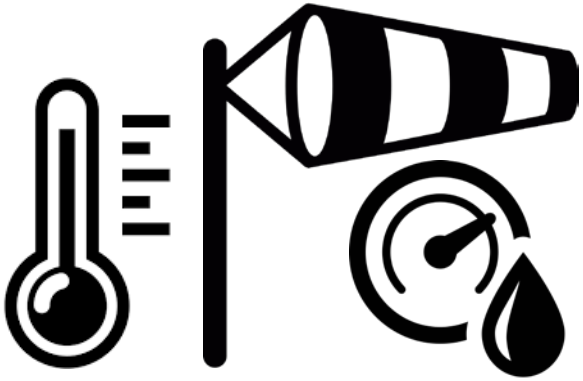
Восстановление потоковых данных онлайн



Предиктивное ТО промышленных агрегатов

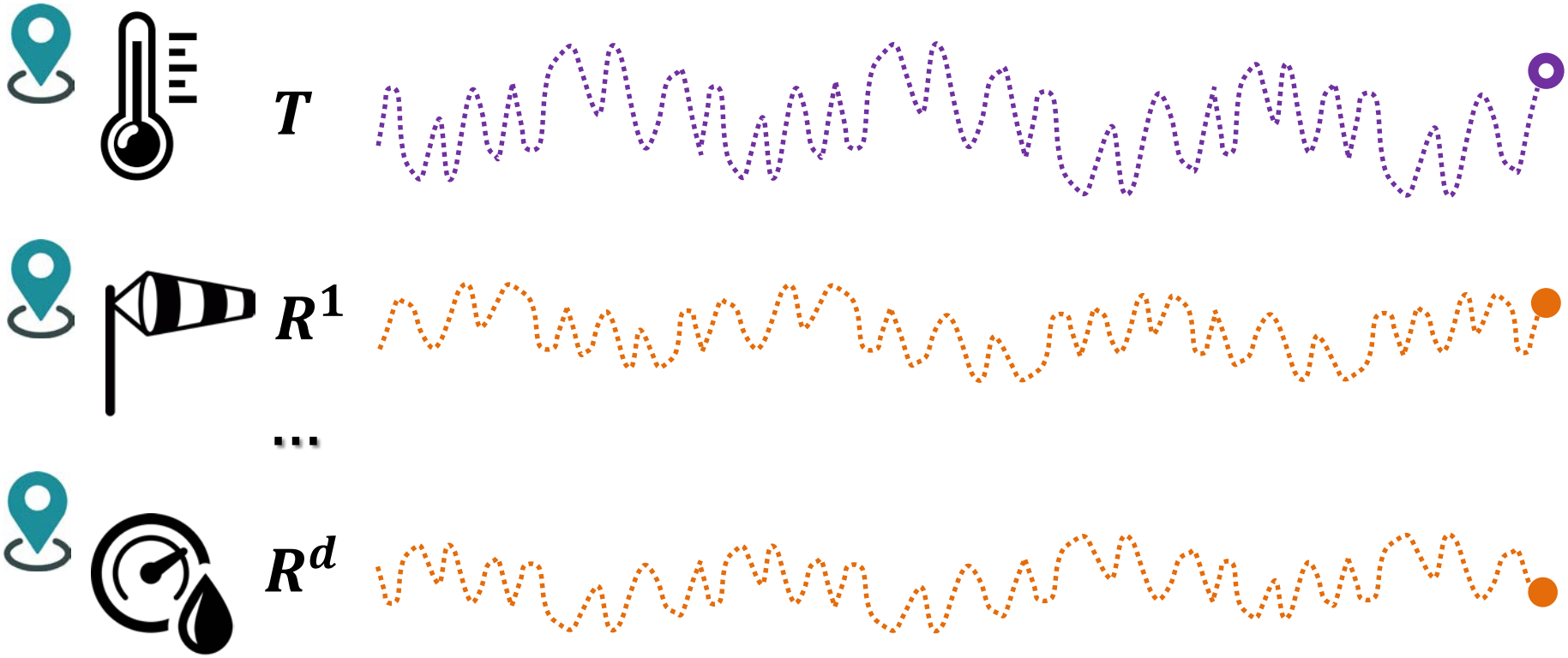
Интернет вещей

Персональная медицина



Мониторинг погоды и моделирование климата

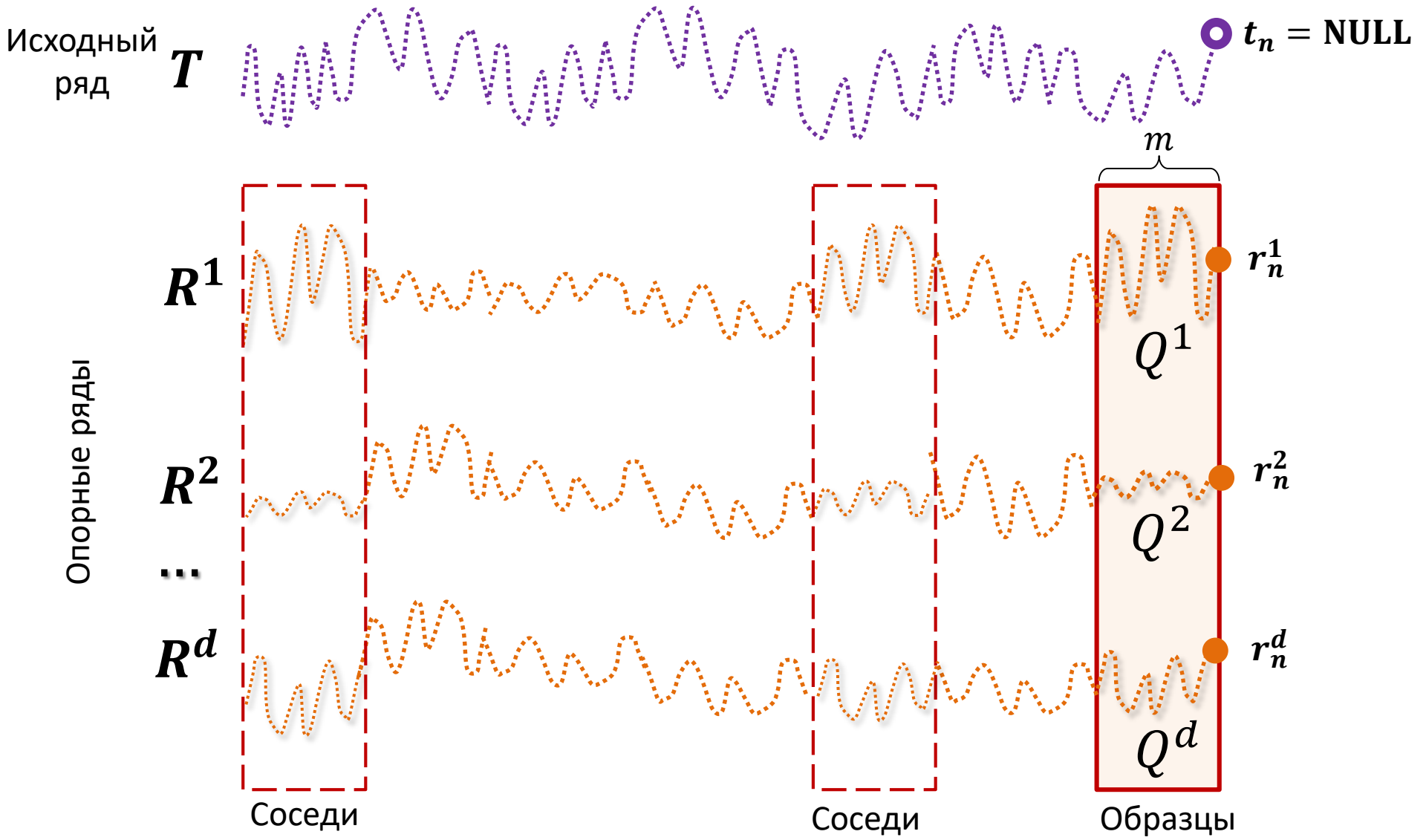
Восстановление по опорным рядам



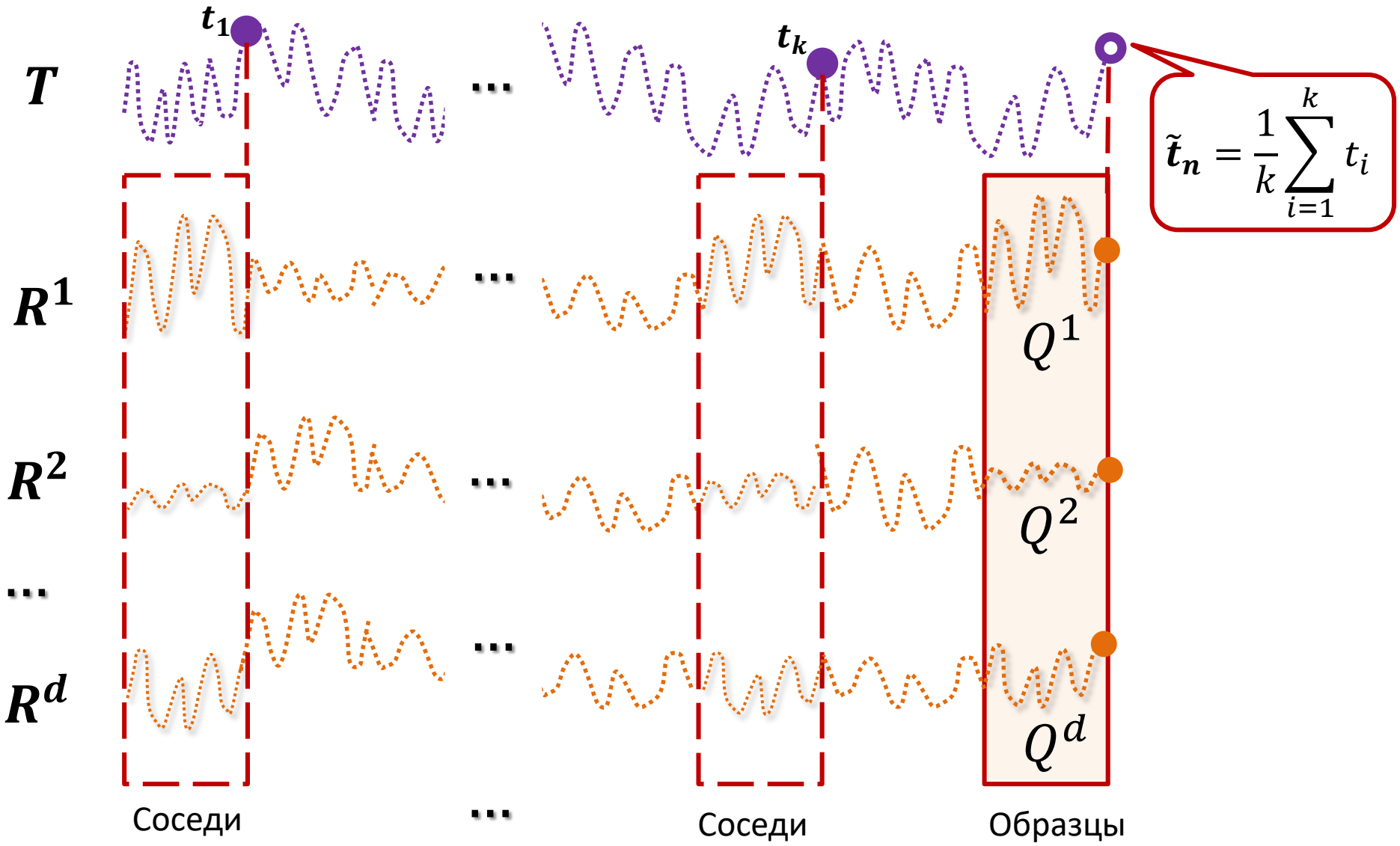
Эвристика

похожие промежутки в исходном ряде возникают в тех же интервалах времени, что и в опорных рядах

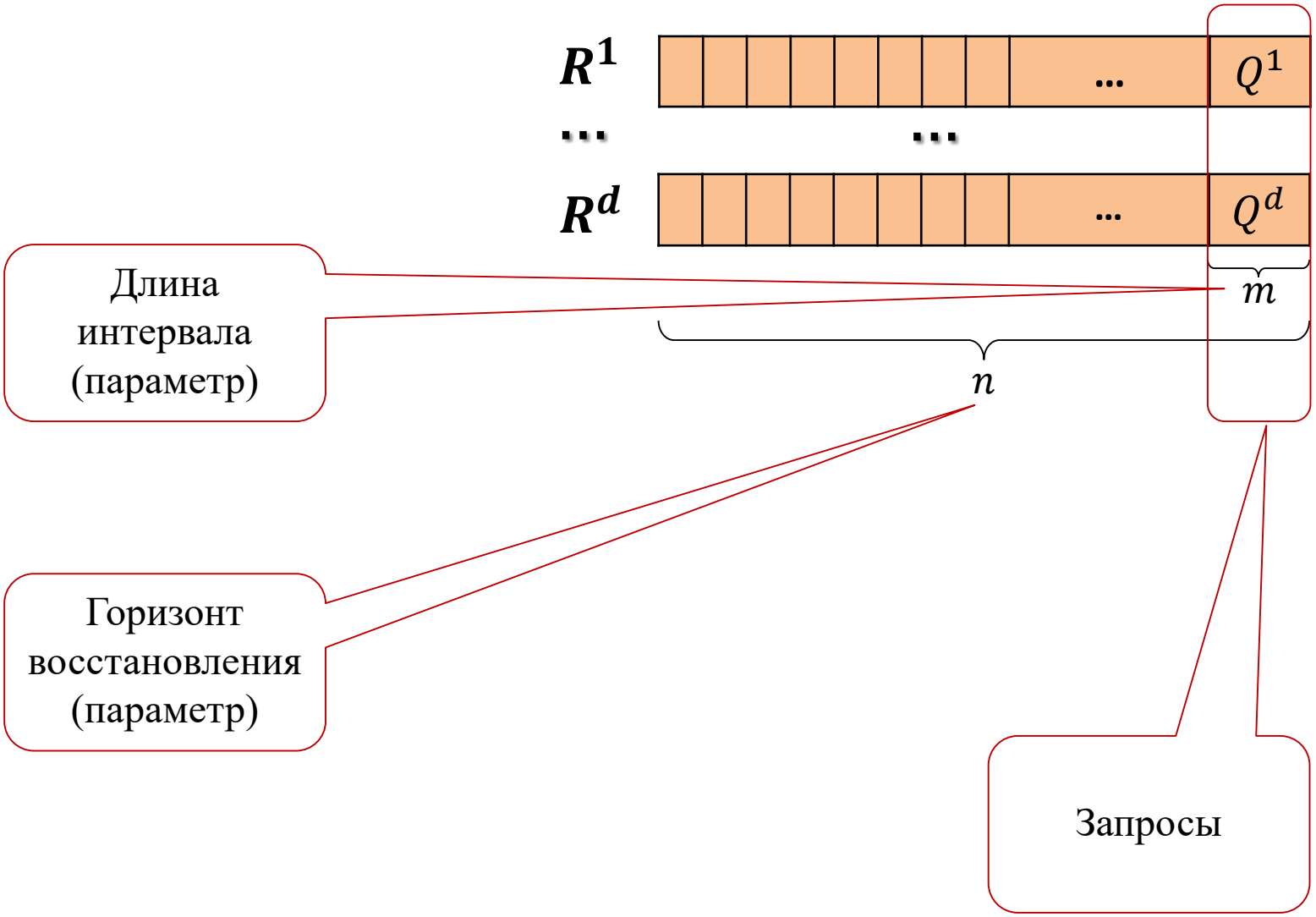
Поиск k ближайших соседей



Реконструкция пропущенного значения

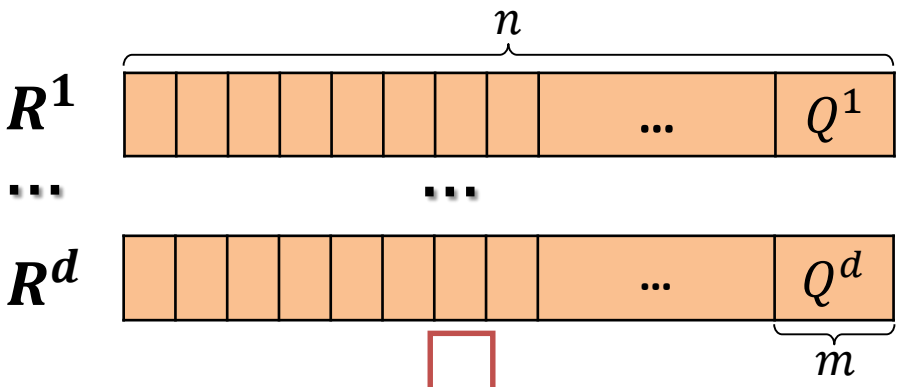


Алгоритм IDK (Imputation by DTW & kNN)



Алгоритм IDK: поиск соседей по образцу

Расстояние между интервалом $R^j[i:m]$ и образцом Q^j в смысле меры DTW (Dynamic Time Warping)

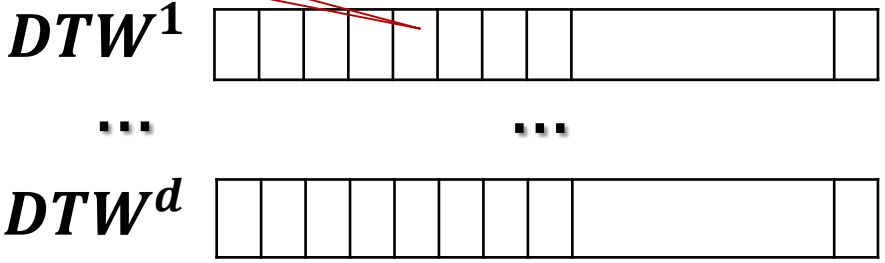


1. Поиск соседей по образцу

for $j := 1$ **to** d **do**

for $i := 1$ **to** $n - m + 1$ **do**

$DTW^j(R^j[i:m], Q^j)$



Алгоритм IDK: скоринг интервалов

1. Поиск соседей по образцу

for $j \coloneqq 1$ **to** d **do**

  for $i \coloneqq 1$ **to** $n - m + 1$ **do**

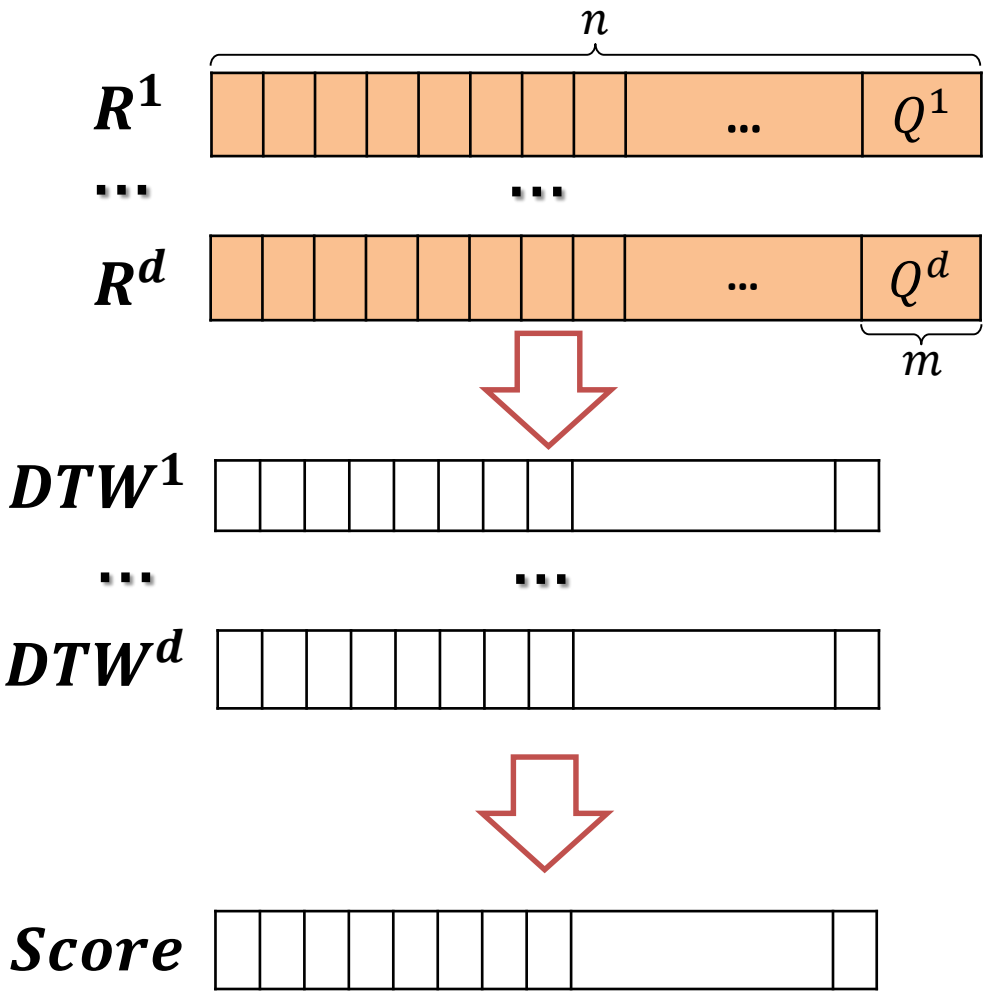
    $DTW^j(R^j[i{:}m], Q^j)$

2. Скоринг интервалов

for $i \coloneqq 1$ **to** $n - m + 1$ **do**

  for $j \coloneqq 1$ **to** d **do**

    $Score(i) \coloneqq Score(i) + \frac{1}{DTW^j(i)+\varepsilon}$



Алгоритм IDK: отбор интервалов

1. Поиск соседей по образцу

for $j \coloneqq 1$ **to** d **do**

  for $i \coloneqq 1$ **to** $n - m + 1$ **do**

    $DTW^j(R^j[i{:}m], Q^j)$

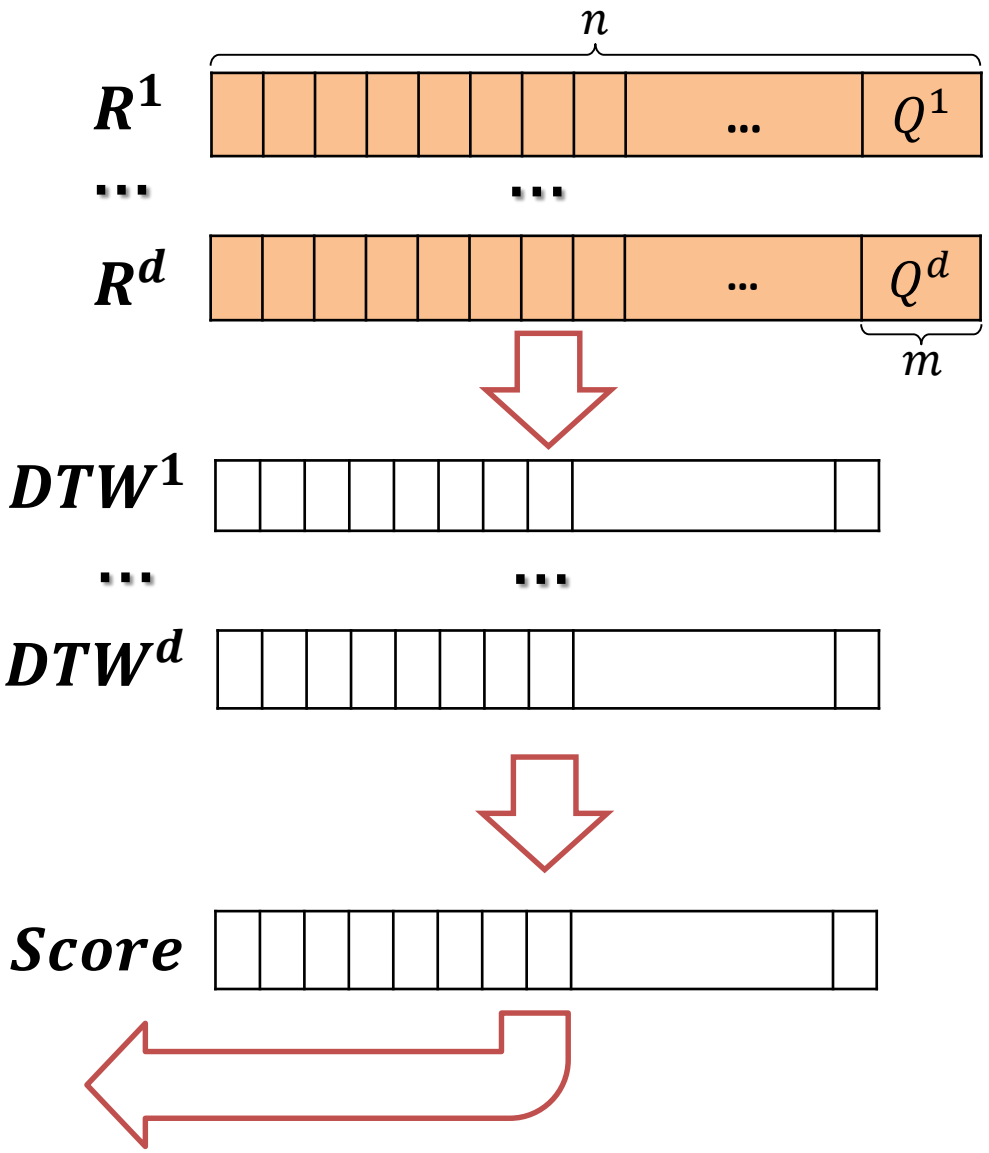
2. Скоринг интервалов

for $i \coloneqq 1$ **to** $n - m + 1$ **do**

  for $j \coloneqq 1$ **to** d **do**

    $Score(i) \coloneqq Score(i) + \frac{1}{DTW^j(i)+\varepsilon}$

3. Отбор TOP-k интервалов, реконструкция

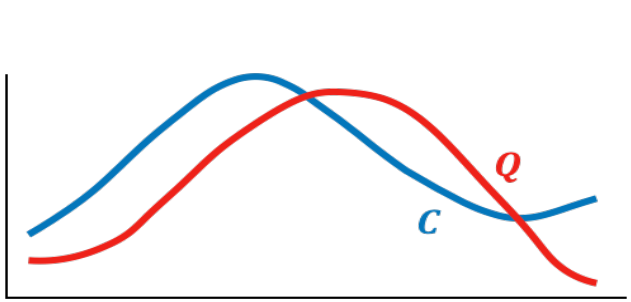


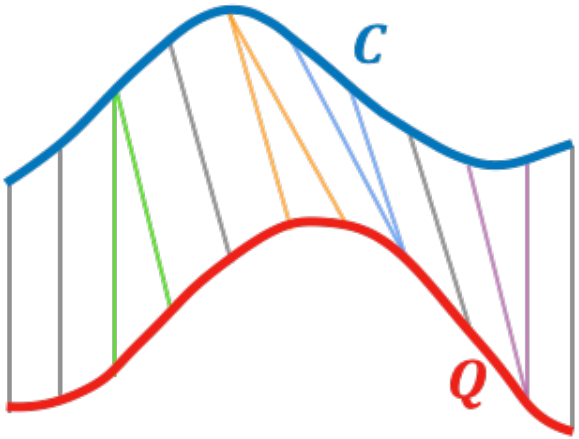
Расстояние DTW (Dynamic Time Warping)

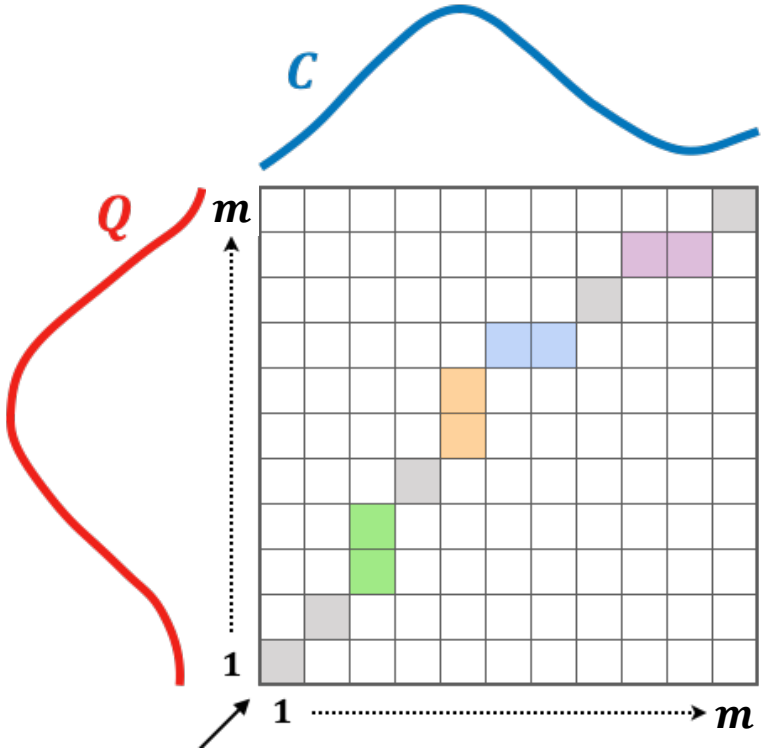
$$\mathbf{DTW}(Q, C) = d(m, m)$$

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

$$d(0,0) = 0, d(i,0) = d(0,j) = +\infty;\ 1 \le i,j \le m$$







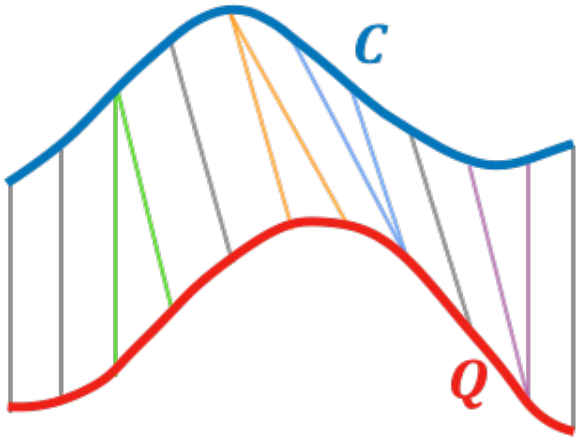
Расстояние DTW: точность *vs.* сложность

Сложность
$\boldsymbol{O(m^2)}$

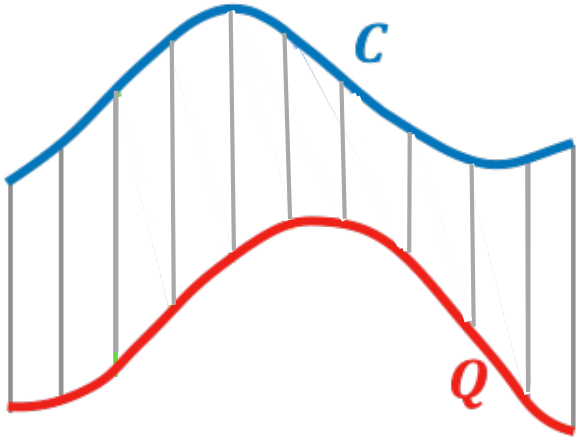
$$\mathbf{DTW}(Q, C) = d(m, m)$$

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

$$d(0,0) = 0, d(i,0) = d(0,j) = +\infty;\ 1 \le i,j \le m$$



DTW лучше определяет схожесть по форме, чем Евклидова метрика



Ускорение подсчета DTW: полоса Сако–Чиба

Сложность $O(mr)$

$$\mathbf{DTW}(Q, C) = d(m, m)$$

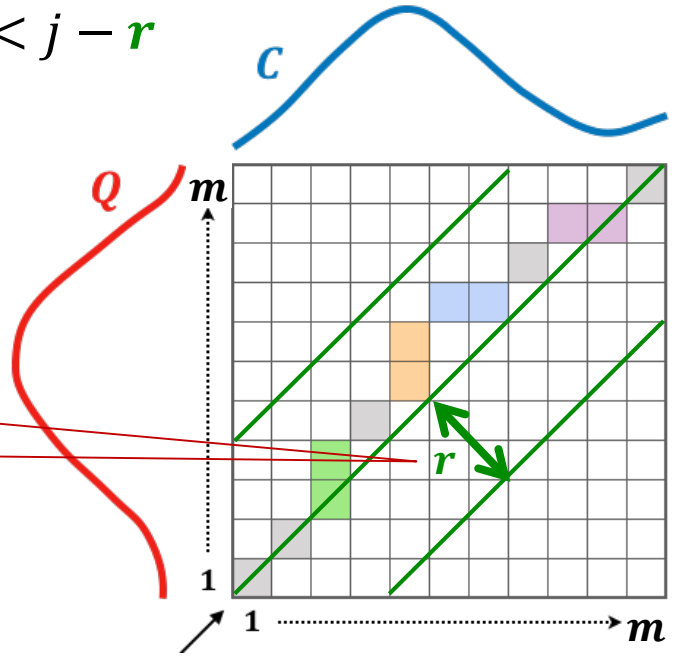
$$d(i,j) = (q_i - c_i)^2 + \min \begin{cases} d(i-1,j) \\ d(i,j-1) \\ d(i-1,j-1) \end{cases}$$

$$d(0,0) = 0, d(i,0) = d(0,j) = +\infty;\ 1 \le i,j \le m;$$

$$\mathbf{0} \le \boldsymbol{r} \le \boldsymbol{m} - \mathbf{1}, j - \boldsymbol{r} \le i \le j + \boldsymbol{r}$$

$$d(i,j) = +\infty, j + \boldsymbol{r} < i < j - \boldsymbol{r}$$

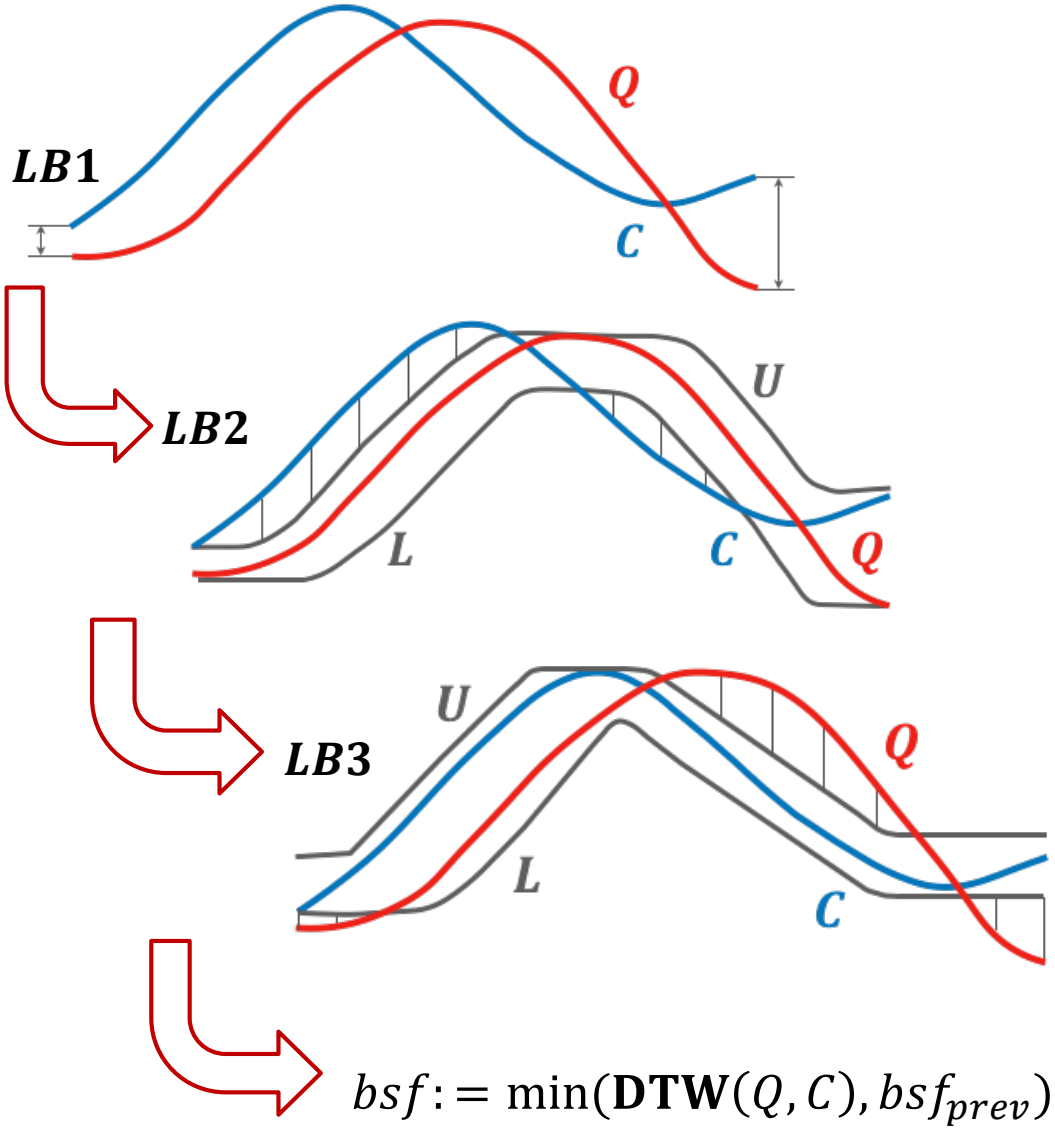
Сокращает сложность за счет огрубления схожести



Ускорение подсчета DTW: отбрасывание

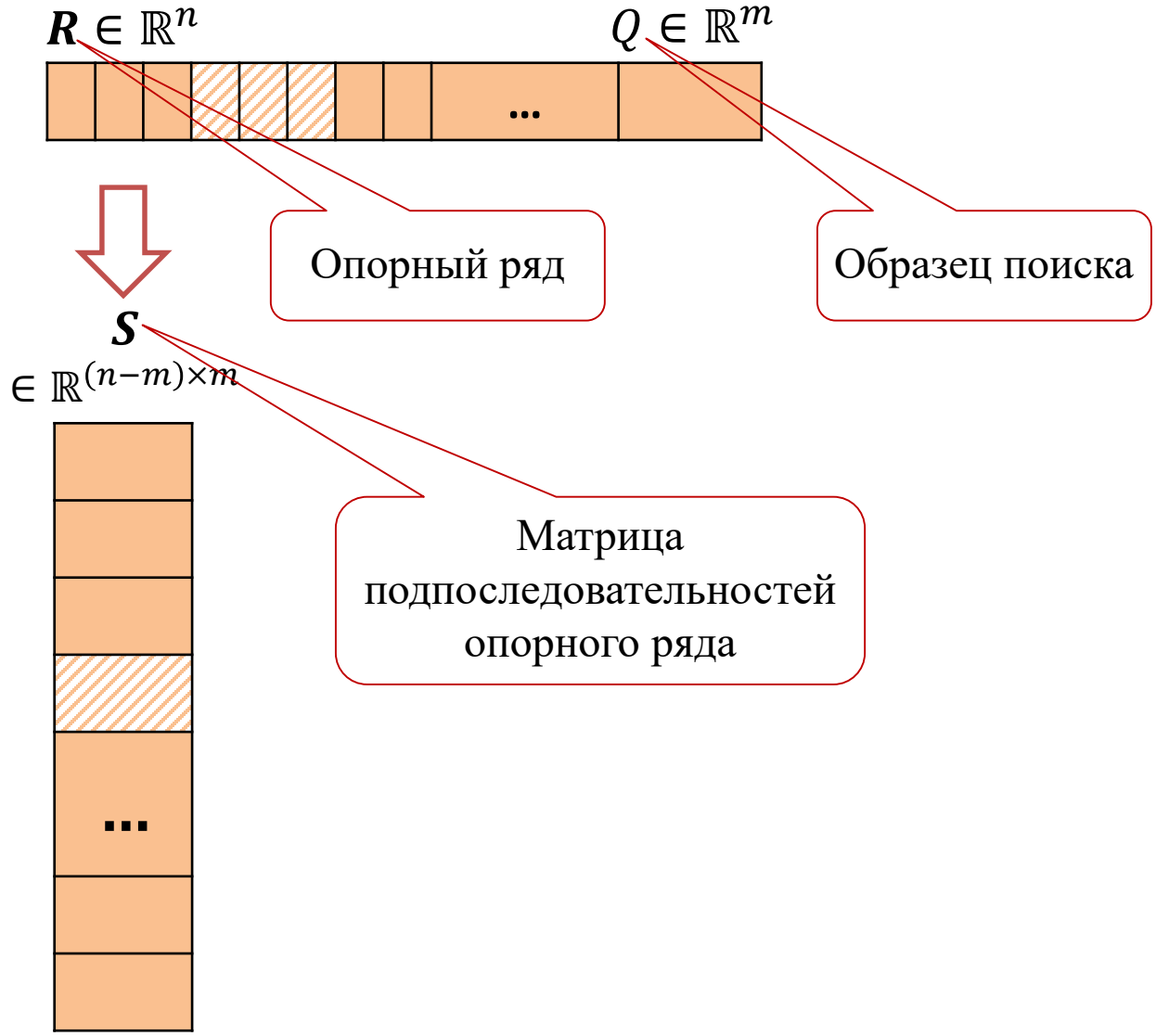
- Нижняя граница
  - функция LB: $\mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+$
    с вычислительной сложностью менее $O(m^2)$
- Текущий минимум DTW
  - bsf (best-so-far)
- Отбрасывание соседей, далеких от образца, без вычисления DTW
  - если $\mathrm{LB}(R[i:m], Q) > bsf$, то $\mathrm{DTW}(R[i:m], Q) > bsf$
- Нормализация
  - Соседей и запрос нужно нормализовать

Отбрасывание: нижние границы

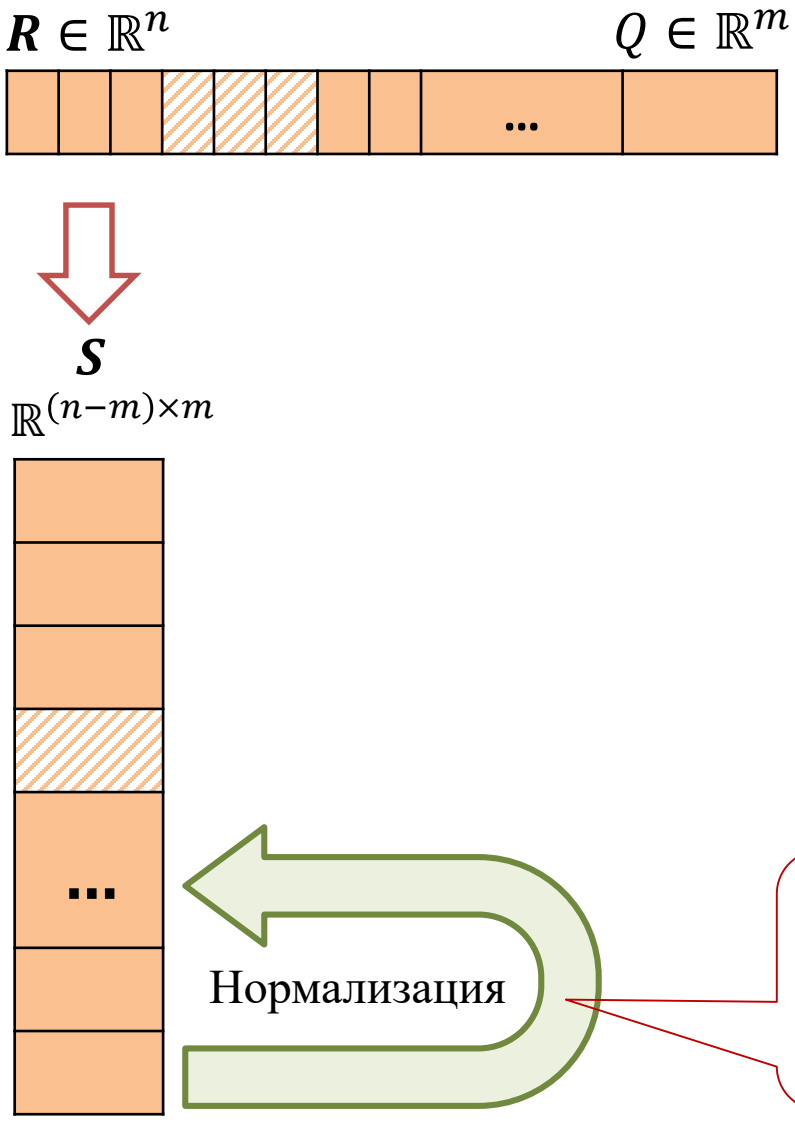


Нижняя граница	Сложность
$\boldsymbol{LB1}(Q, C) = (q_1 - c_1)^2 + (q_m - c_m)^2$	$O(1)$
$\boldsymbol{LB2}(Q, C) = \sum_{i=1}^{m} \begin{cases} (c_i - u_i)^2, & c_i > u_i \\ (c_i - \ell_i)^2, & c_i < \ell_i \\ 0, & otherwise \end{cases}$ $u_i = \max_{i-r \le k \le i+r} q_k$ $\ell_i = \min_{i-r \le k \le i+r} q_k$	$O(m)$
$\boldsymbol{LB3}(Q, C) = \boldsymbol{LB2}(C, Q)$	$O(m)$
$\mathbf{DTW}(Q, C)$	$O(mr)$

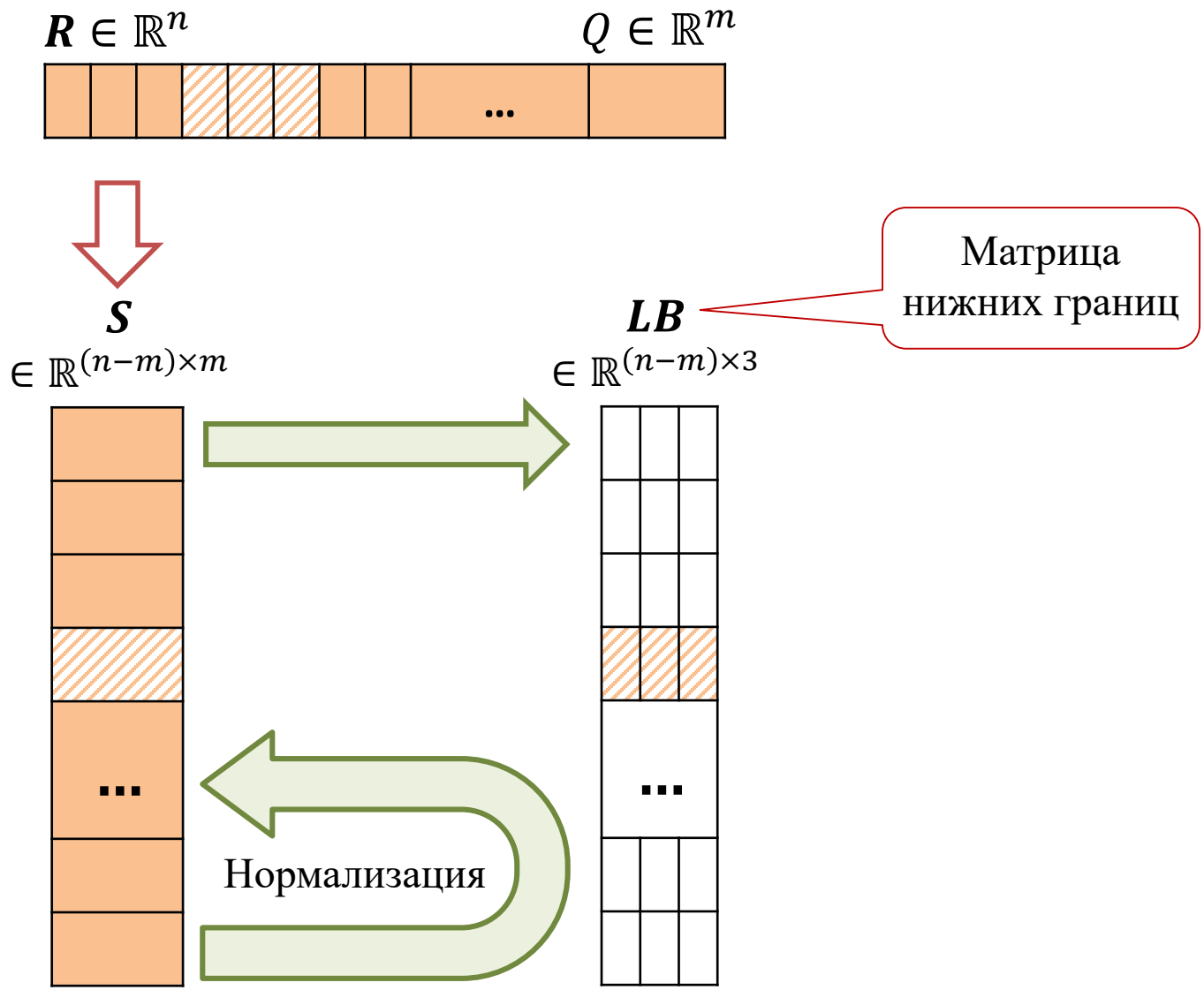
Параллельный поиск соседей по образцу



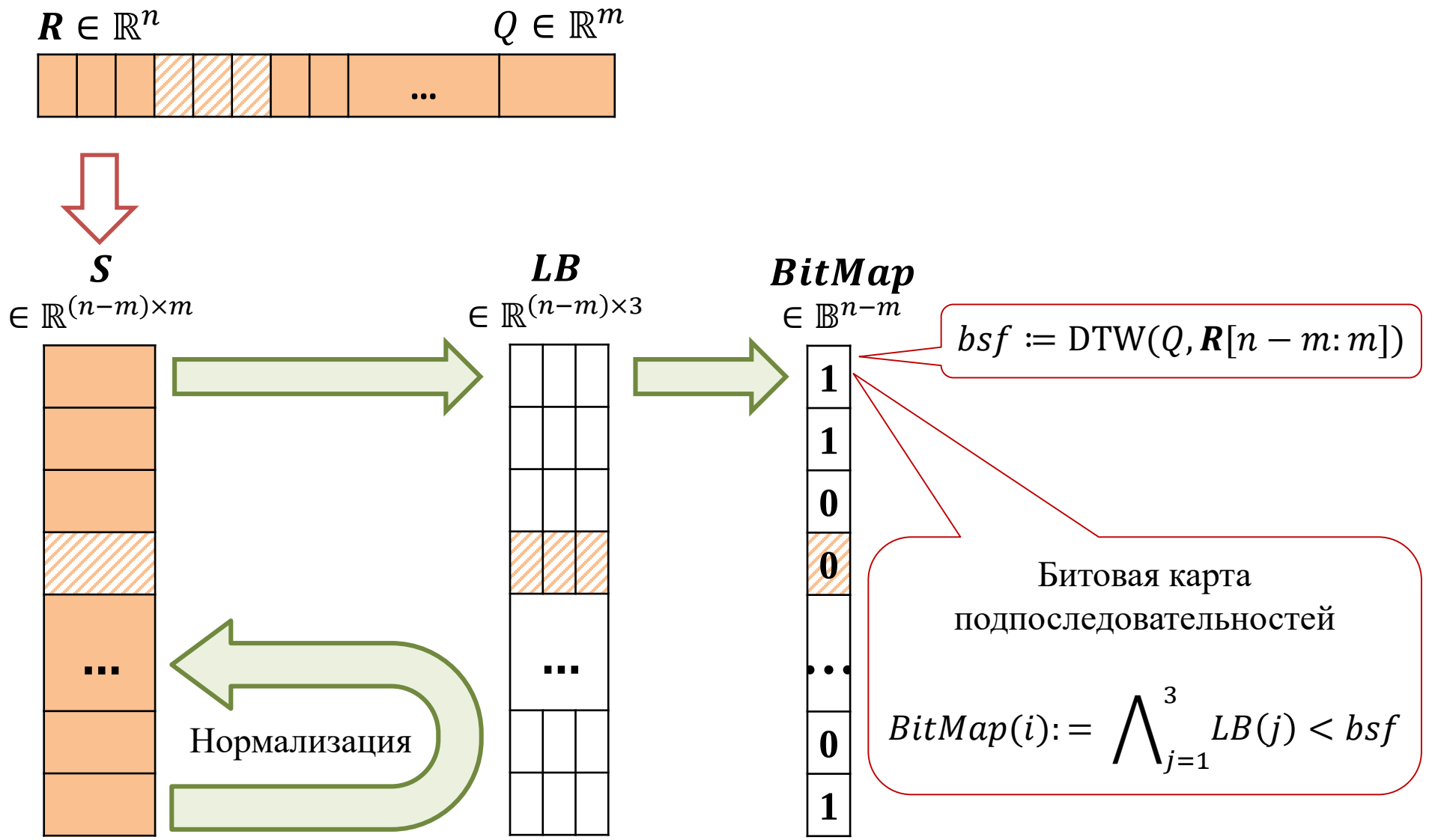
Параллельный поиск соседей по образцу



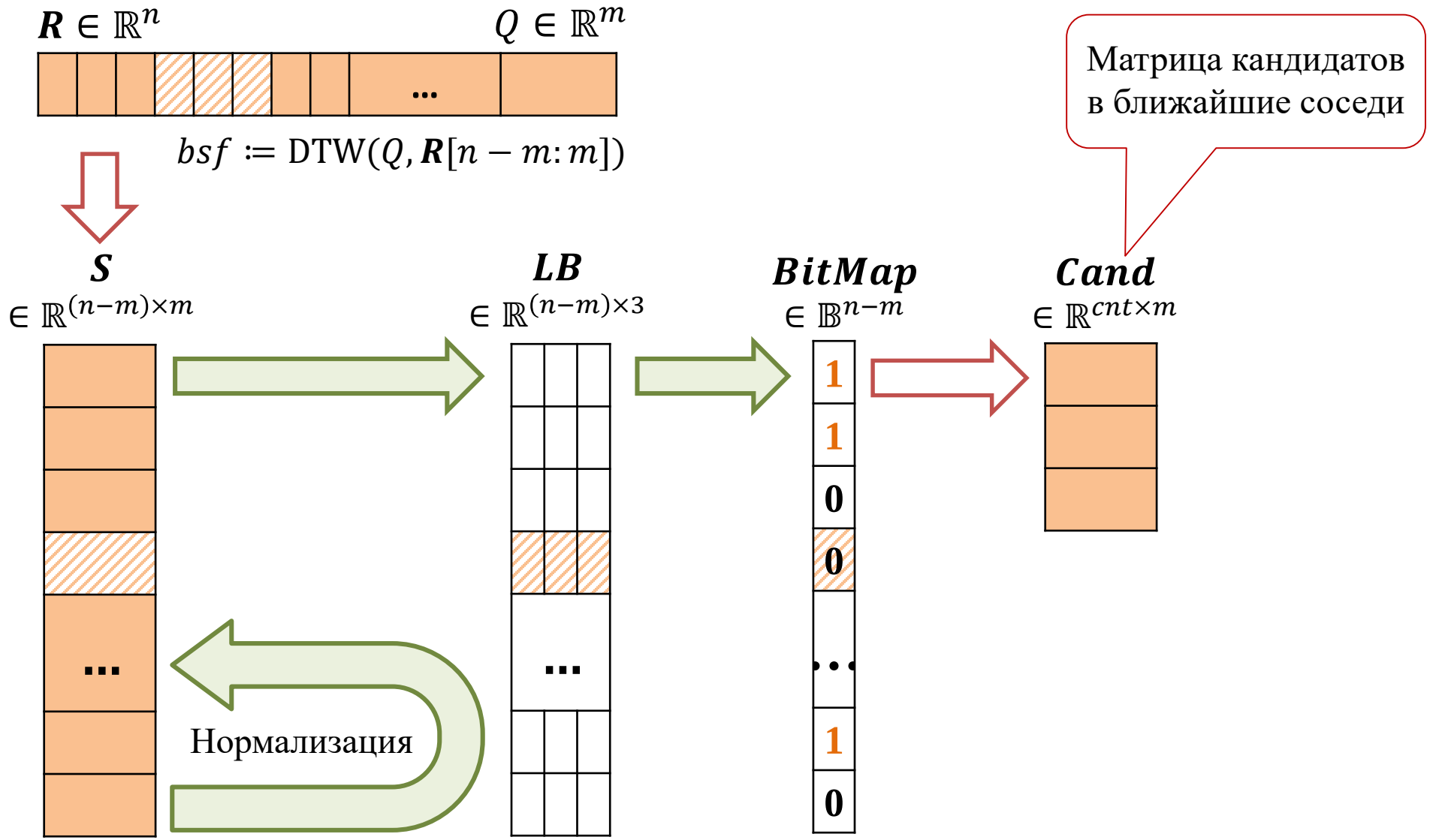
Параллельный поиск соседей по образцу



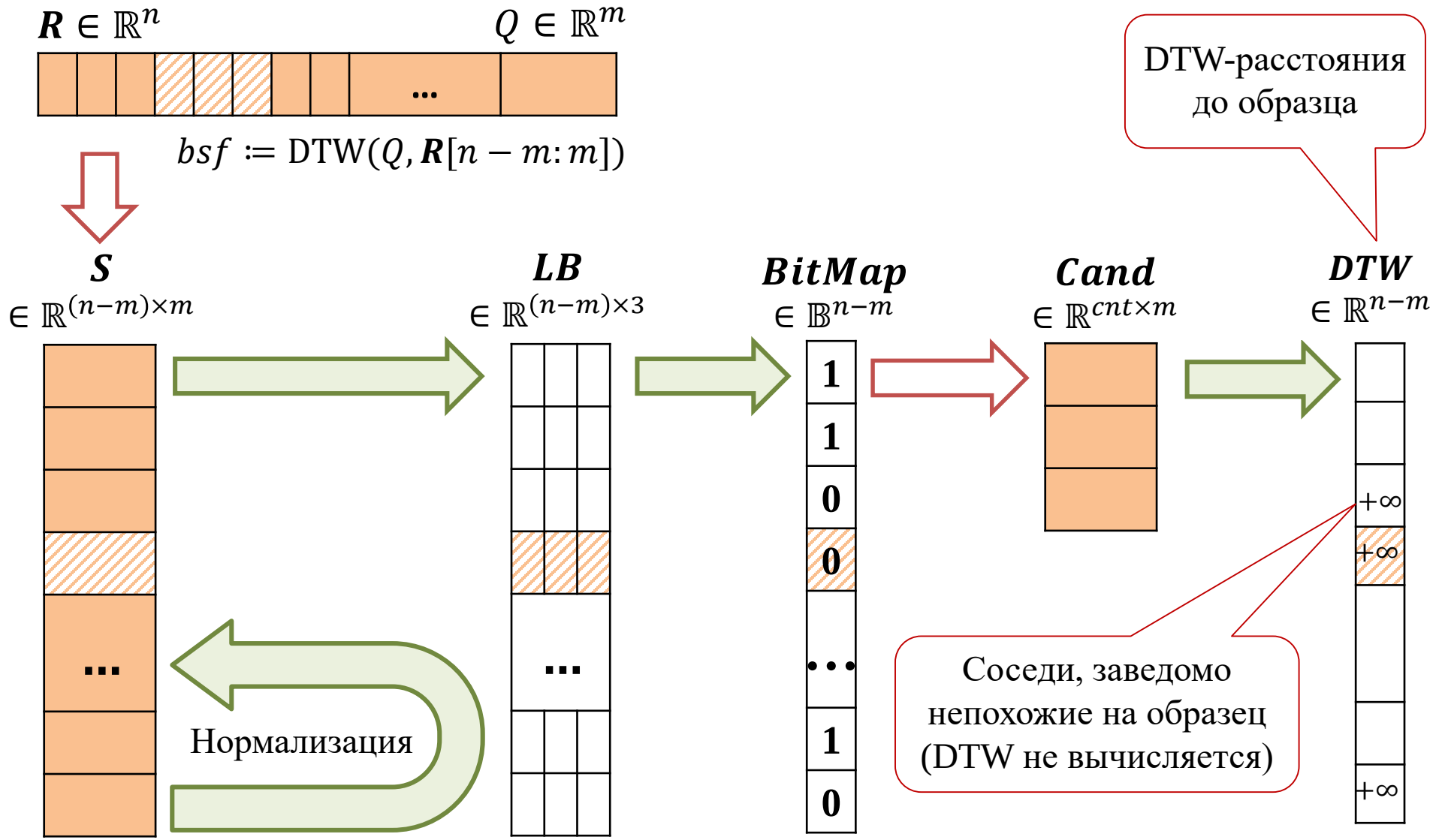
Параллельный поиск соседей по образцу



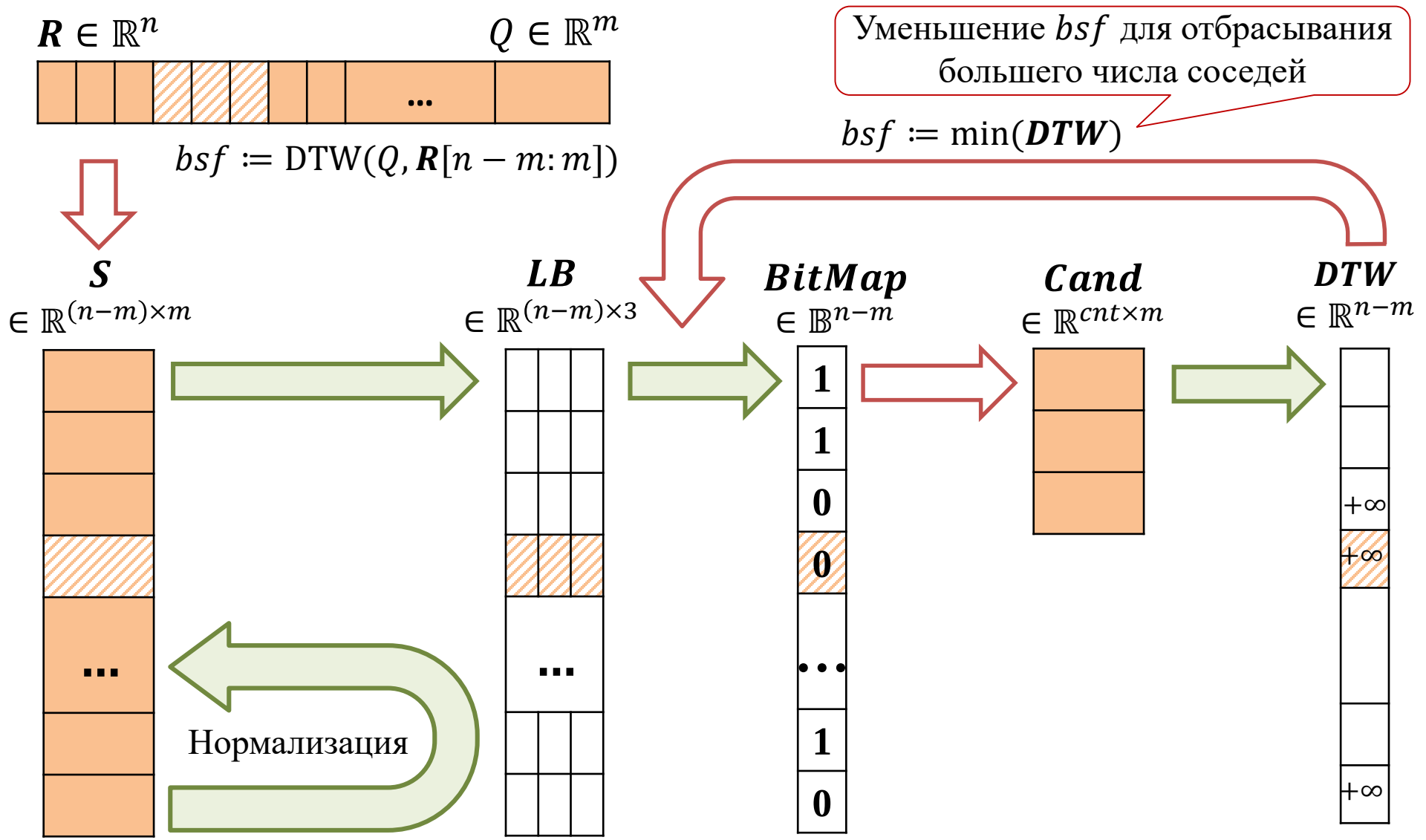
Параллельный поиск соседей по образцу



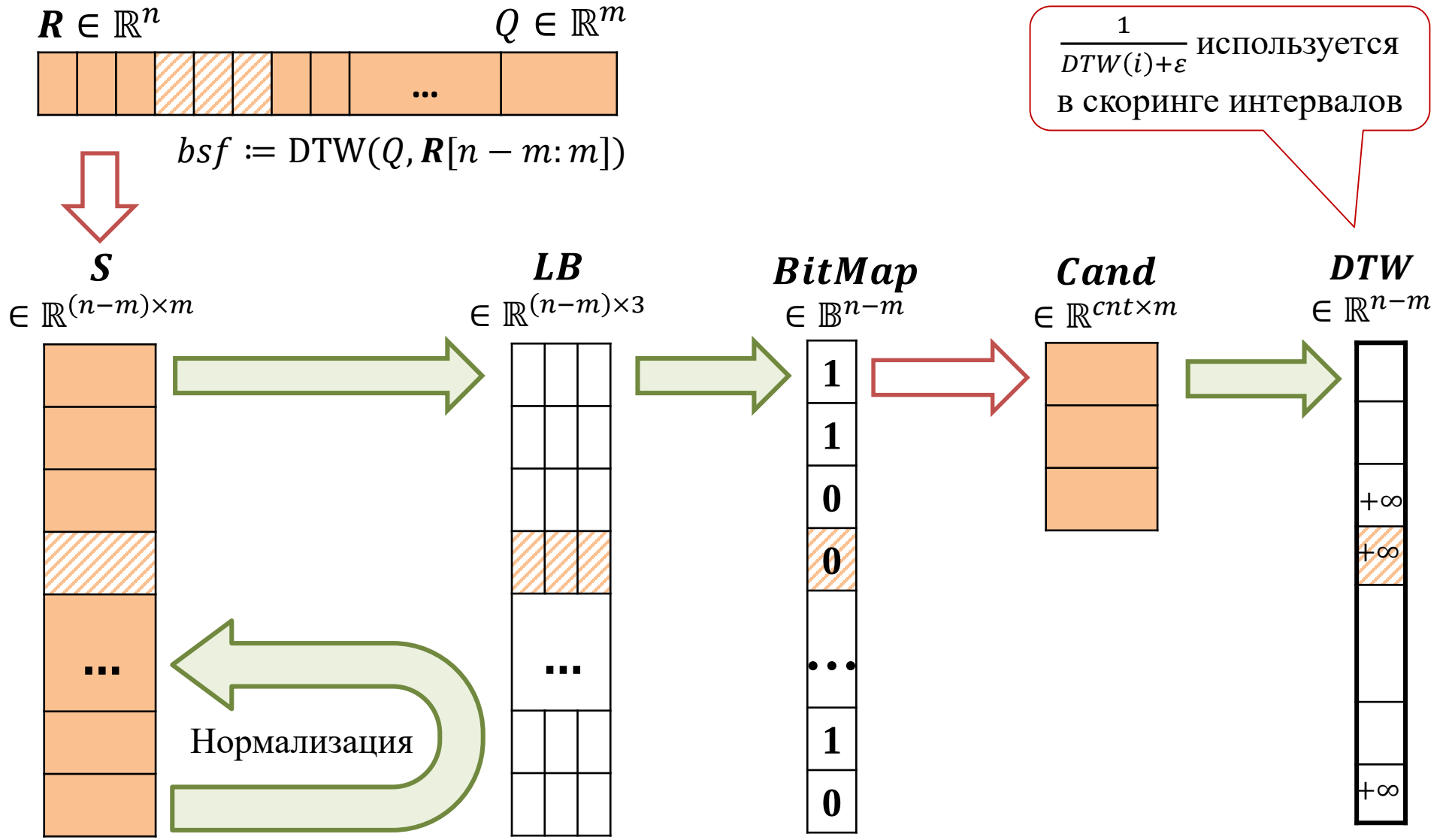
Параллельный поиск соседей по образцу



Параллельный поиск соседей по образцу



Параллельный поиск соседей по образцу



Эксперименты: платформа и данные

- Аппаратная платформа:

Процессор	Intel Xeon Gold 6254
Ядра	18 ядер по 4 GHz
Память	192 Gb

- Точность восст-я:

$$RMSE = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(x_i - \tilde{x}_i)^2}$$

$$h = 1000$$

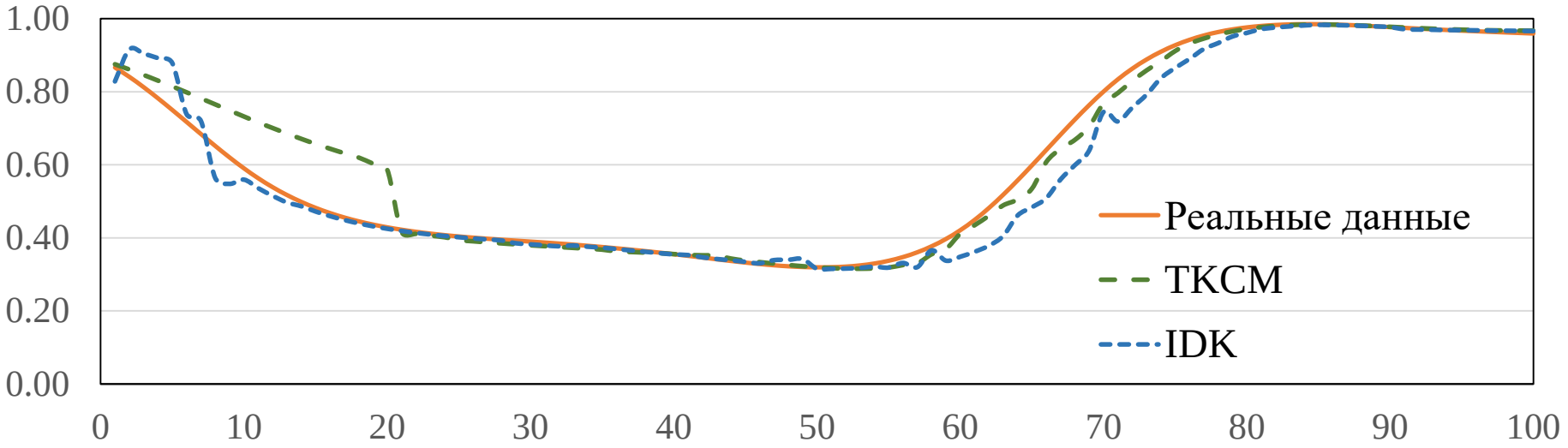
- Наборы данных:

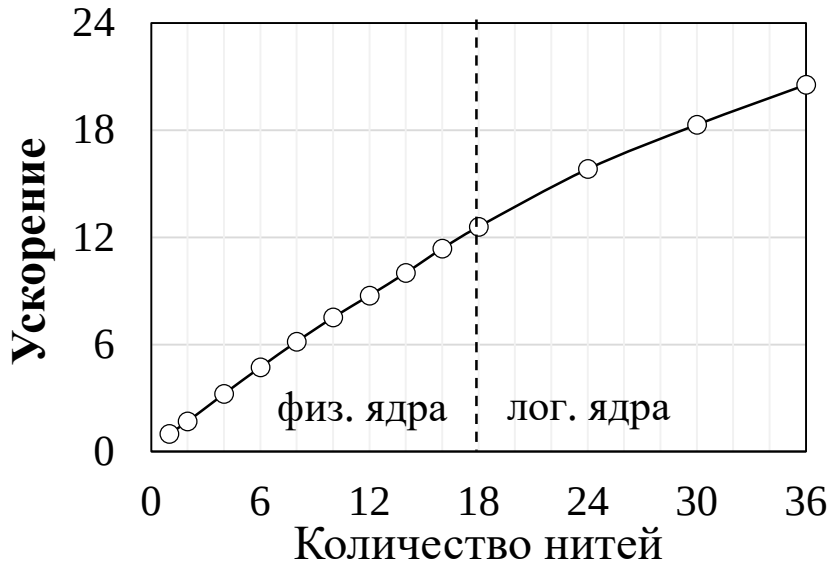
Набор	Длина ряда, n	К-во опорных рядов, d	К-во ближ. соседей, k	Длина запроса, m	Полоса Сако–Чиба, r
Chlorine	4 209	3	3	100	25
Marel Carnot	50 000	3	3	100	25
Walk run	210 000	3	3	50	12

- Аналог: TKCM*

* Wellenzohn K. *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series. Proc. of EDBT 2017. P. 330–341. DOI: 10.5441/002/edbt.2017.30

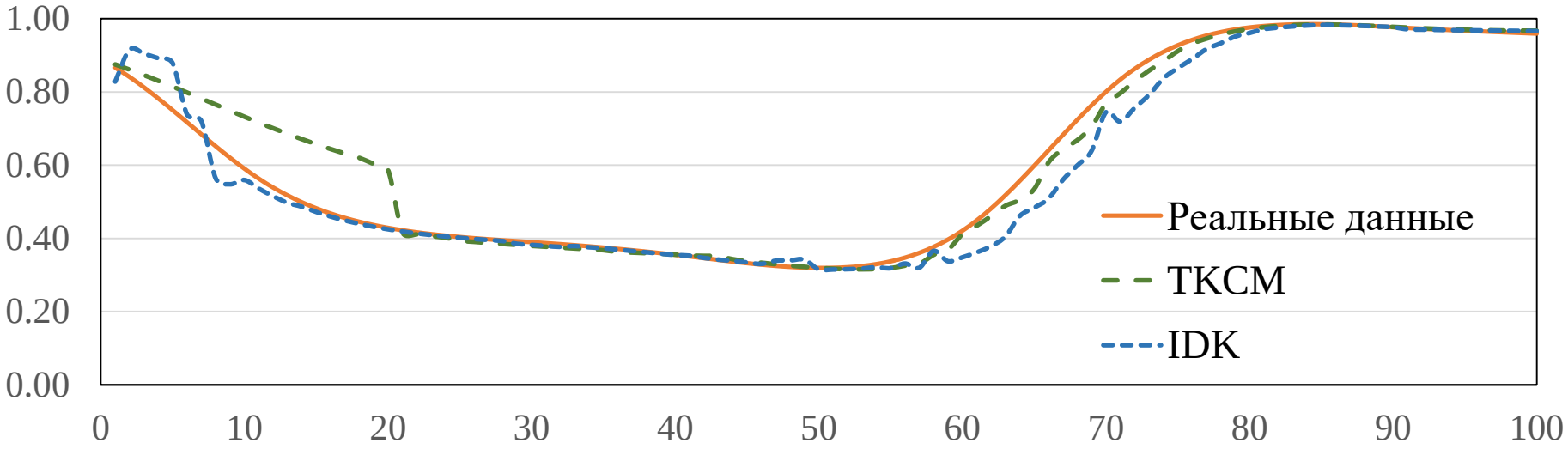
Эксперименты: набор Chlorine

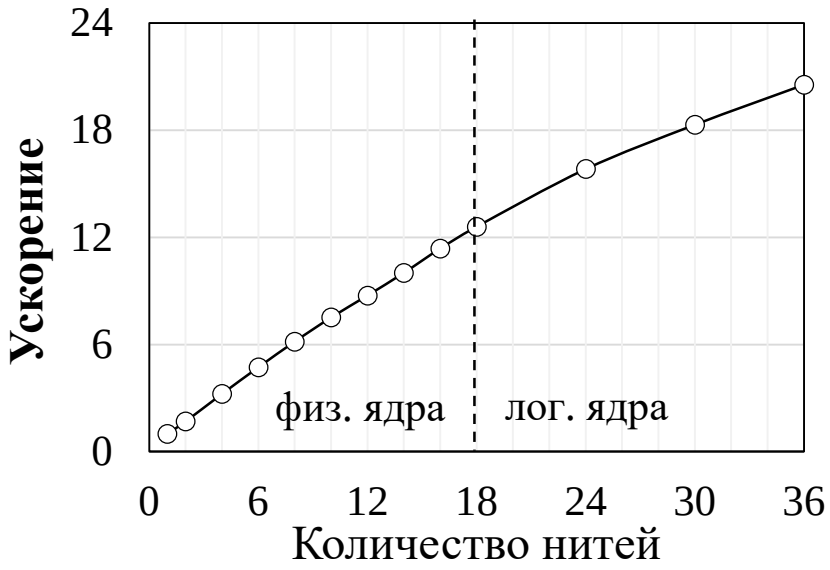




Алгоритм	RMSE	Runtime, s
TKCM	0.062	0.013
IDK	**0.049**	**0.006**

Эксперименты: набор Chlorine

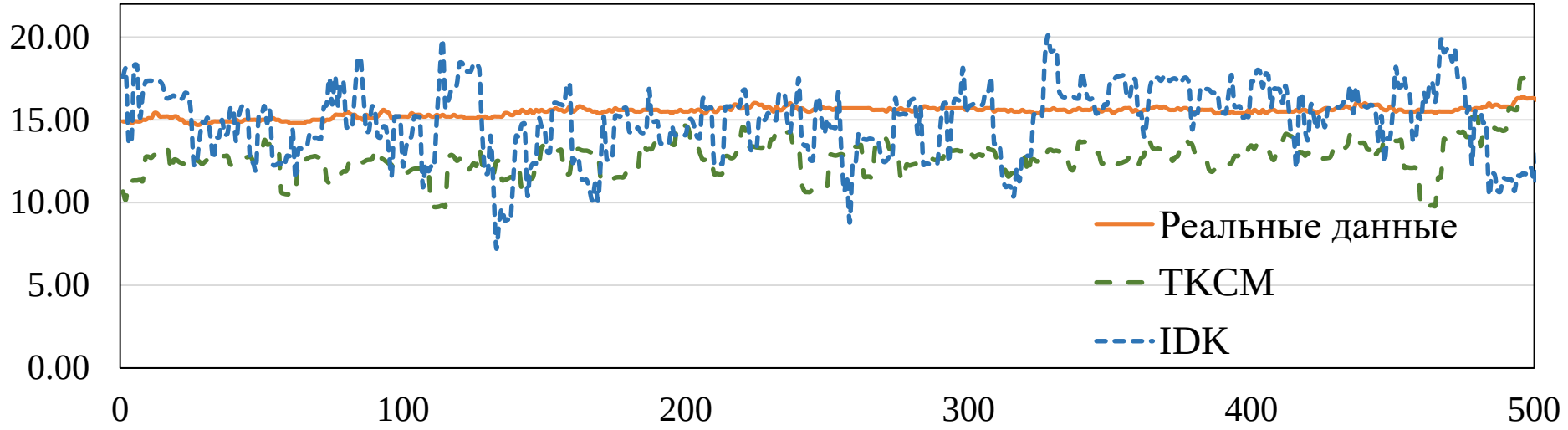


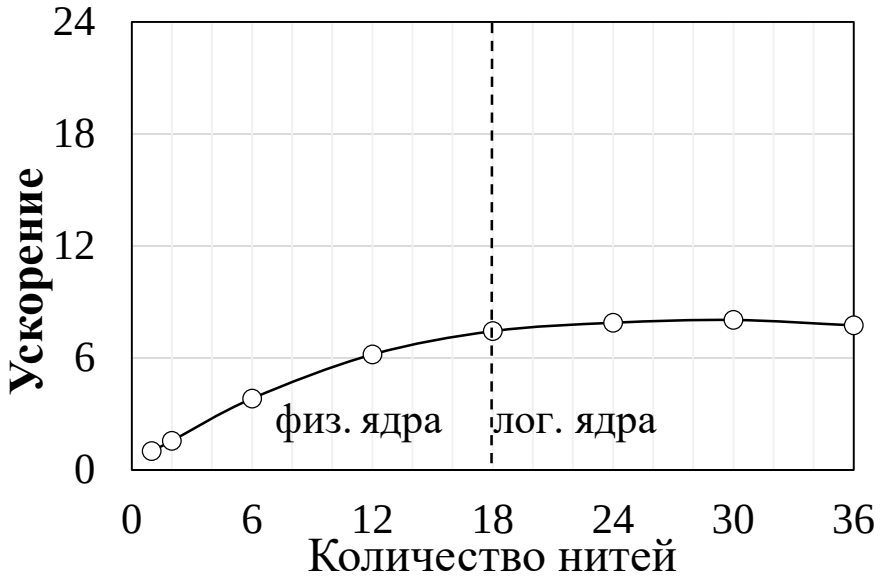


Алгоритм	RMSE	Runtime, s
TKCM	0.062	0.013
IDK	**0.049**	**0.006**

Отброшено 98% соседей
без вычисления DTW

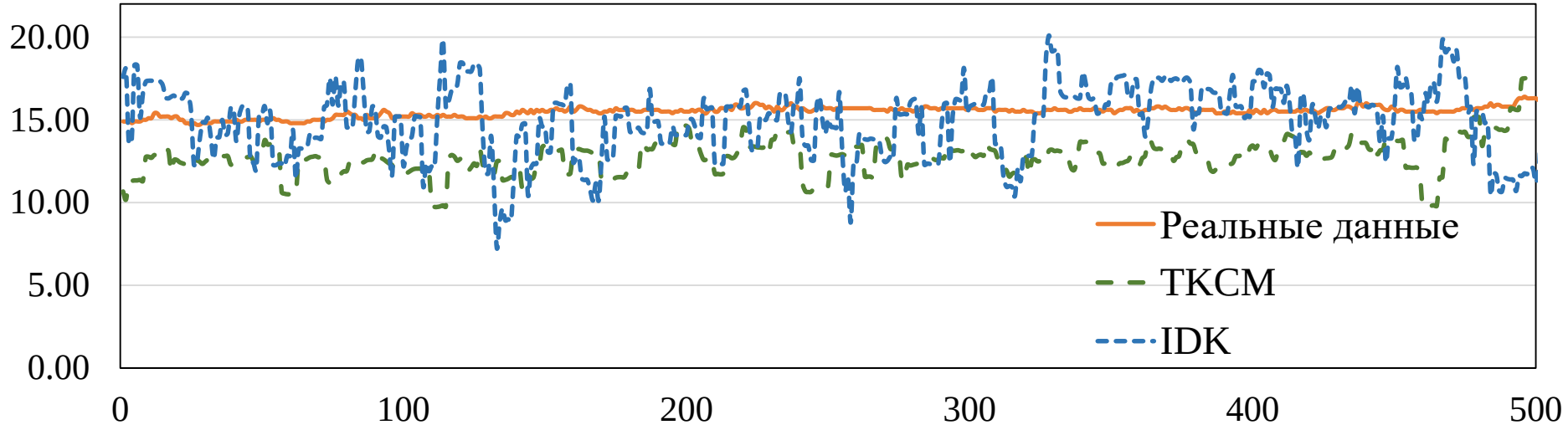
Эксперименты: набор Marel Carnot

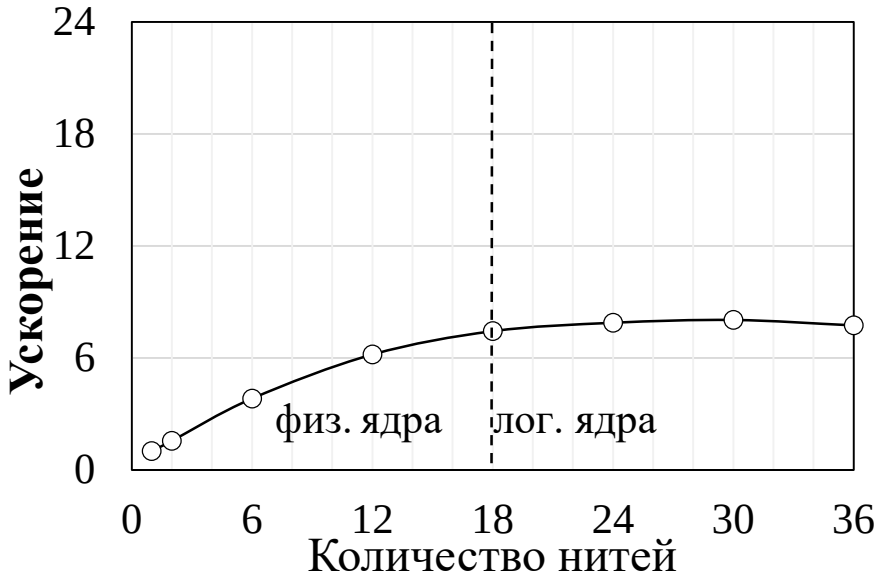




Алгоритм	RMSE	Runtime, s
TKCM	2.497	**0.057**
IDK	**2.434**	**0.054**

Эксперименты: набор Marel Carnot

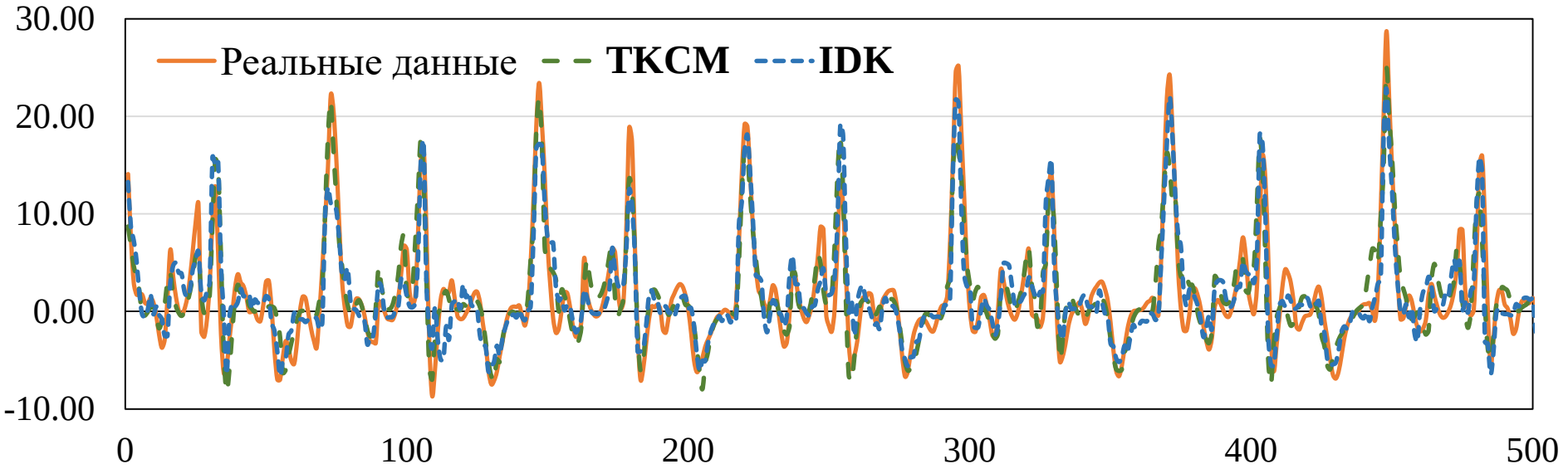


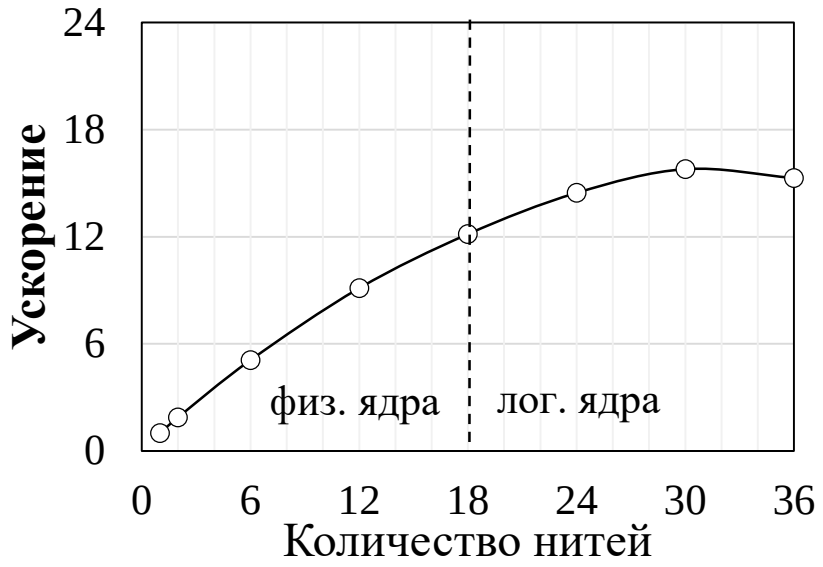


Алгоритм	RMSE	Runtime, s
TKCM	2.497	**0.057**
IDK	**2.434**	**0.054**

Отброшено 75% соседей
без вычисления DTW

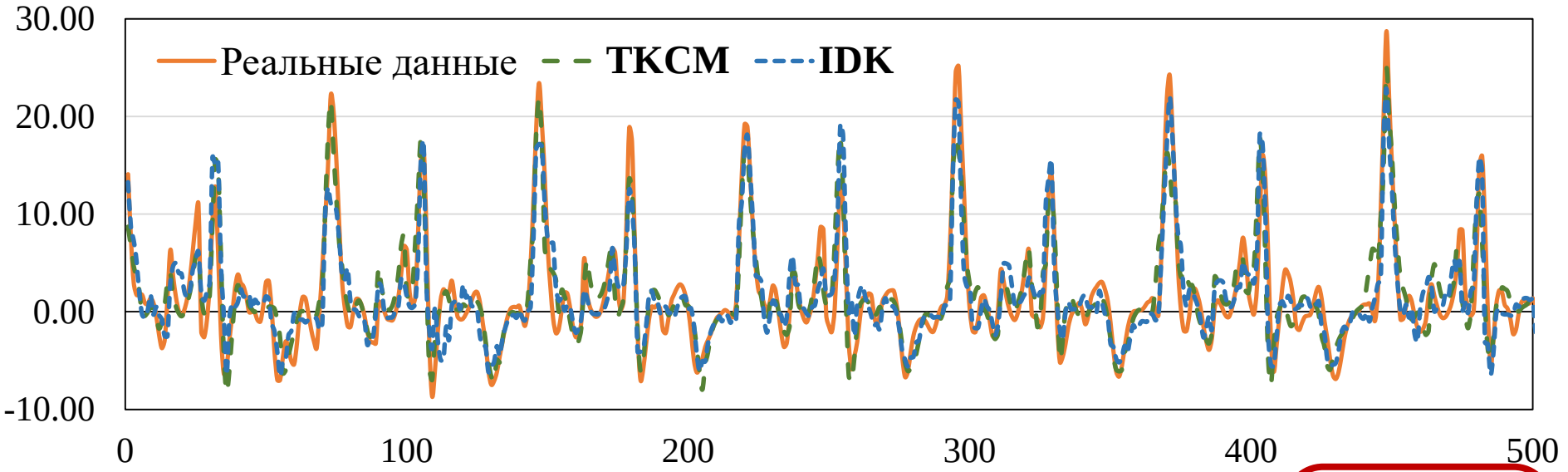
Эксперименты: набор Walk run

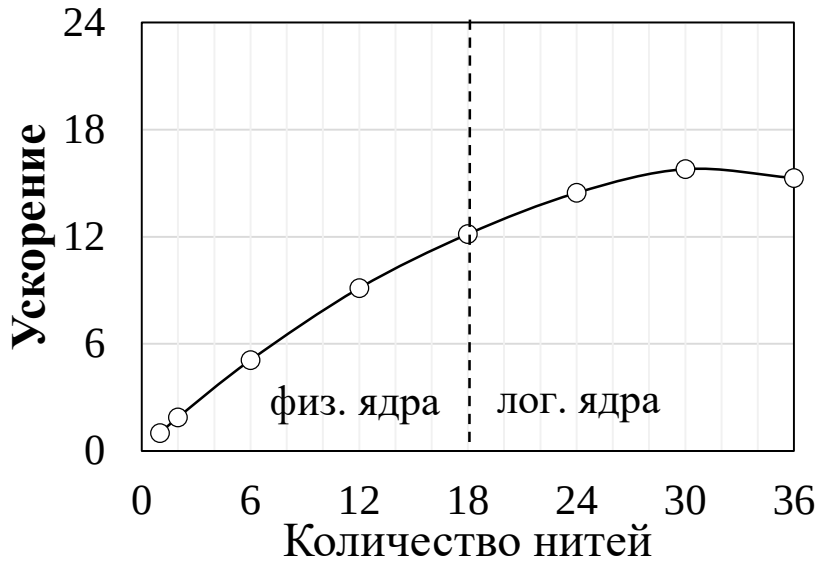




Алгоритм	RMSE	Runtime, s
TKCM	**2.785**	0.242
IDK	2.937	**0.080**

Эксперименты: набор Walk run

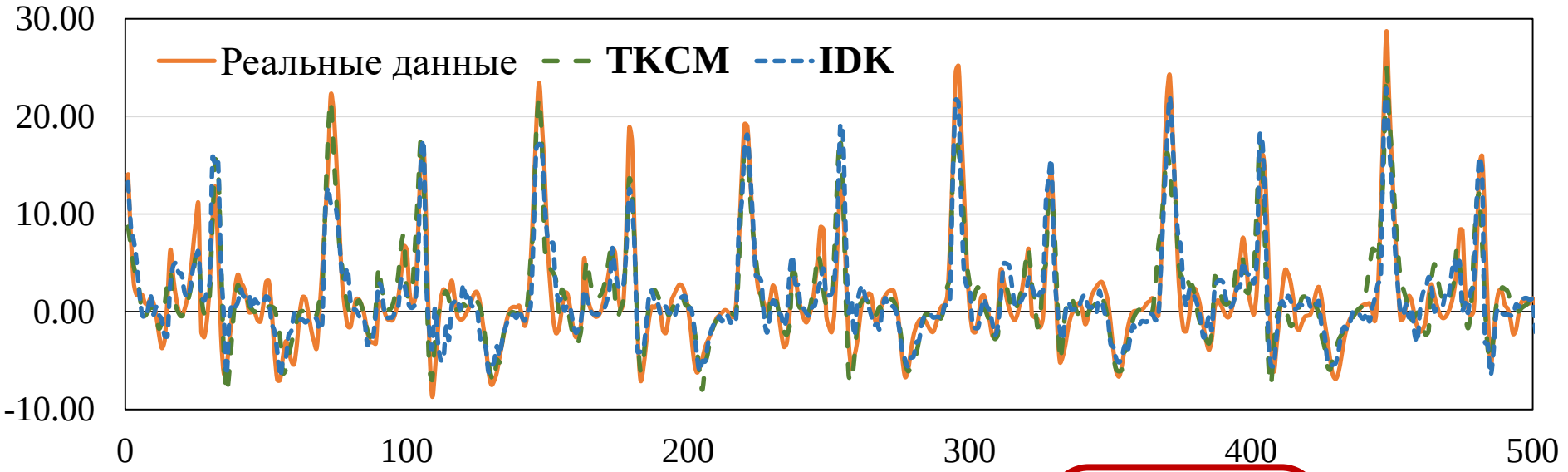


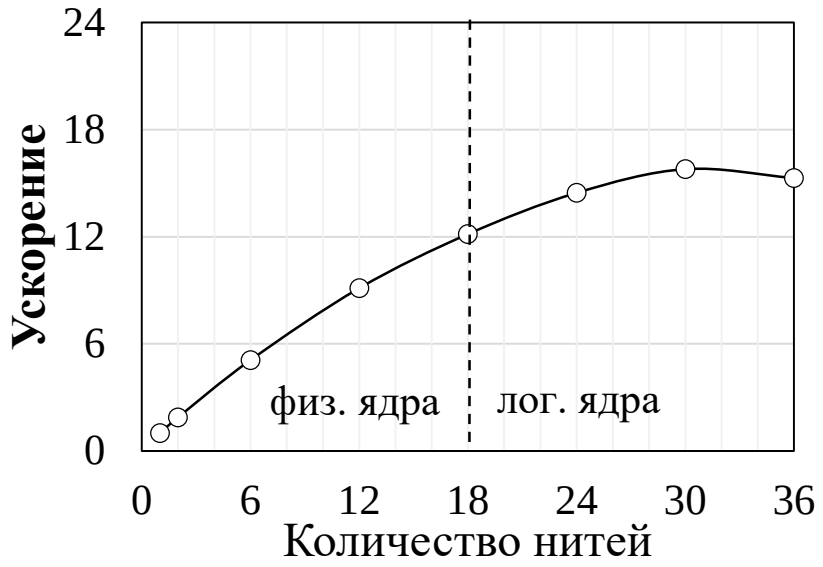


Алгоритм	RMSE	Runtime, s
TKCM	**2.785**	0.242
IDK	2.937	**0.080**

Отброшено 97% соседей
без вычисления DTW

Эксперименты: набор WalkRun





Алгоритм	RMSE	Runtime, s
TKCM	**2.785**	0.242
IDK	2.937	**0.080**

DTW лучше определяет схожесть по форме
Нормализация – ошибаемся по амплитуде

Заключение

- IDK – новый параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени на многоядерном процессоре
- Эксперименты показывают хорошие масштабируемость и точность восстановления IDK
- Будущие исследования: версия IDK для GPU

Спасибо за внимание! Вопросы?

Андрей Николаевич Полуянов andrey.poluyanov@gmail.com
Михаил Леонидович Цымблер mzym@susu.ru