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

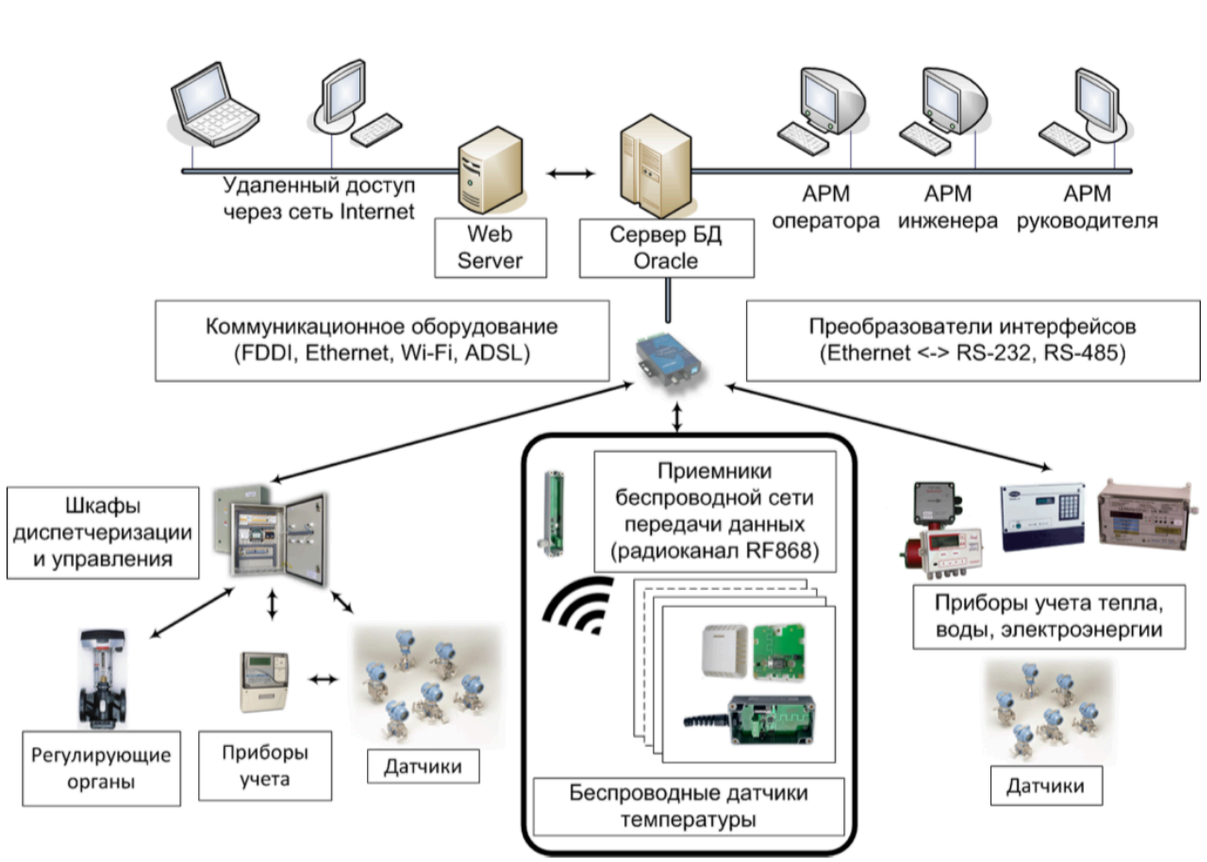
Поиск аномалий временного ряда на графическом процессоре

Я.А. Краева, М.Л. Цымблер

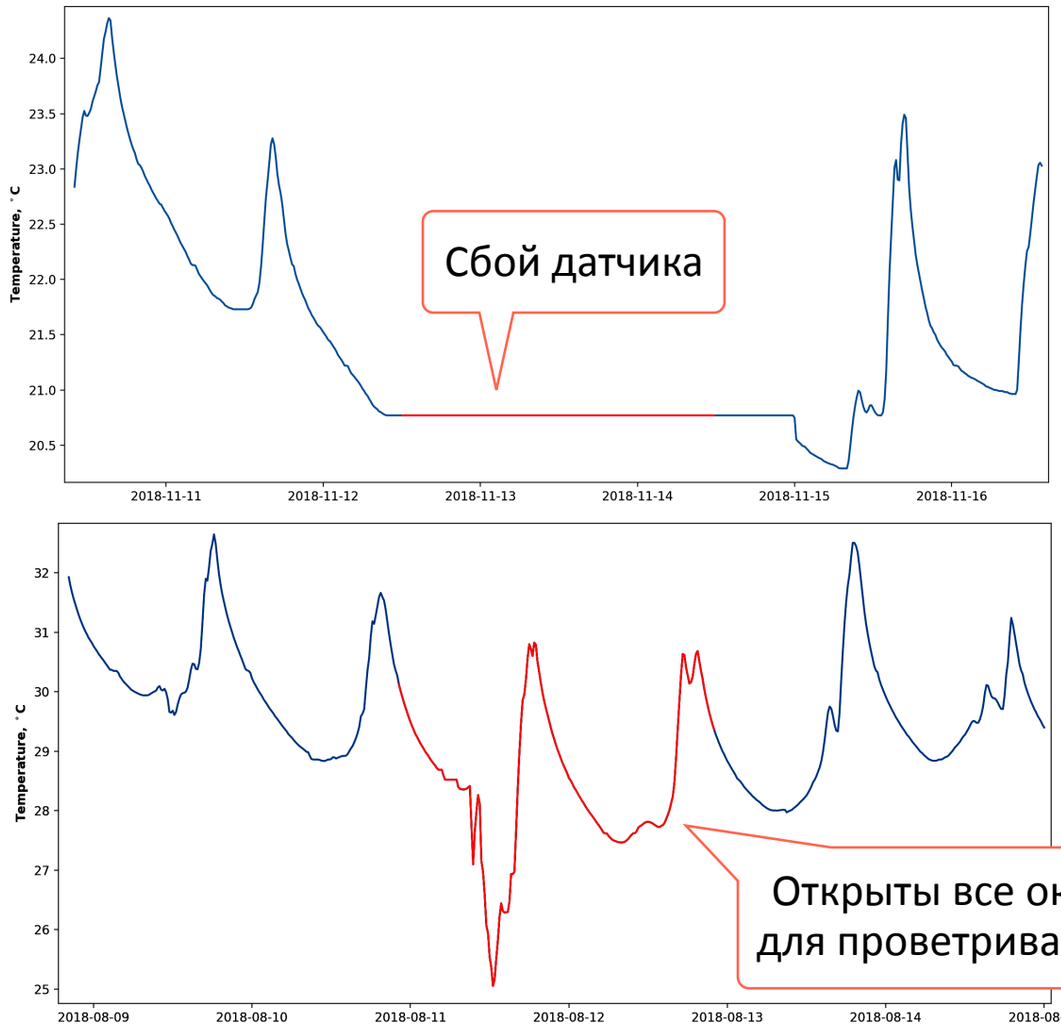
Южно-Уральский государственный университет (Челябинск)

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

Аномалии во временных рядах



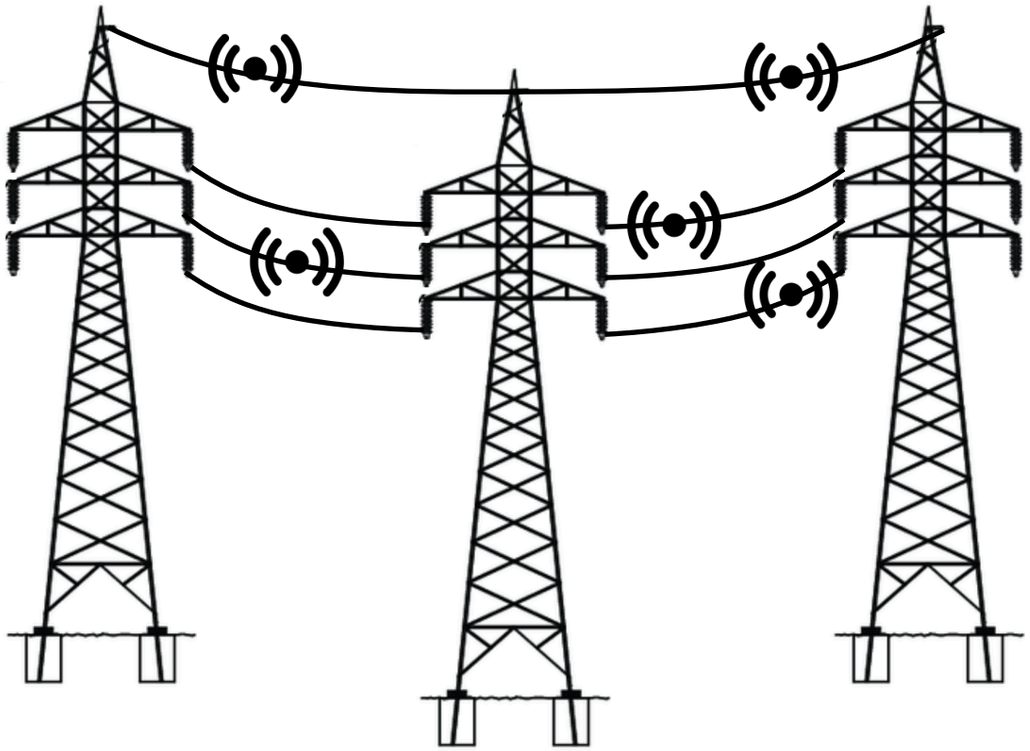
Структура интеллектуальной системы управления теплоснабжением зданий кампуса ЮУрГУ

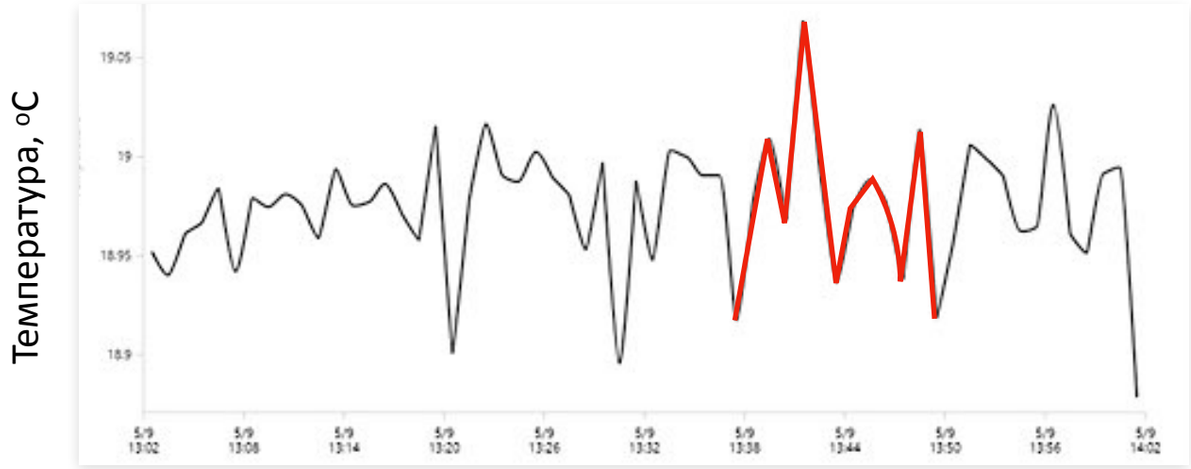


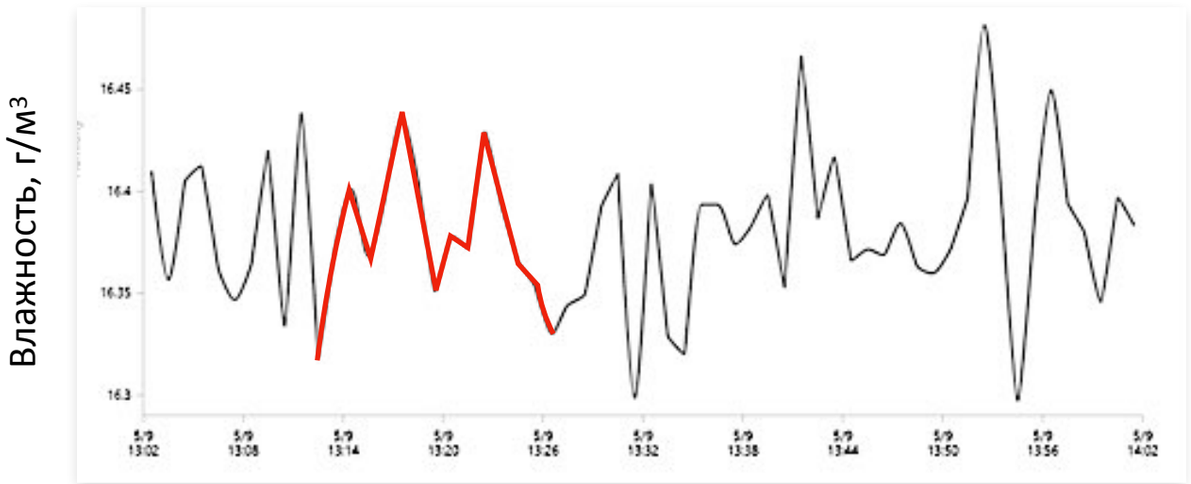
- Поиск аномалий в показаниях температурных IoT-датчиков позволяет выявить некорректное их поведение в системе умного управления отоплением зданий кампуса ЮУрГУ[1)]
- Дискретность показаний датчика: **4 раза в час.**

[1)] Цымблер М.Л., Краева Я.А., Латыпова Е.А., Иванова Е.В., Шнайдер Д.А., Басалаев А.А. Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий. Вестник ЮУрГУ. Серия: Вычислительная математика и информатика 10(3): 16–36. 2021.

Аномалии в больших временных рядах







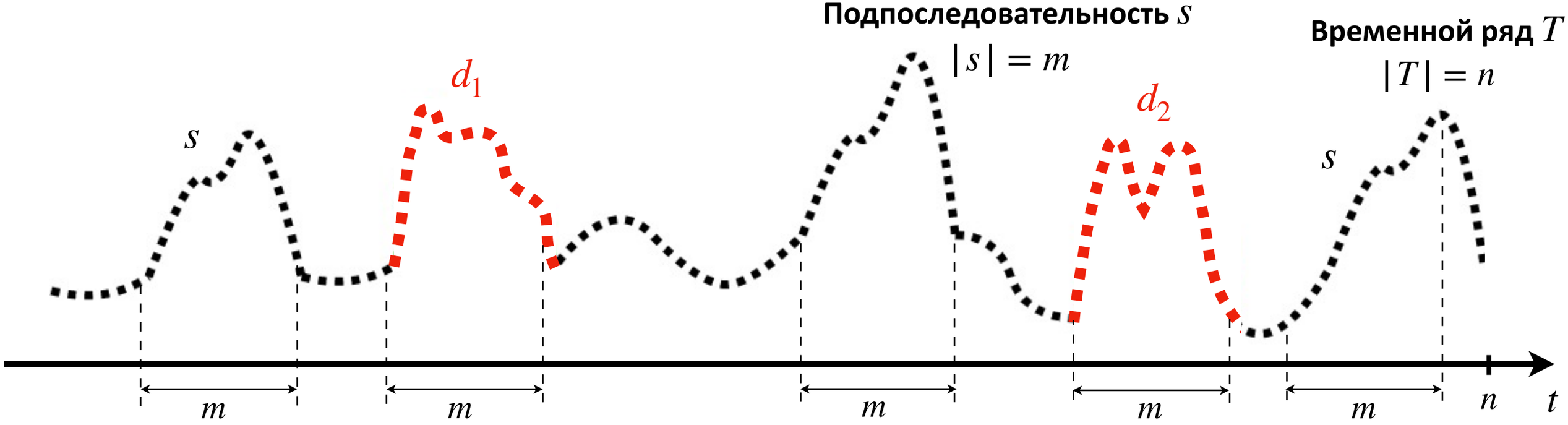
- Поиск аномалий в показаниях **более 10^4 датчиков** позволяет оперативно обнаружить повреждения в ЛЭП[1)]
- Дискретность показаний датчика: **240 раз в сек**.

[1)] Leon R. A., Vittal V., Manimaran G. Application of Sensor Network for Secure Electric Energy Infrastructure. IEEE Transactions on Power Delivery 22(2): 1021–1028. 2007.

Постановка задачи

- **Диссонанс**[1)] – подпоследовательность ряда, расстояние от которой до наиболее похожей на нее подпоследовательности не ниже порога r
- **Дано:** временной ряд T, длина диссонанса m, порог r
- **Найти:** $D = \{d_1, d_2, \dots\}$,

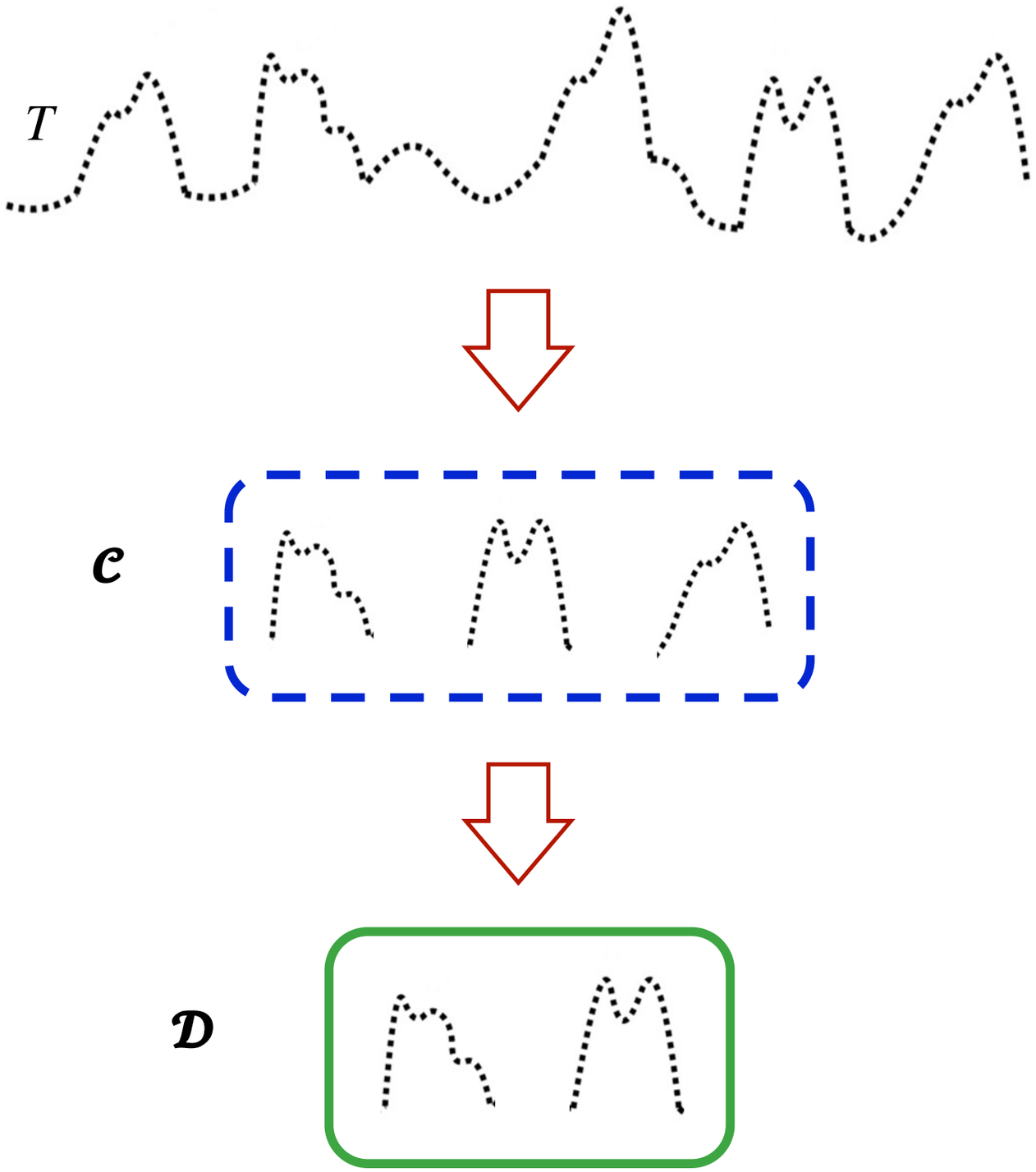
$$d_i \in D \Leftrightarrow \forall s \in T \min_{s \cap d_i = \varnothing} \mathrm{ED}(d_i, s) \geq r$$



Количество подпоследовательностей s длины m в ряде T длины n: $N = n - m + 1$

1) Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: finding unusual time series in terabyte sized datasets. Knowl. Inf. Syst. 17(2): 241–262. 2008.

Основная идея поиска диссонансов



1. Отбор

За одно сканирование ряда сформировать **множество кандидатов** в диссонансы

2. Очистка

За одно сканирование ряда **отбросить кандидатов**, которые не являются диссонансами

Параллельный алгоритм для GPU

Вход:

T – временной ряд

m – длина подпоследовательности

Выход:

D – множество диссонансов

1: $D \leftarrow \emptyset$

2: $r \leftarrow 2\sqrt{m}$

3: $nnDist \leftarrow -\infty$

4: $Bitmap_{cand} \leftarrow$ TRUE; $Bitmap_{subs} \leftarrow$ TRUE

5: $\{\bar{\mu}, \bar{\sigma}\} \leftarrow$ **ParPrecompute**(T, m)

6: **while** $nnDist < 0$ **do**

7: $C \leftarrow$ **ParSelect**$(T, \bar{\mu}, \bar{\sigma}, Bitmap_{cand}, Bitmap_{subs}, nnDist, m, r^2)$

8: $D \leftarrow$ **ParRefine**$(T, C, \bar{\mu}, \bar{\sigma}, Bitmap_{cand}, nnDist, m, r^2)$

9: $r \leftarrow 0.5 \cdot r$

10: **return** D

1. Предварительная обработка данных

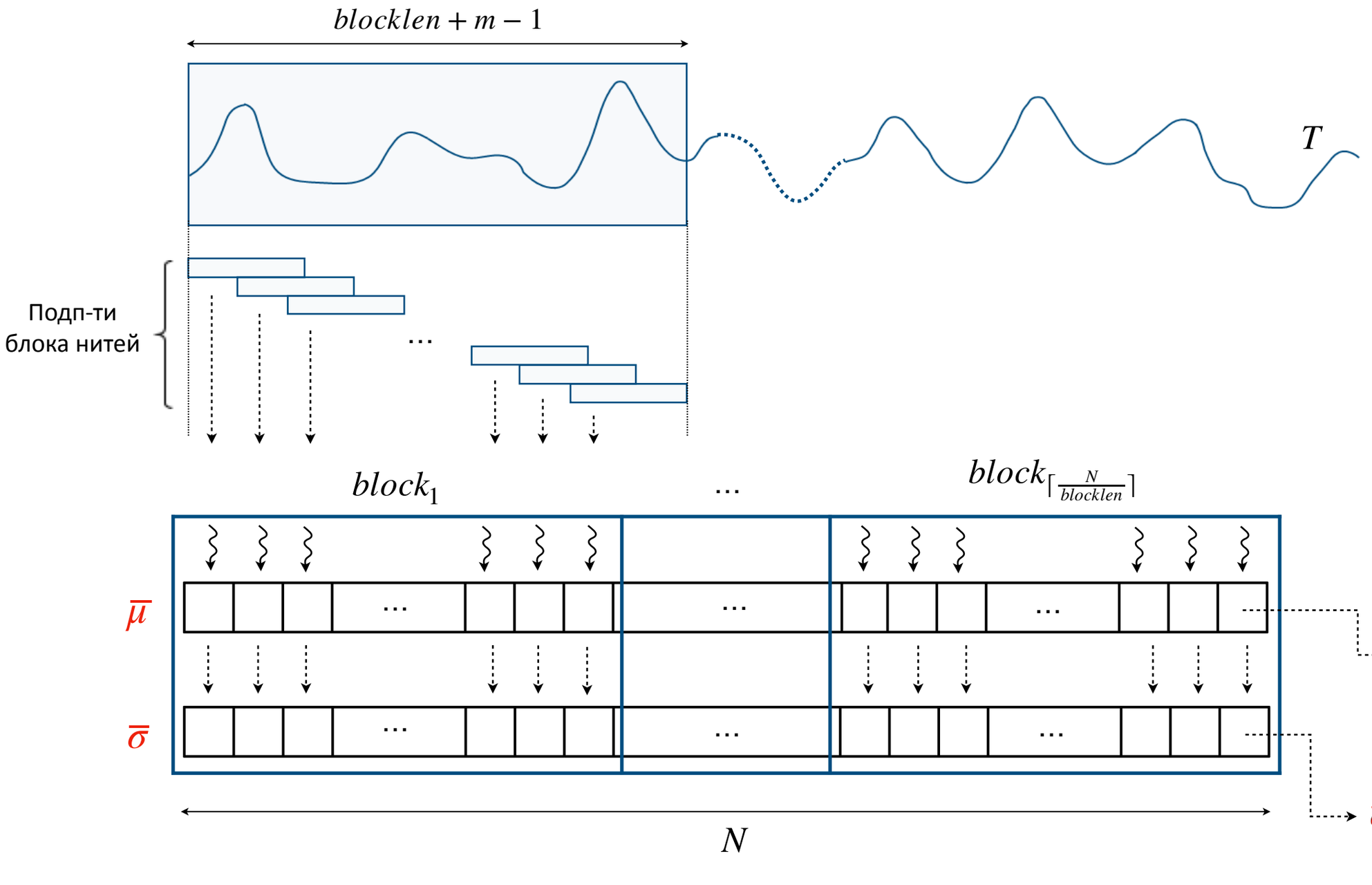
2. Отбор кандидатов

3. Очистка кандидатов

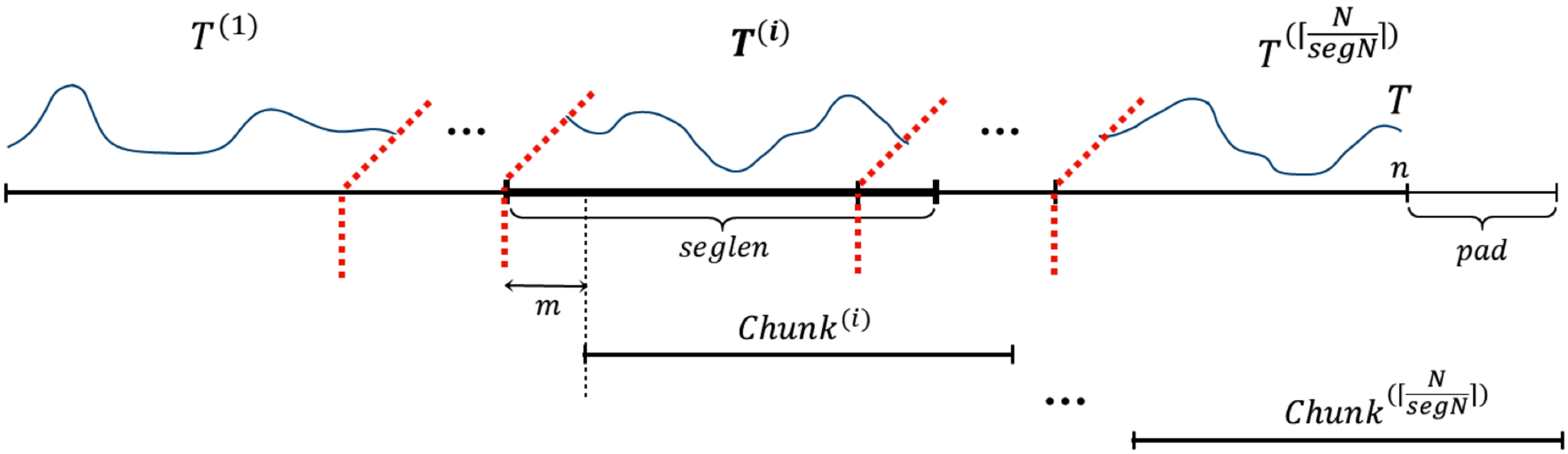
Предварительная обработка данных

Предварительно вычислим $\bar{\mu}$ и $\bar{\sigma}$ для нормализованной евклидовой метрики:

$$\mathrm{ED}^2_{\mathrm{norm}}(T_{i,m}, T_{j,m}) = 2m\left(1 - \frac{QT_{i,j} - m\mu_i\mu_j}{m\sigma_i\sigma_j}\right).$$

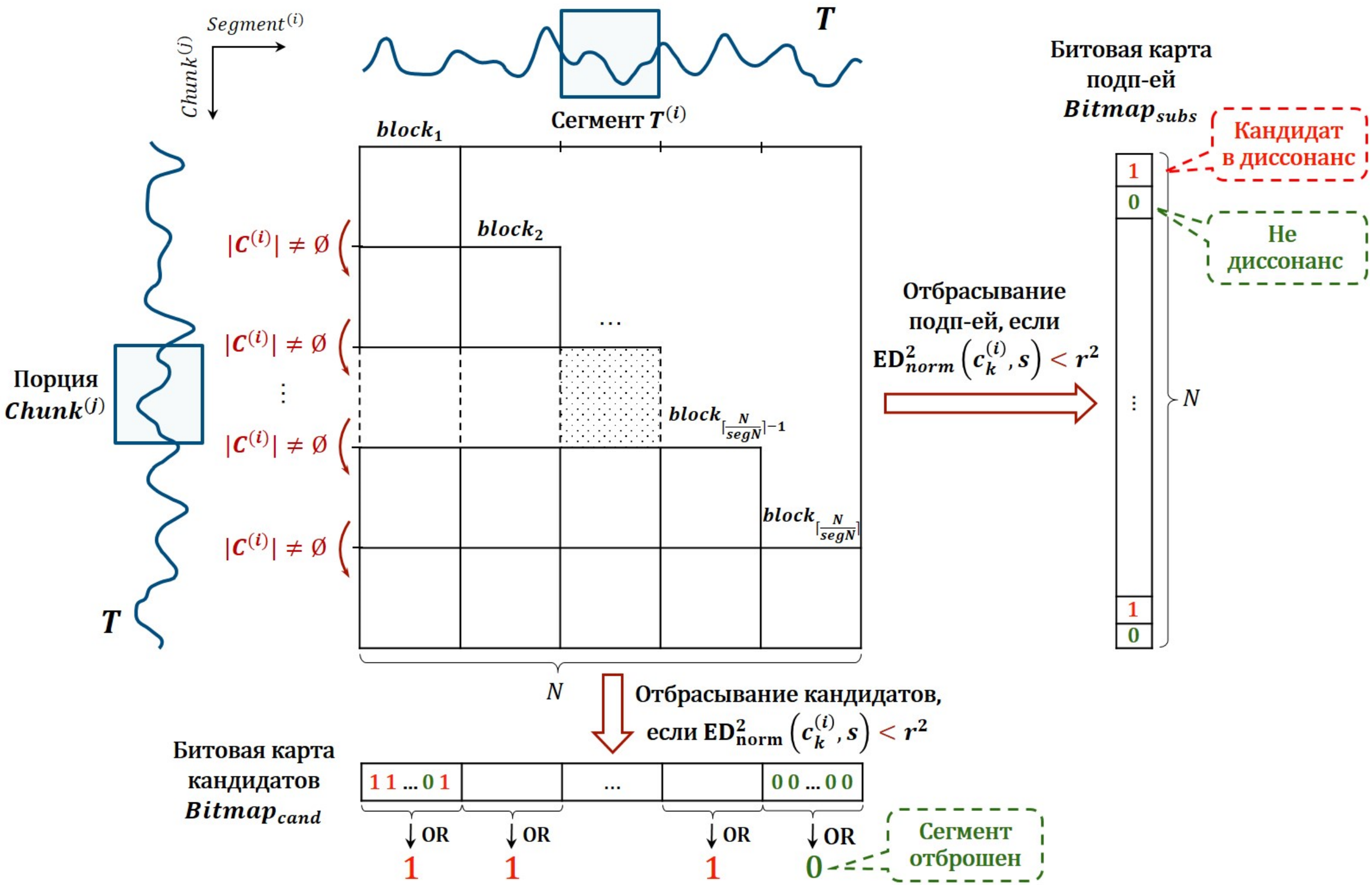


Сегментирование временного ряда для отбора кандидатов

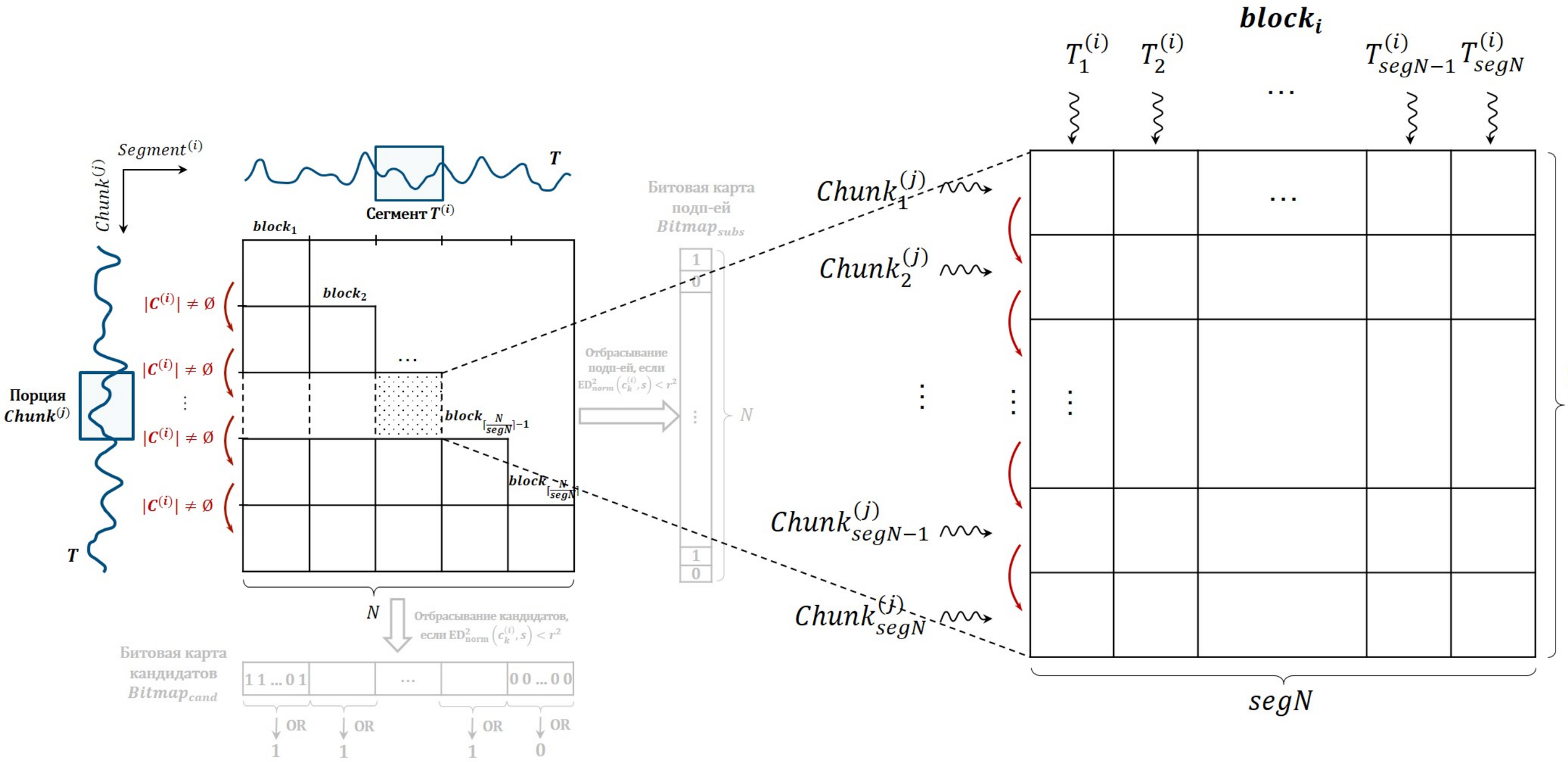


Переменная	Семантика
$seglen$	Длина сегмента, $seglen = segN + m - 1$
$segN$	Количество кандидатов в сегменте (параметр) кратно размеру варпа ($warp\ size = 32$)
$T^{(i)}$	Сегмент кандидатов
$Chunk^{(j)}$	Элементы ряда для отбрасывания кандидатов на очередной итерации
pad	Количество фиктивных элементов ряда

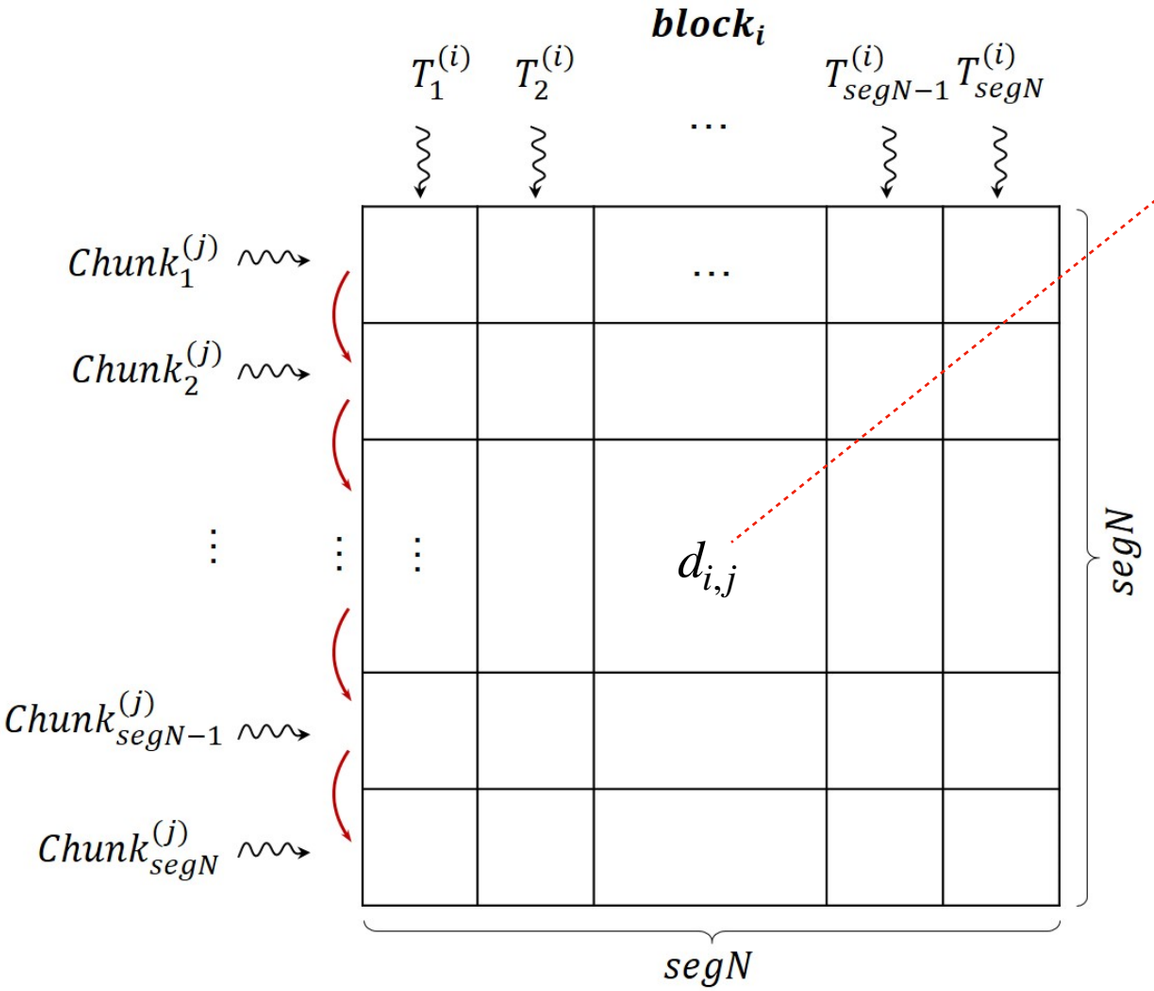
Отбор кандидатов в диссонансы: блочное распараллеливание



Отбор кандидатов в диссонансы: распараллеливание по нитям



Отбор кандидатов в диссонансы: распараллеливание по нитям

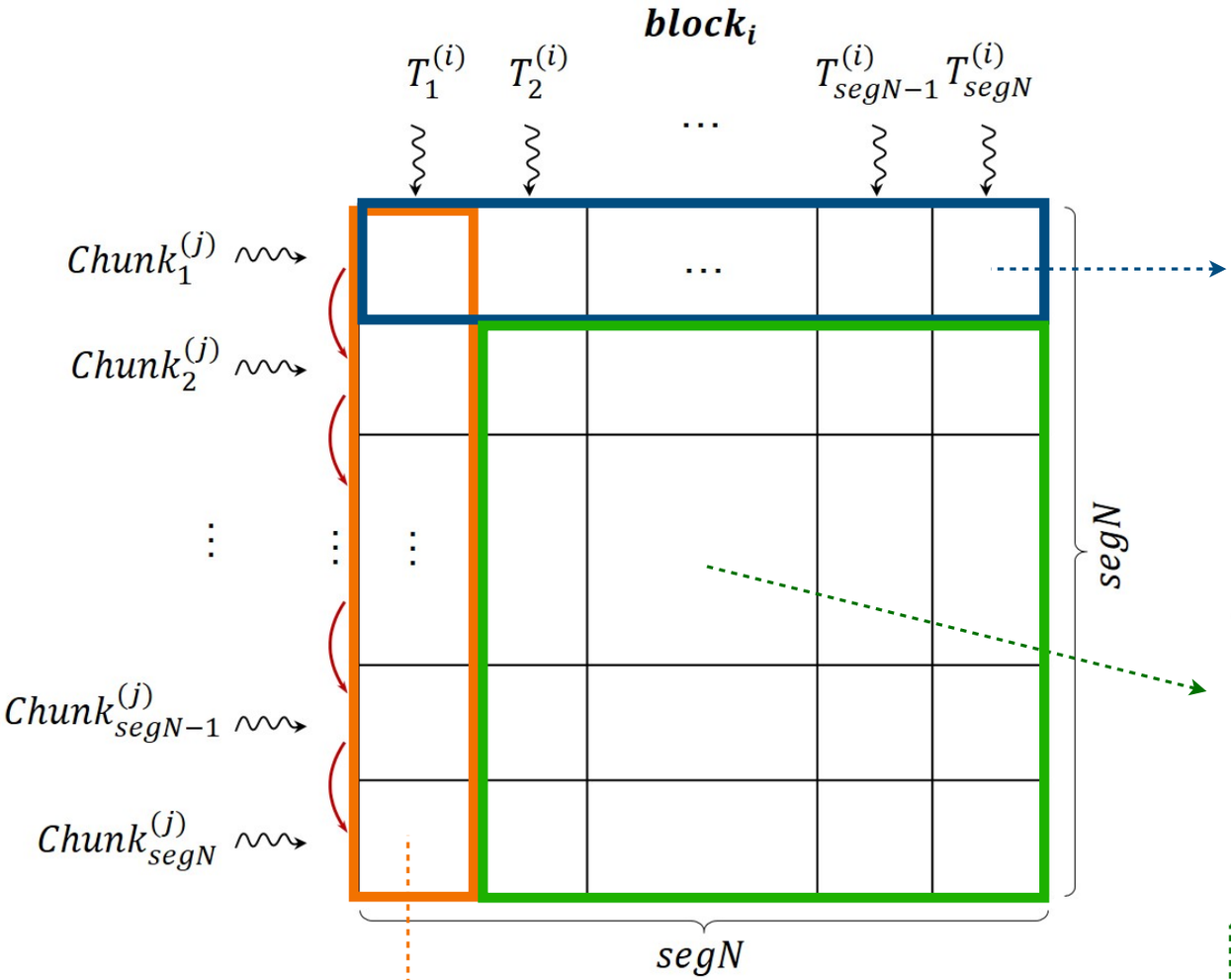


Расстояние между двумя подп-ями $T_{i,m}$ и $T_{j,m}$:

$$d_{i,j} = ED^2_{norm}(T_{i,m}, T_{j,m}) = 2m\left(1 - \frac{QT_{i,j} - m\mu_i\mu_j}{m\sigma_i\sigma_j}\right)$$

Скалярное произведение

Отбор кандидатов в диссонансы: распараллеливание по нитям

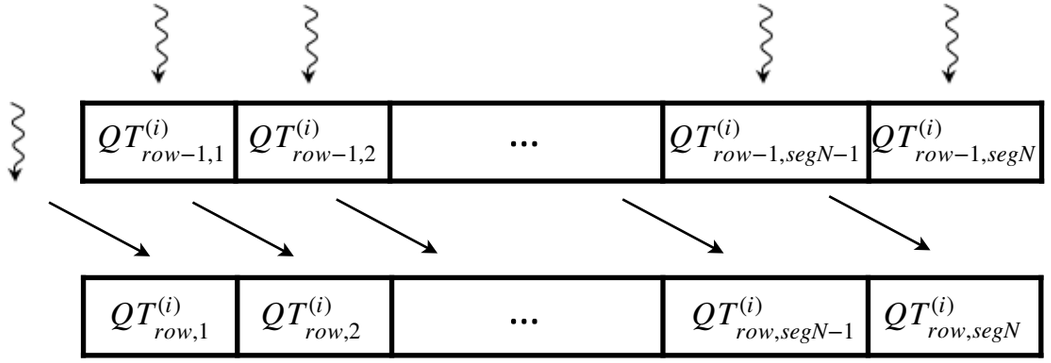


Расстояние между двумя подп-ями $T_{i,m}$ и $T_{j,m}$:

$$d_{i,j} = ED^2_{norm}(T_{i,m}, T_{j,m}) = 2m\left(1 - \frac{QT_{i,j} - m\mu_i\mu_j}{m\sigma_i\sigma_j}\right)$$

Скалярное произведение

$$QT^{(i)}_{1,tid} = \sum_{k=1}^{m} T^{(i)}_{tid}[k] \cdot Chunk^{(j)}_{1}[k]$$



Вычисленное QT на предыдущей итерации

$$QT^{(i)}_{row,tid} = QT^{(i)}_{row-1,tid-1} - T^{(i)}_{tid-1}[1] \cdot Chunk^{(j)}_{tid-1}[1] + T^{(i)}_{tid}[m] \cdot Chunk^{(j)}_{tid}[m]$$

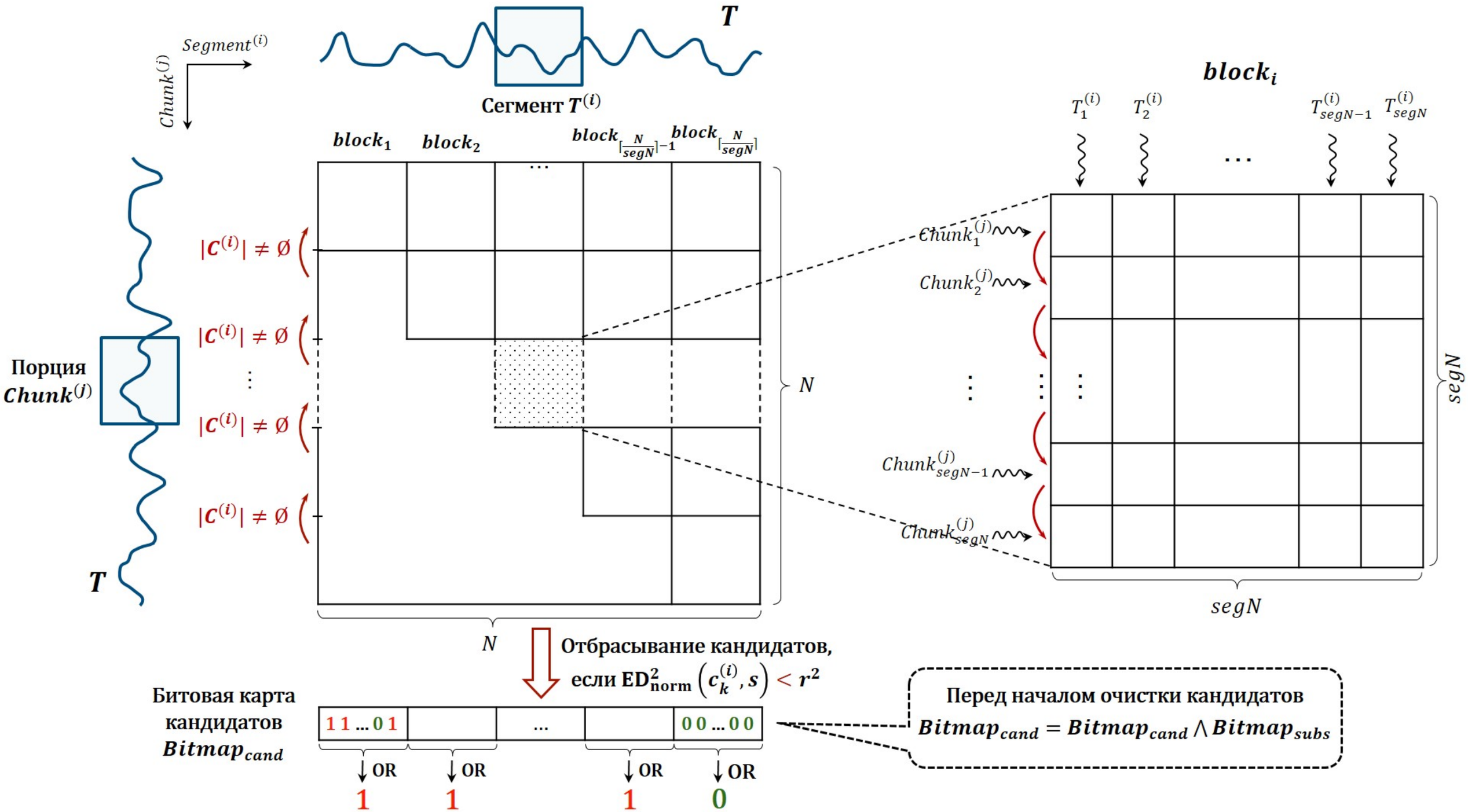
Вычислительная сложность составляет $O(1)$ вместо $O(m)$!

$$QT^{(i)}_{tid,1} = \sum_{k=1}^{m} T^{(i)}_{1}[k] \cdot Chunk^{(j)}_{tid}[k]$$

Очистка кандидатов

Блочное распараллеливание

Распараллеливание по нитям



Эксперименты

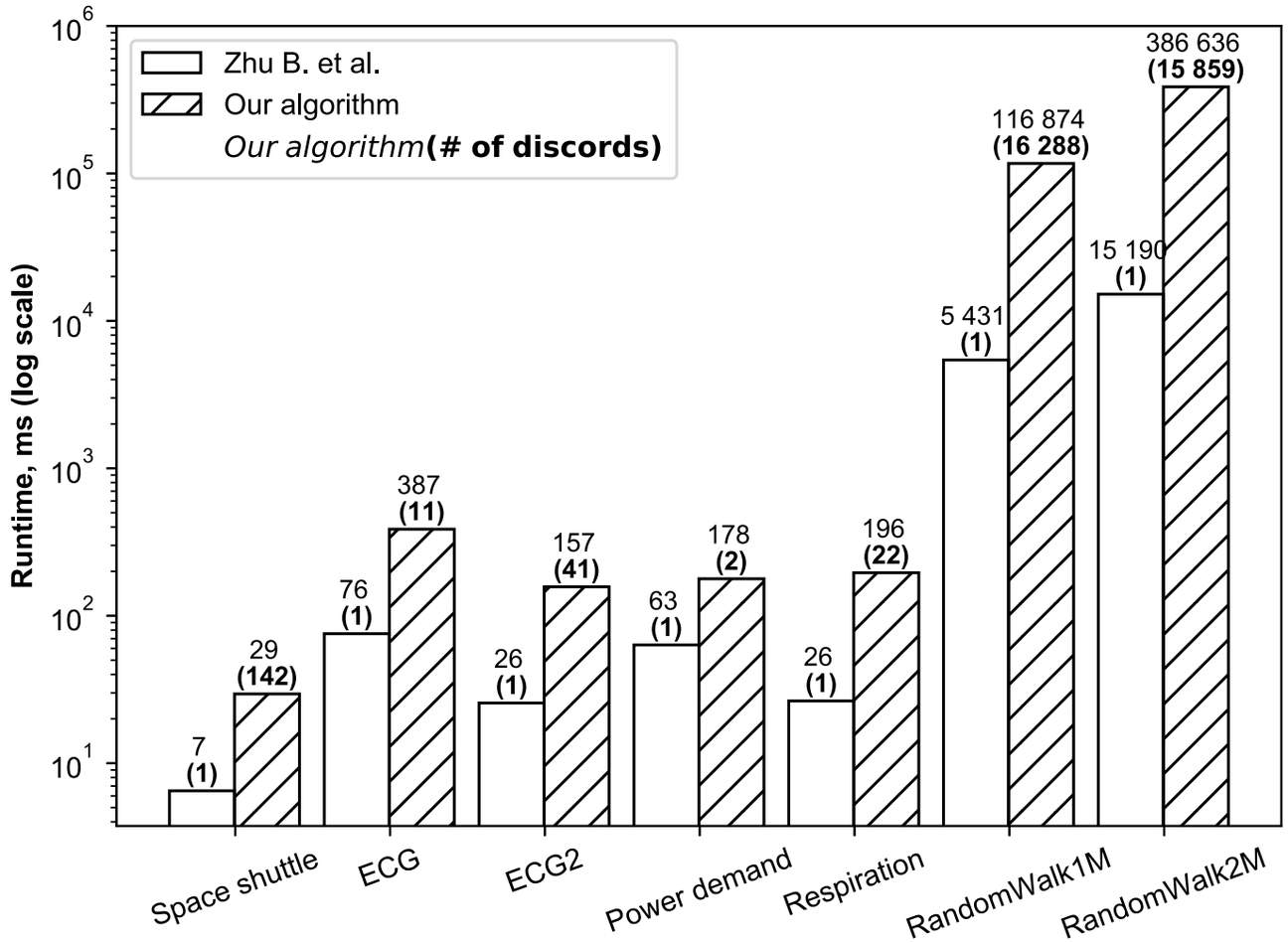
- **Аппаратная платформа**
  - Графический процессор: **NVIDIA Tesla V100 SXM2**
  - Кол-во физ. ядер: **5 120 (84 мультипроцессора)**
  - Тактовая частота: **1.312 ГГц**
  - Пиковая произв-ть: **15.7 TFLOPS**
  - Оперативная память: **32 Гб**
- **Данные**

Временной ряд	**Длина ряда,** n	**Длина диссонанса,** m	**Семантика**
Space shuttle	5 000	150	Показания датчика тока соленоида на космическом корабле NASA[1)]
ECG	45 000	200	ЭКГ взрослого пациента[2)]
ECG2	21 600	400	
Power demand	33 220	750	Годовое энергопотребление здания[3)]
Respiration	24 125	250	Дыхание человека по расширению грудной клетки[4)]
RandomWalk1M	$1 \cdot 10^7$	512	Синтетический ряд (модель случайных блужданий)[5)]
RandomWalk2M	$2 \cdot 10^7$	512	

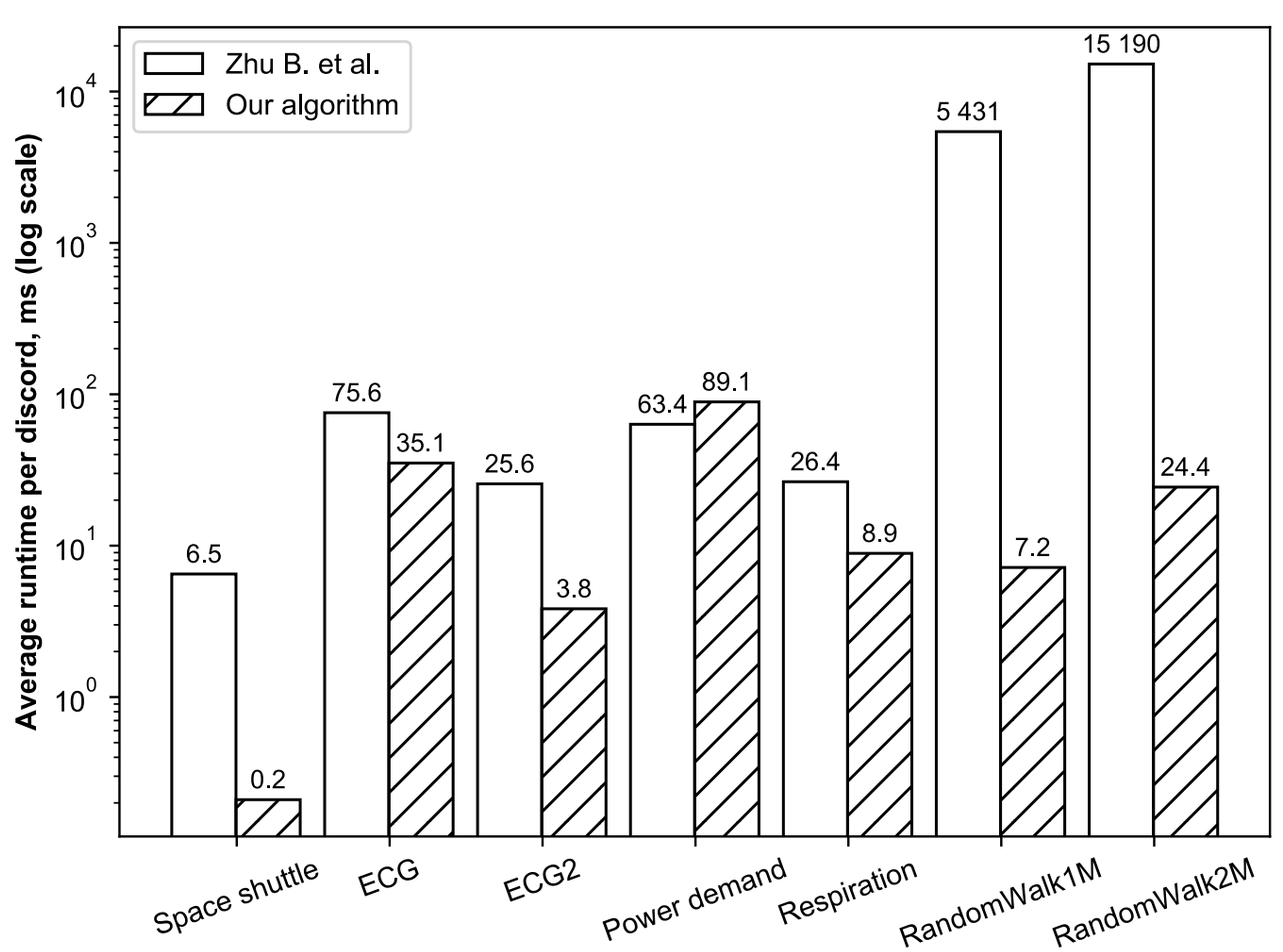
[1)] Ferrell B., Santuro S. NASA shuttle valve data 2005. URL: http://www.cs.fit.edu/~pkc/nasa/data/.
[2)] Goldberger A., et al. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circulation 101(23): 215–220. doi: 10.1161/01.CIR.101.23.e215.
[3)] van Wijk J.J., van Selow E.R. Cluster and calendar based visualization of time series data. INFOVIS'99: 4–9. doi: 10.1109/INFVIS.1999.801851.
[4)] Keogh E., Lin J., Fu A. HOT SAX: Finding the most unusual time series subsequence: Algorithms and applications. Proc. 5th IEEE Int. Conf. Data Mining 2004: 440–449. URL: http://www.cs.ucr.edu/~eamonn/discords/.
[5)] Pearson K. The problem of the random walk. Nature 72(394). doi: 10.1038/072342a0.

Производительность

Время выполнения
(поиск **всех** диссонансов)



Среднее время выполнения
(поиск **одного** диссонанса)



Предложенный алгоритм опережает аналог по среднему времени на поиск одного диссонанса: на реальных данных – от 2 до 32 раз, на синтетических данных – более чем в 750 раз

Zhu B., Jiang Y., Gu M., Deng Y. A GPU Acceleration framework for motif and discord based pattern mining. IEEE Transactions on Parallel and Distributed Systems 32(8): 1987–2004. 2021. doi: 10.1109/TPDS.2021.3055765.

Заключение

- Предложен новый параллельный алгоритм поиска аномалий временного ряда для графического процессора, который по результатам экспериментов опережает аналог
- Будущие исследования:
  - Разработка распределенной версии алгоритма поиска аномалий временного ряда в заданном диапазоне длин

Спасибо за внимание! Вопросы?

Яна Александровна Краева
kraevaya@susu.ru

Работы по теме исследования

Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: finding unusual time series in terabyte sized datasets. Knowl. Inf. Syst. 17(2): 241–262. 2008.	MR-DRAG разработан на основе парадигмы MapReduce.
Huang T., Zhu Y., Mao Y., et al. Parallel discord discovery. PAKDD 2016. LNCS 9652. Springer, 2016. P. 233–244.	PDD использует парадигму мастер-рабочие и платформу кластера Spark.
Wu Y., Zhu Y., Huang T., et al. Distributed discord discovery: Spark based anomaly detection in time series. Proc. of the 17th IEEE Int. Conf. on High Performance Computing and Communications. IEEE Press, 2015. P. 154–159.	DDD разработан для вычислительного кластера на основе использования Apache Spark и HDFS.
Zymbler M., Polyakov A., Kipnis M. Time series discord discovery on Intel Many-core systems. PCT 2019. CCIS 1063. P. 168–182. 2019.	Алгоритм разработан для ускорителей архитектур Intel MIC и NVIDIA GPU.
Zimmerman Z., Kamgar K., Senobari N.S., et al. Matrix Profile XIV: Scaling time series motif discovery with GPUs to break a quintillion pairwise comparisons a day and beyond. SoCC 2019. P. 74–86. 2019.	SCAMP разработан на основе концепции матричного профиля для GPU.
Zymbler M., Grents A., Kraeva Ya., Kumar S. A Parallel Approach to Discords Discovery in Massive Time Series Data. Computers, Materials & Continua 66(2): 1867–1876. 2021.	Алгоритм разработан для кластерной системы с вычислительными узлами на базе многоядерных ускорителей Intel Xeon Phi.
Zhu B., Jiang Y., Gu M., Deng Y. A GPU Acceleration framework for motif and discord based pattern mining. IEEE Transactions on Parallel and Distributed Systems 32(8): 1987–2004. 2021.	**Алгоритм поиска Top-1 диссонанса временного ряда разработан для GPU.**

Отбор кандидатов

Сканировать ряд $\boldsymbol{T}$:

текущая подпоследовательность $\boldsymbol{s}$

Кандидат := TRUE

для всех $\boldsymbol{c_i} \in \boldsymbol{\mathcal{C}}$

если $\mathrm{ED}(\boldsymbol{s}, \boldsymbol{c_i}) < r$ **and** $\boldsymbol{s} \cap \boldsymbol{c_i} = \emptyset$ то

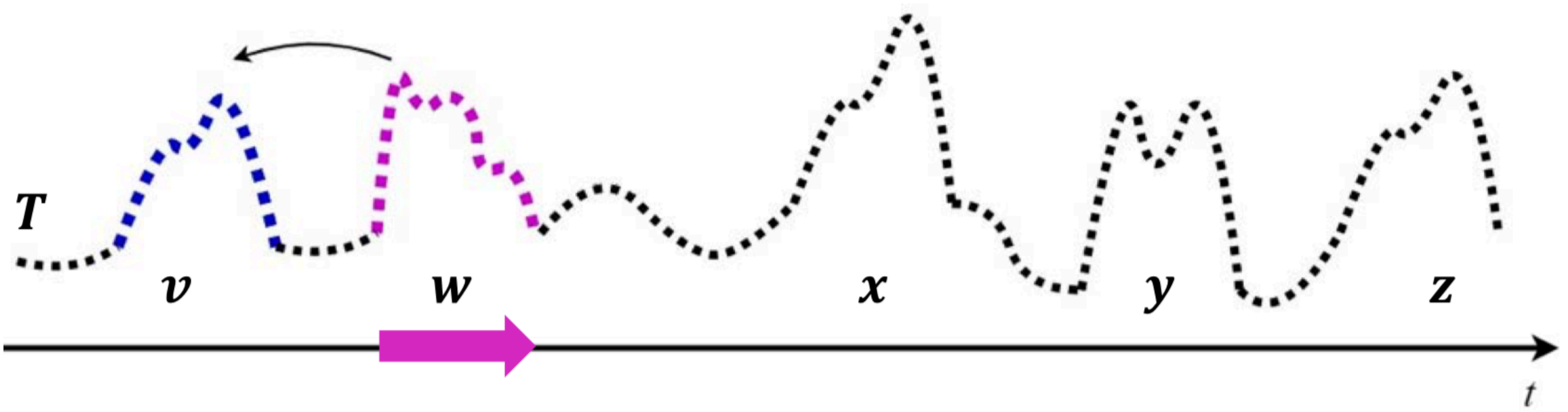
$\boldsymbol{\mathcal{C}} := \boldsymbol{\mathcal{C}} \setminus \boldsymbol{c_i}$; Кандидат := FALSE

если Кандидат = TRUE то $\boldsymbol{\mathcal{C}} := \boldsymbol{\mathcal{C}} \cup \boldsymbol{s}$

$\boldsymbol{\mathcal{C}} = \{\boldsymbol{v}\}$

$\mathrm{ED}(\boldsymbol{w}, \boldsymbol{v}) \geq r$

$\boldsymbol{\mathcal{C}} = \{\boldsymbol{v}, \boldsymbol{w}\}$



Отбор кандидатов

Сканировать ряд T:

текущая подпоследовательность s

Кандидат := TRUE

для всех $c_i \in \mathcal{C}$

если $\mathrm{ED}(s, c_i) < r$ **and** $s \cap c_i = \emptyset$ то

$\mathcal{C} := \mathcal{C} \setminus c_i$; Кандидат := FALSE

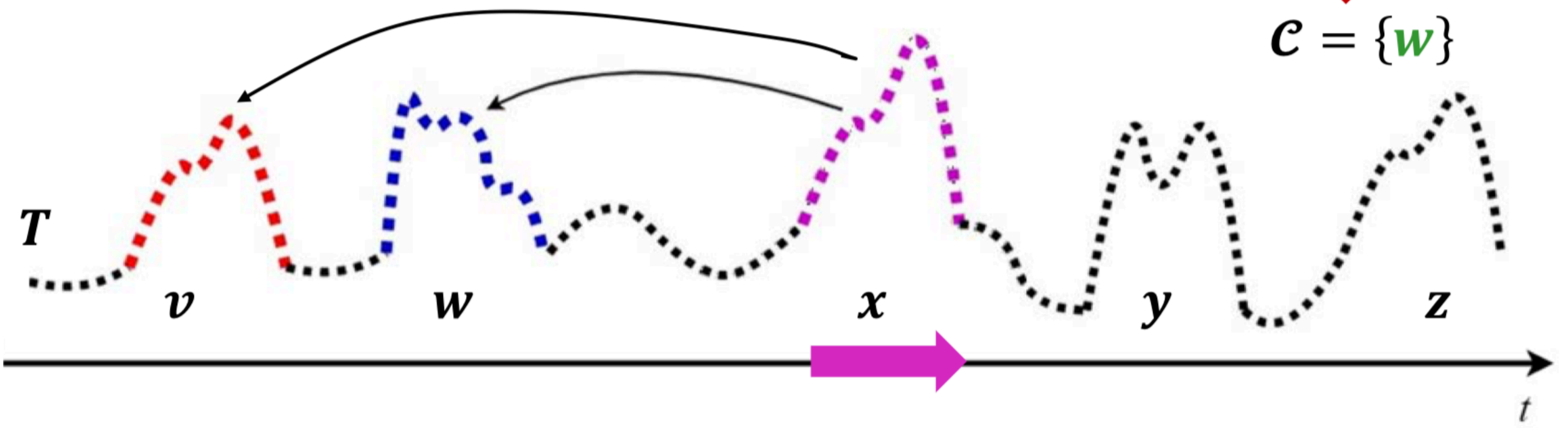
если Кандидат = TRUE то $\mathcal{C} := \mathcal{C} \cup s$

$\mathcal{C} = \{v, w\}$

$\mathrm{ED}(x, v) < r$

$\mathrm{ED}(x, w) \geq r$

$\mathcal{C} = \{w\}$



Отбор кандидатов

Сканировать ряд $\boldsymbol{T}$:

текущая подпоследовательность $\boldsymbol{s}$

Кандидат := TRUE

для всех $\boldsymbol{c_i} \in \boldsymbol{\mathcal{C}}$

если $\mathrm{ED}(\boldsymbol{s}, \boldsymbol{c_i}) < r$ **and** $\boldsymbol{s} \cap \boldsymbol{c_i} = \emptyset$ то

$\boldsymbol{\mathcal{C}} := \boldsymbol{\mathcal{C}} \setminus \boldsymbol{c_i}$; Кандидат := FALSE

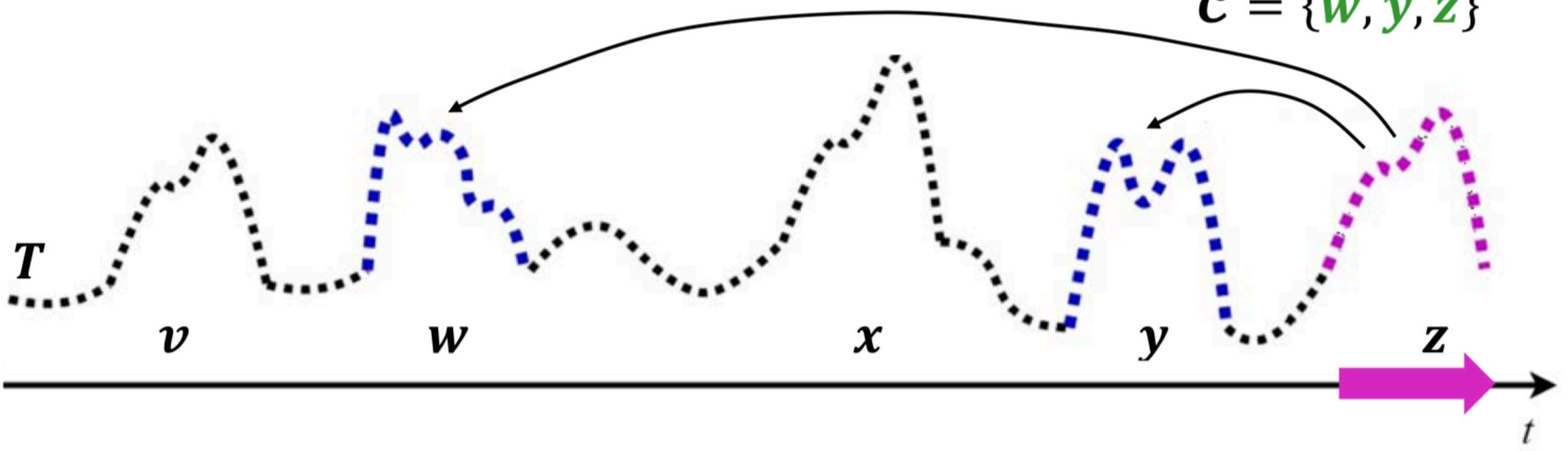
если Кандидат = TRUE то $\boldsymbol{\mathcal{C}} := \boldsymbol{\mathcal{C}} \cup \boldsymbol{s}$

$\boldsymbol{\mathcal{C}} = \{\boldsymbol{w}, \boldsymbol{y}\}$

$\mathrm{ED}(\boldsymbol{z}, \boldsymbol{w}) \geq r$

$\mathrm{ED}(\boldsymbol{z}, \boldsymbol{y}) \geq r$

$\boldsymbol{\mathcal{C}} = \{\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{z}\}$



Очистка кандидатов

$\mathcal{D} := \mathcal{C}$

Сканировать ряд T:

текущая подпоследовательность s

<u>для всех</u> $d_i \in \mathcal{D}$

<u>если</u> $\mathrm{ED}(s, d_i) < r$ **and** $s \cap d_i = \emptyset$ <u>то</u>

$\mathcal{D} := \mathcal{D} \setminus d_i$

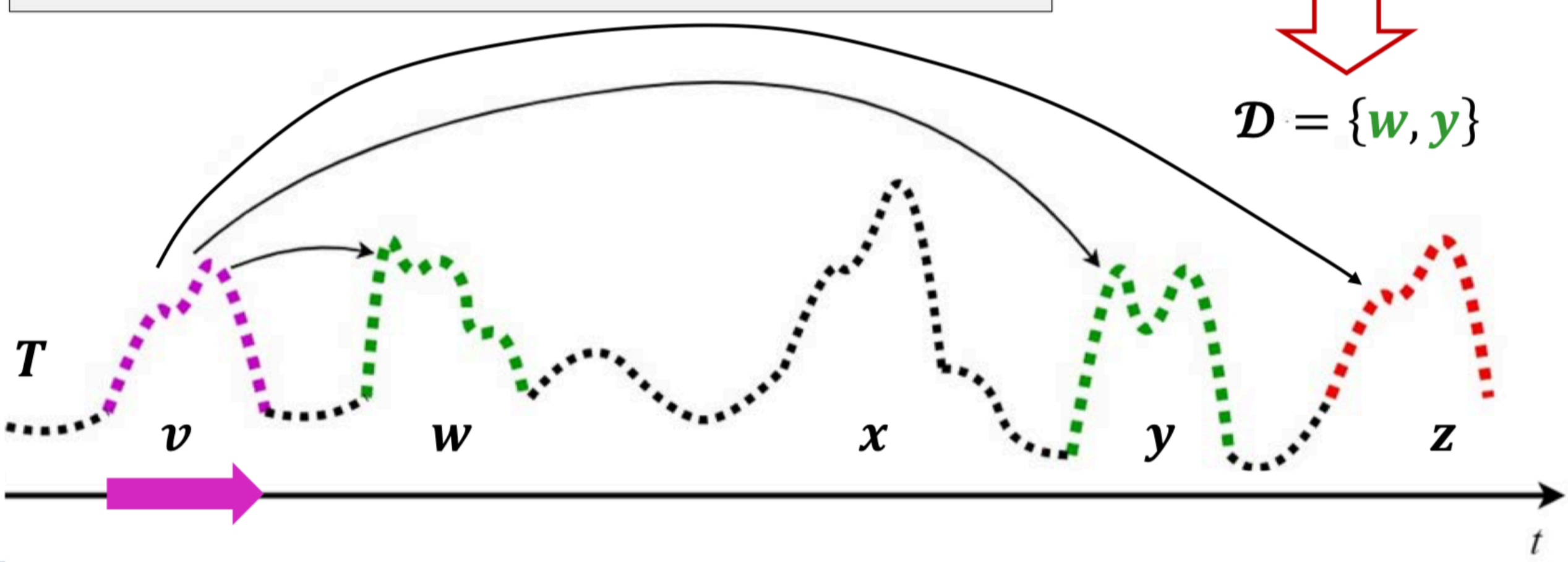
$\mathcal{D} = \{w, y, z\}$

$\mathrm{ED}(v, w) \geq r$

$\mathrm{ED}(v, y) \geq r$

$\mathrm{ED}(v, z) < r$

$\mathcal{D} = \{w, y\}$



Очистка кандидатов

$\mathcal{D} \coloneqq \mathcal{C}$

Сканировать ряд T:

текущая подпоследовательность s

<u>для всех</u> $d_i \in \mathcal{D}$

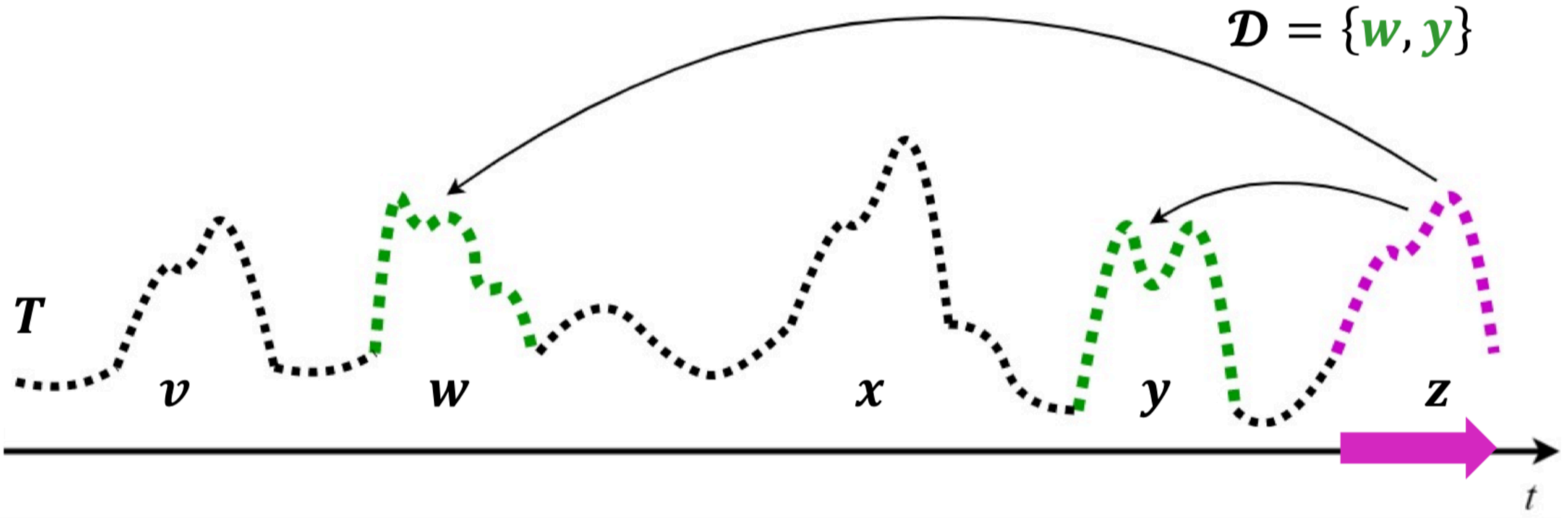
<u>если</u> $\mathrm{ED}(s, d_i) < r$ **and** $s \cap d_i = \emptyset$ <u>то</u>

$\mathcal{D} \coloneqq \mathcal{D} \setminus d_i$

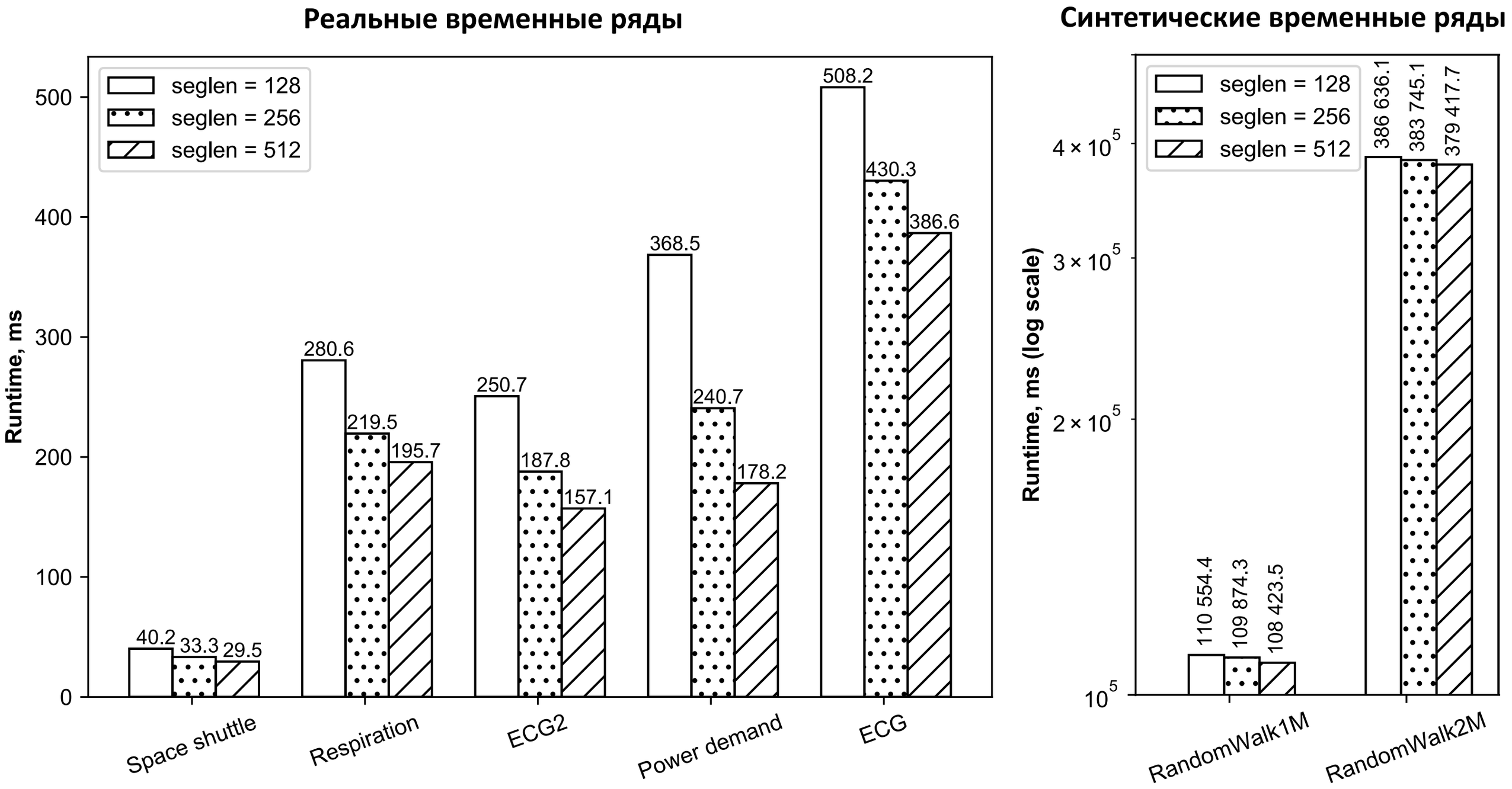
$\mathcal{D} = \{w, y\}$

$\mathrm{ED}(z, w) \geq r$

$\mathrm{ED}(z, y) \geq r$



Производительность: влияние длины сегмента



Производительность алгоритма пропорциональна длине сегмента

Структуры данных