Параллельный алгоритм поиска локально похожих подпоследовательностей временного ряда для ускорителей на базе архитектуры Intel MIC*

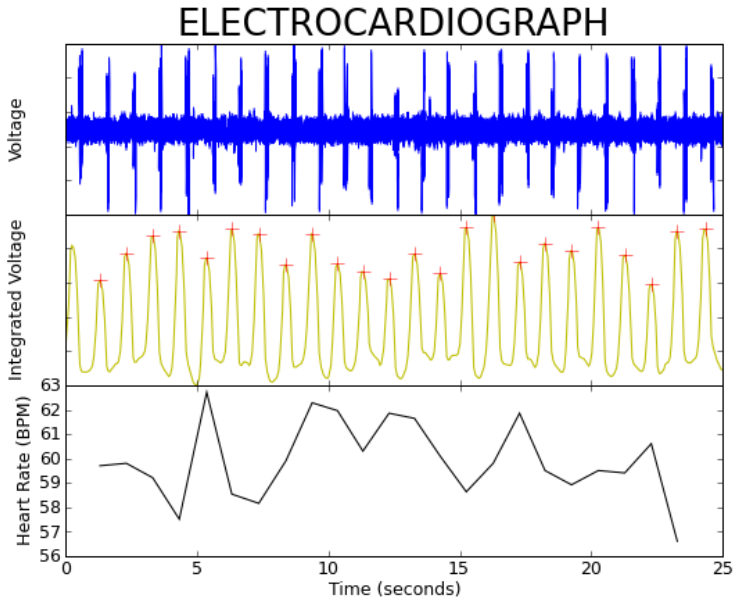
Александр Мовчан, Михаил Цымблер

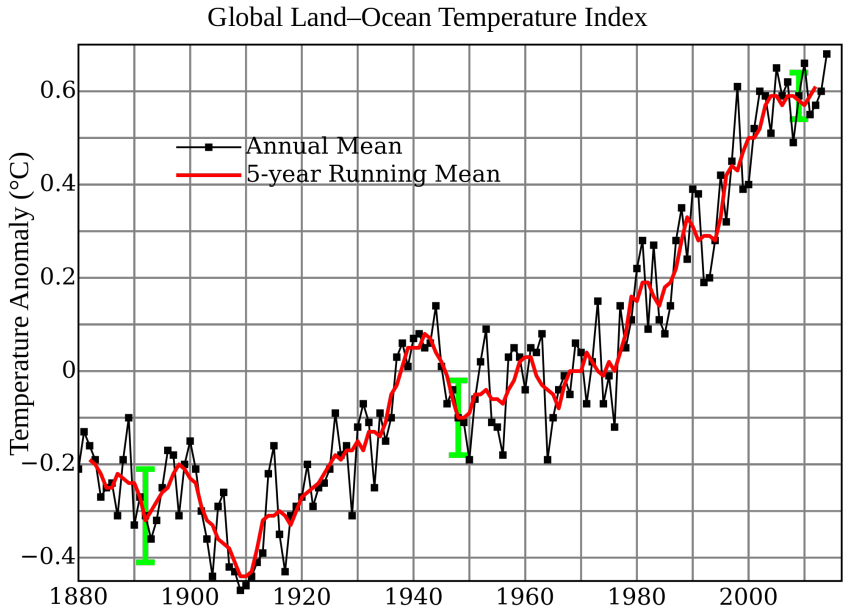
Южно-Уральский государственный университет (НИУ)

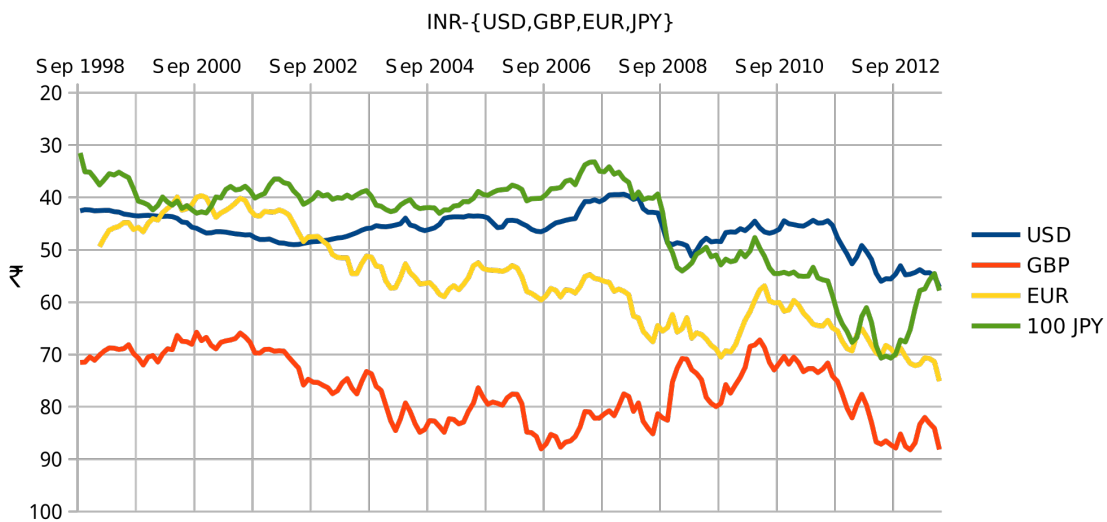
Суперкомпьютерные дни в России

* Работа выполнена при финансовой поддержке Минобрнауки России в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014–2020 годы» (Соглашение №14.574.21.0035 от 17.06.2014, идентификатор RFMEFI57414X0035).

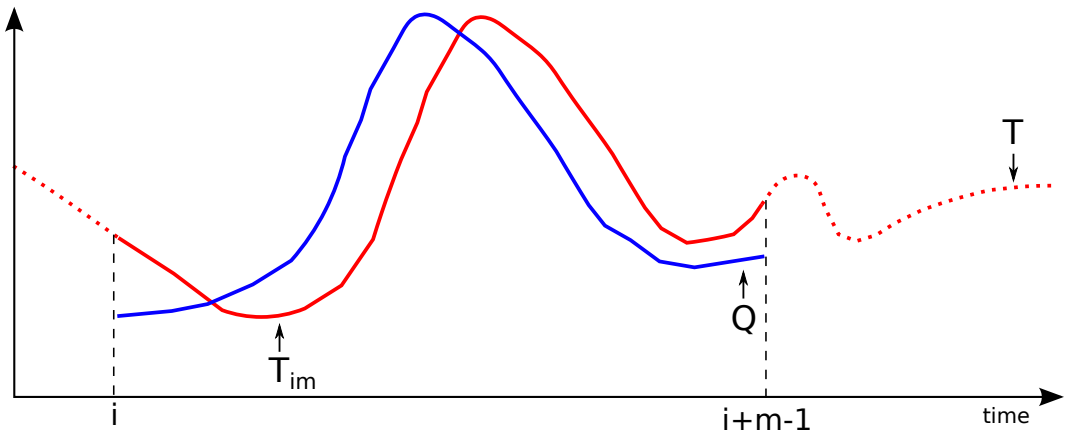
Временные ряды







Обозначения



- *Временной ряд* T
  - $T = t_1, t_2, \ldots, t_N$, где $t_i \in \mathbb{R}$
  - N – длина временного ряда
- *Запрос* Q
  - Q – временной ряд, искомый в T
  - n – длина запроса, $n \ll N$
- *Подпоследовательность* $T_{i\,m}$
  - $T_{i\,m} = t_i, t_{i+1}, \ldots, t_{i+m-1}$
  - $1 \leq i \leq N$ и $i + m \leq N$

Поиск похожих подпоследовательностей

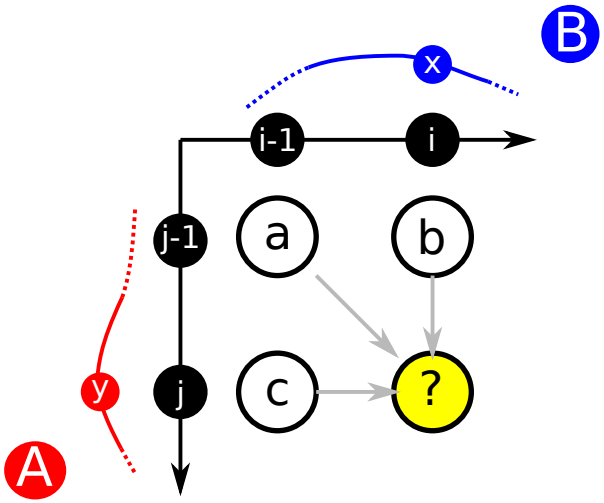
Best-match search: результатом является подпоследовательность $T_{i\,n} \in T$, для которой выполнены следующие условия:

- $\forall m, 1 \leq m \leq N - n, D(T_{i\,n}, Q) < D(T_{m\,n}, Q)$

Local-best-match search: результатом является множество подпоследовательностей $\{T_{i\,m} \in T\}$, для которых выполнены следующие условия:

- $m = n$;
- $D(T_{i\,m}, Q) < \mathcal{E}$;
- $i = \underset{j \in \{i-1, i, i+1\}}{\operatorname{argmin}} D(T_{j\,m}, Q)$.

Динамическая трансформация шкалы времени (DTW)

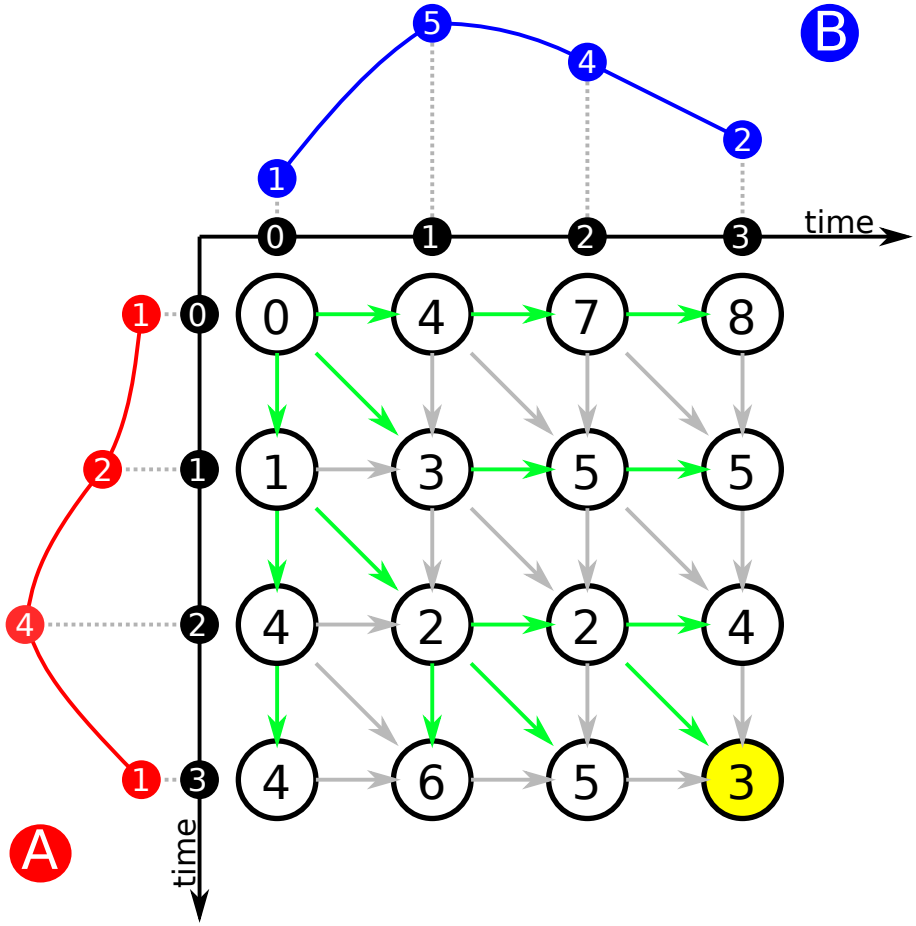


$$d(0,0) = 0$$

$$d(i,j) = |x-y| + min \begin{cases} d(i-1,j) \\ d(i,j-1) \\ d(i-1,j-1) \end{cases}$$

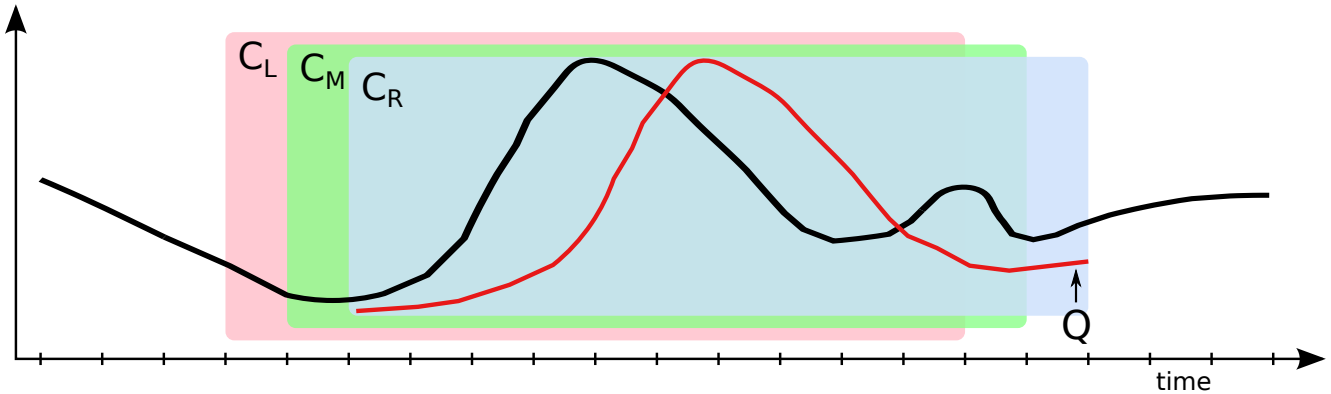
$$= |x-y| + min(a,b,c)$$

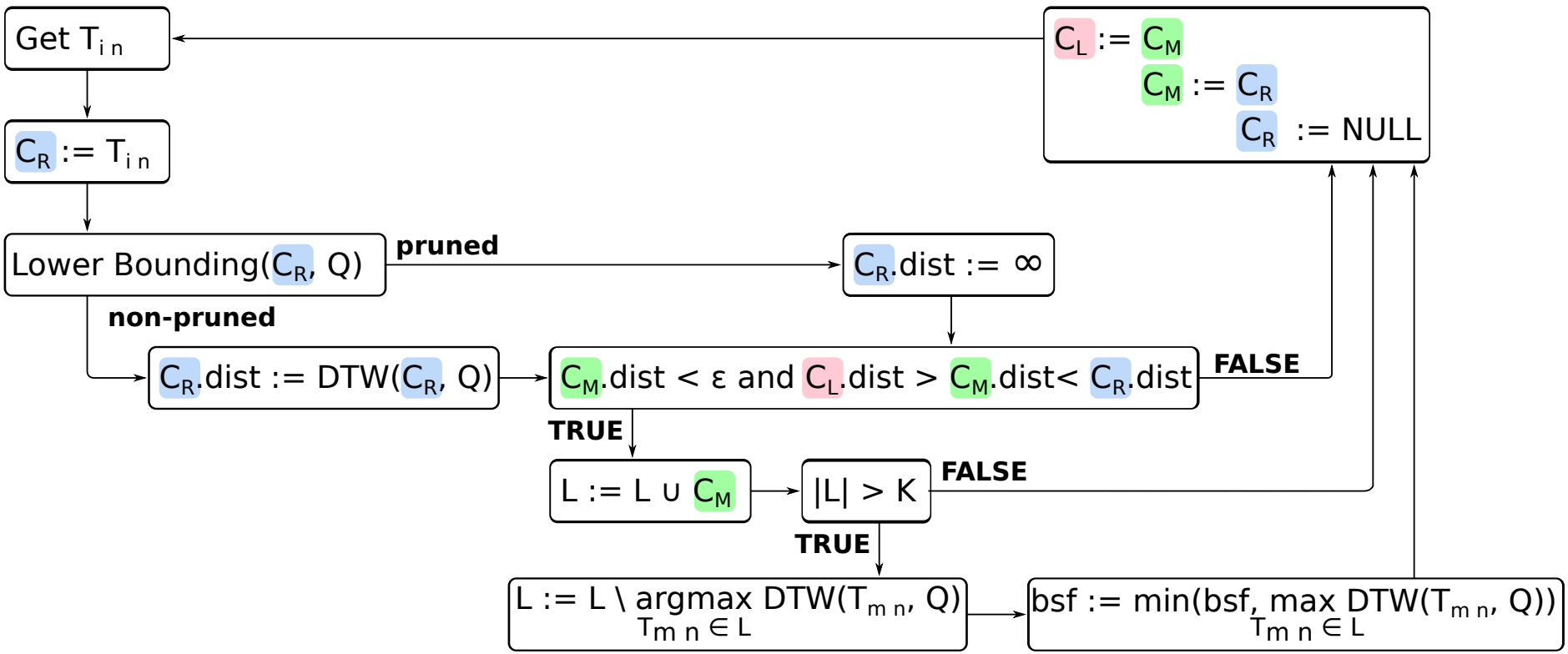
$$\mathrm{DTW(A,\ B)} = \mathrm{d(N,\ N)}$$



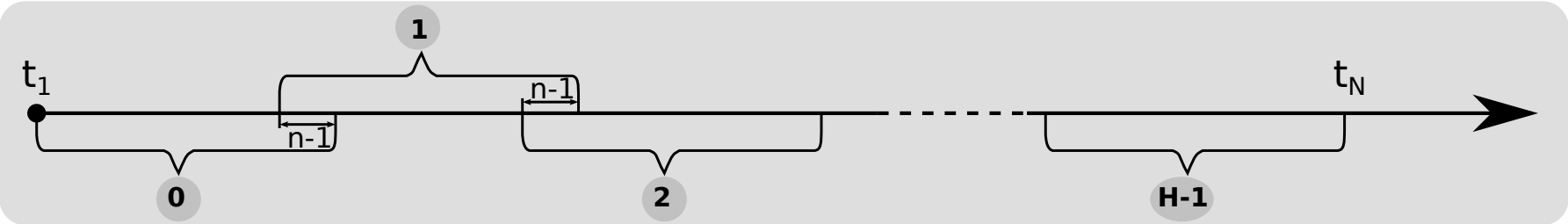
$$\mathrm{DTW(A,\ B)} = \mathrm{d(3,\ 3)} = 3$$

Последовательный алгоритм UCR-DTW-lbm





Распределение временного ряда по нитям



- T разбивается на H равных по длине *сегментов*

$$H = \lceil \frac{N}{P \cdot S} \rceil \cdot P$$

где
P – количество OpenMP нитей,
S – максимальная длина сегмента (параметр алгоритма, $S = 10^6$),
$n \ll S < N$.

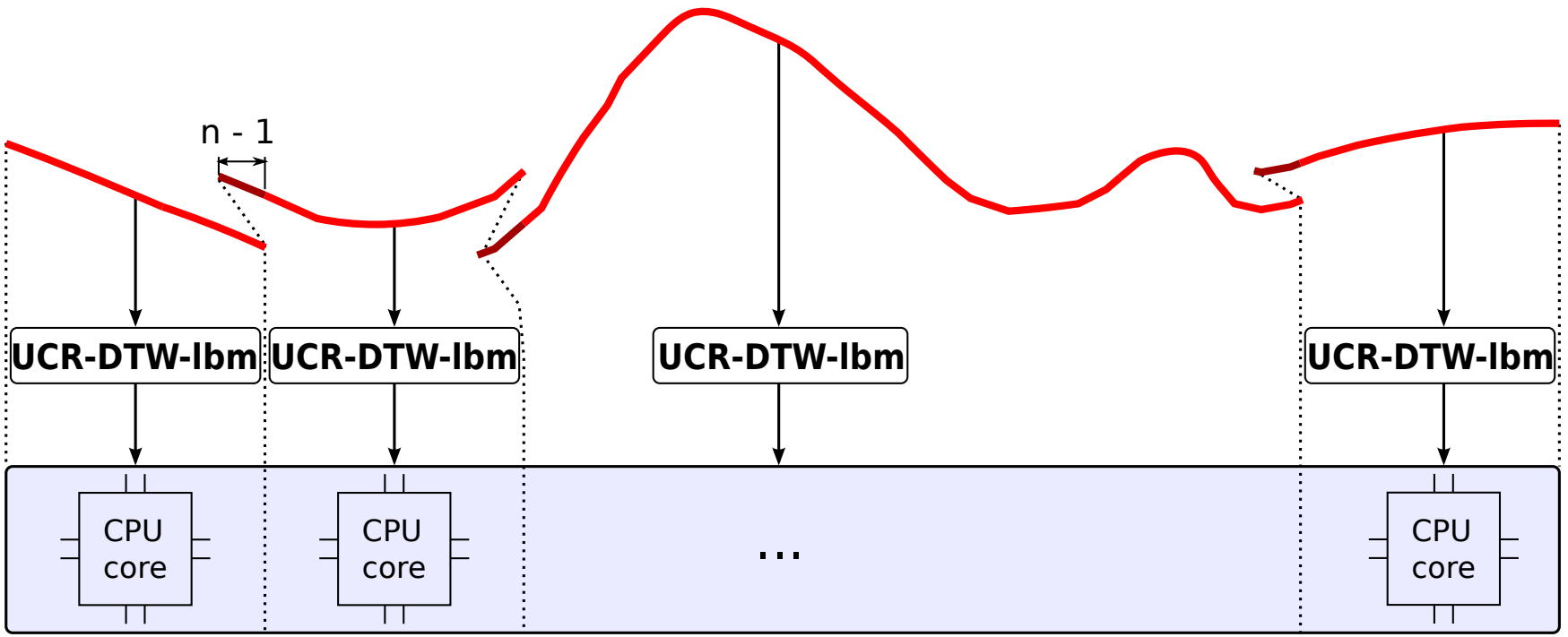
- k-th сегмент, $0 \leq k \leq H - 1$, – подпоследовательность $T_{s\,\ell}$

$$s = \begin{cases} 1 & , k = 0 \\ k \cdot \lfloor \frac{N}{H} \rfloor - n + 2 & , else \end{cases}$$

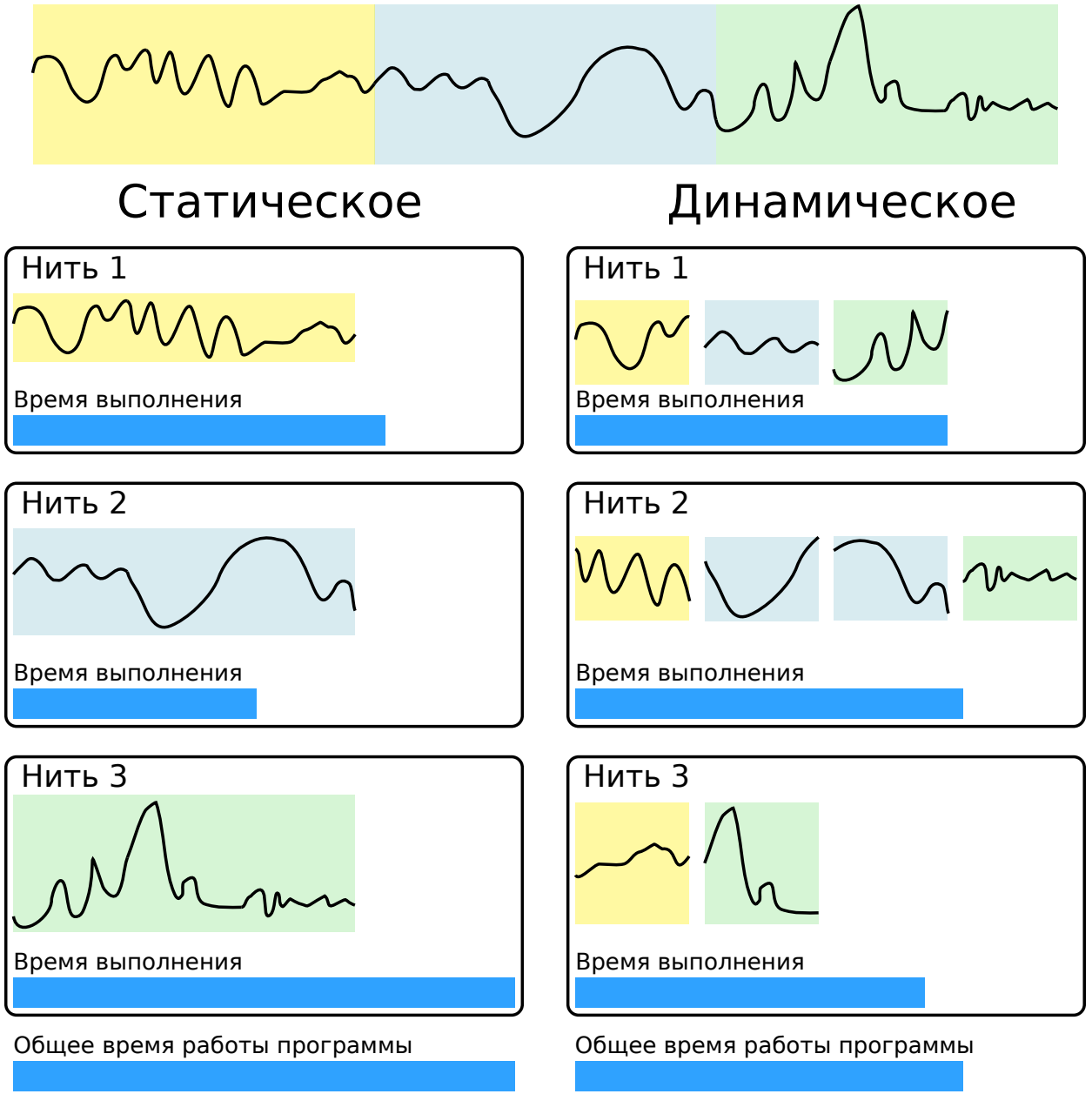
$$\ell = \begin{cases} \lfloor \frac{N}{H} \rfloor & , k = 0 \\ \lfloor \frac{N}{H} \rfloor + n - 1 + (N \ mod \ H) & , k = H - 1 \\ \lfloor \frac{N}{H} \rfloor + n - 1 & , else \end{cases}$$

где
n – длина запроса.

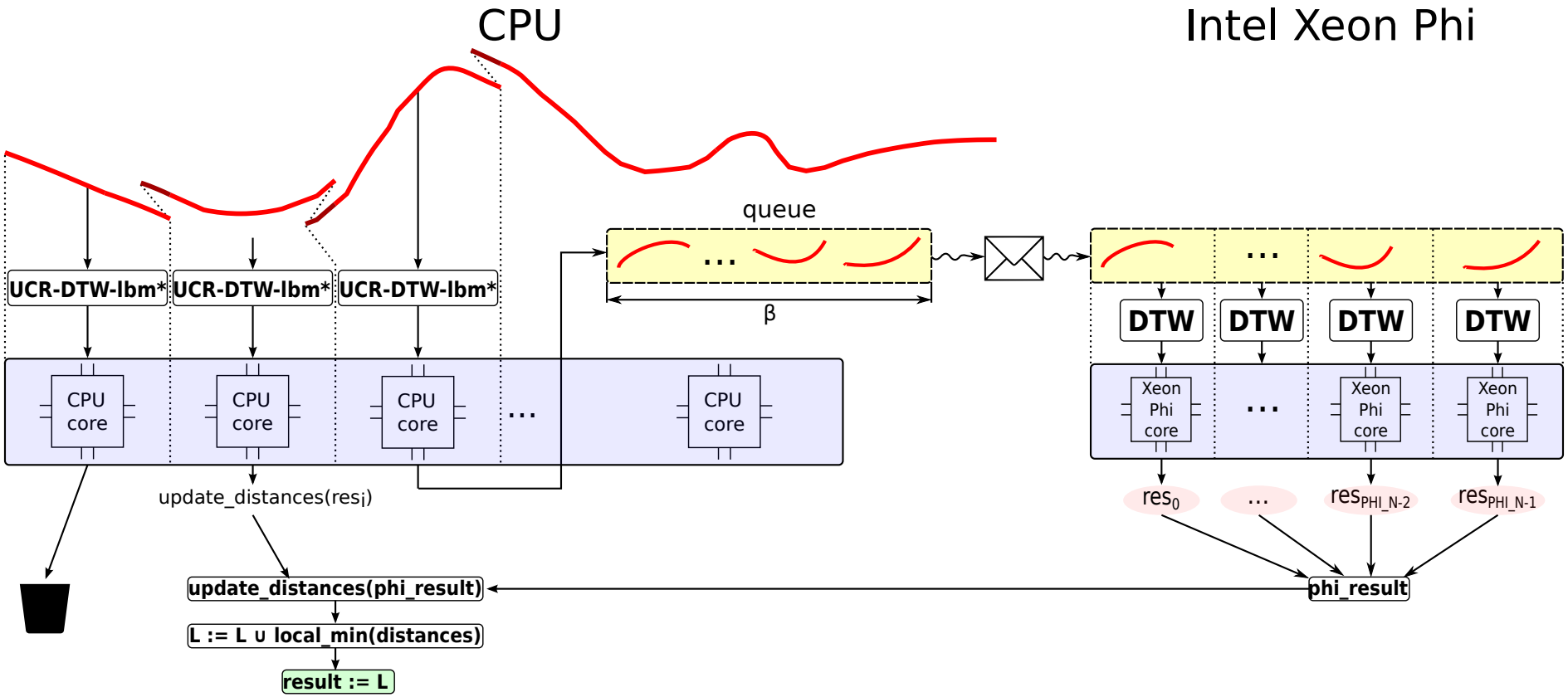
Параллельный алгоритм для CPU



Динамическое и статическое распределение сегментов



Параллельный алгоритм для сопроцессора



Эксперименты: аппаратное обеспечение

Спецификации	Процессор	Сопроцессор
Модель	Intel Xeon X5680	Intel Xeon Phi SE10X
Количество ядер	6	61
Тактовая частота, ГГц	3.33	1.1
Количество нитей на ядро	2	4
Пиковая производительность, TFLOPS	0.371	1.076
Размер памяти, Gb	24	8
Кэш, Mb	12	30.5

Эксперименты: данные и параметры

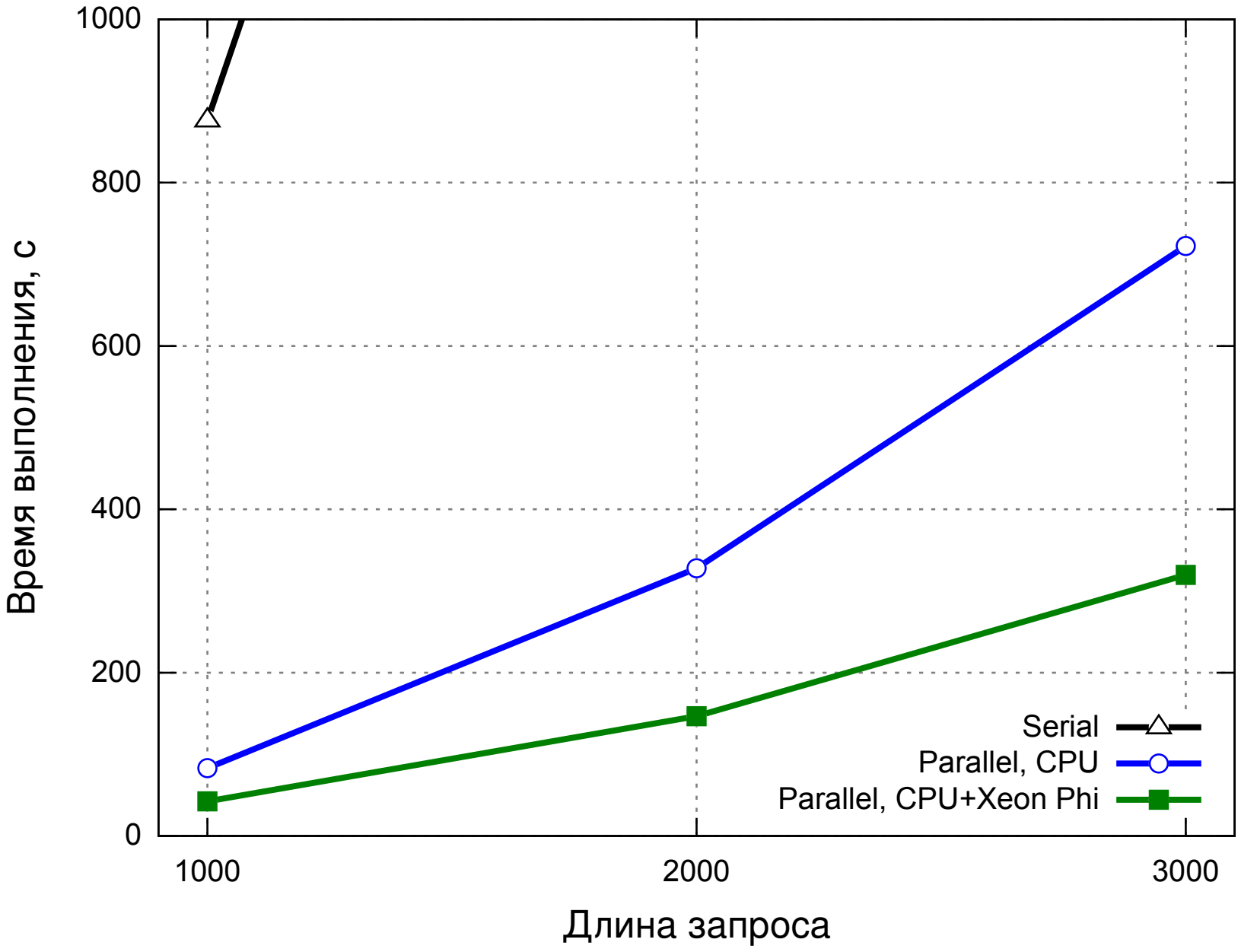
Временной ряд	Категория	Длина
Случайный (Pure Random)	синтетический	10^6
Случайные блуждания (Random Walk)	синтетический	10^8
ЭКГ (ECG)*	реальный	$2 \cdot 10^7$

Пороговое значение $\mathcal{E} = 2 + D_{min}$, где D_{min} — предварительно вычисленное расстояние до самой похожей подпоследовательности.

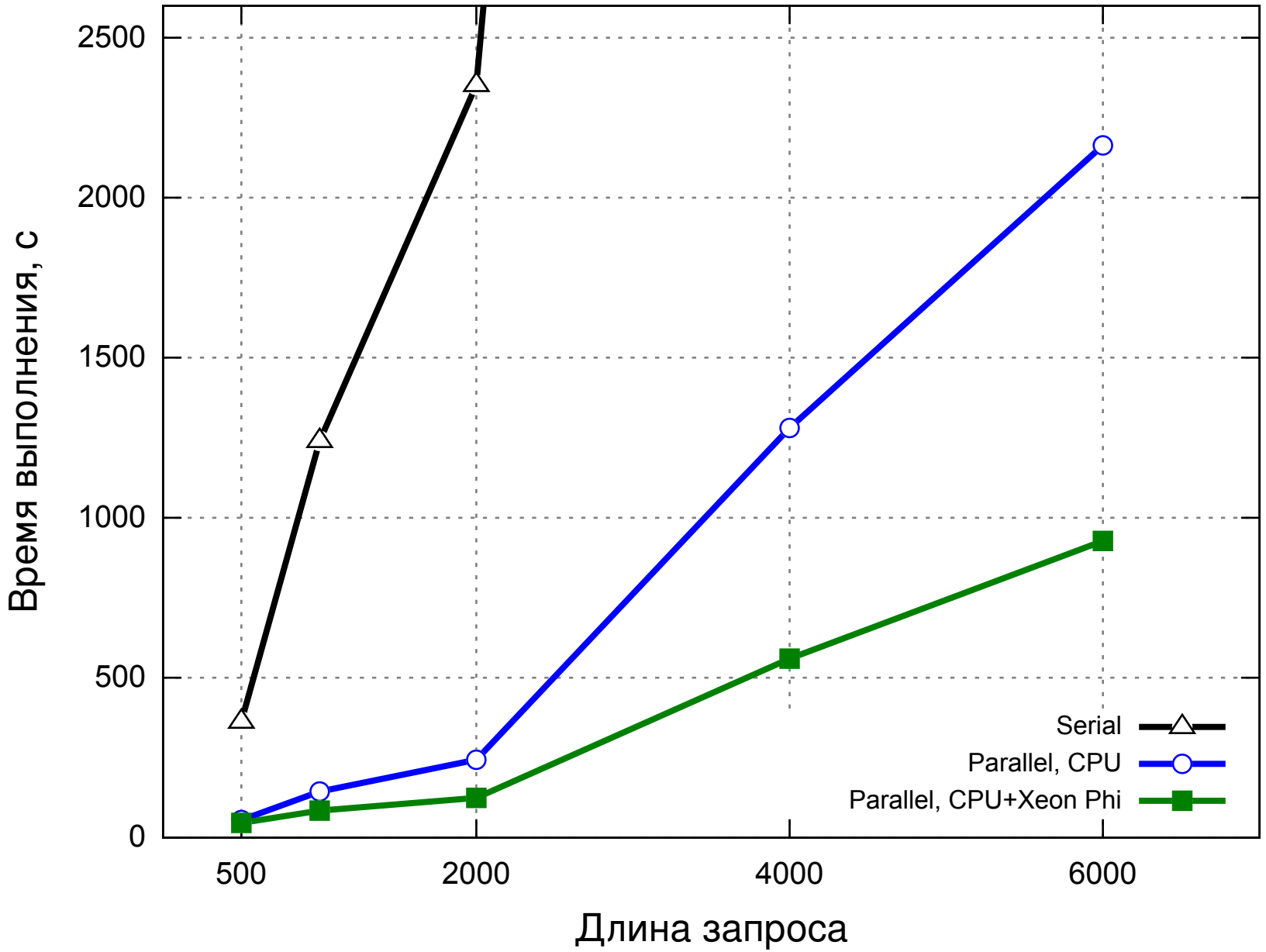
Мощность результирующего множества $K = 10000$.

* Rakthanmanon T., et al. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping // ACM SIGKDD, 2012. P. 262–270.

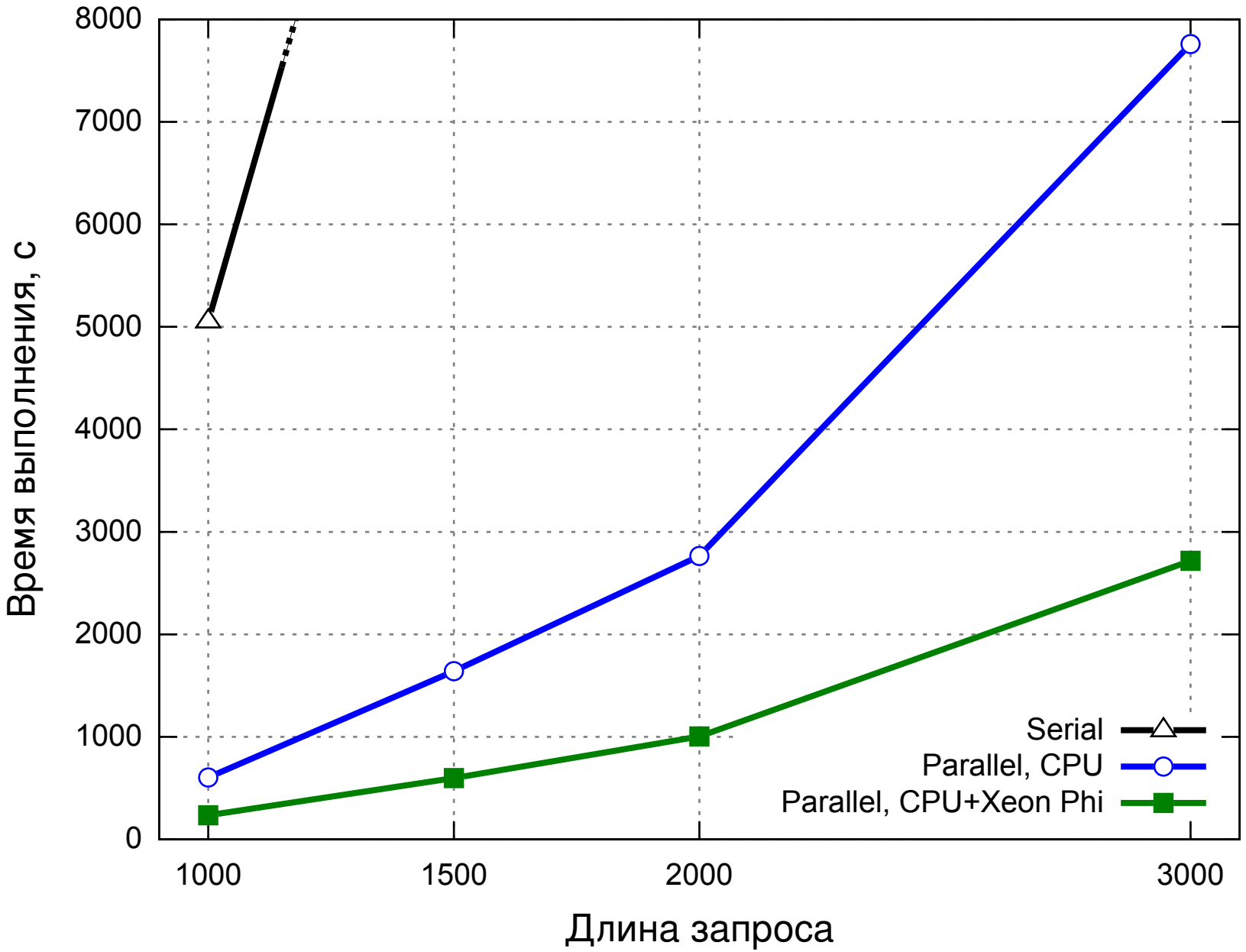
Производительность – PURE RANDOM



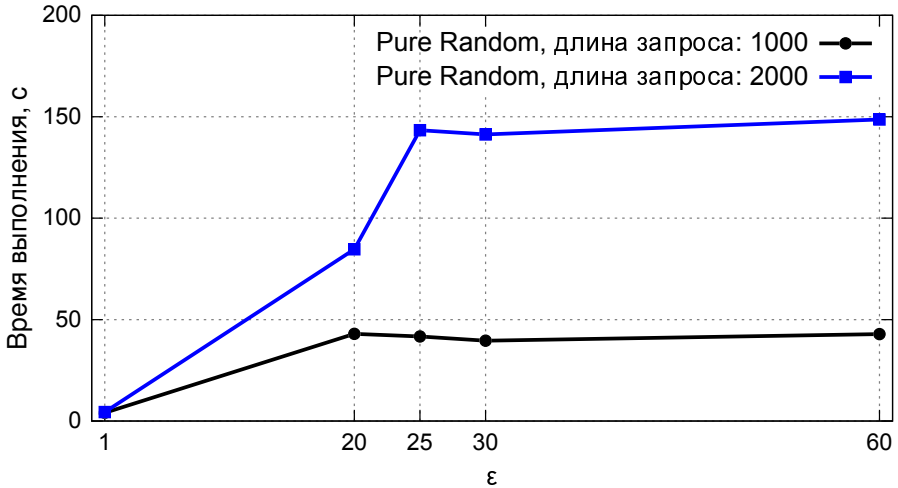
Производительность – RANDOM WALK



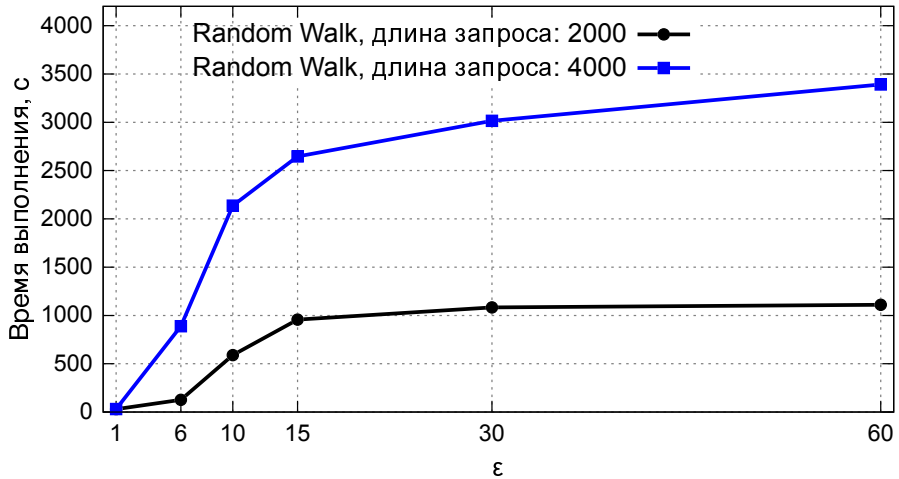
Производительность – ECG



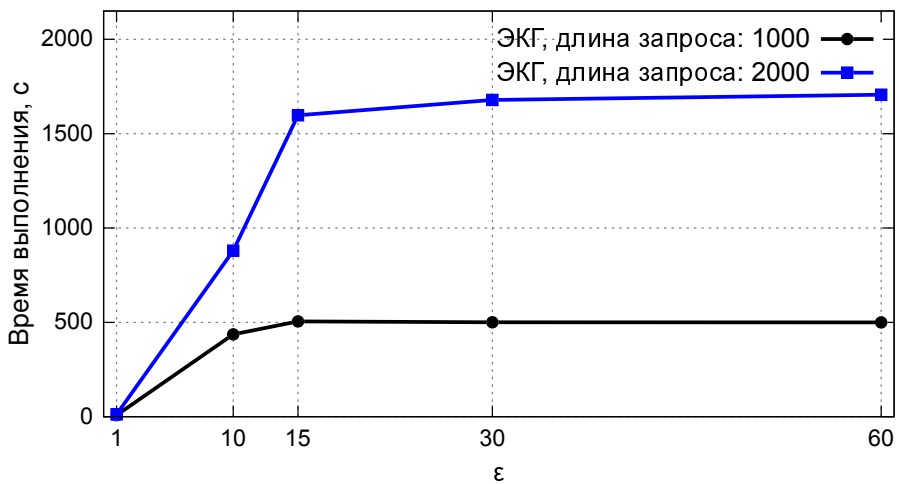
Влияние порогового значения $\mathcal{E}$



(a) PURE RANDOM



(b) RANDOM WALK

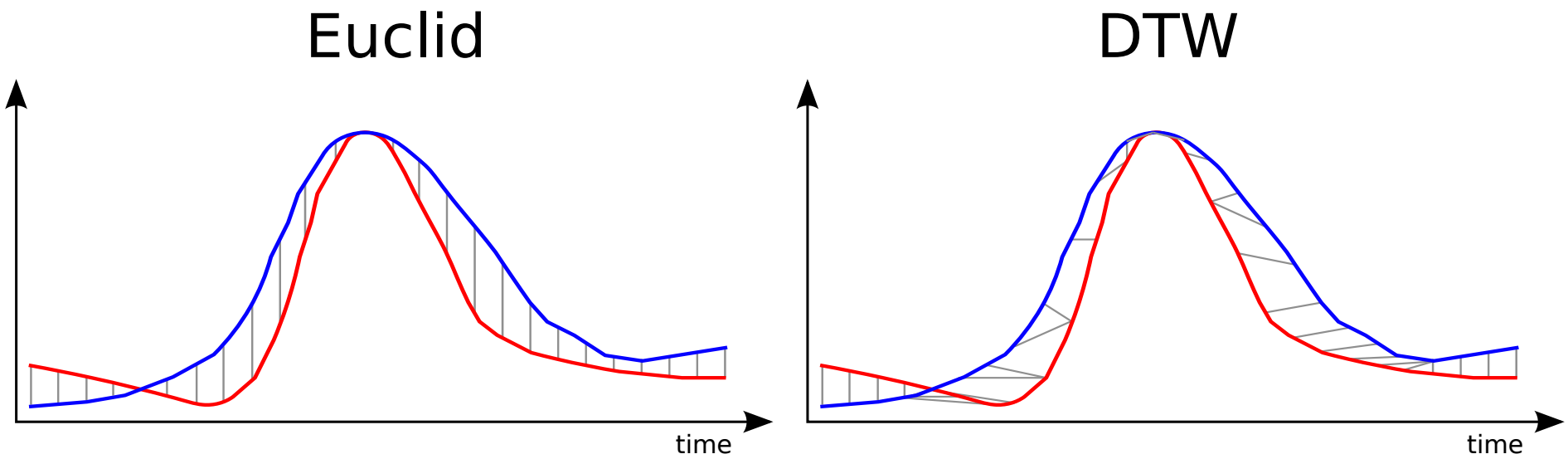


(c) ECG

Заключение

- Разработан параллельный алгоритм поиска локально похожих подпоследовательностей временного ряда для сопроцессоров Intel Xeon Phi.
- Эксперименты, проведенные на синтетических и реальных данных, показали эффективность разработанного алгоритма.
- Дальнейшие исследования:
  - исследование динамического изменения порогового значение (расстояние до самой похожей подпоследовательности + некоторое число, являющееся параметром алгоритма);
  - использование нескольких сопроцессоров Intel Xeon Phi;
  - расширение разработанного алгоритма для кластерной системы, каждый вычислительный узел которой оснащен сопроцессором Intel Xeon Phi.

Динамическая трансформация шкалы времени

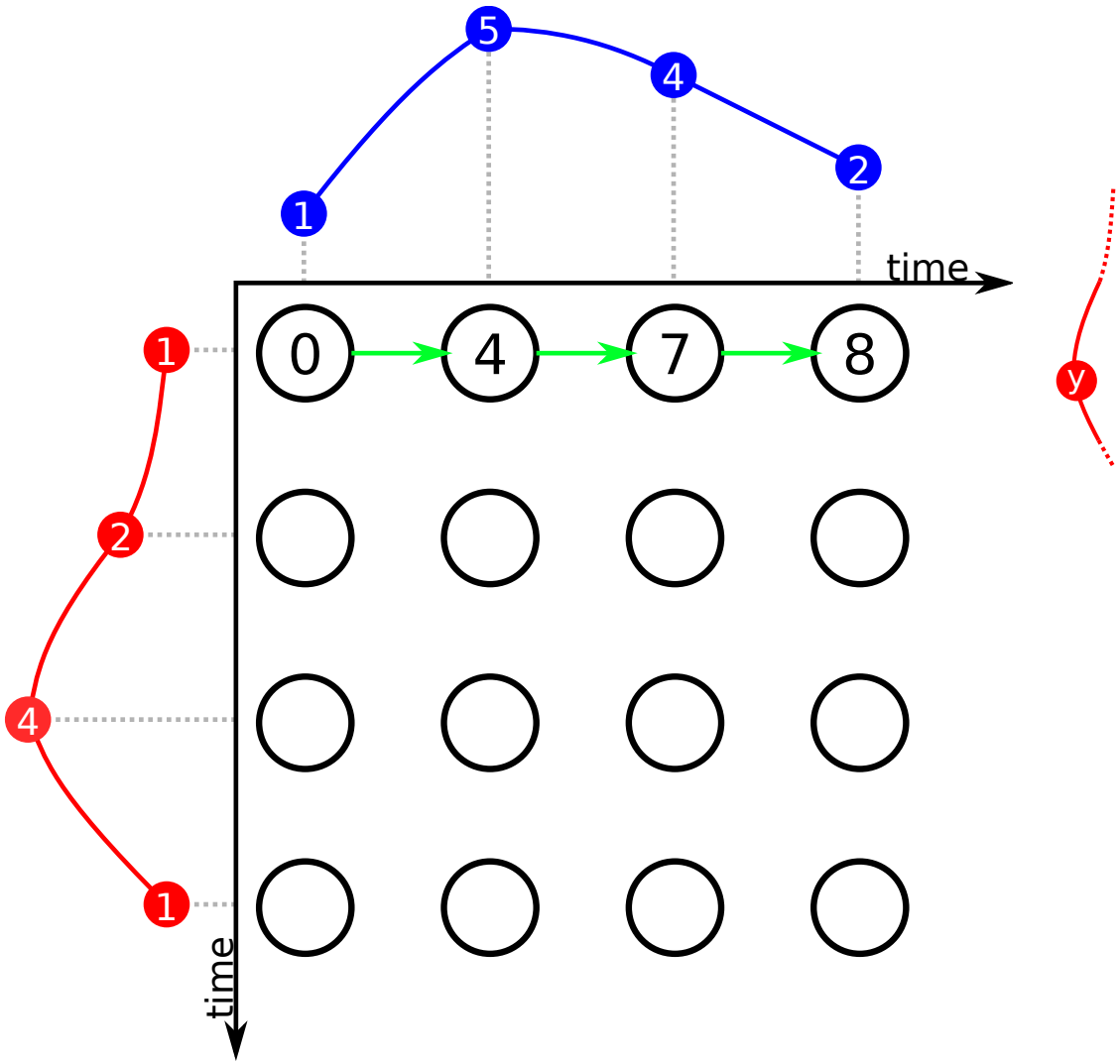


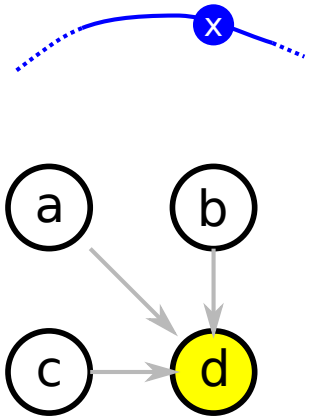
$$DTW(X,Y) = d(N,N),$$

$$d(i,j) = |x_i - y_j| + min \begin{cases} d(i-1,j) \\ d(i,j-1) \\ d(i-1,j-1), \end{cases}$$

$$d(0,0) = 0; d(i,0) = d(0,j) = \infty; i = 1,2,\ldots,N; j = 1,2,\ldots,N.$$

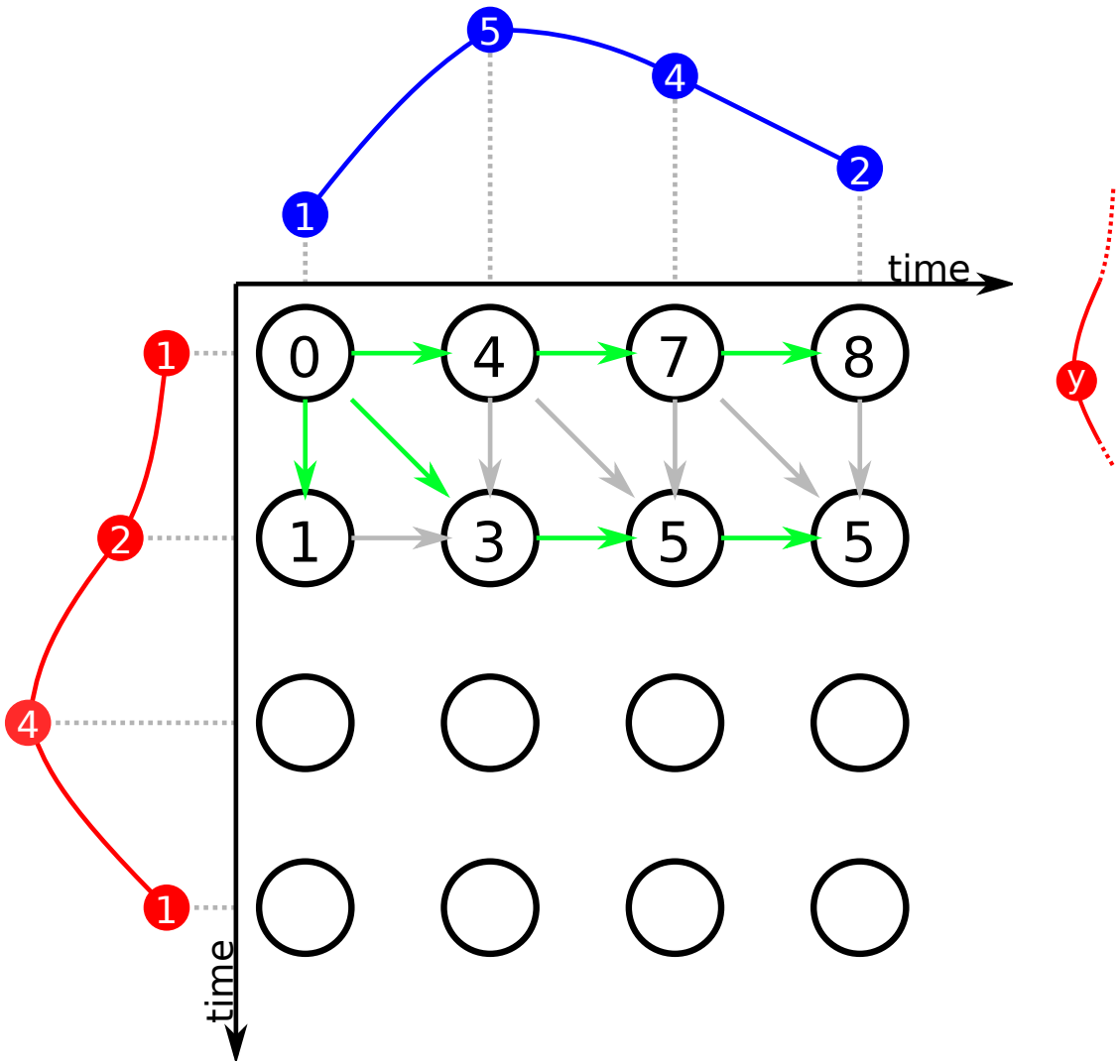
Динамическая трансформация шкалы времени

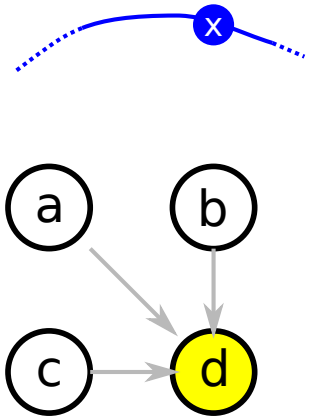




d = cost + min(a, b, c)
cost = |x - y|

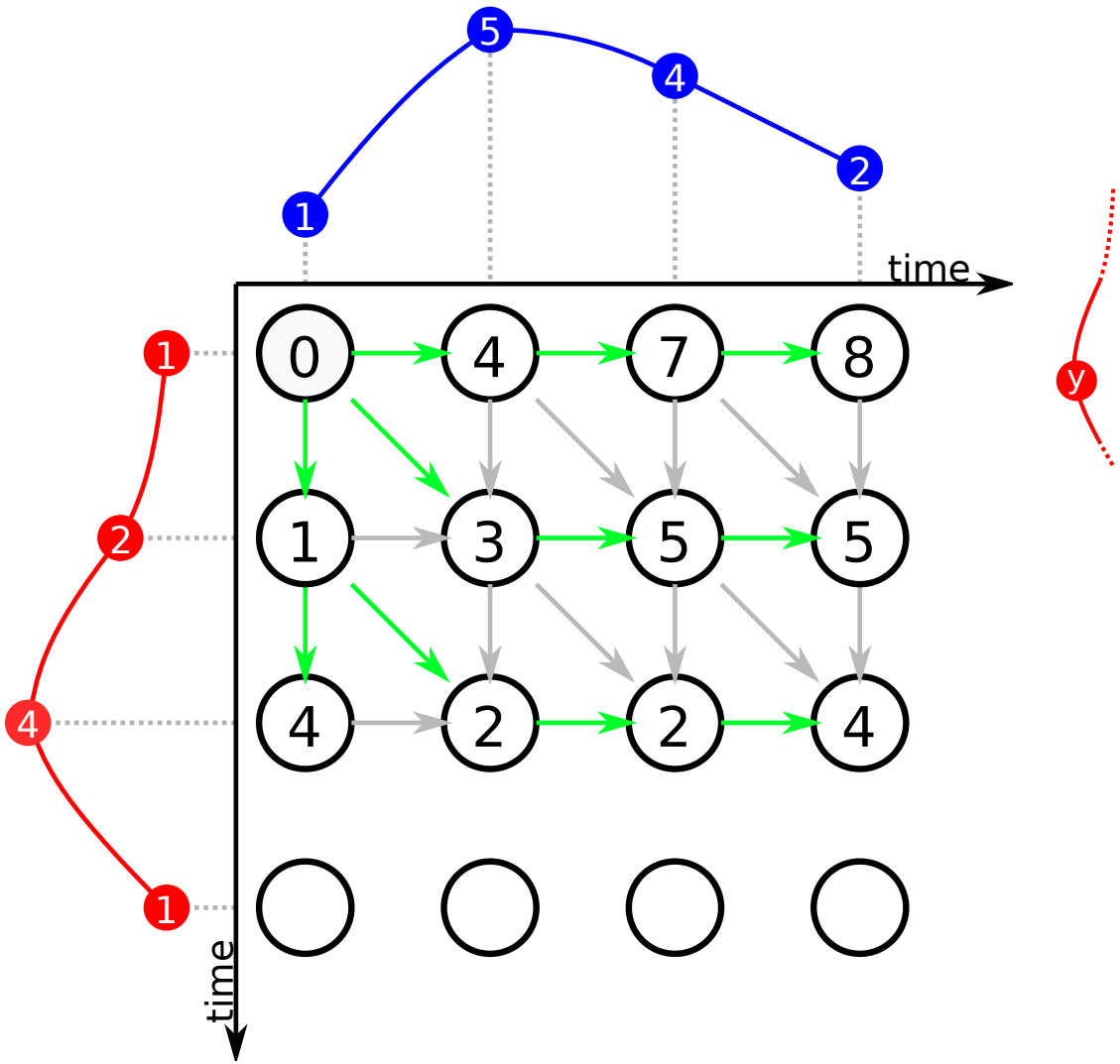
Динамическая трансформация шкалы времени

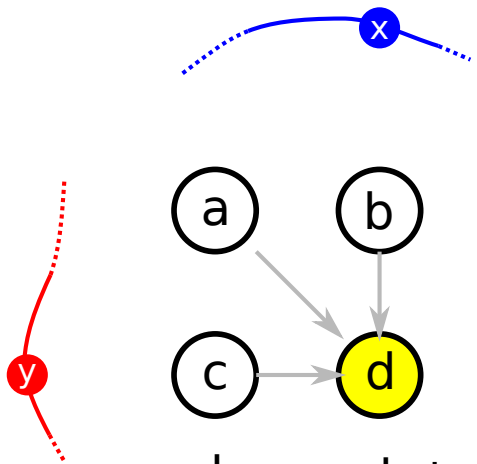




d = cost + min(a, b, c)
cost = |x - y|

Динамическая трансформация шкалы времени

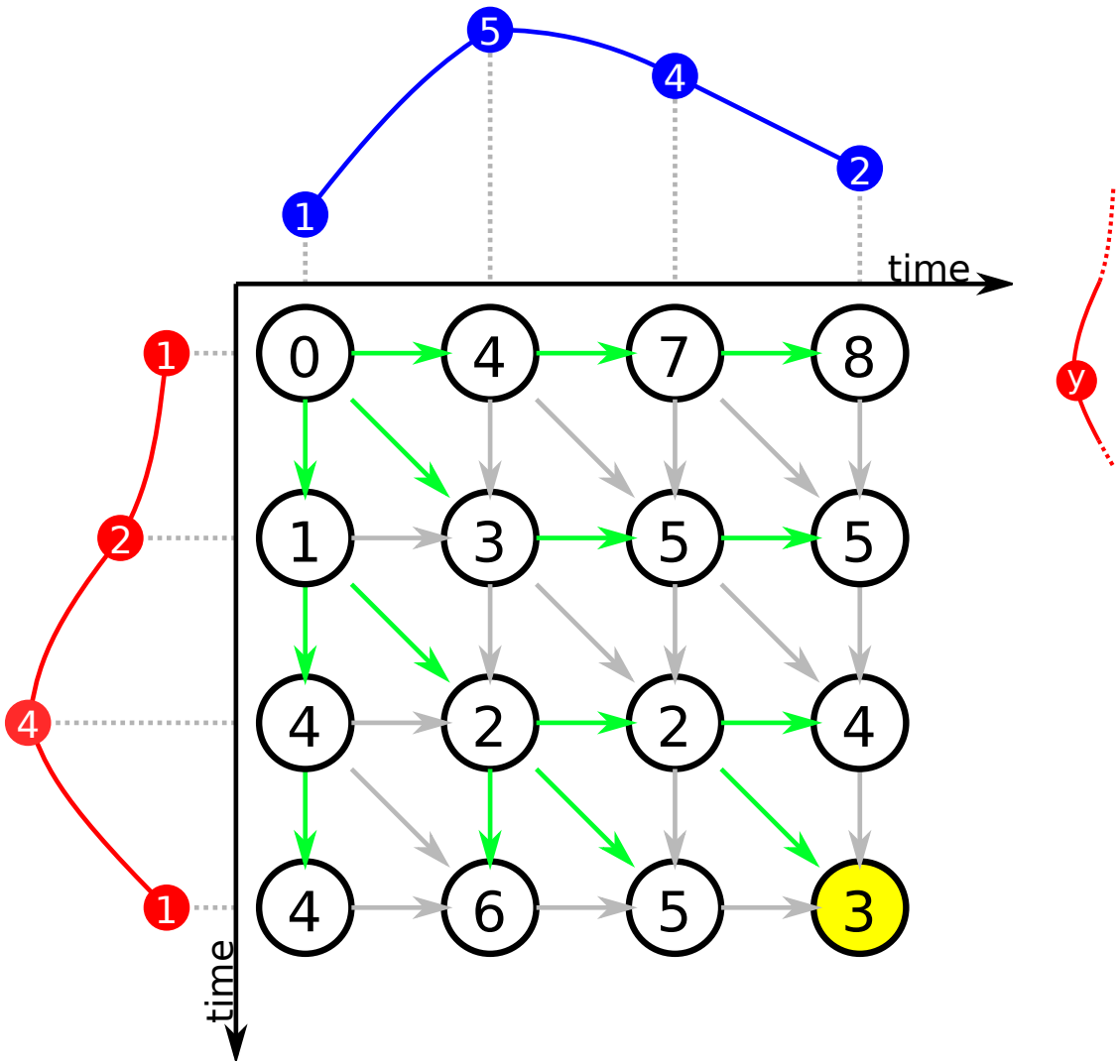


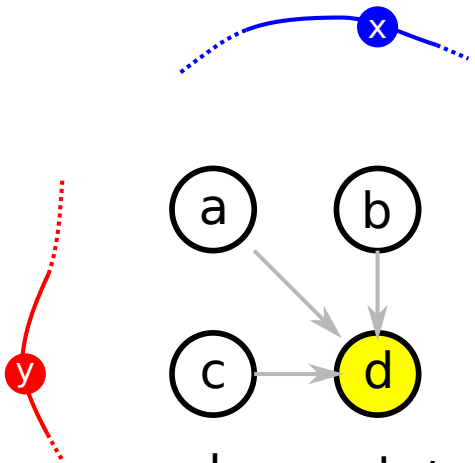


$d = cost + \min(a, b, c)$

$cost = |x - y|$

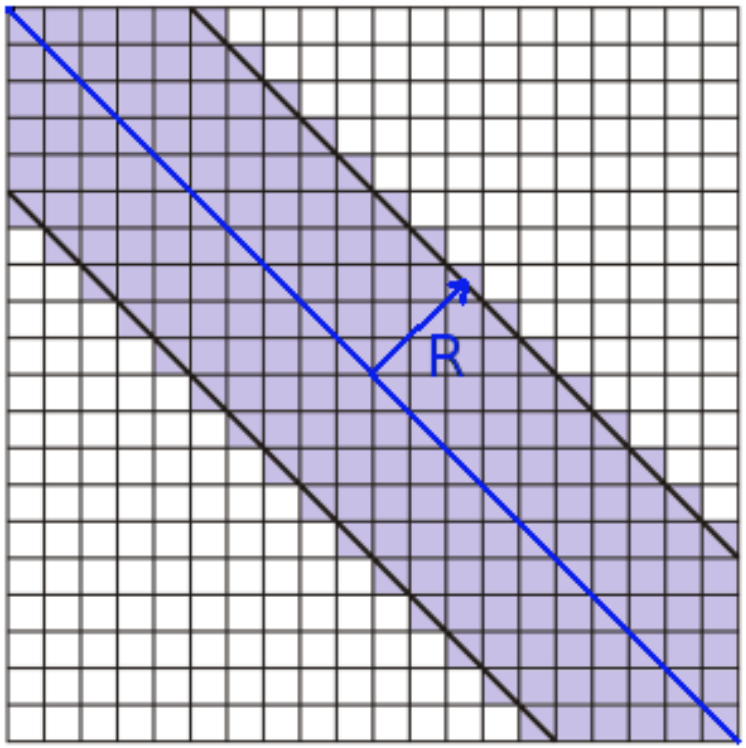
Динамическая трансформация шкалы времени



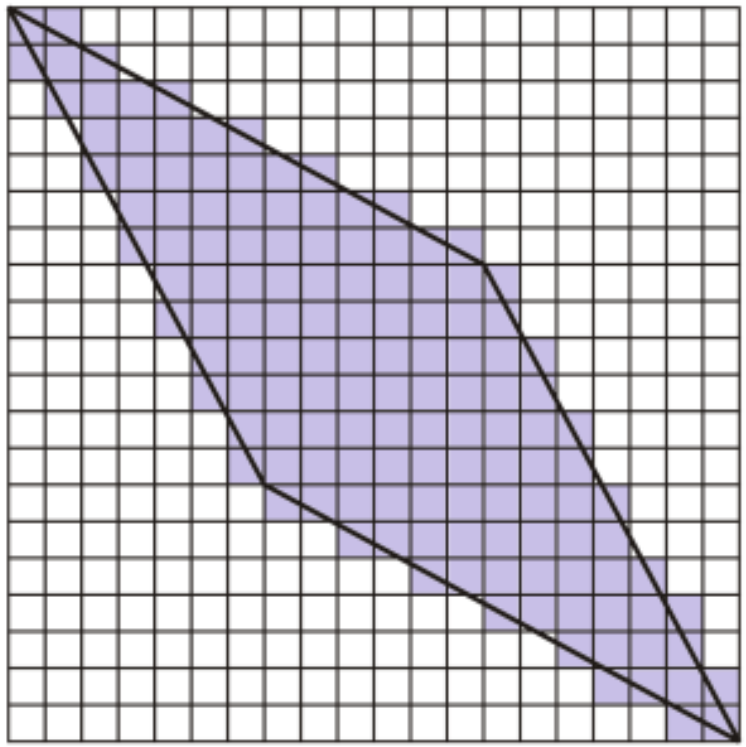


d = cost + min(a, b, c)
cost = |x - y|

Ограничение DTW



Sakoe-Chiba band



Itakura parallelogram

Оценки DTW

- $LB_{Kim} = \sqrt{(t_0 - q_0)^2 + (t_{n-1} - q_{n-1})^2}$
  Сложность: $O(1)$.
- LB_{Keogh}
  Для запроса Q строятся последовательности U и L.
  $u_i = max(q_{i-R}, q_{i+R})$, $l_i = min(q_{i-R}, q_{i+R})$,

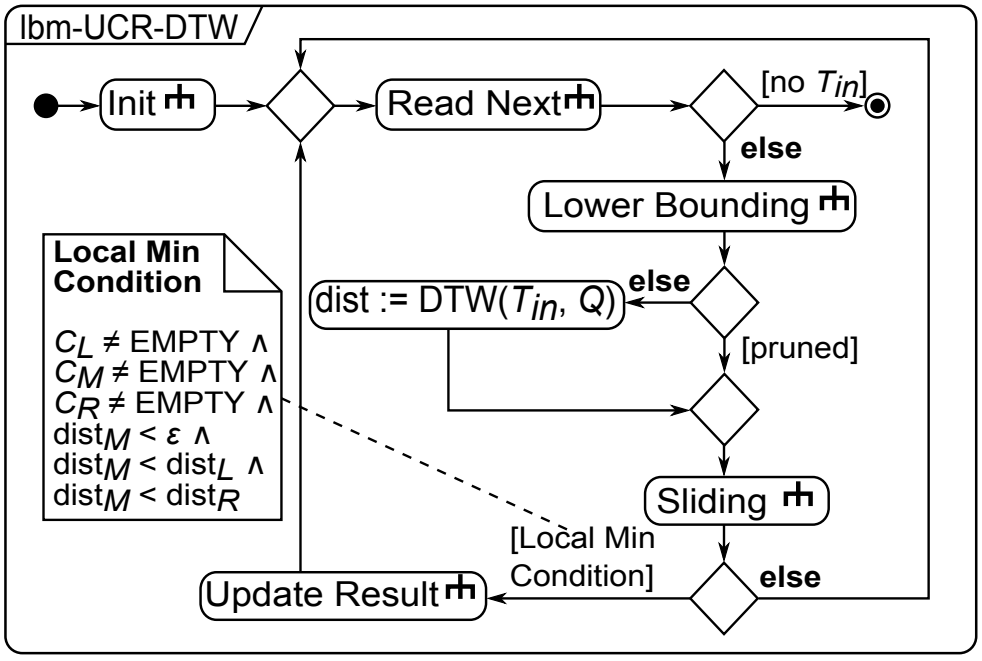
$$LB_{Keogh}(Q, C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - u_i)^2 & \text{if } c_i > u_i \\ (c_i - l_i)^2 & \text{if } c_i < l_i \\ 0 & \text{otherwise} \end{cases}}$$

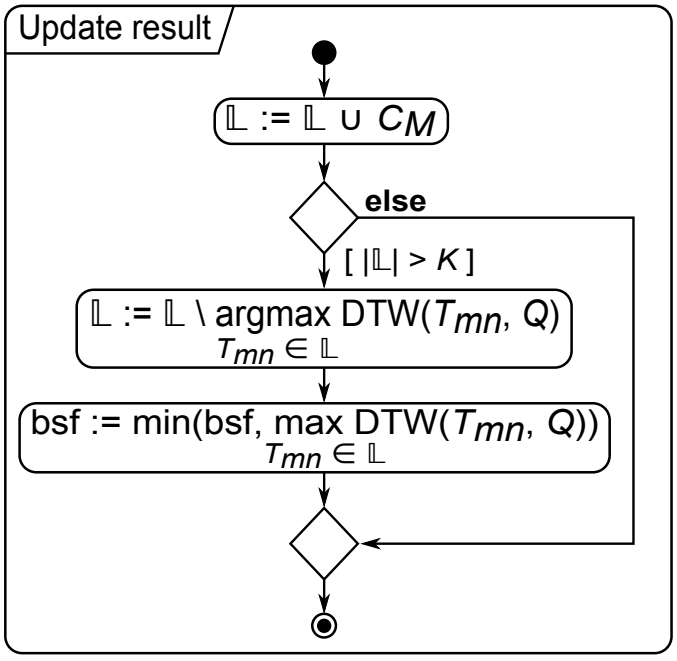
  Сложность: $O(n)$.
- $LB_{KeoghEC}$
  Для подпоследовательности C строятся последовательности U и L.
  $u_i = max(c_{i-R}, c_{i+R})$, $l_i = min(c_{i-R}, c_{i+R})$,

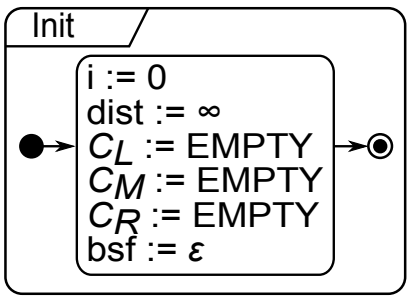
$$LB_{Keogh}(Q, C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - u_i)^2 & \text{if } q_i > u_i \\ (q_i - l_i)^2 & \text{if } q_i < l_i \\ 0 & \text{otherwise} \end{cases}}$$

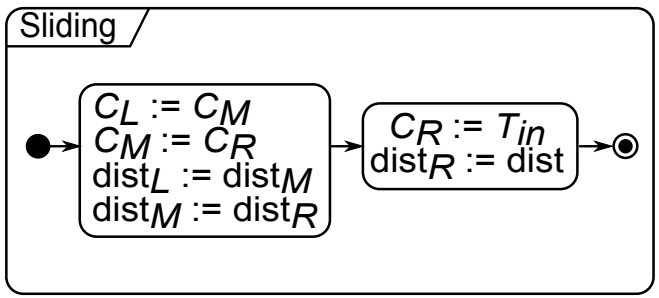
  Сложность: $O(n)$.

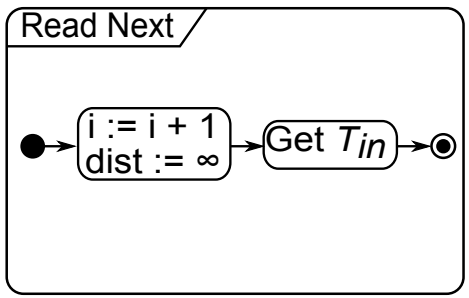
Последовательный алгоритм

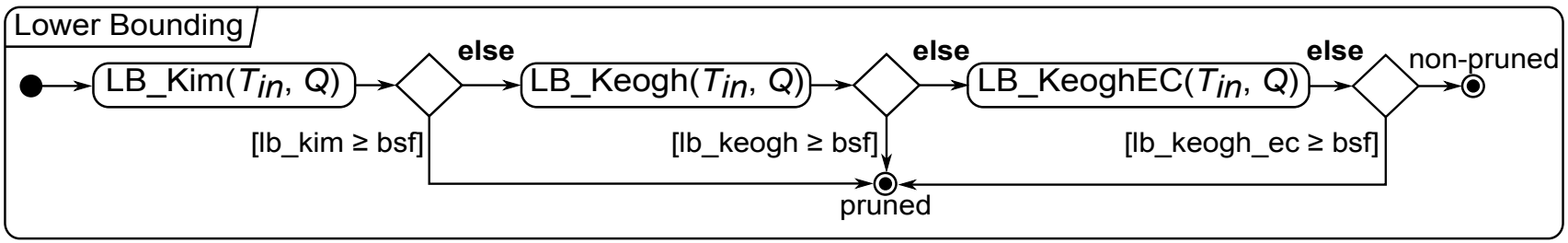




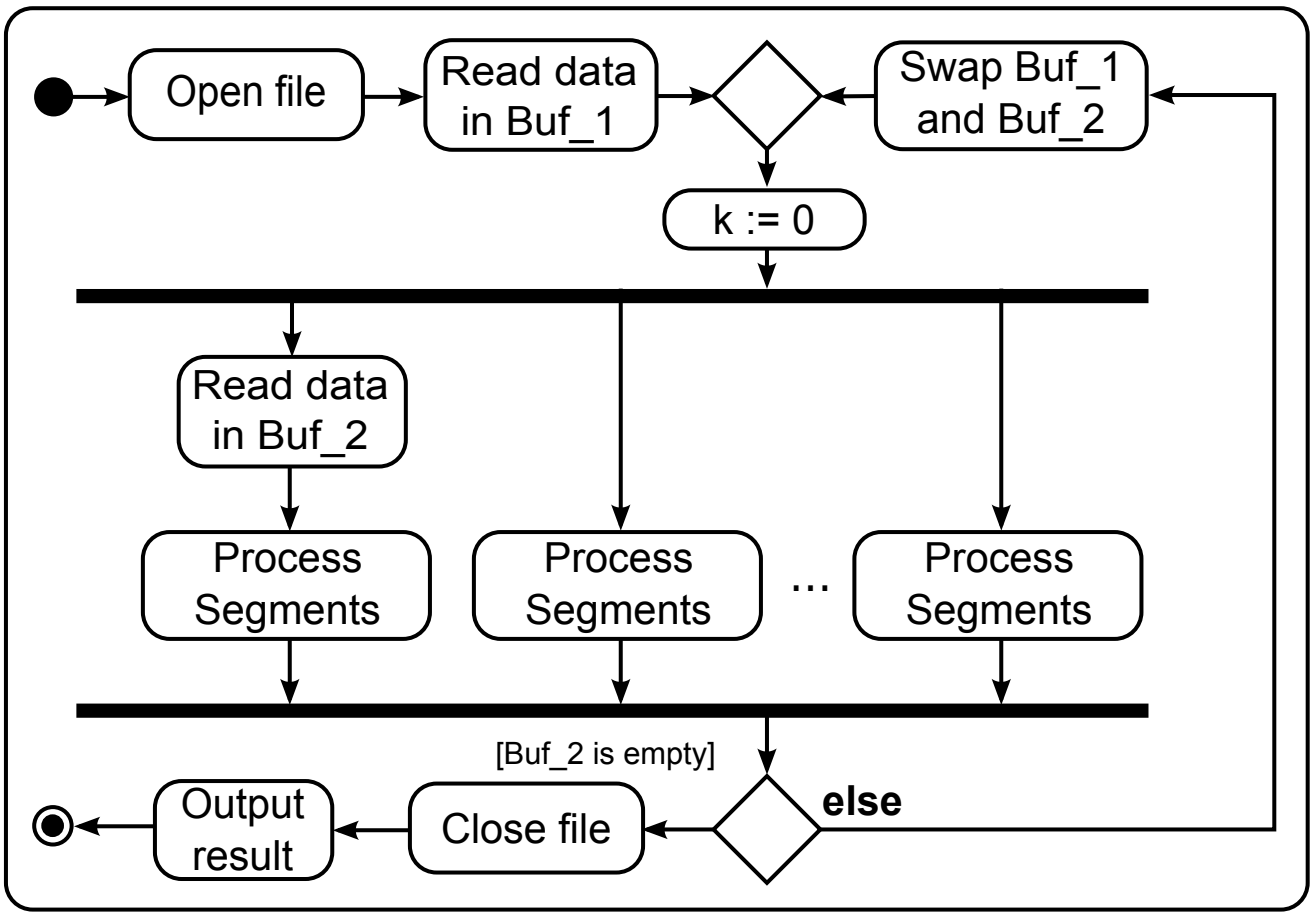


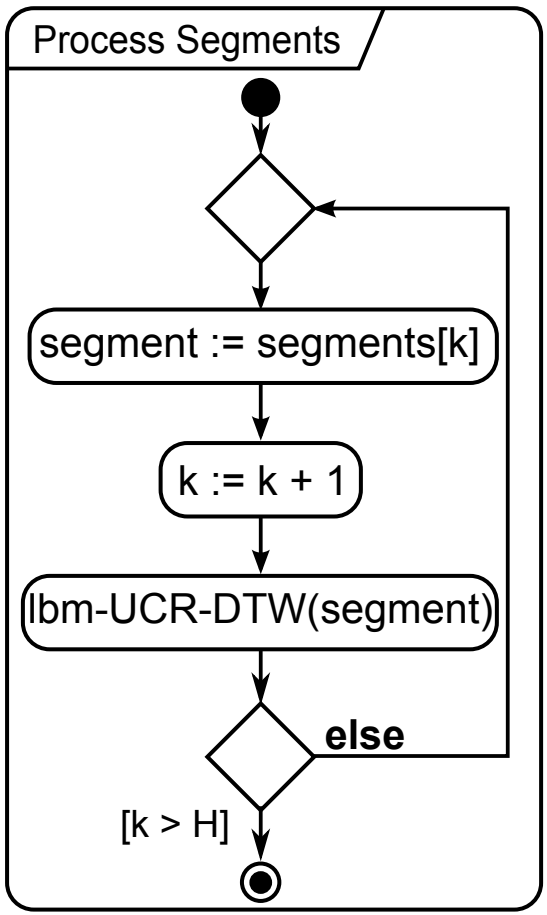




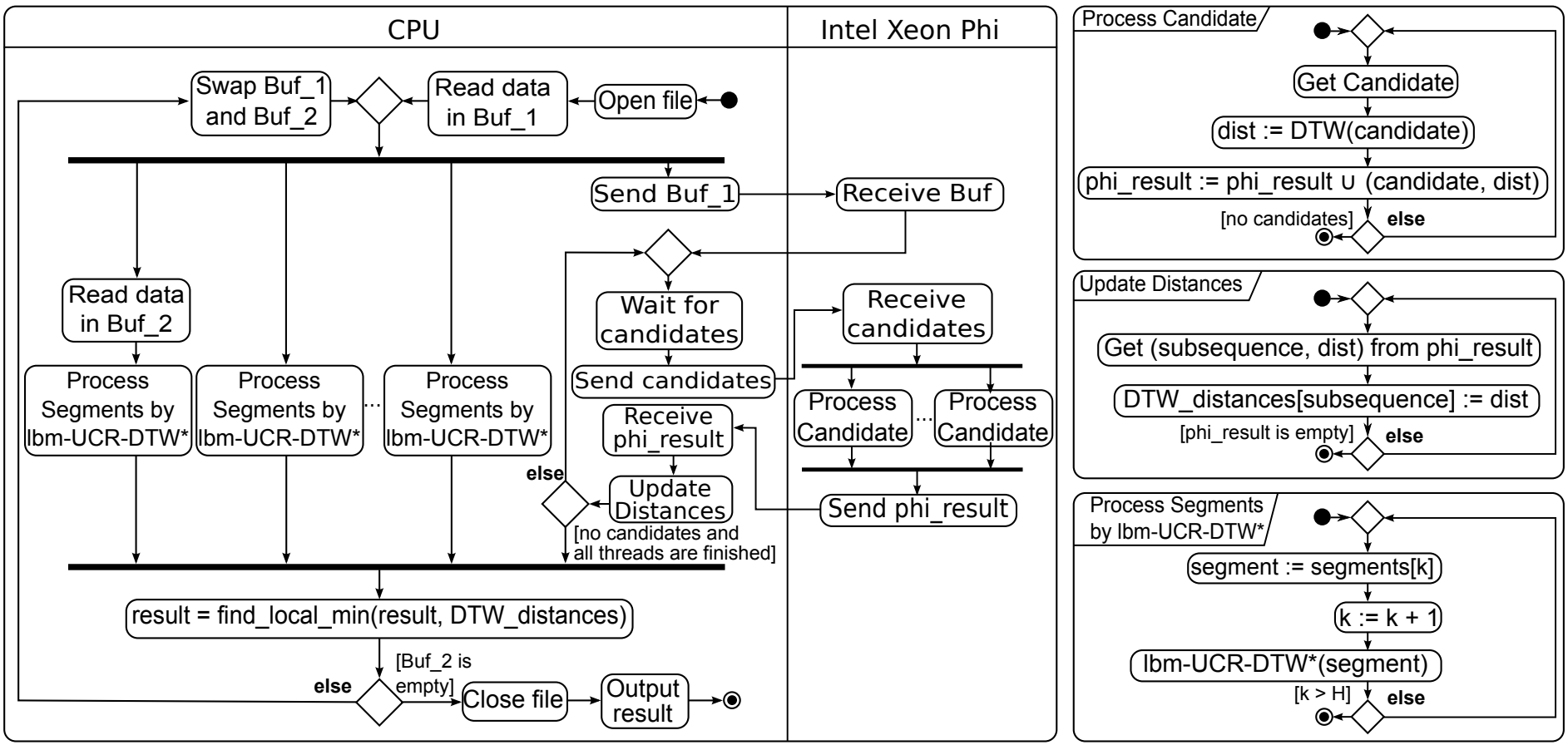


Параллельный алгоритм для процессора

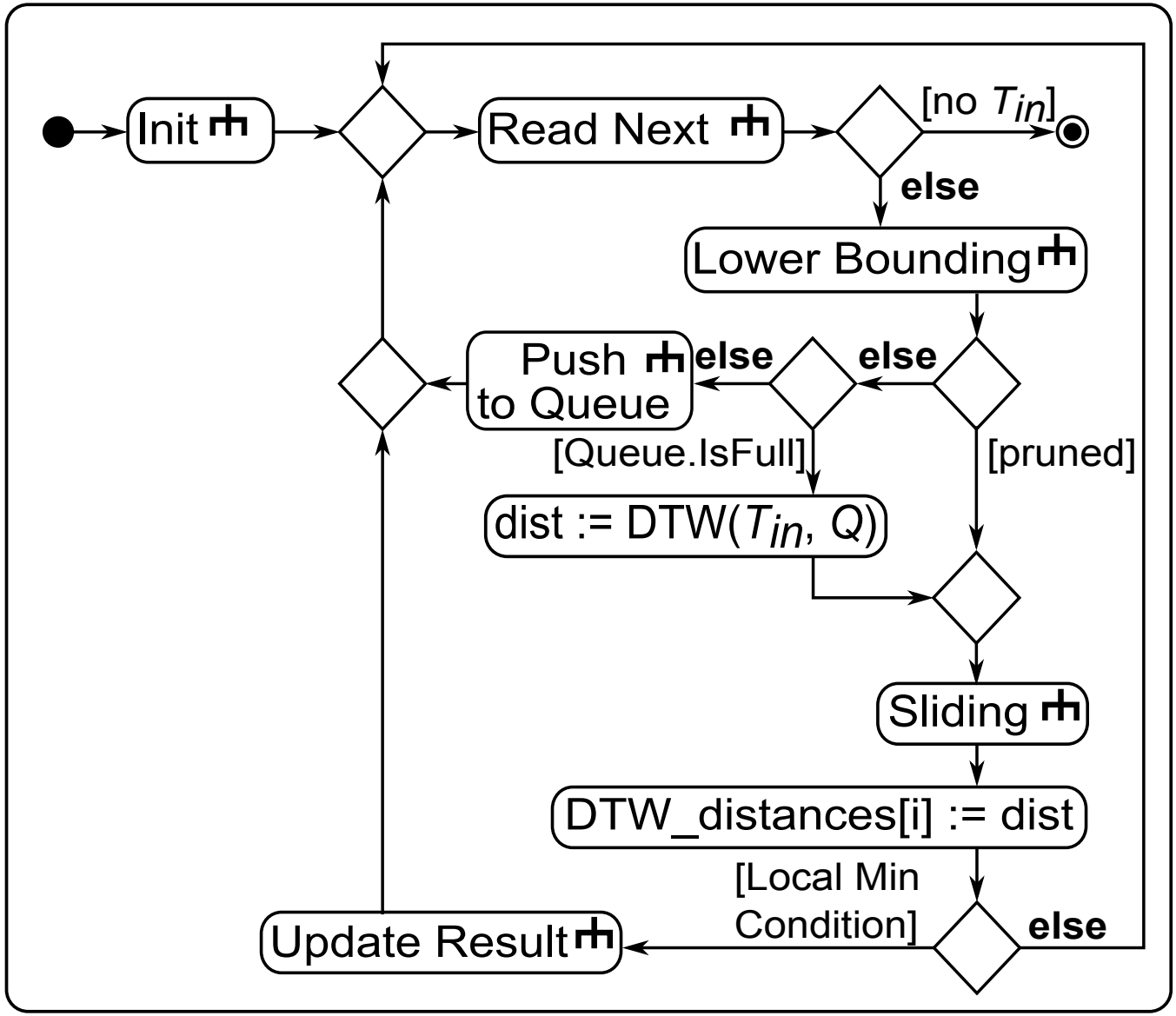


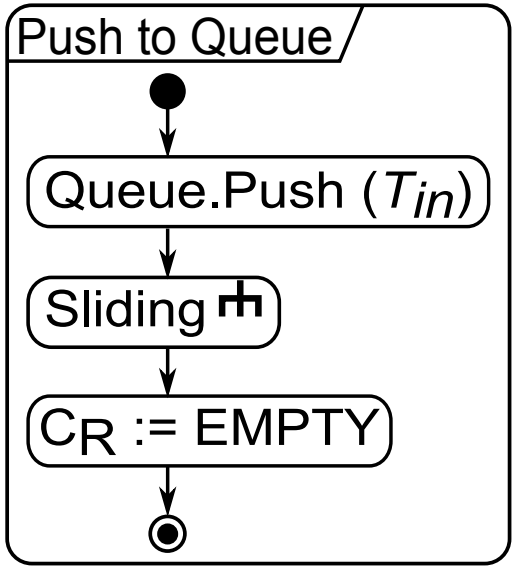


Параллельный алгоритм для сопроцессора



UCR-DTW-lbm*


Init
Read Next
[no T_{in}]
else
Lower Bounding
Push to Queue
else
else
[Queue.IsFull]
[pruned]
dist := DTW(T_{in}, Q)
Sliding
DTW_distances[i] := dist
[Local Min Condition]
else
Update Result


Push to Queue
Queue.Push (T_{in})
Sliding
C_R := EMPTY