Scientific seminar
SUSU Big Data and Machine Learning Lab

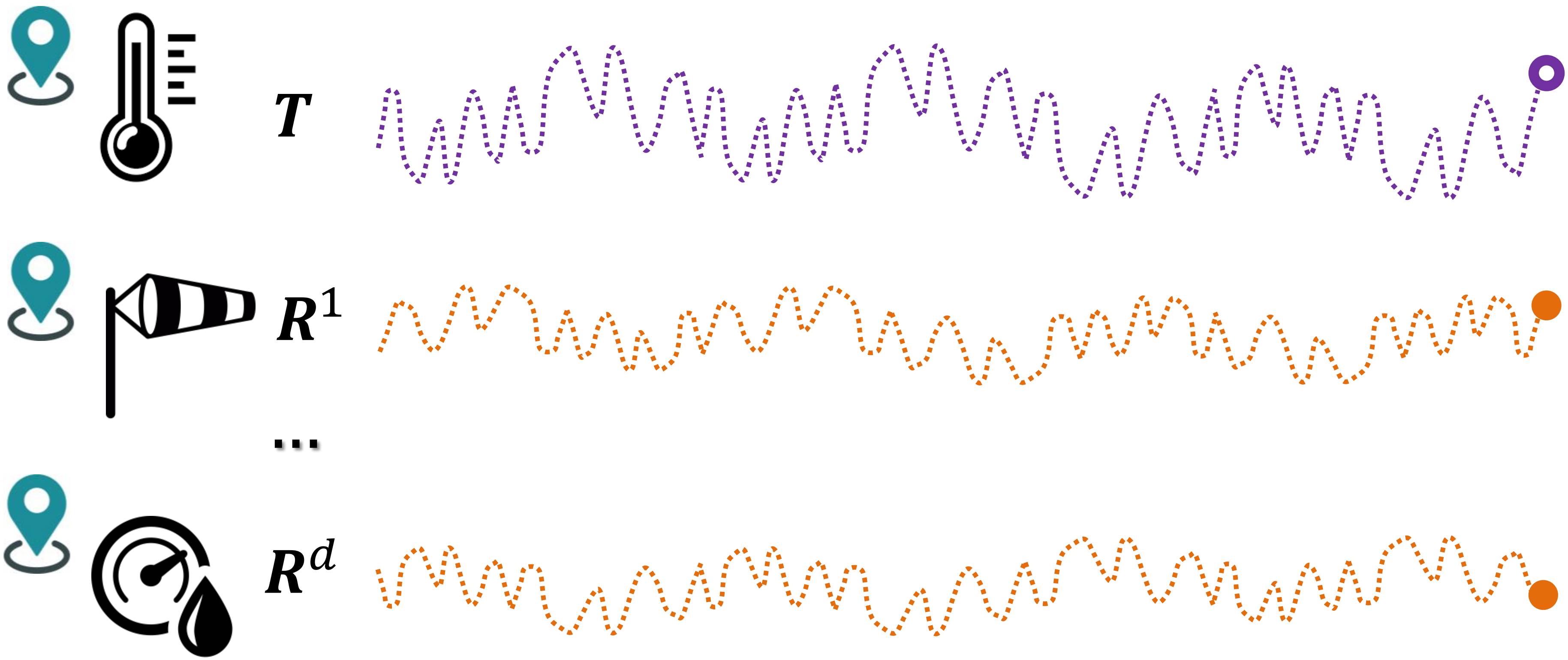
Parallel algorithm for imputation missing values of time series

Mikhail Zymbler[1], Andrey Poluyanov[2], Yana Kraeva[1]

[1] South Ural State University, Chelyabinsk, Russia,
[2] Sobolev Institute of Mathematics SB RAS, Omsk, Russia

Chelyabinsk—2022

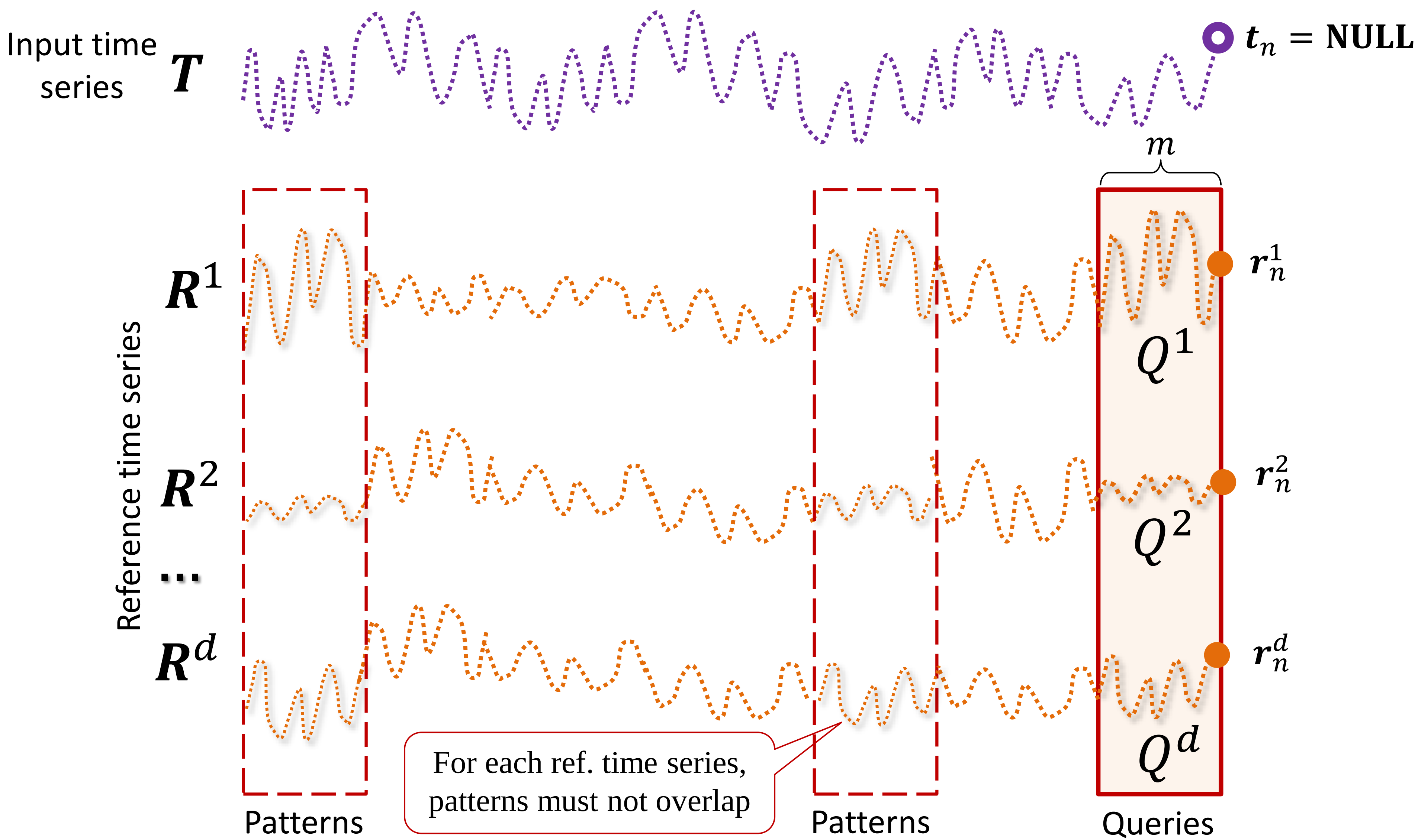
Imputation through reference time series



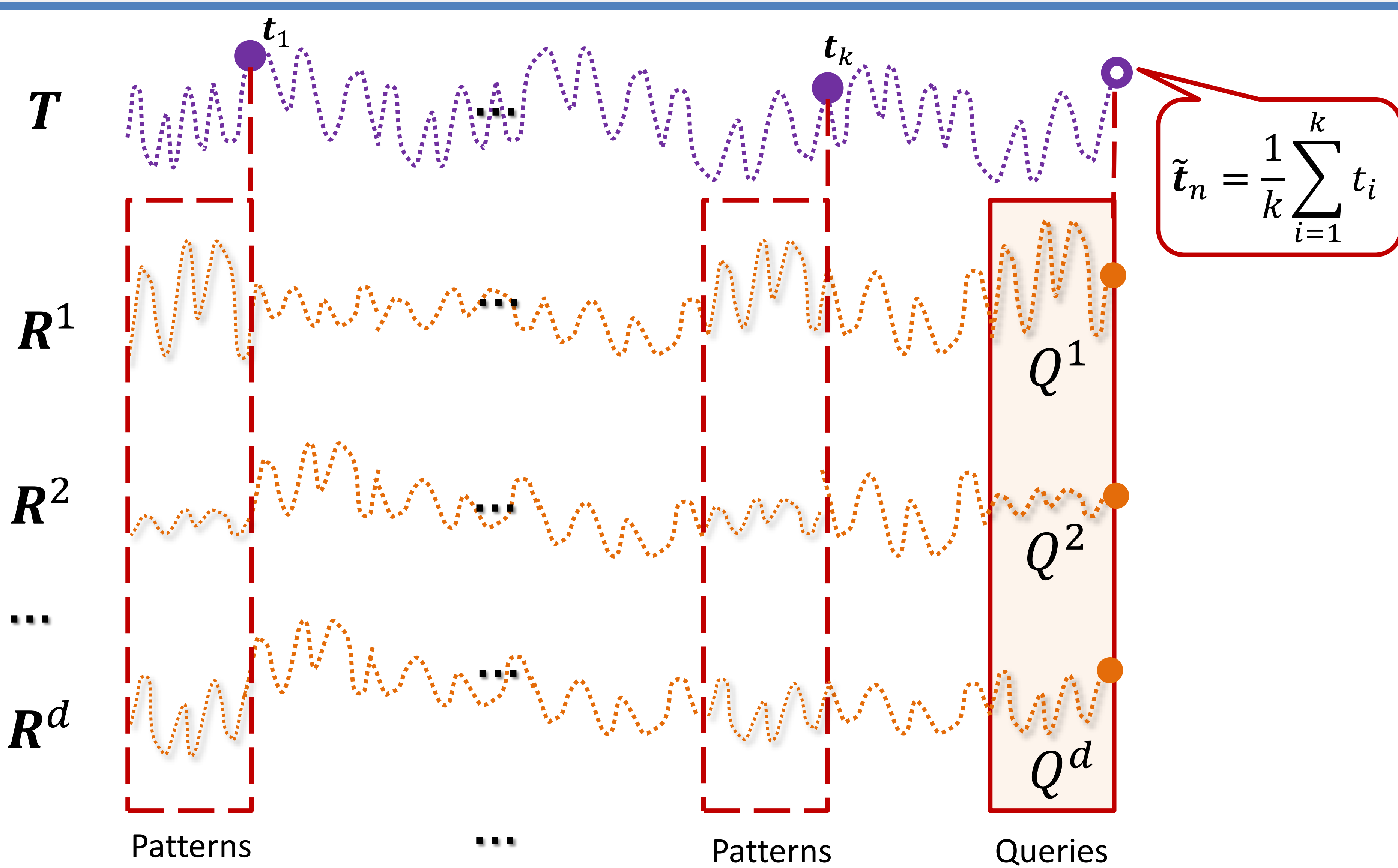
Heuristics

the time series undergo imputation and the reference time series behave similarly in the same subsequences

Imputation: Find k patterns



Imputation: Reconstruction of missing value



ParaDI: Parallel DTW-based Imputation*

- DTW (Dynamic Time Warping) distance measure
  – DTW is used in the pattern search, since it is the best to measure similarity of subsequences by their shapes
- Pruning through the lower bounds (LB)
  – LBs allow to discard clearly dissimilar subsequences without calculation of DTW
- Fork-join parallelism
  – OpenMP for x86, OpenACC for GPU

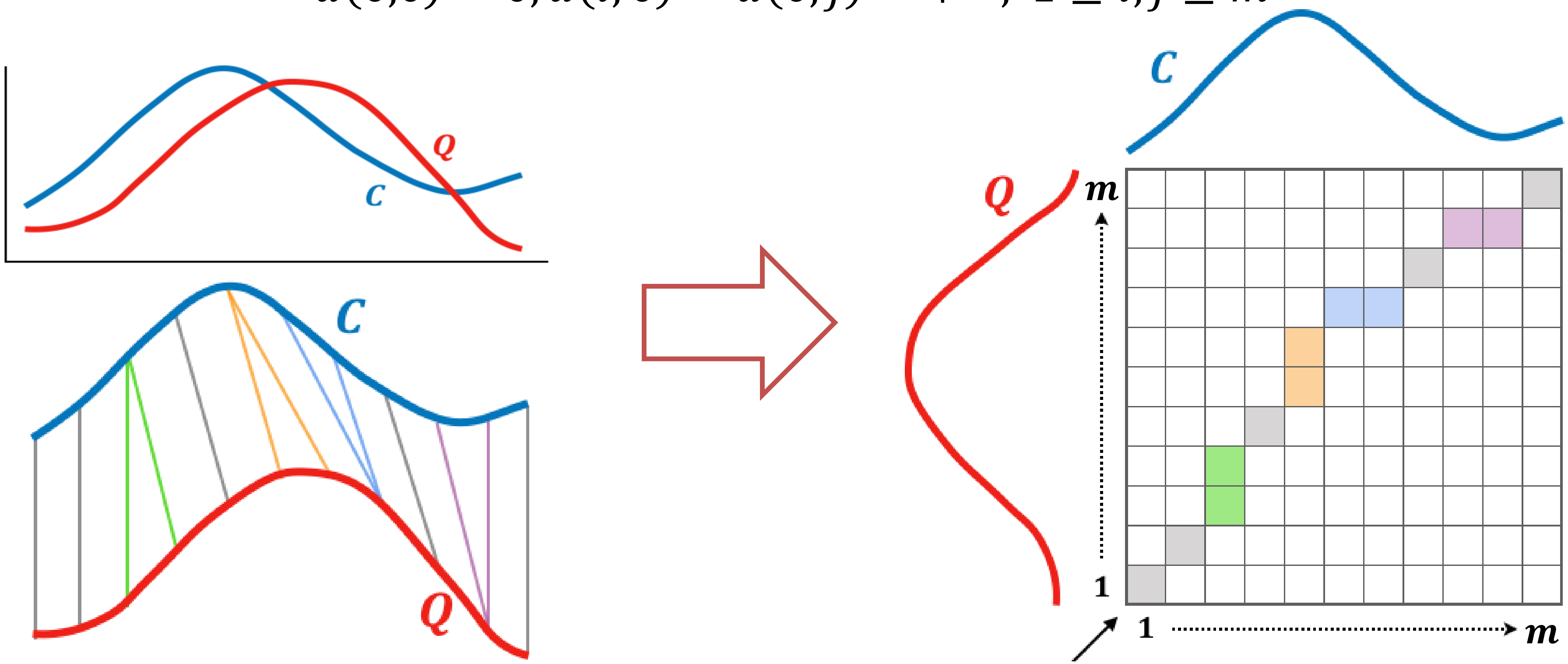
* Zymbler M.L., Poluyanov A.N., Kraeva Ya.A. Parallel Algorithm for Real-time Sensor Data Recovery for a Many-core Processor. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 3. P. 68–89. (in Russian) DOI: 10.14529/cmse220305.

DTW (Dynamic Time Warping) distance measure*

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

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

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



* Berndt D.J., Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series. KDD & AAAI Workshop 1994. TR-WS-94-03. P. 359-370.

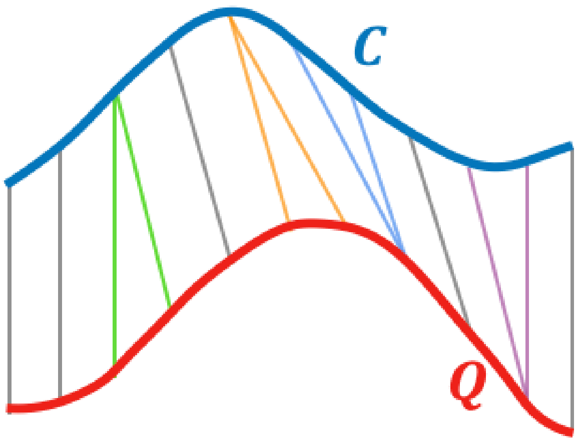
DTW: complexity *vs.* accuracy

Complexity
$\boldsymbol{O(m^2)}$

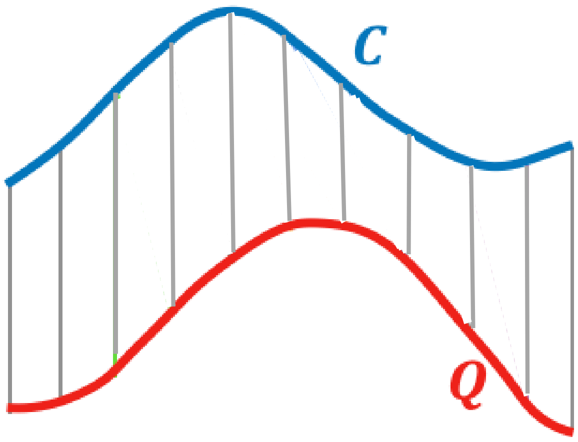
$$\mathbf{DTW}(Q, C) = d(m, m)$$

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

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



DTW is better to measure the similarity by shape than the Euclidean distance



DTW acceleration: Sakoe–Chiba band*

Complexity
$\boldsymbol{O(mr)}$

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

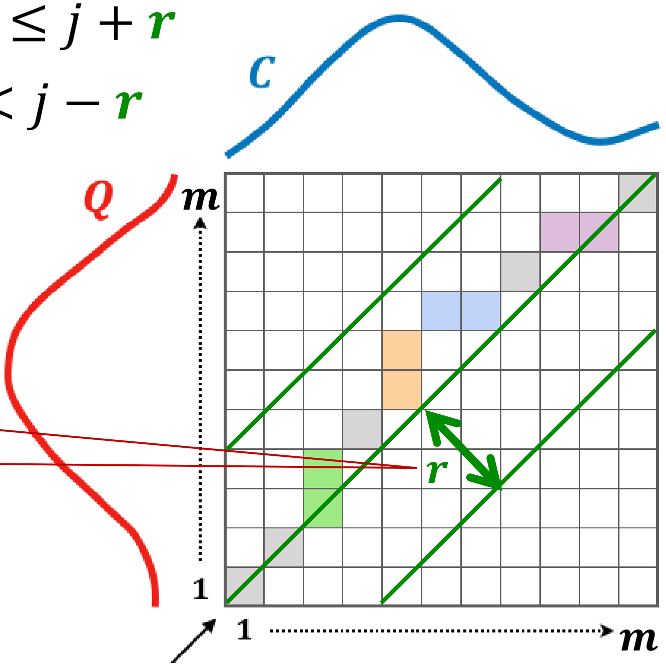
$$d(i,j) = (q_i - c_i)^2 + \min \begin{cases} d(i-1, j) \\ d(i, j-1) \\ d(i-1, j-1) \end{cases}$$

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

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

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

Complexity *vs.* accuracy trade-off



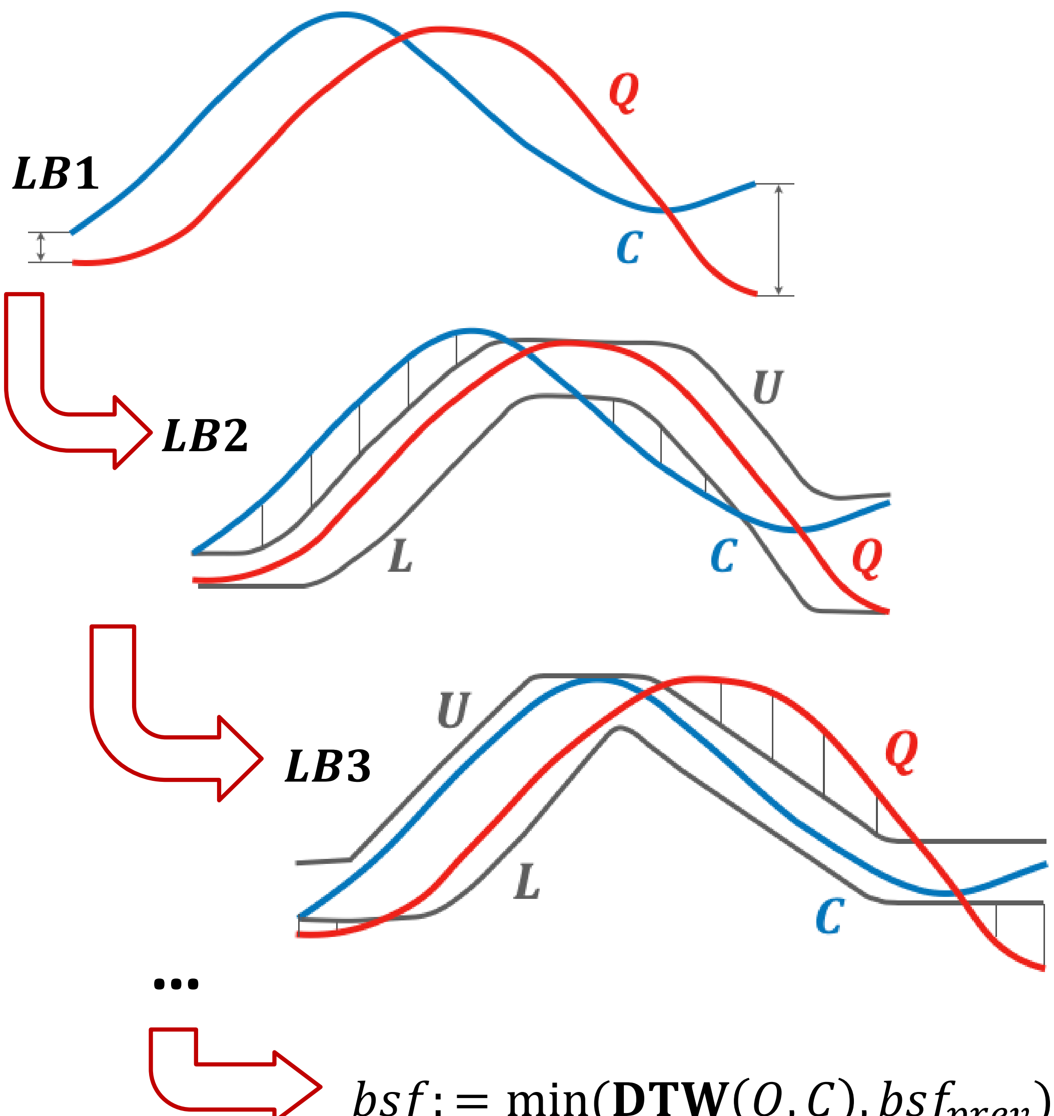
* Sakoe H., Chiba S. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Trans. on Acoustics, Speech, and Signal Processing. 1978. Vol. 26. P. 43-49.

DTW acceleration: Lower bounding*

- Lower bound function (LB)
  - $\mathrm{LB}: \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+$, complexity is lower than $O(m^2)$
    $\forall\, R[i{:}m], Q{:}\ \mathrm{LB}(R[i{:}m], Q) \leq \mathrm{DTW}(R[i{:}m], Q)$
- Lower bounding pruning
  - Best-so-far minimum of DTW: bsf
  - if $\mathrm{LB}(R[i{:}m], Q) > bsf$, then $\mathrm{DTW}(R[i{:}m], Q) > bsf$,
    so $R[i{:}m]$ is clearly dissimilar to Q,
    and we don't need to calculate DTW
  - It works only if $R[i{:}m]$ and Q are z-normalized
  - LBs can be applied in a cascade

* Rakthanmanon T., *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.

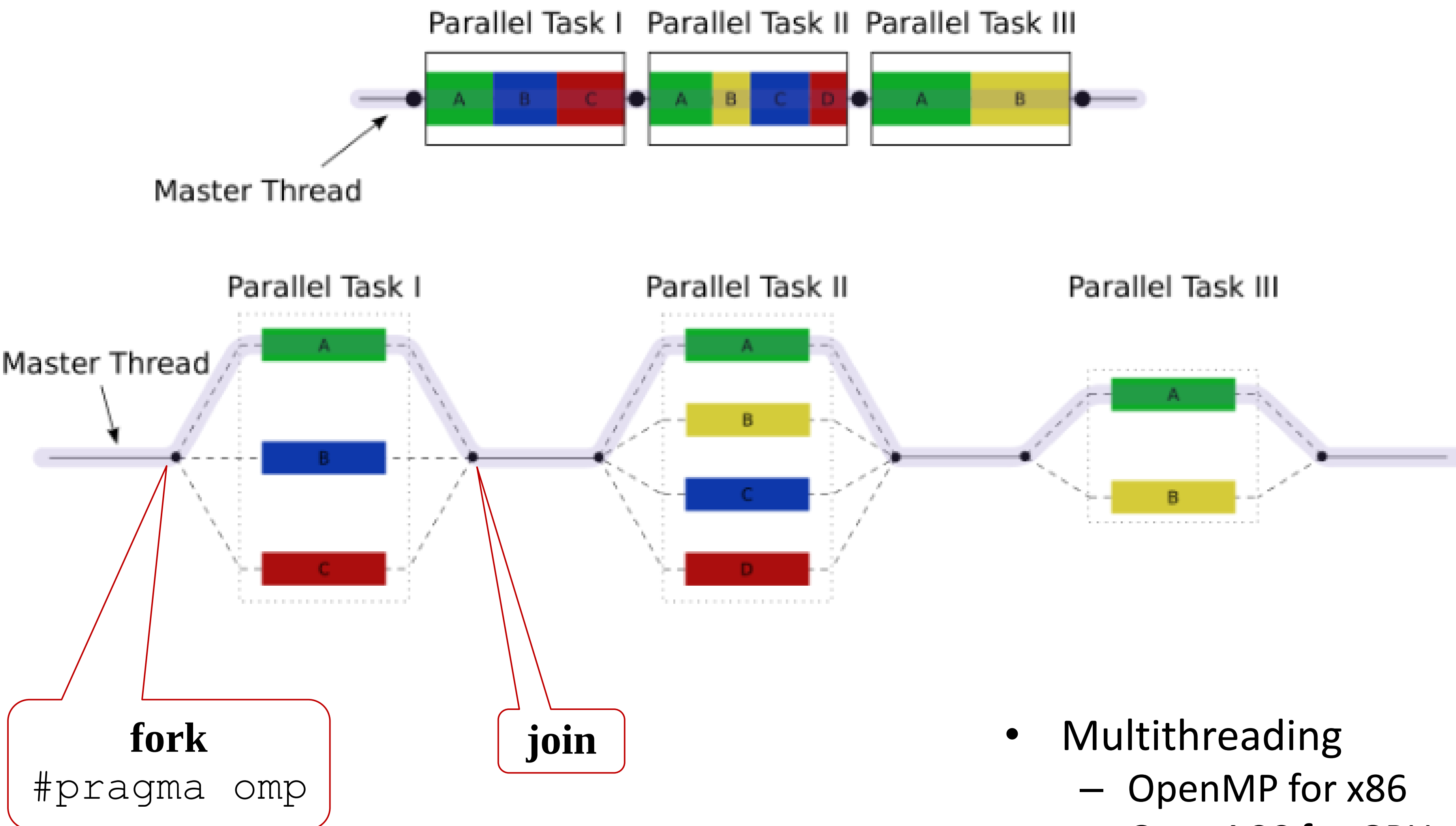
Lower bounding: Cascade of LBs



$bsf := \min(\mathbf{DTW}(Q,C), bsf_{prev})$

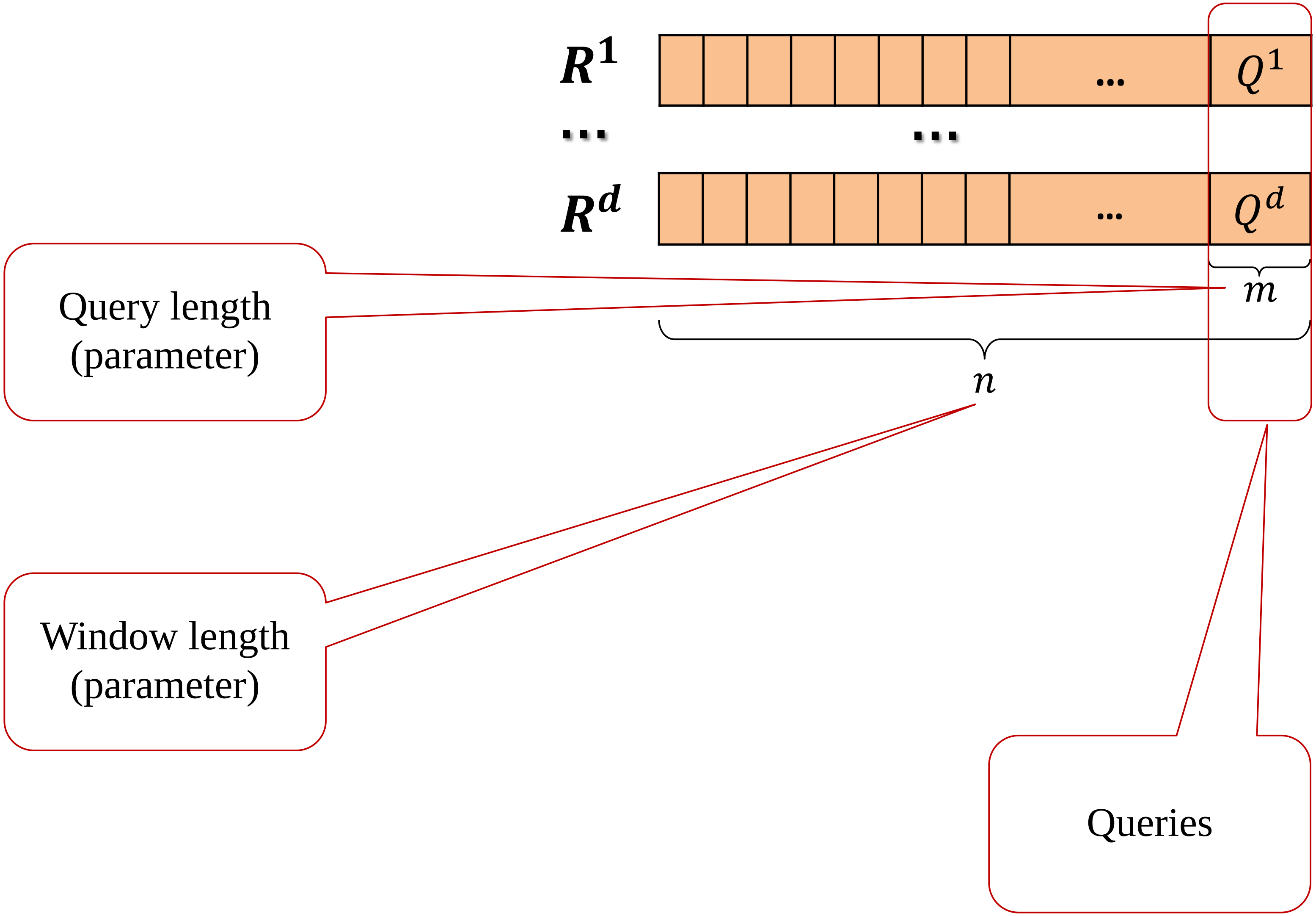
Lower bound	Complexity
LB_KimFL $\boldsymbol{LB1}(Q,C) = (q_1 - c_1)^2 + (q_m - c_m)^2$	$O(1)$
LB_KeoghEC $\boldsymbol{LB2}(Q,C) = \sum_{i=1}^{m} \begin{cases} (c_i - u_i)^2, & c_i > u_i \\ (c_i - \ell_i)^2, & c_i < \ell_i \\ 0, & otherwise \end{cases}$ $u_i = \max_{i-r \le k \le i+r} q_k$ $\ell_i = \min_{i-r \le k \le i+r} q_k$	$O(m)$
LB_KeoghEQ $\boldsymbol{LB3}(Q,C) = \boldsymbol{LB2}(C,Q)$	$O(m)$
LB_Yi, LB_PAA, LB_FTW, …	…
$\mathbf{DTW}(Q,C)$	$O(mr)$

Fork-join parallelism



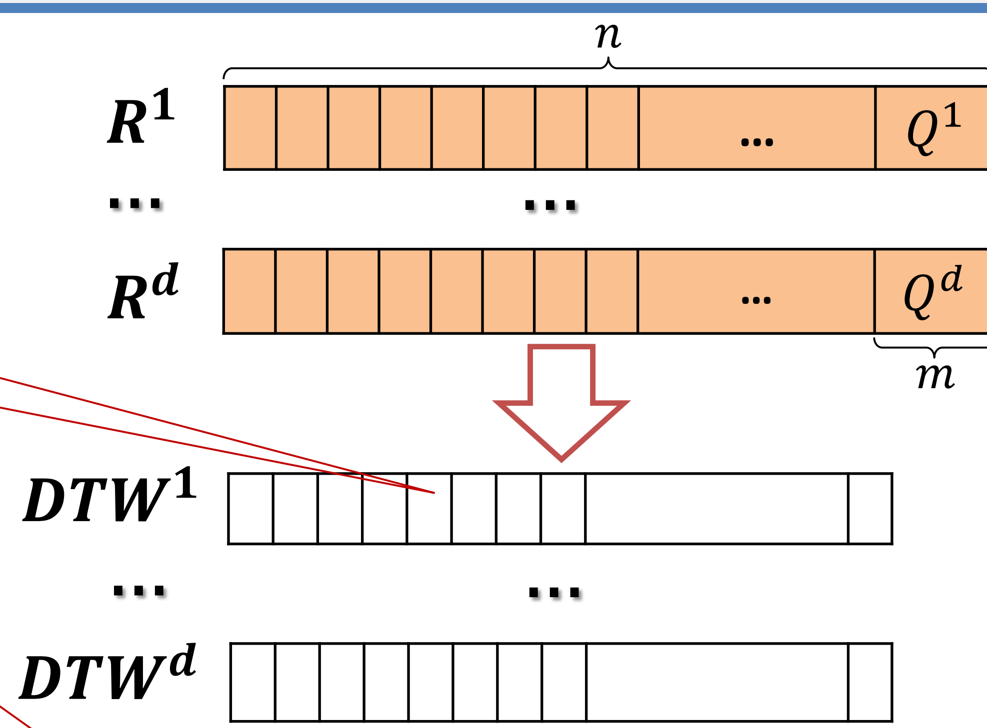
- Multithreading
  - OpenMP for x86
  - OpenACC for GPU

ParaDI: Reference time series



ParaDI: Pattern search

Distance between
the interval $R^i[j:m]$ and the query Q^i
w.r.t. the DTW distance measure



1. Pattern search

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

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

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

Green means **parallel through OpenMP**

ParaDI: Scoring

1. Pattern search

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

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

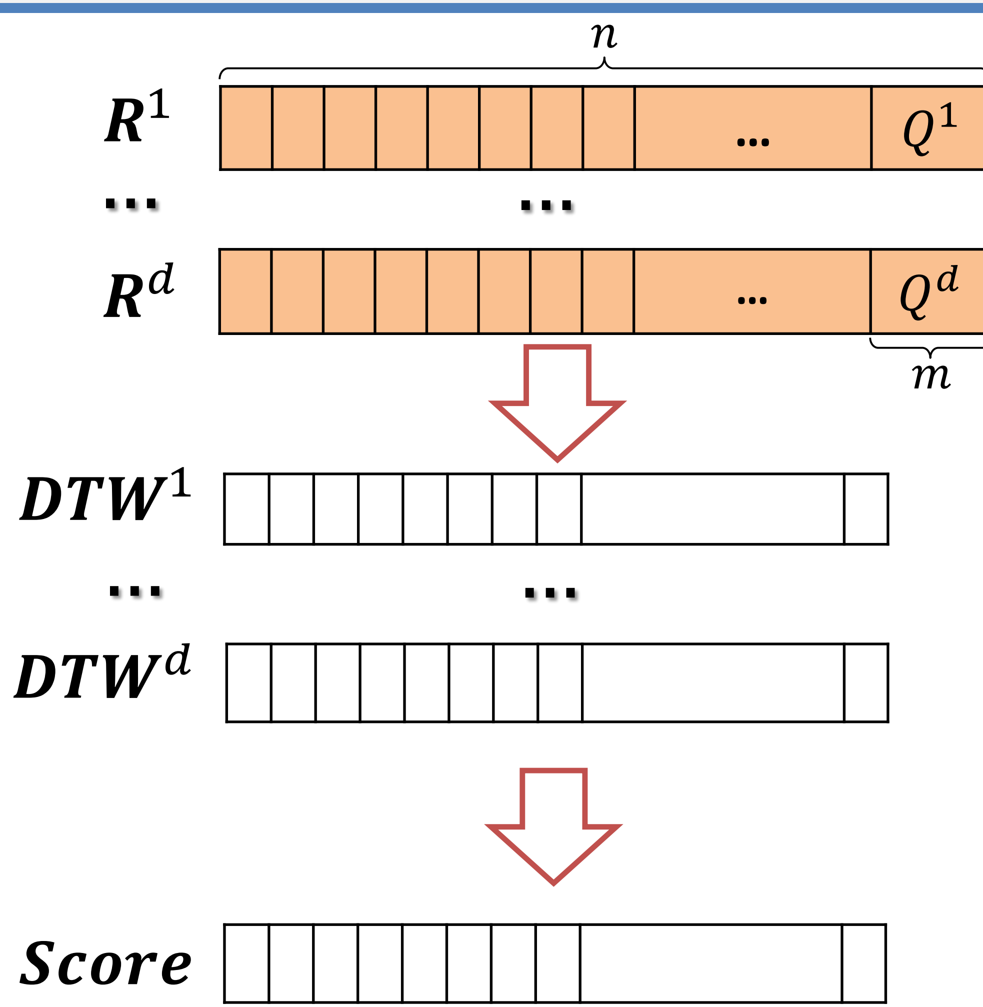
    $DTW^i(R^i[j:m], Q^i)$

2. Scoring

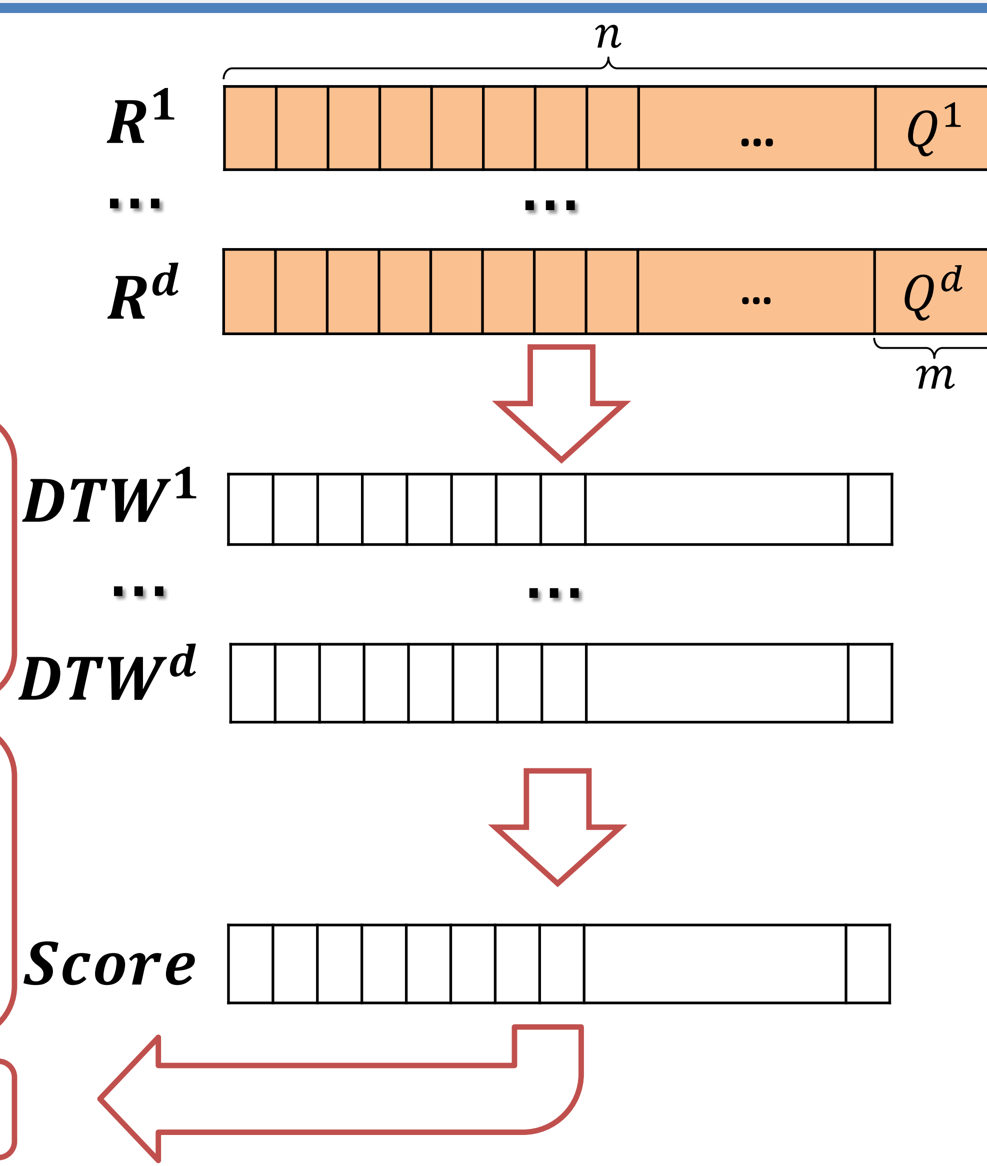
for $j := 1$ **to** $n - m + 1$ **do**

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

    $Score(j) := Score(j) + \frac{weight(j,i)}{DTW^i(j)+\varepsilon}$



ParaDI: Reconstruction



1. Pattern search

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

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

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

2. Scoring

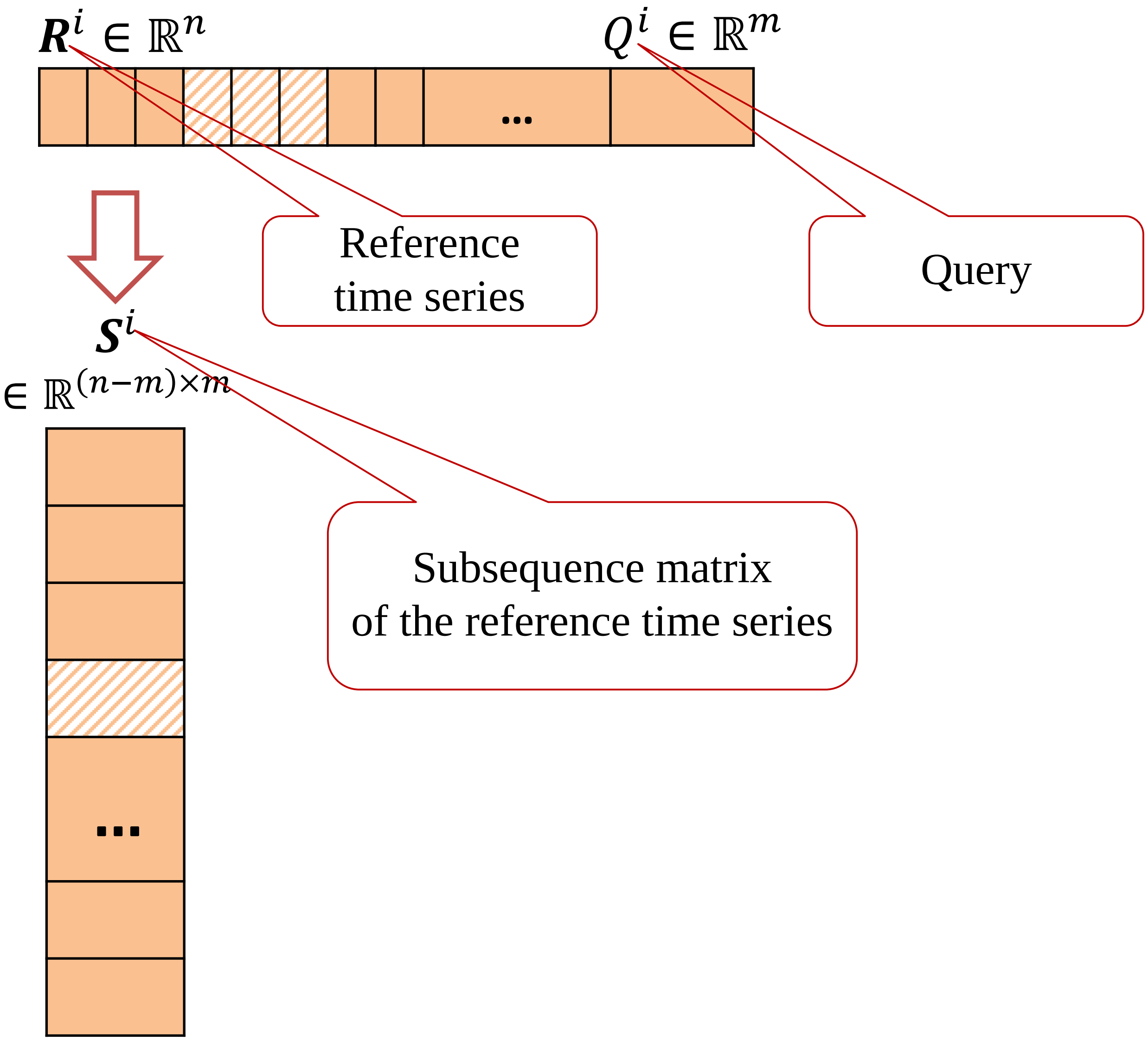
for $j := 1$ **to** $n - m + 1$ **do**

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

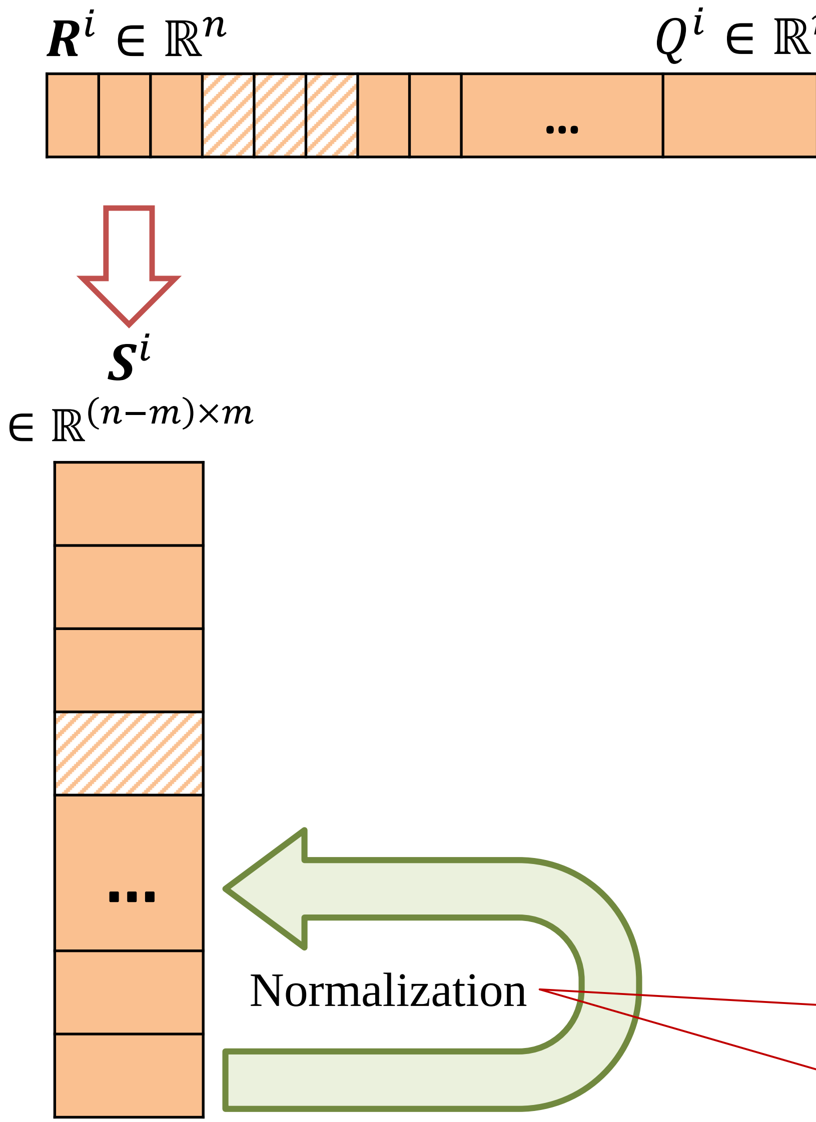
$Score(j) := Score(j) + \frac{weight(j,i)}{DTW^i(j)+\varepsilon}$

3. Selection of TOP-k intervals, reconstruction

ParaDI: Pattern search



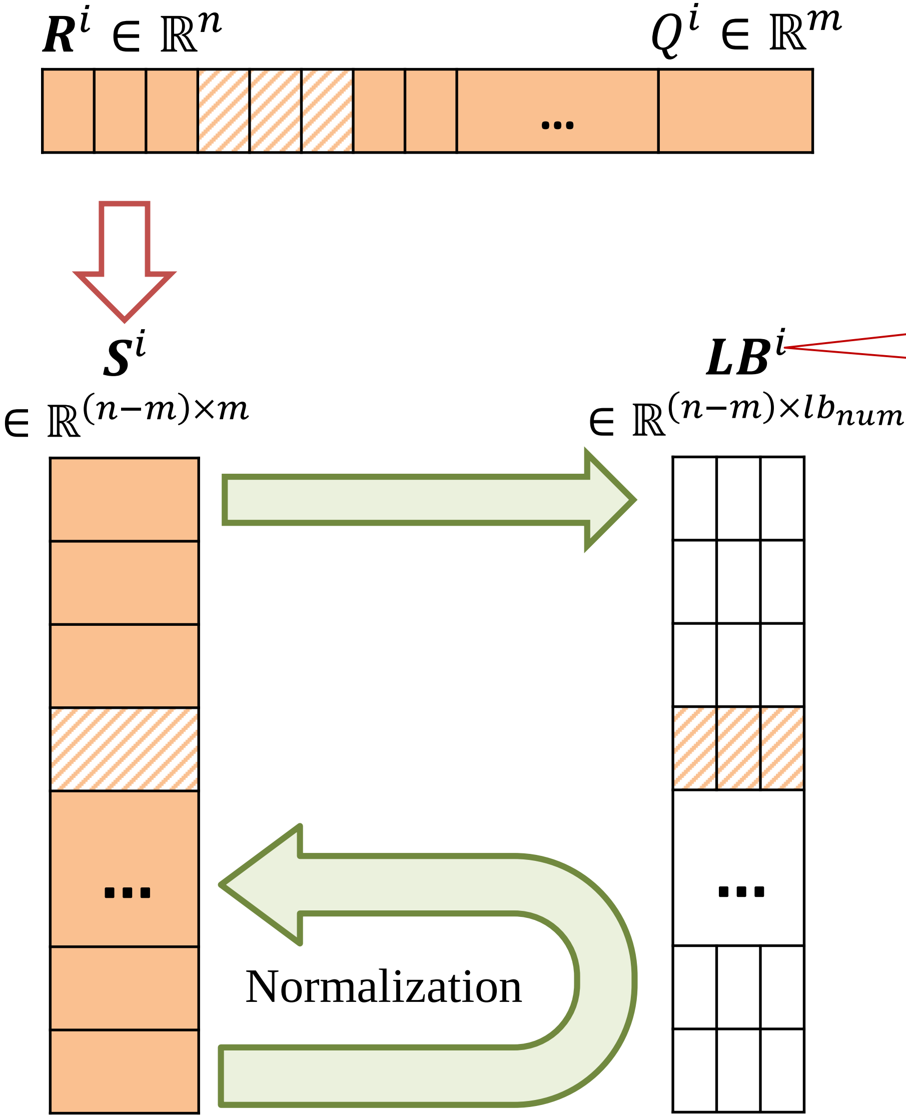
ParaDI: Normalization



$$R[1{:}m] \Rightarrow \hat{R}\,[1{:}m], \quad \hat{r}_i = \frac{r_i - \mu}{\sigma},$$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} r_i, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m} r_i^2 - \mu^2$$

ParaDI: Lower bounds

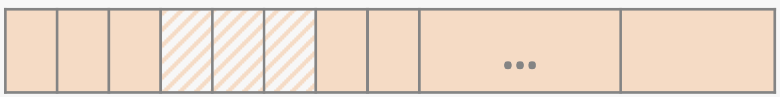


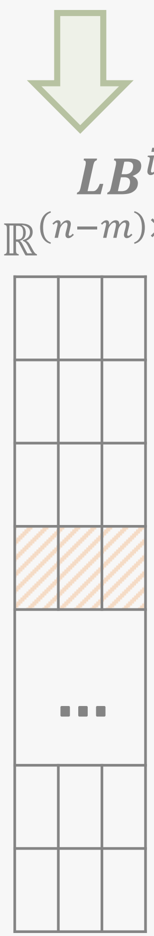
Matrix
of lower bounds

Ahead calculation of all LBs
for all subsequences
is significantly **redundant**
(cf. LB cascade in serial search)
but
it is **parallel**

ParaDI: Initializing the bsf threshold

$\boldsymbol{R}^i \in \mathbb{R}^n$ $\quad Q^i \in \mathbb{R}^m$



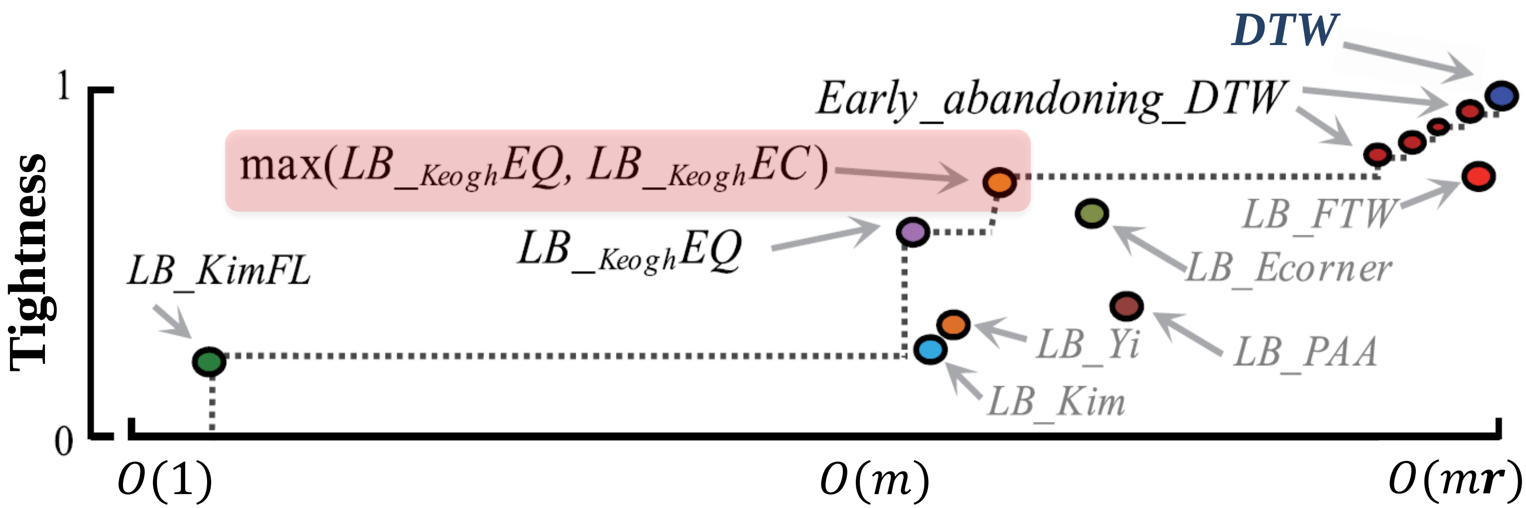


$\boldsymbol{LB}^i \in \mathbb{R}^{(n-m)\times lb_{num}}$

$$bsf_{init} \coloneqq \mathrm{DTW}(Q^i, C)$$

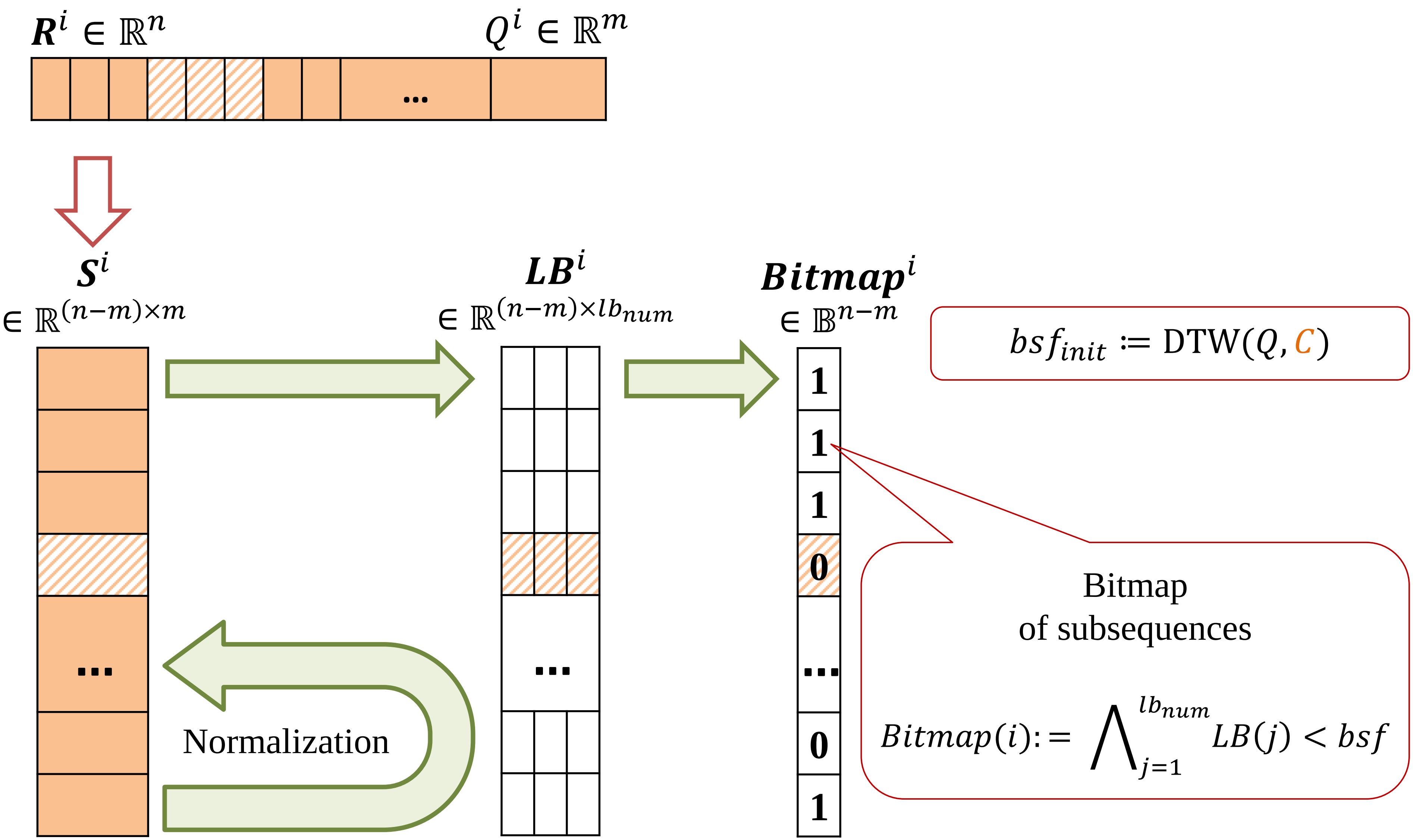
$$C = \arg \min_{1\le p\le n-m} \max_{1\le j\le lb_{num}} \boldsymbol{LB}_j^i(Q^i, \boldsymbol{R}[p{:}m])$$

$$Tightness^* = \frac{\boldsymbol{LB}(C,Q)}{\boldsymbol{DTW}(C,Q)}$$

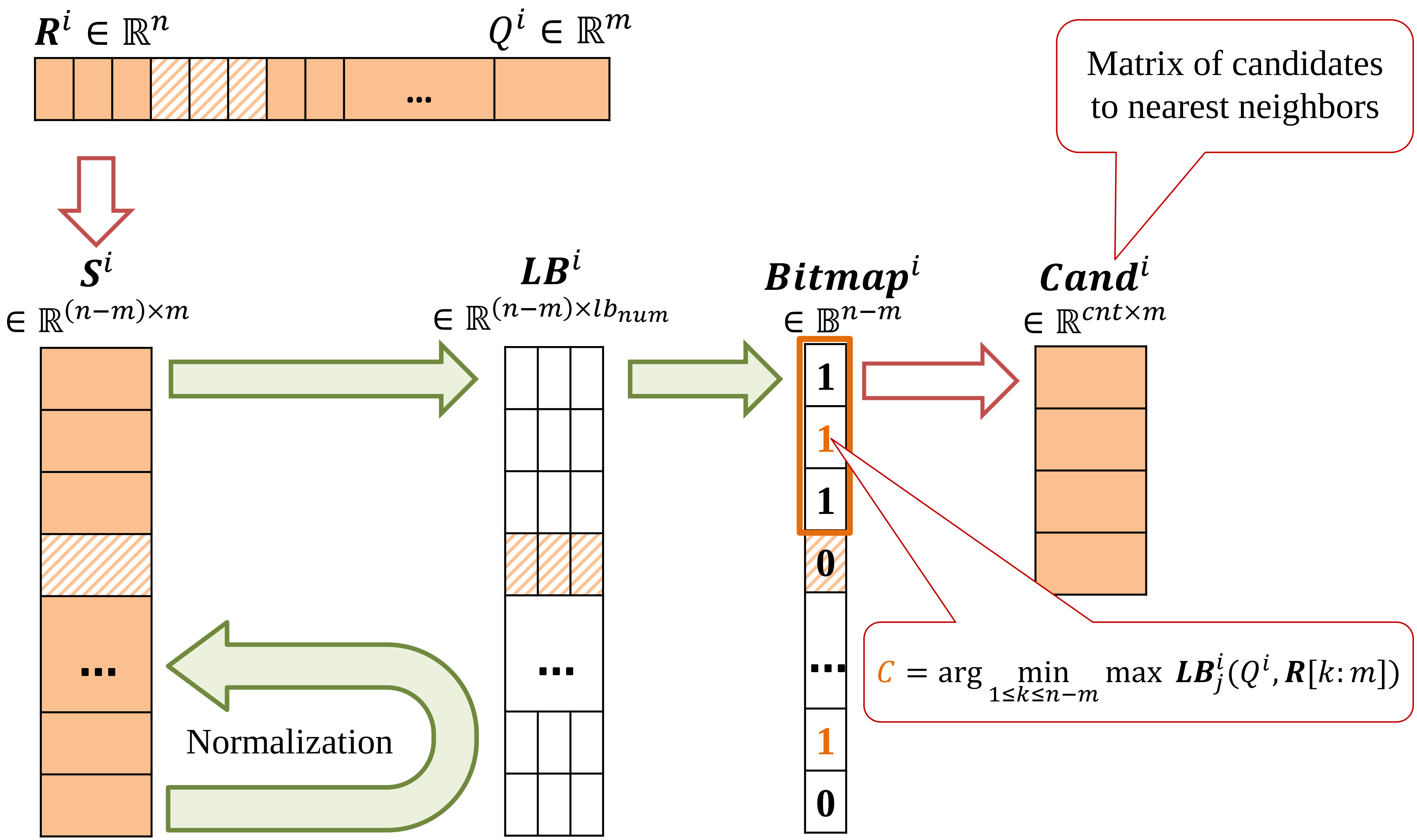


* Rakthanmanon T., *et al.* Addressing big data time series: Mining trillions of time series subsequences under Dynamic Time Warping. ACM Trans. Knowl. Discov. Data. 2013. Vol. 7, no. 3. 10:110:31. DOI: 10.1145/2500489.

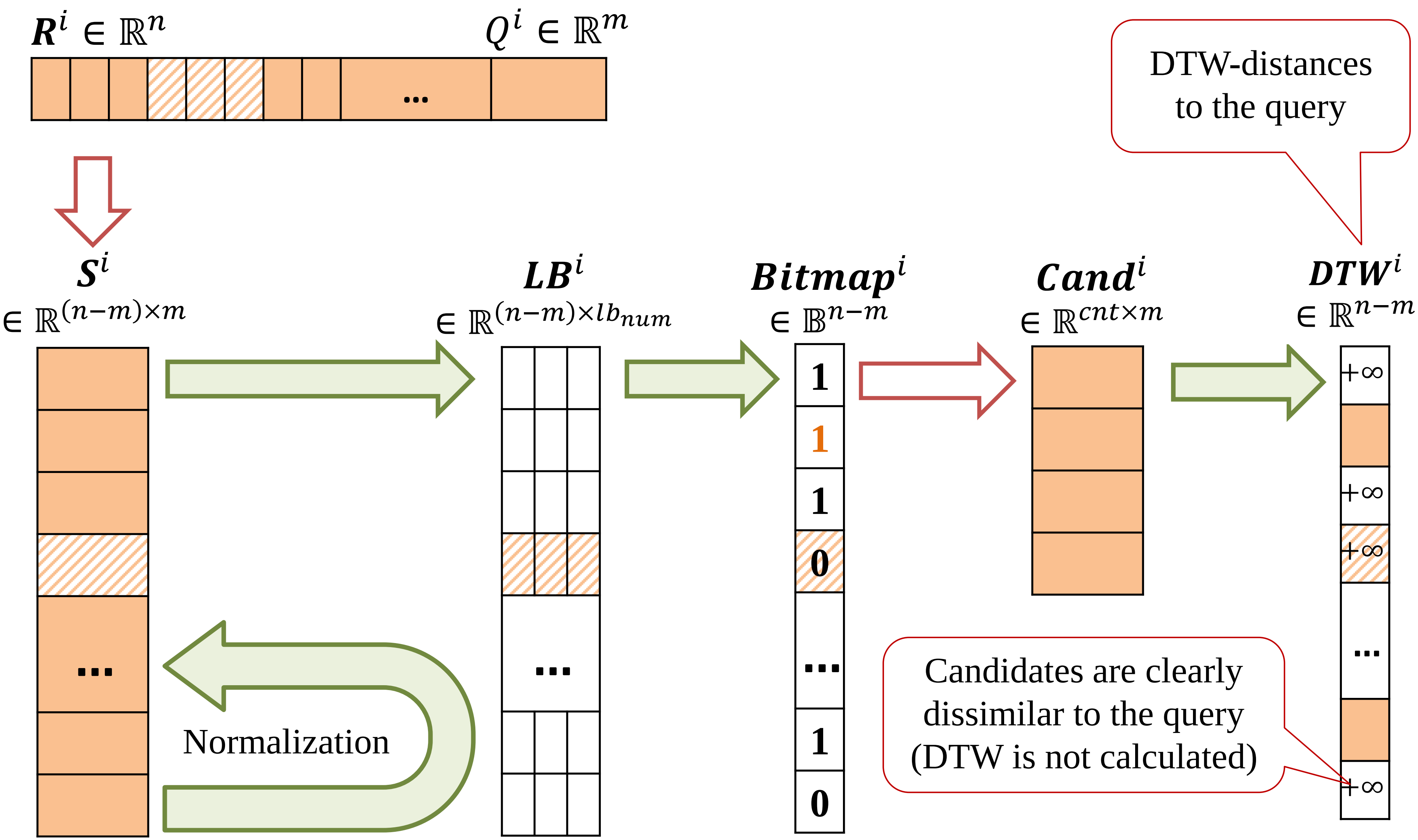
ParaDI: Lower bounding



ParaDI: Candidate matrix



ParaDI: DTW calculation



ParaDI: vectorization in DTW calculation

```
double DTW(a: array [1..m], b: array [1..m], r: int) {
   cost := array [1..m]
   cost_prev := array [1..m]

   for i := 1 to m
      cost[i] = infinity
      cost_prev[i] = infinity

   cost_prev[1] = dist(a[1], b[1])

   for j := max(2, i-r) to min(m, i+r)
      cost_prev[j] := cost_prev[j-1] + dist(a[1], b[j])

   for i := 2 to m
      for j := max(1, i-r) to min(m, i+r)
         c := d(a[i], b[j])
         cost[j] := c + min(cost[j-1], cost_prev[j-1], cost_prev[j])
      swap(cost, cost_prev)

   return cost_prev[m]
}
```

Cannot be vectorized by a compiler, ☹
but can be rewritten to be vectorized ☺

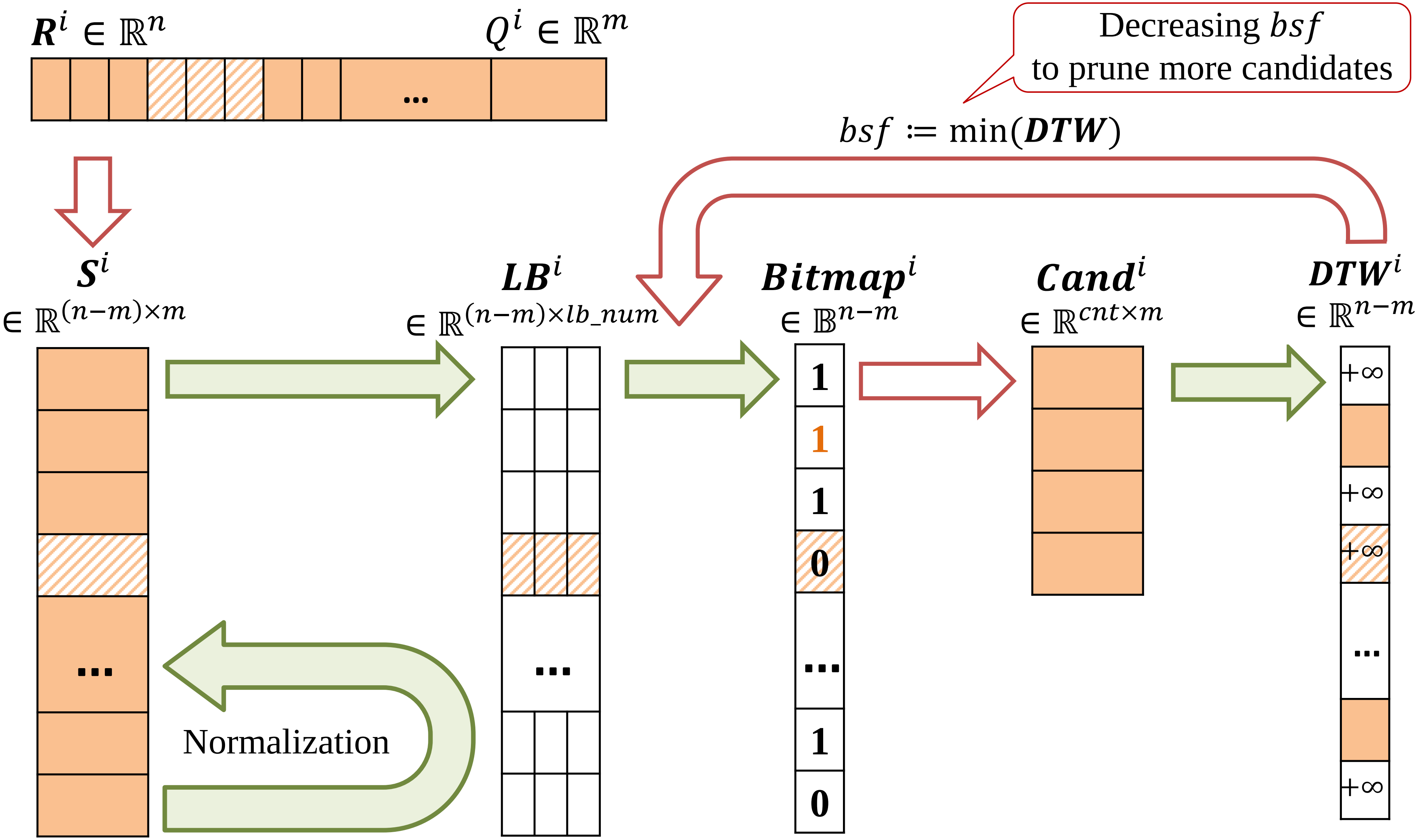
ParaDI: vectorization in DTW calculation

```
double DTW(a: array [1..m], b: array [1..m], r: int) {
    cost := array [1..m]
    cost_prev := array [1..m]
    for i := 1 to m
        cost[i] = infinity
        cost_prev[i] = infinity
    cost_prev[1] = dist(a[1], b[1])
    for j := max(2, i-r) to min(m, i+r)
        cost_prev[j] := cost_prev[j-1] + dist(a[1], b[j])
    for i := 2 to m
        for j := max(1, i-r) to min(m, i+r)
            cost[j] = min(cost_prev[j-1], cost_prev[j])
        for j := max(1, i-r) to min(m, i+r)
            c := dist(a[i], b[j])
            cost[j] := c + min(cost[j-1], cost[j])
        swap(cost, cost_prev)
    return cost_prev[m]
}
```

Vectorized by the compiler ☺

Not vectorized by the compiler ☹

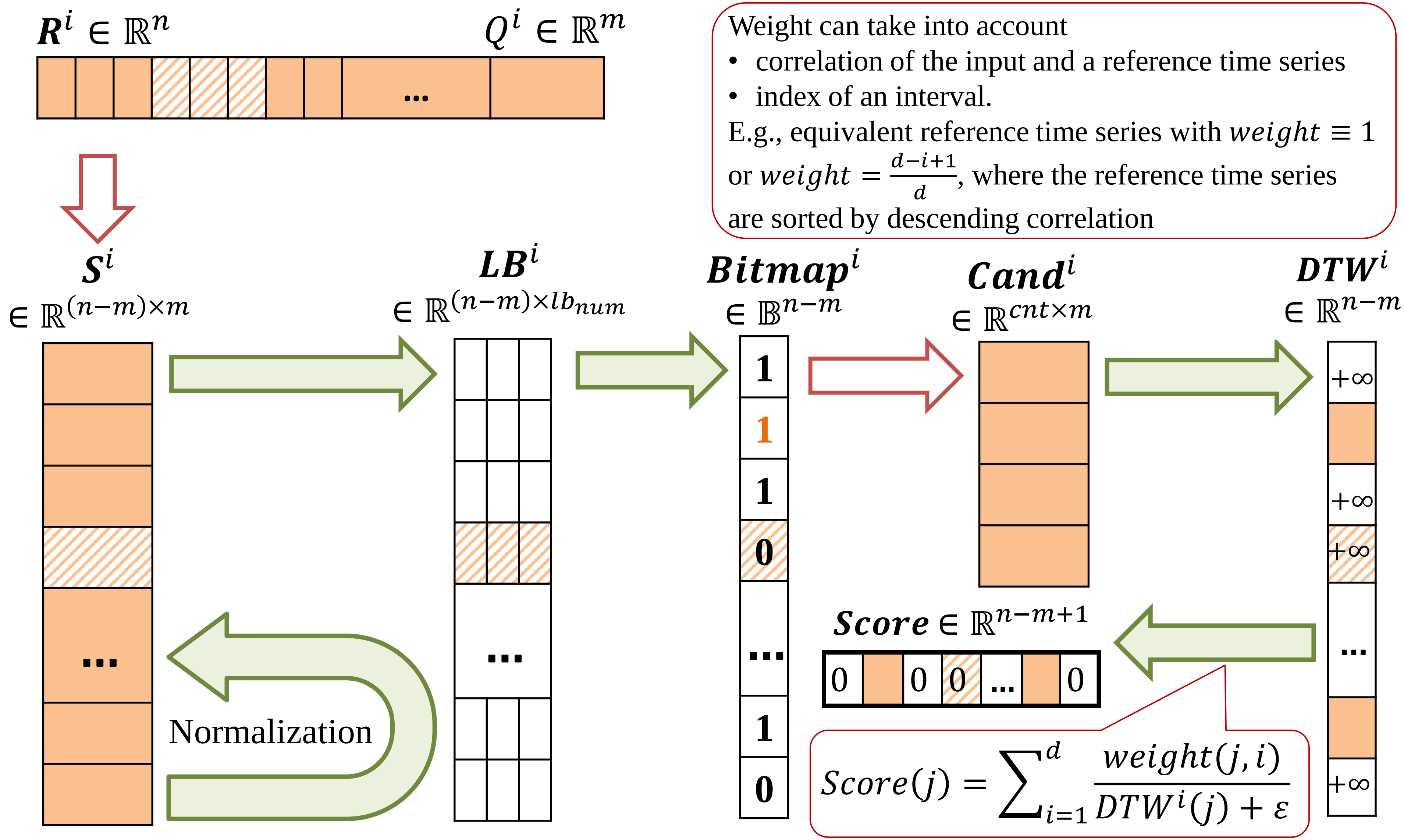
ParaDI: Improving the bsf threshold



ParaDI: Scoring

Weight can take into account
- correlation of the input and a reference time series
- index of an interval.

E.g., equivalent reference time series with $weight \equiv 1$ or $weight = \frac{d-i+1}{d}$, where the reference time series are sorted by descending correlation



ParaDI: Experiments

- **Hardware** (SUSU SSL): Intel Xeon E5-2687W v2 (8 cores @3.40 GHz)
- **Data**

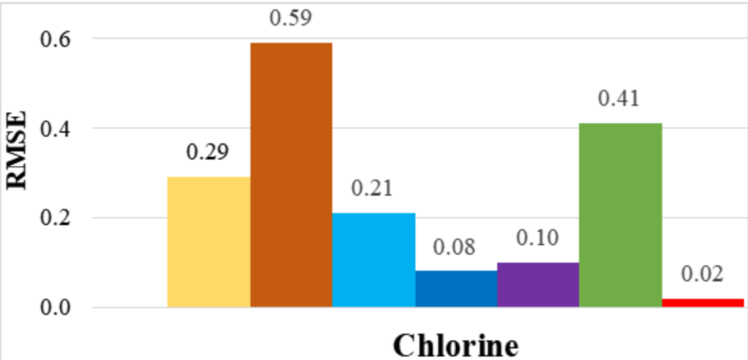
Dataset	# t.s., $d+1$	Length, $n \cdot 10^3$	Domain
BAFU	10	50	Water discharge in Swiss rivers
Chlorine	50	1	Simulation of the chlorine concentration in a drinking water system
Climate	10	5	Weather in locations of North America
MADRID	10	25	Road traffic (AVR statistics) in Madrid
MAREL	10	50	Characteristics of sea water in the English Channel

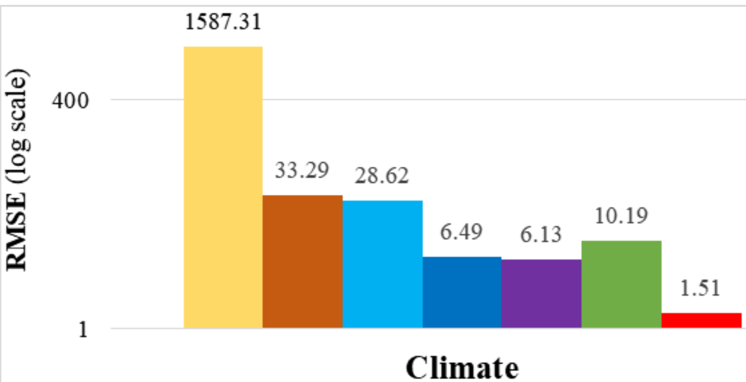
- **Rivals**: ORBITS[1], OGD-Impute[2], SPIRIT[3], SAGE[4], TKCM[5]
- **Setup** under ORBITS[1] framework:
  - Scenario: imputation of last 10% points; cases for max and min number of ref. time series
  - Rivals: best recommended parameters
  - ParaDI: $m = 50$, $k = 3$, $r = 0.25m$

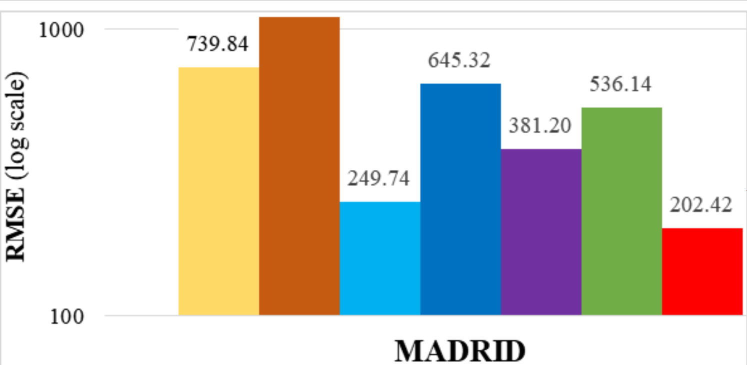
[1] Khayati M., *et al.* ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams. Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294-306.
[2] Anava O., *et al.* Online Time Series Prediction with Missing Data. Proc. ICML 2015. P. 2191-2199.
[3] Papadimitriou S., *et al.* Streaming Pattern Discovery in Multiple Time-Series. VLDB 2005. P. 697-708.
[4] Balzano L., *et al.* Streaming PCA and Subspace Tracking: The Missing Data Case. Proc. of IEEE. 2018. Vol. 106, no. 8. P. 1293-1310.
[5] Wellenzohn K., *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series. EDBT 2017. P. 330-341.

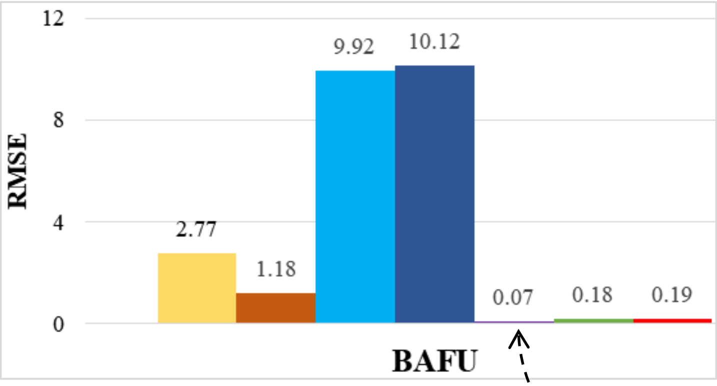
Experiments: accuracy

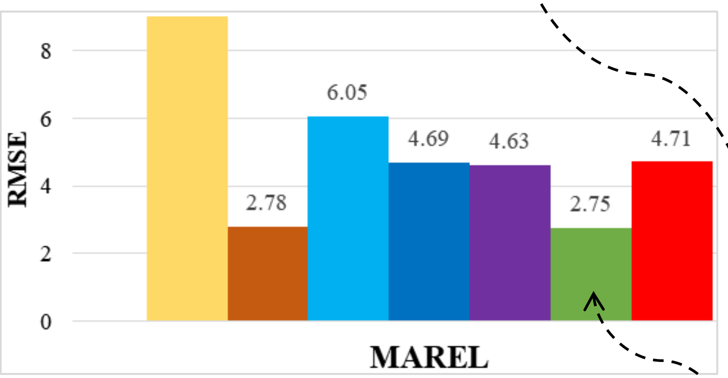
$$\text{RMSE} = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(t_i - \tilde{t}_i)^2}$$











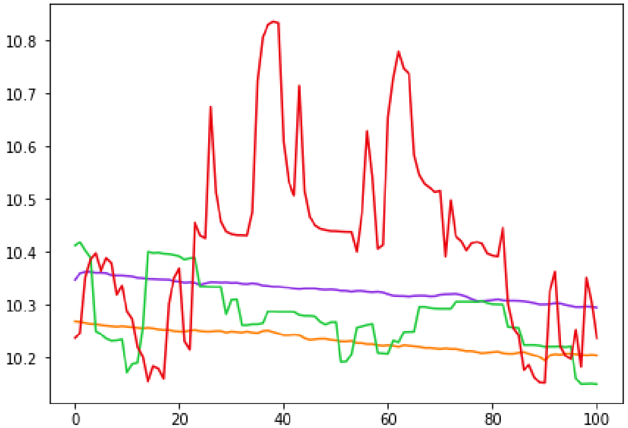
ParaDI is **ahead** of many but **not all** rivals:

SPIRIT on BAFU, TKCM on MAREL

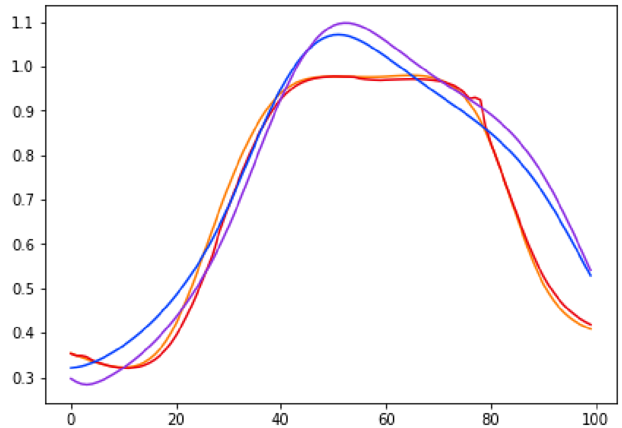
SAGE | OGD-Impute | ORBITS (k=2) | ORBITS (k=3)

SPIRIT | TKCM | **ParaDI**

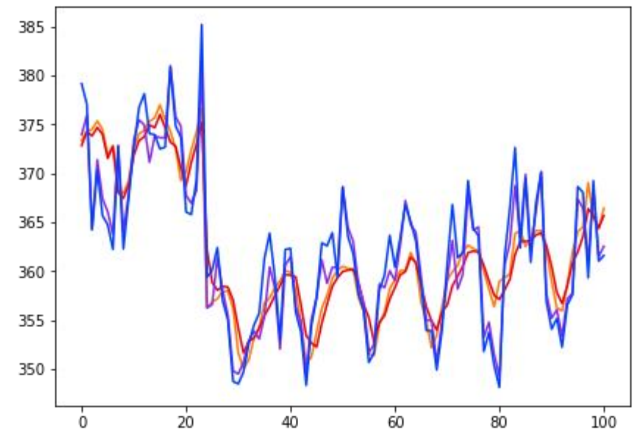
Experiments: imputation of the first 100 points*



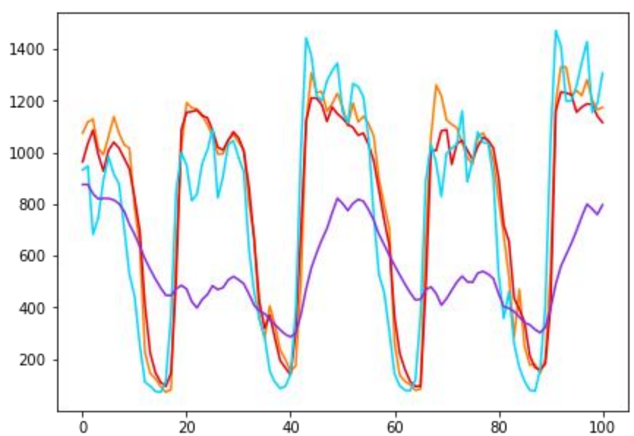
BAFU



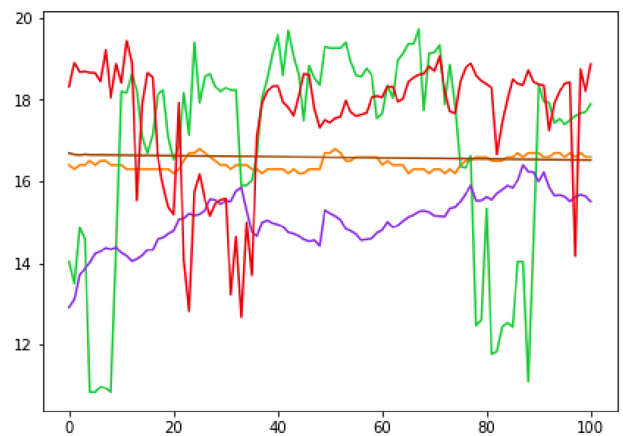
Chlorine



Climate



MADRID



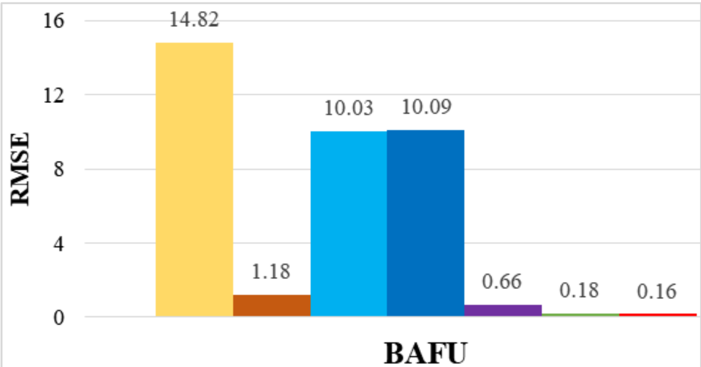
MAREL

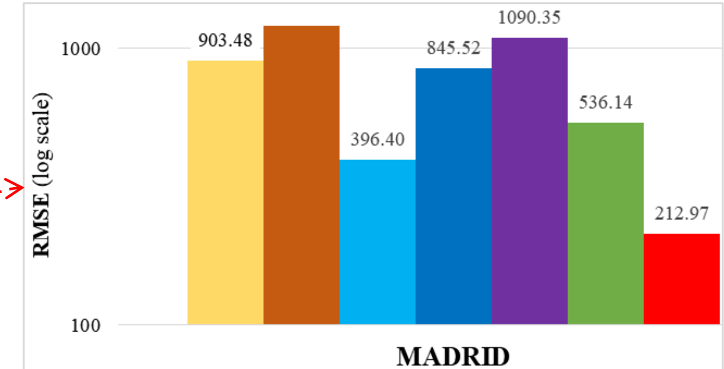
- SAGE
- OGD-Impute
- SPIRIT
- TKCM
- ORBITS (k=2)
- ORBITS (k=3)
- **ParaDI**
- **real data**

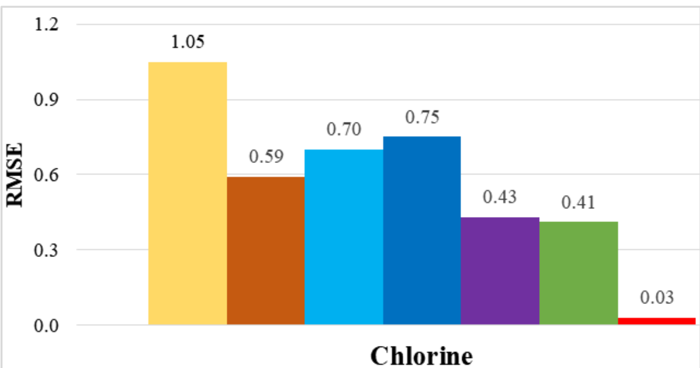
* We show the real data and the results of top-3 algorithms w.r.t. accuracy

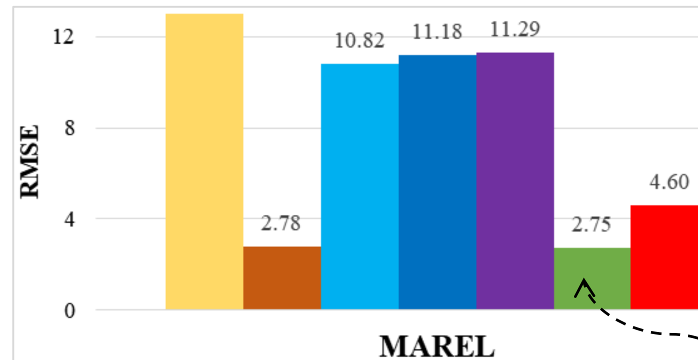
Experiments: accuracy (min # ref. t.s.)

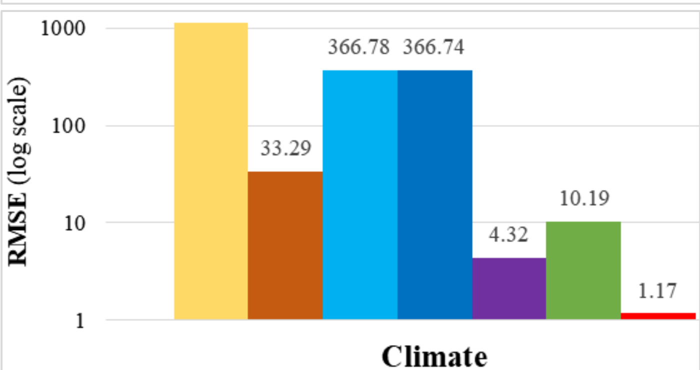
$$\text{RMSE} = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(t_i - \tilde{t}_i)^2}$$







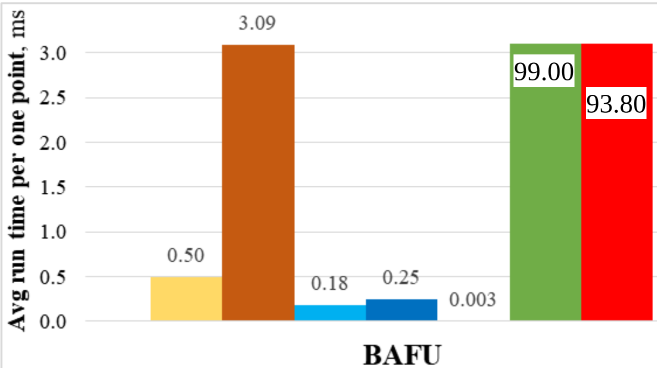


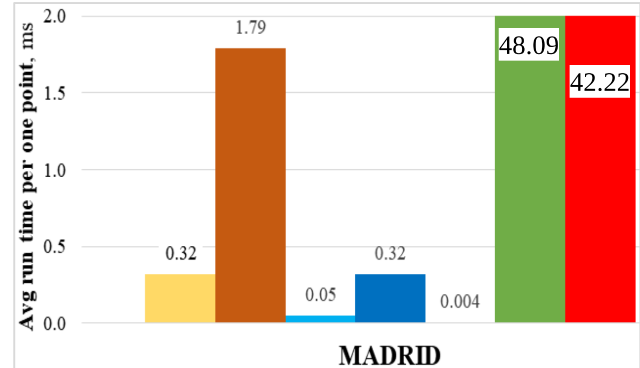


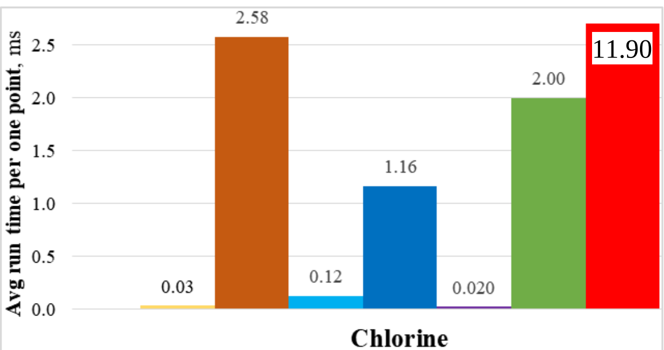
ParaDI is **ahead** of all but **one** rival: TKCM on MAREL

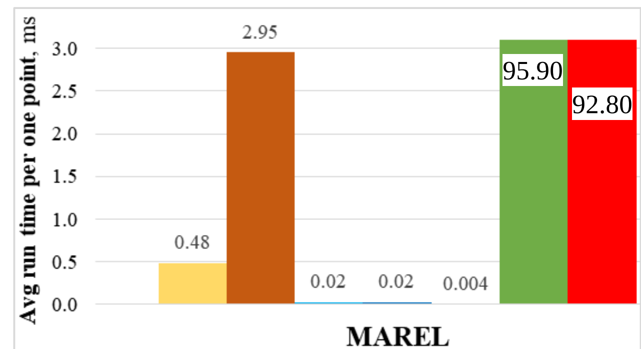
SAGE | OGD-Impute | ORBITS (k=2) | ORBITS (k=3)
SPIRIT | TKCM | **ParaDI**

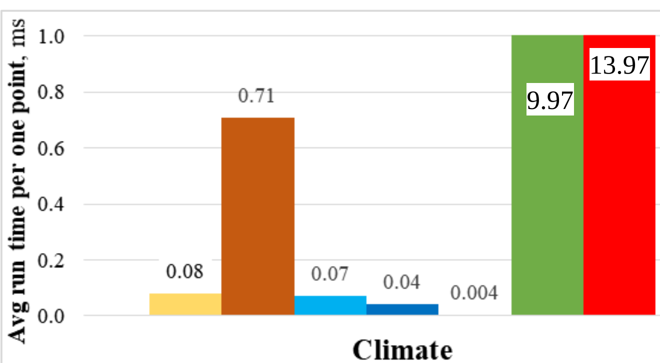
Experiments: performance







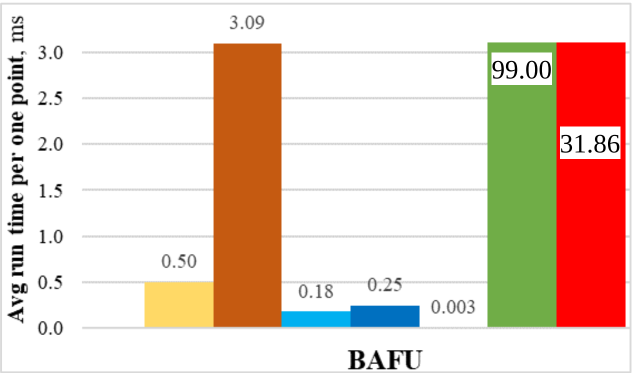


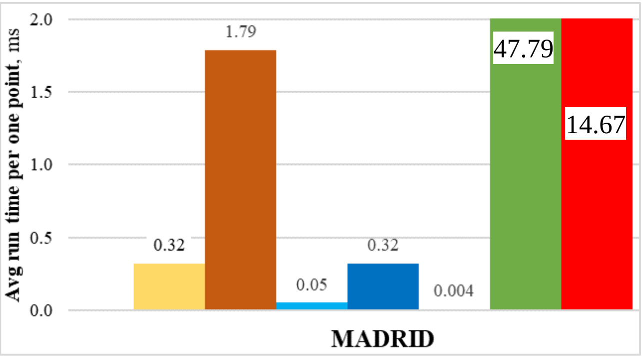


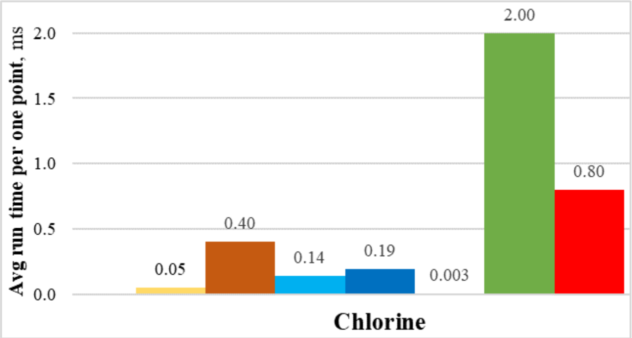
ParaDI is **inferior to all but one rival** (TKCM)

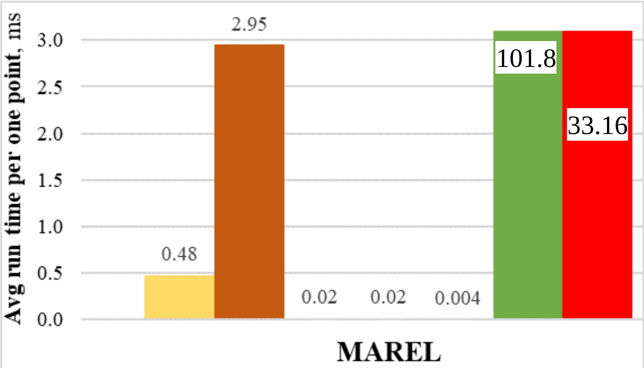
SAGE OGD-Impute ORBITS (k=2) ORBITS (k=3)
SPIRIT TKCM **ParaDI**

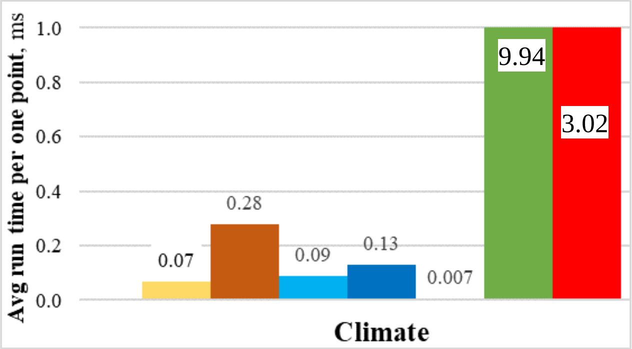
Experiments: performance (min # ref. t.s.)







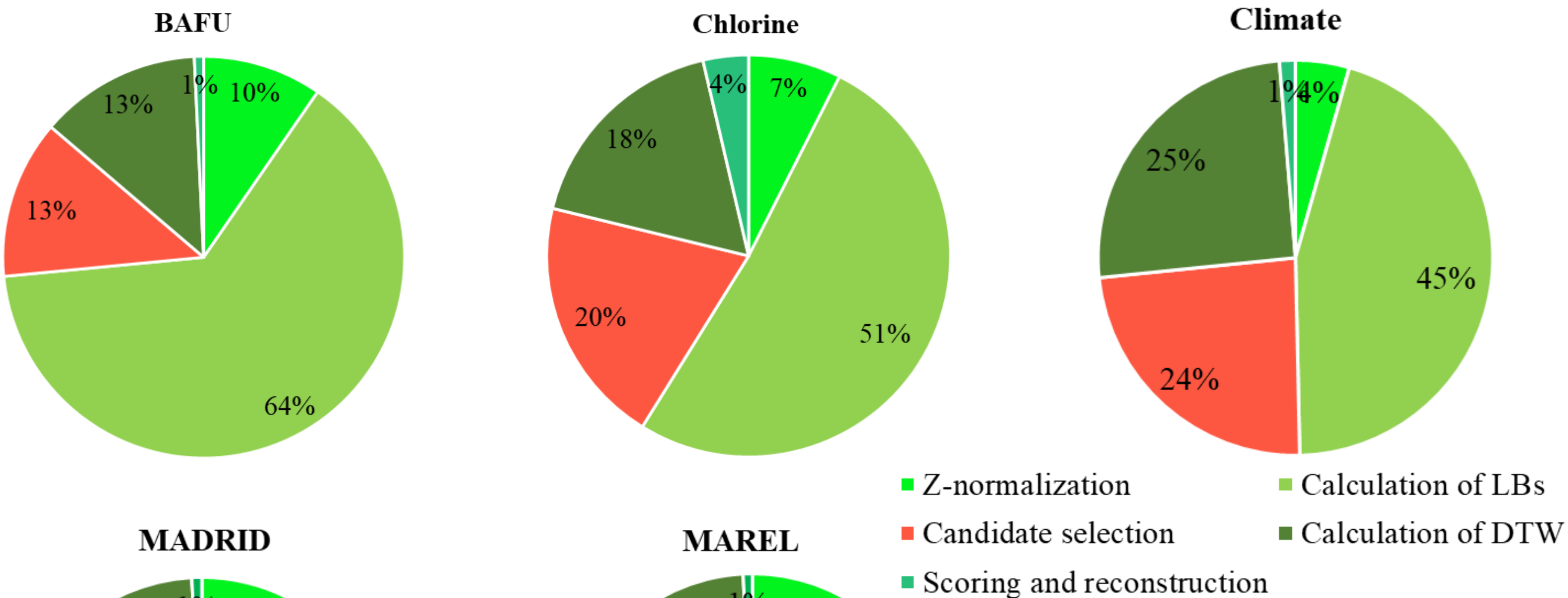


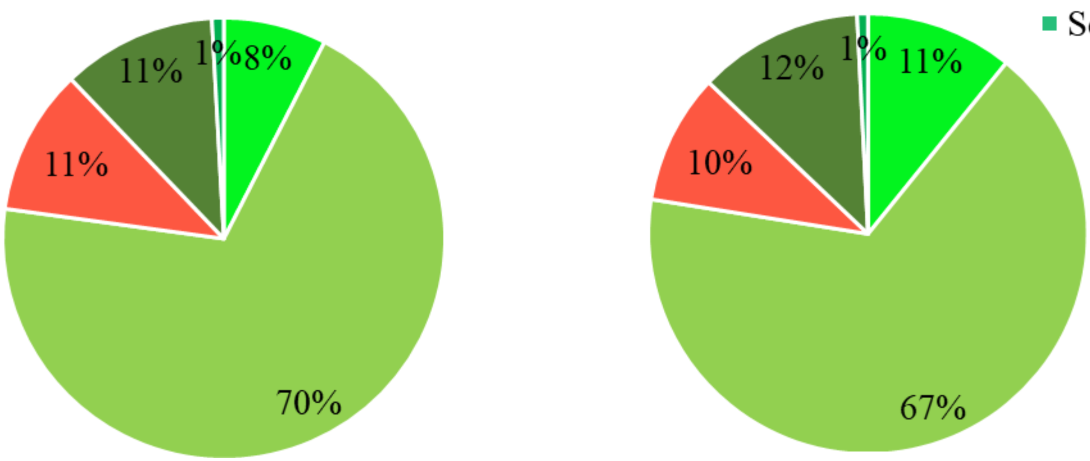


ParaDI is **inferior to all but one rival** (TKCM), SPIRIT is the fastest

SAGE | OGD-Impute | ORBITS (k=2) | ORBITS (k=3)
SPIRIT | TKCM | **ParaDI**

Experiments: runtime spent by phases

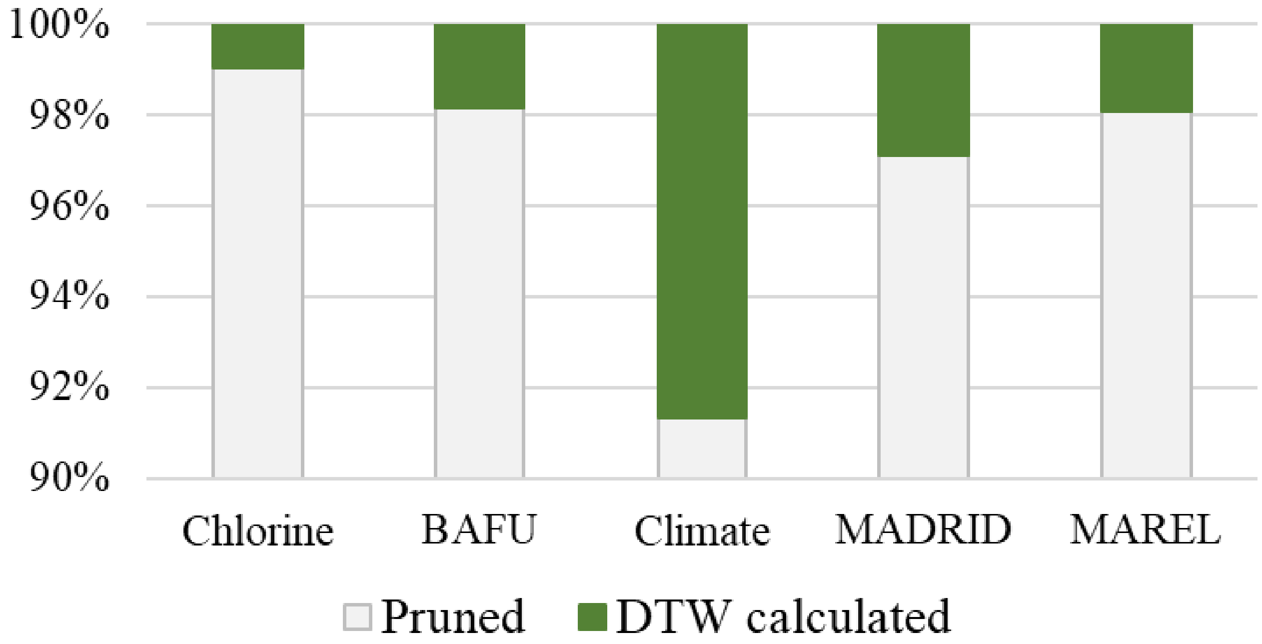




Top-3
time consuming phases:

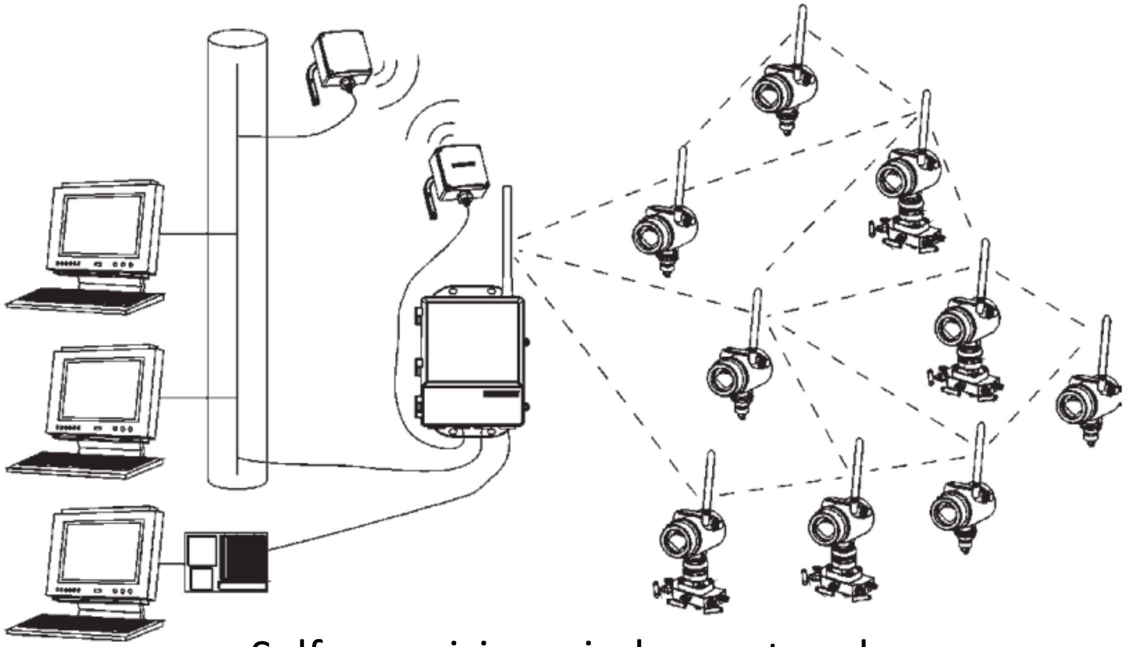
1. **Calculation of LBs**
2. **Calculation of DTW**
3. **Candidate selection**

Experiments: efficiency of lower bounding



From 91% to 99% calculations of DTW are pruned

ParaDI: as for real-time mode…



Self-organizing wireless network

Wireless Gateway 1420*		**ParaDI**, **ms**	
# sensors	min data update time, **s**	worst	best
100	8	93.8	0.8
50	4		
12	1		

ParaDI fits real-time mode

* Emerson temperature sensors catalogue 2021. URL: https://www.c-o-k.ru/library/catalogs/emerson/110477.pdf

ParaDI: *pro et contra*

- Pros
  - ahead of many (but not all) analogs w.r.t. accuracy
  - fits for real-time mode
  - needs a minimum number of reference time series
- Cons
  - memory overhead
  - missing values in a reference time series
- Scope
  - stationary time series (the mean and variance are relatively constant)

Conclusions and further research

- ParaDI is novel parallel algorithm for imputation missing values of time series
  - Accuracy: ahead of many but not all analogs
  - Speed: inferior to all but one analog, but still fits for real-time
  - Scope: stationary time series
- Future works
  - Avoiding redundant calculations when sliding in z-norm, LBs
  - Calibration phase: (semi-)automatic choice of n, m, r, k
  - GPU version: calculation of LBs
  - Extensive experiments: impact of n, m, r, k
  - Case of missing values in a reference time series

Thank you for paying attention! Questions?

Mikhail Zymbler (mzym@susu.ru), Andrey Poluyanov, Yana Kraeva