Efficient Search of the Best Warping Window for Dynamic Time Warping





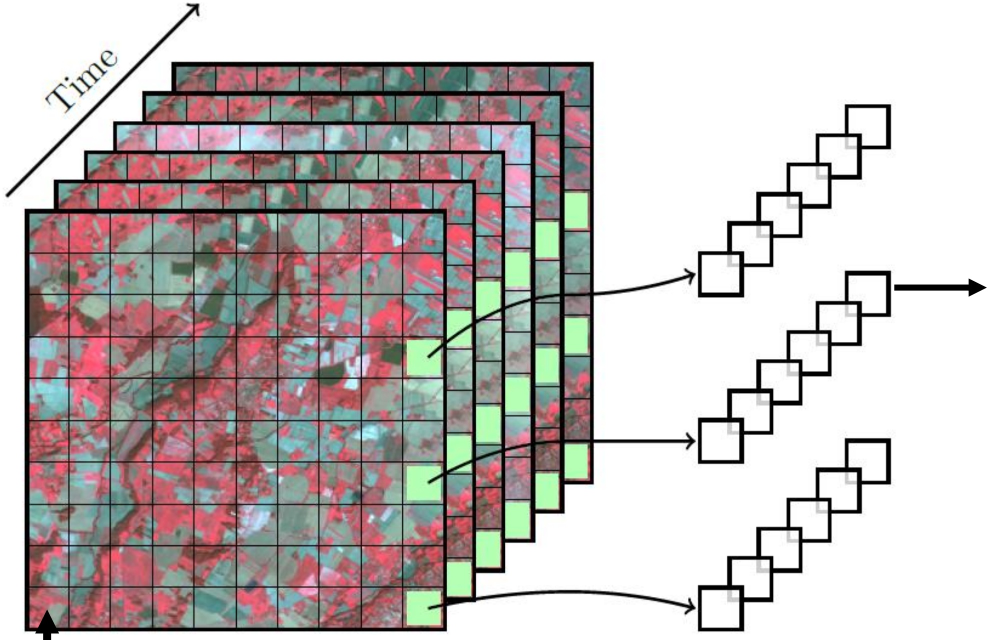
CW. Tan   M. Herrmann   G. Forestier   G.I. Webb   F. Petitjean

2018 SIAM International Conference on DATA MINING

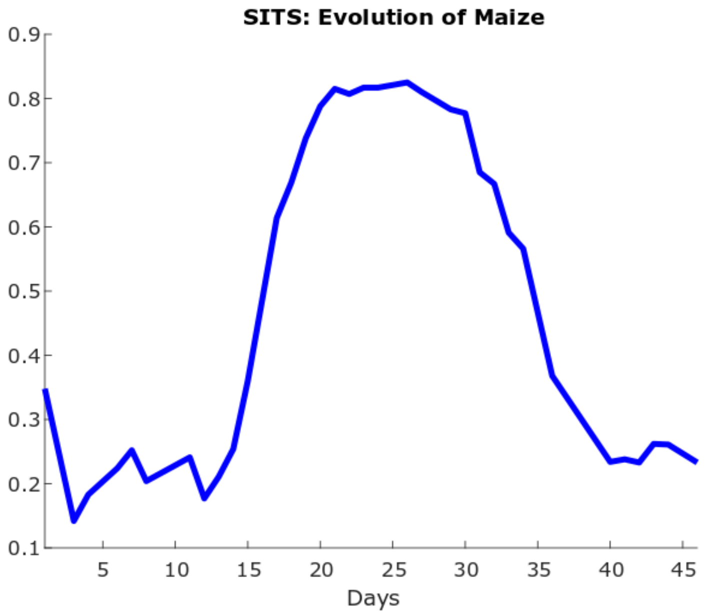
3 May 2018

What is a Time Series?

- Collection of observations made sequentially, more intuitive **visually**
- Many data can be transformed into time series → **Satellite Image Time Series**



**0.348,0.245,0.142,0.183,0.203,
0.224,0.252,0.204,0.216,0.229,
0.241,0.177,0.211,0.254,0.360,
0.487,0.614,0.669,0.738,0.788,
0.815,0.807,0.817,0.817,0.821,
0.825,0.810,0.796,0.783,0.777,
0.685,0.667,0.591,0.566,0.467,
0.368,0.335,0.301,0.268,0.234,
0.238,0.233,0.262,0.261,0.247,
0.233**



Satellite Images

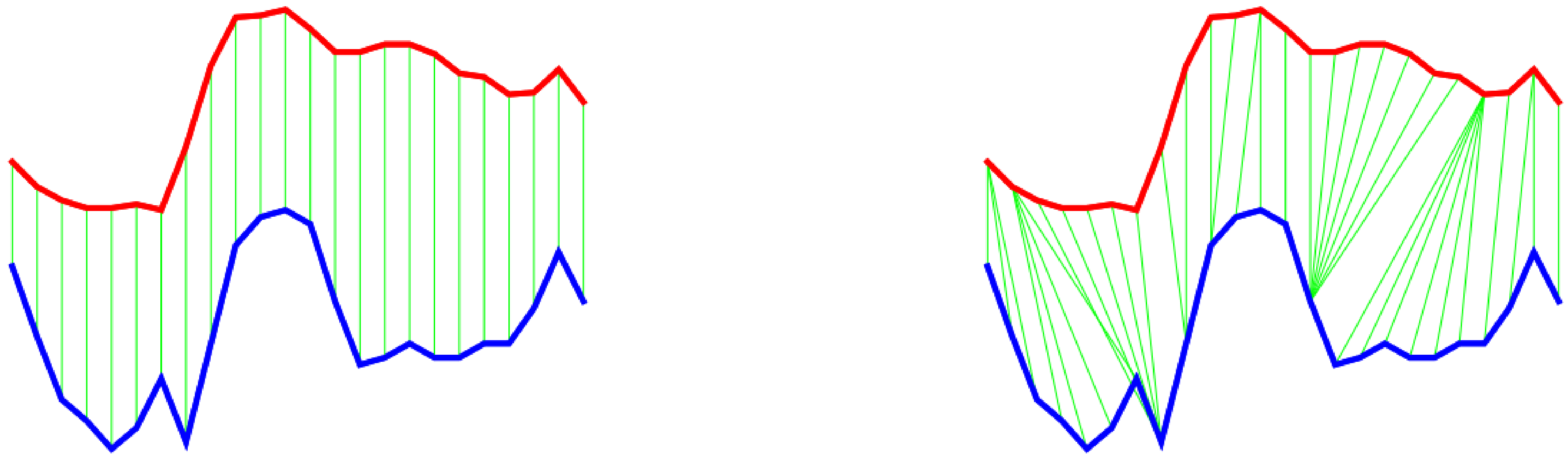
Array of numbers

Time series

Every pixel represents a geographic area (Lat, Lon) on Earth

Dynamic Time Warping

- a.k.a. **DTW** – similarity function to align time series $O(L^2)$
- Nearest Neighbour Algorithm (NN-DTW) – Hard to beat [1]
- Used in many fields: Finance, Engineering, Speech Recognition, …



Euclidean Distance
One-to-one alignment

Dynamic Time Warping
Nonlinear alignment

[1] Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606-660..

Dynamic Time Warping

- Aligns two time series Q and C using Dynamic Programming
  - Build a cost matrix and solve:

$$D^{Q,C}(i,j) = \delta\left(q_i, c_j\right) + \min\begin{cases} D^{Q,C}(i-1, j-1) \\ D^{Q,C}(i-1, j) \\ D^{Q,C}(i, j-1) \end{cases}$$

  - where $\delta\left(q_i, c_j\right) = L_p\mathrm{-norm}$

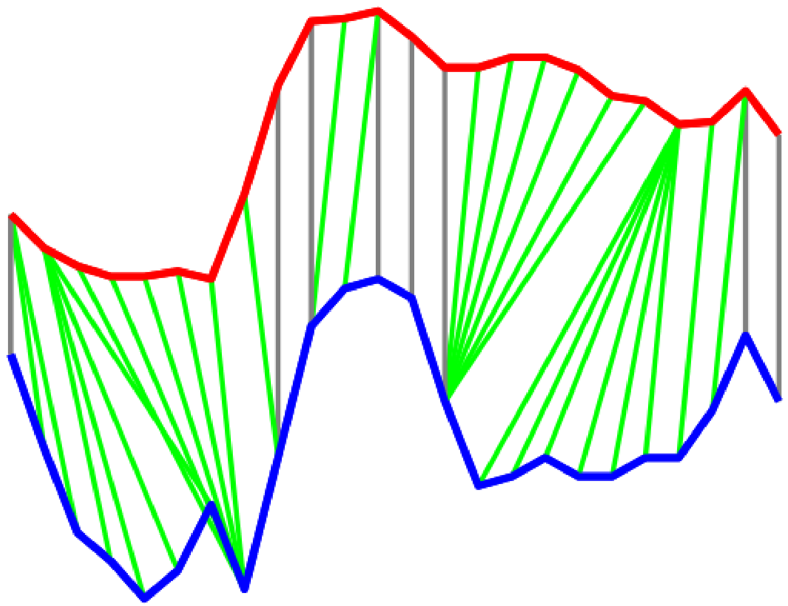
$$\mathrm{DTW}(Q,C) = \left(D^{Q,C}(m,n)\right)^{\frac{1}{p}}$$

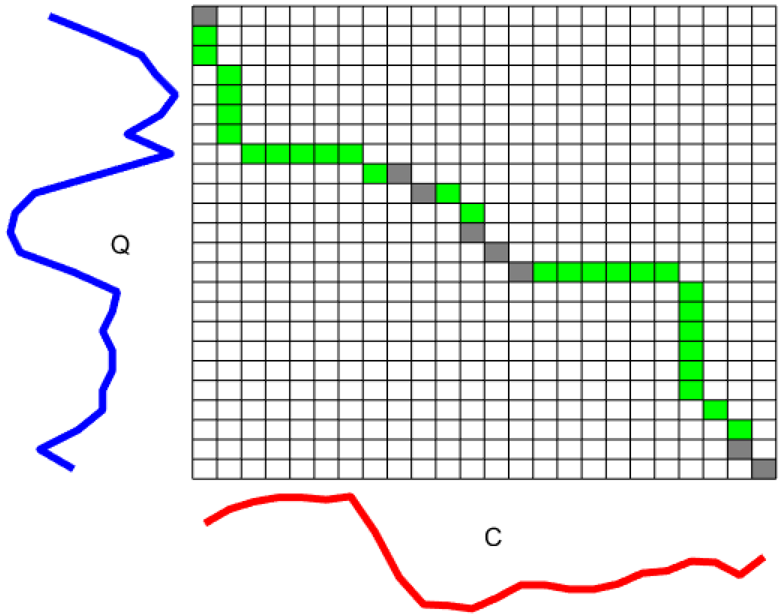
Dynamic Time Warping

- Every possible alignment of Q and C is a warping path, $\vec{p}$

$$\vec{p} = [w_1, \ldots, w_K]$$

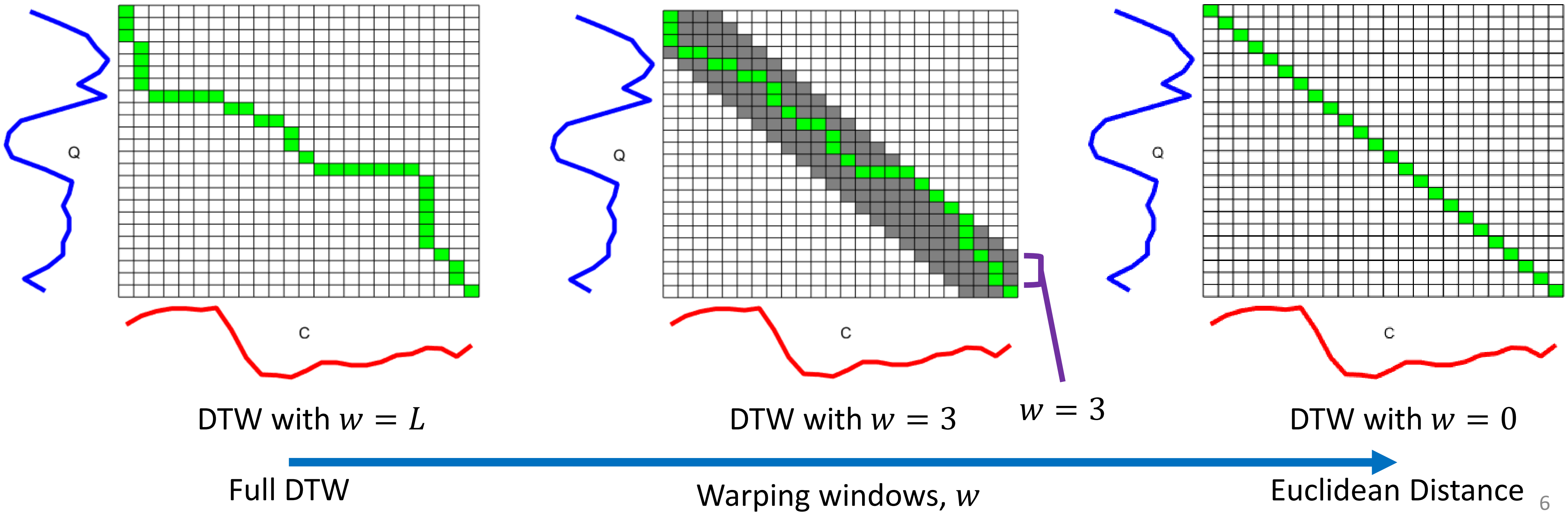
- $w_k = (i, j)$ represents an association of $q_i \leftrightarrow c_j$ aligned by DTW
- $\mathrm{DTW}(Q, C)$ finds the cheapest warping path ("*best*")





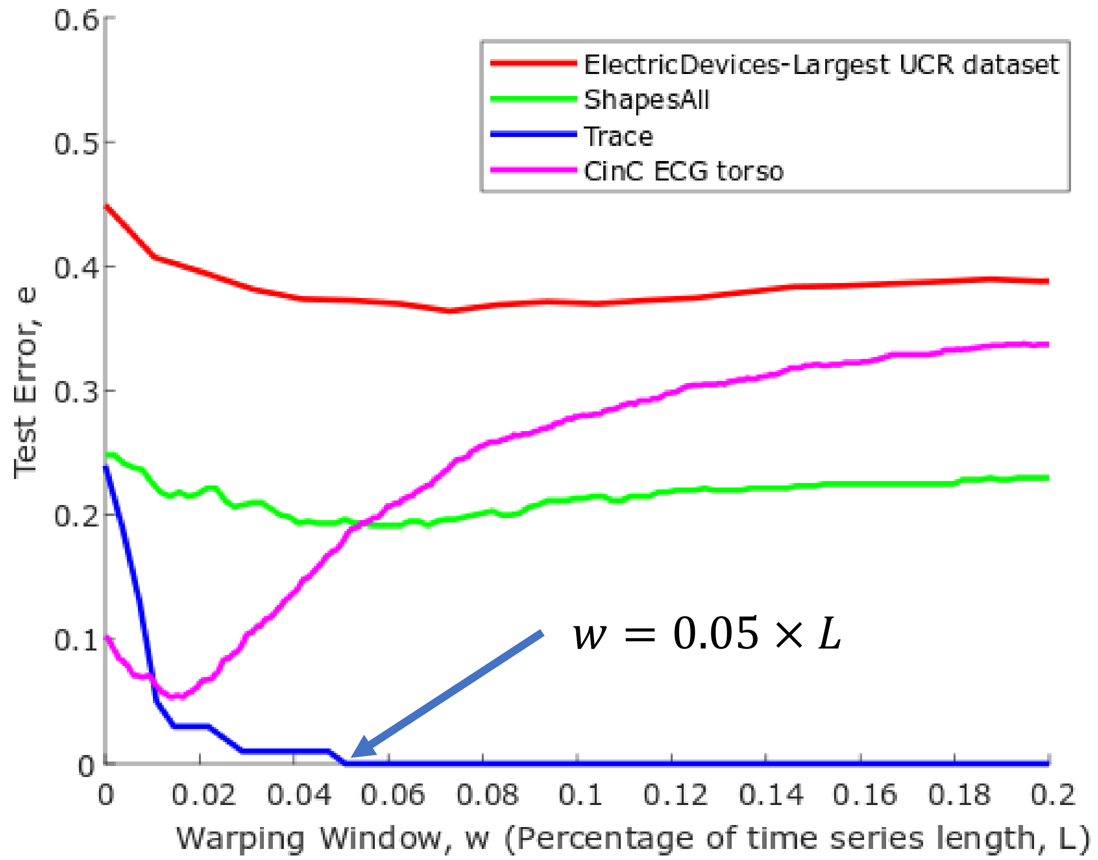
Warping Window

- Warping Window, w is a global constraint on the alignment of DTW such that the elements of Q and C can only be mapped if they are less than w apart, $w = \{0, \dots, L\}$



Why learn the best warping window?

- **Strong** influence on accuracy
  - On **CinC ECG torso** dataset, error rate reduced from 35% to 7%
- **Outperforms** all existing time series classification (TSC) methods
  - **State of the art** – **COTE** and **EE** learn the best warping window for **DTW**
- **Speedup** DTW
  - Smaller w means we don't need to compute the full DTW matrix



Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606-660..

How to learn the best warping window?

```
for w = 0 to L do
    error = 0
    for each s in T do
        nn_s = nn_search(s, T\s, w)
        if nn_s.class ≠ s.class then error++

    if error < bestError then
        bestWW = w
        bestError = error
```

Parameter to NN-DTW algorithm

Leave One Out Cross Validation (LOO-CV)
Can be any ***NN-DTW*** algorithm

Nearest Neighbour – DTW Search

- Naïve DTW Search

$bestDist = \infty$

for each c in T **do**

$dtwDist = DTW(q, c, w)$

if $dtwDist < bestDist$ **then**

$bestDist = dtwDist$

$nnIndex = c.index$

- Lower Bound DTW Search

$bestDist = \infty$

for each c in T **do**

$lbDist = lowerBound(q, c, w)$

if $lbDist < bestDist$ **then**

$dtwDist = DTW(q, c, w)$

if $dtwDist < bestDist$ **then**

$bestDist = dtwDist$

$nnIndex = c.index$

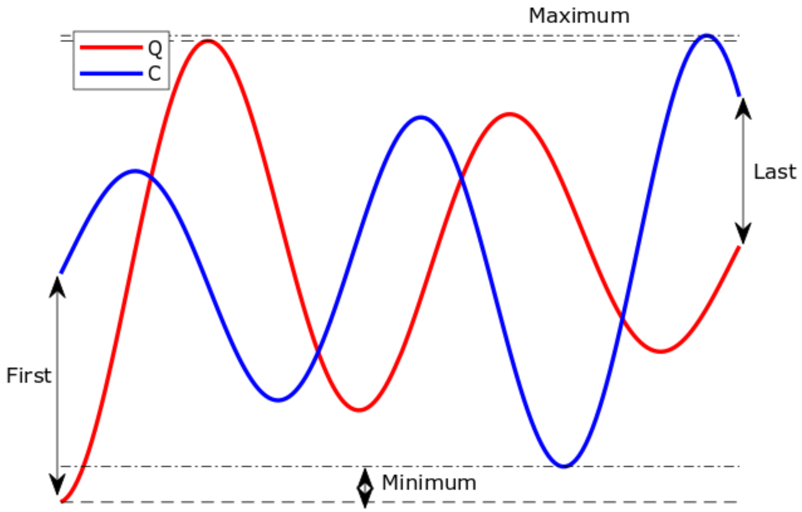
LB Kim
LB Keogh

[1] Kim, S. W., Park, S., & Chu, W. W. (2001). An index-based approach for similarity search supporting time warping in large sequence databases. In *Data Engineering, 2001. Proceedings. 17th International Conference on* (pp. 607-614). IEEE.
[2] Keogh, E., & Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and information systems*, *7*(3), 358-386.

DTW Lower Bounds

- **LB Kim**

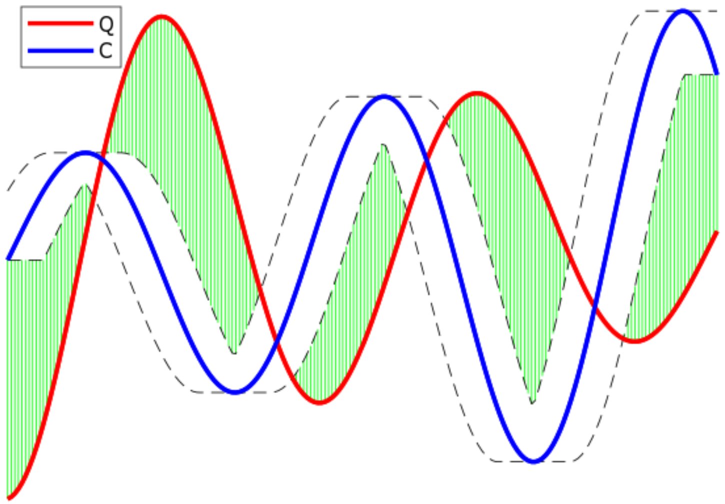
$$\text{LB\_Kim}(Q,C) = \max\begin{cases} |q_1 - c_1| \\ |q_L - c_L| \\ |q_{\max} - c_{\max}| \\ |q_{\min} - c_{\min}| \end{cases}$$



Kim, S. W., Park, S., & Chu, W. W. (2001). An index-based approach for similarity search supporting time warping in large sequence databases. In *Data Engineering, 2001. Proceedings. 17th International Conference on* (pp. 607-614). IEEE.

- **LB Keogh**

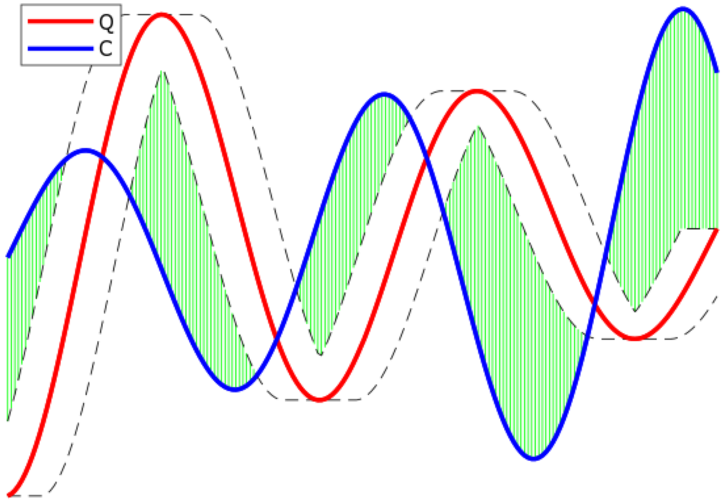
$$\text{LB\_Keogh}_w(Q,C) = \sum_{i=1}^{L}\begin{cases} (q_i - U_i)^2, \text{if } q_i > U_i \\ (q_i - L_i)^2, \text{if } q_i < L_i \\ 0, \text{otherwise} \end{cases}$$



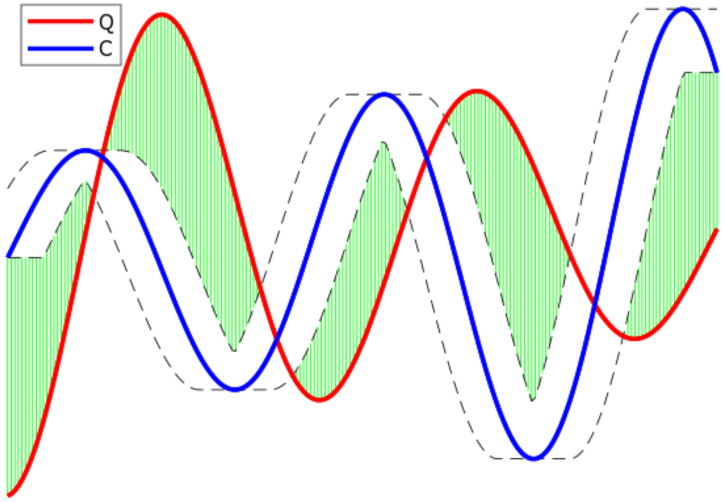
Keogh, E., & Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and information systems*, *7*(3), 358-386.

Reversing Query/Candidate in LB Keogh

Envelope on Q
$\text{LB\_Keogh}_w(Q, C)$



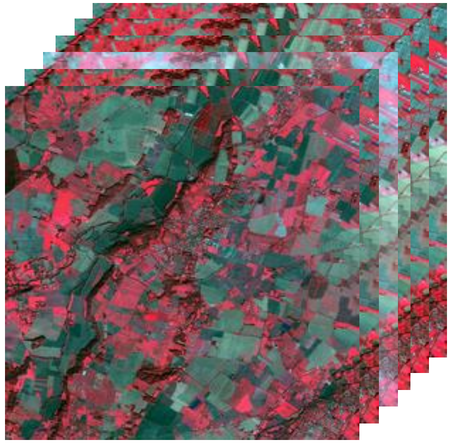
Envelope on C
$\text{LB\_Keogh}_w(C, Q)$



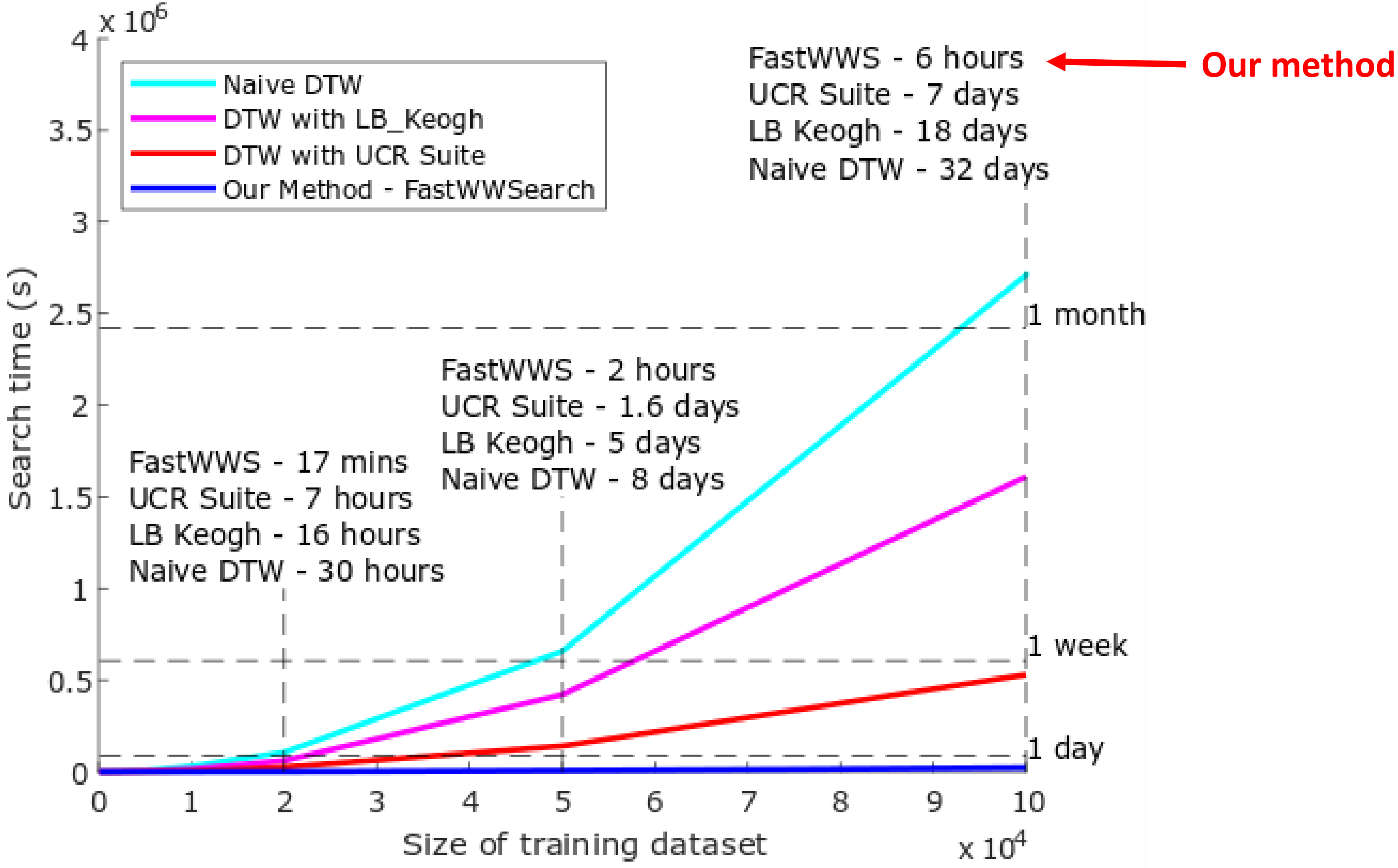
- $\max\left(\text{LB\_Keogh}_w(Q, C), \text{LB\_Keogh}_w(C, Q)\right)$
- Increase tightness of LB Keogh
- Envelopes can be pre-computed
- We will show how we utilised all these “tricks” in our algorithm

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., ... & Keogh, E. (2012, August). Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 262-270). ACM.

Naïve approach learns the best warping window requires $\theta(N^2L^3)$ operations



Satellite Image Time Series
$N = 1{,}000{,}000$
$L = 46$



Efficiently Search for the Best Warping Window of Any Time Series Dataset

Related Methods

- **UCR Suite**
  - Improve efficiency of NN-DTW by **minimising DTW computations**
  - **4 optimisation techniques**
    - Early abandoning Z-Normalisation
    - Reordering early abandoning
    - Reversing query and candidate in LB Keogh
    - Cascading lower bounds
  - Did not use to learn warping window but can be repurposed for this task

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., ... & Keogh, E. (2012, August). Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 262-270). ACM.

- **Pruned DTW**
  - Improve efficiency of DTW
  - Compute an upper bound to **minimise the computations** by skipping the cells **of the cost matrix** that are larger
  - Uses the **DTW value with smaller w as the upper bound** to prune DTW with larger w
  - Improvement for warping window search is minimal

Silva, D. F., & Batista, G. E. (2016, June). Speeding up all-pairwise dynamic time warping matrix calculation. In *Proceedings of the 2016 SIAM International Conference on Data Mining* (pp. 837-845). Society for Industrial and Applied Mathematics.

Fast Warping Window Search for DTW

- a.k.a. **FastWWS** - An exact method
  - **LazyAssessNN**
  - **FastFillNNTable**
- Use links between different values of the loops

These loops are independent

All optimisation in the literature occurs here

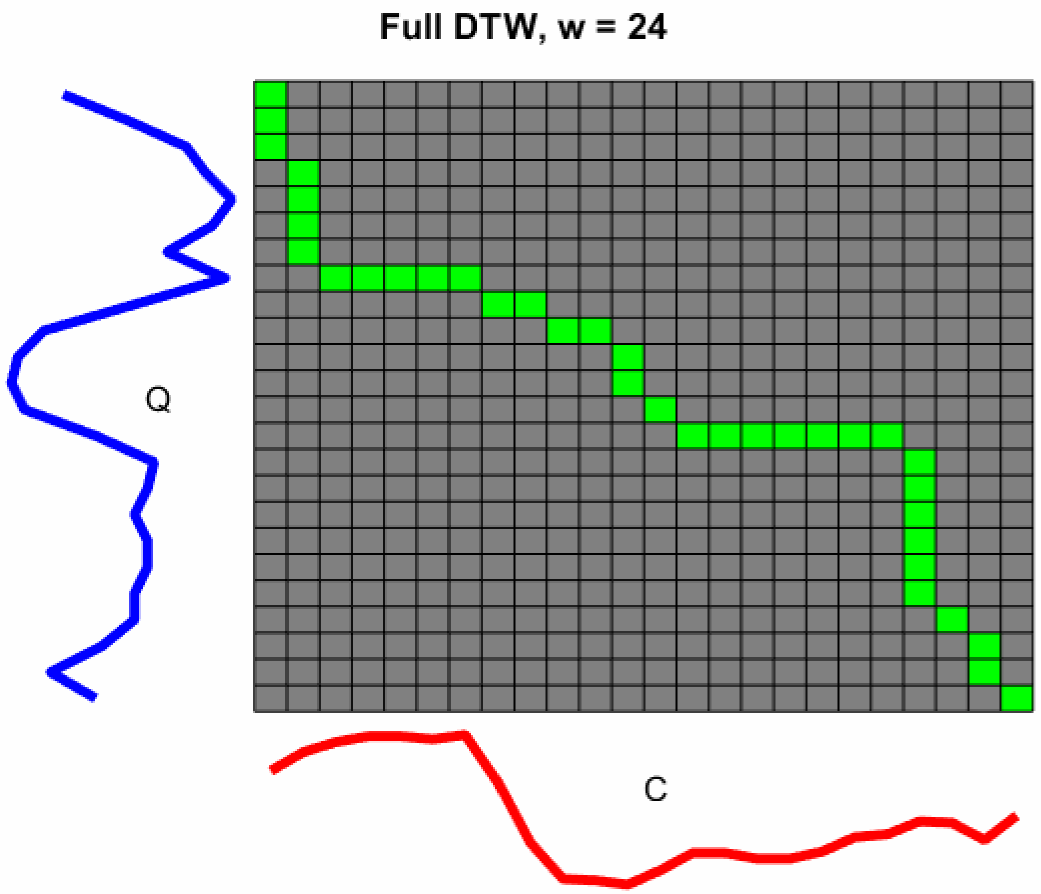
```
for w = 0 to L do
    error = 0
    for each s in T do
        nn_s = nn_search(s, T\s, w)
        if nn_s.class ≠ s.class then error++

    if error < bestError then
        bestWW = w
        bestError = error
```

(1) For each warping window, w
(2) Find the nearest neighbour nn of each time series s in $T\backslash s$

Properties for FastWWS

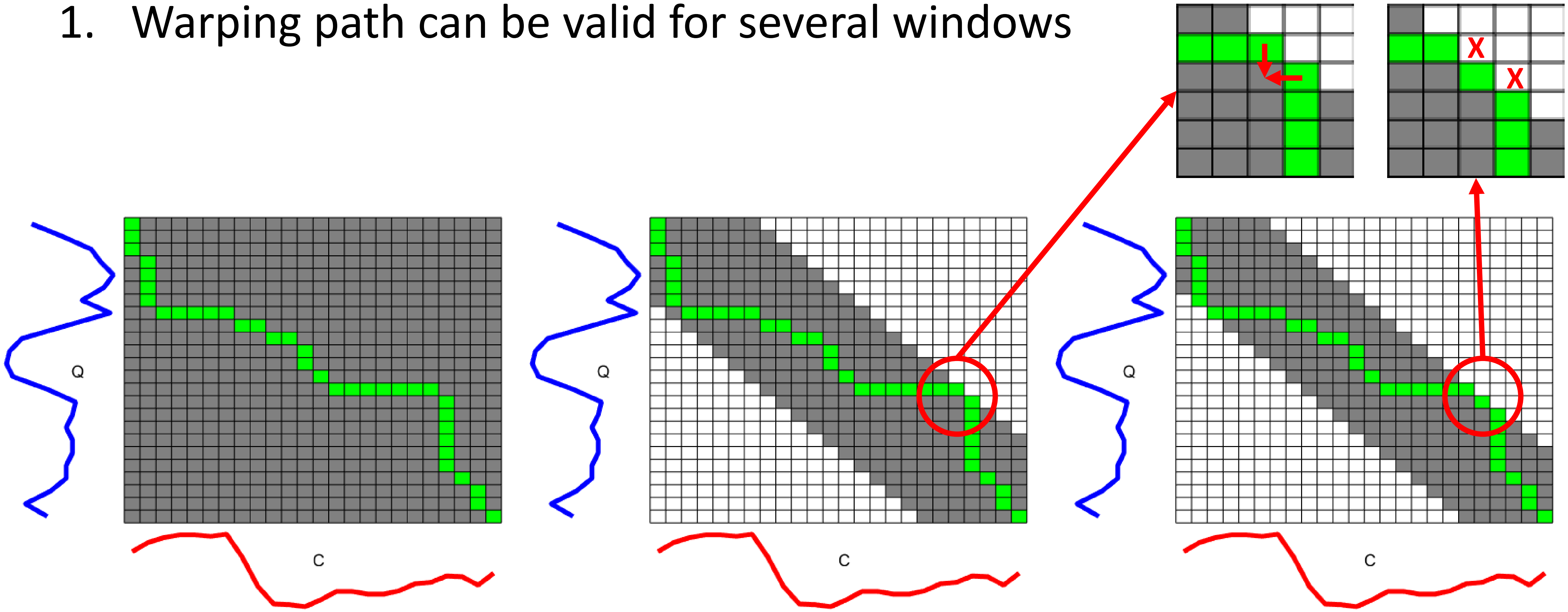
1. Warping path can be valid for several windows
   - w has a "validity"
   - skip computations of all valid w
   - Example:
     - Warping path is valid to $w = 6$
     - $\mathrm{DTW}_{24}(Q, C) = \mathrm{DTW}_6(Q, C)$
     - Skip all DTW from $w = [24, \dots, 6]$

w	...	4	5	6	7	...	23	24
$\mathbf{DTW_w(Q, C)}$	...	8.82	8.36	8.04	8.04	...	8.04	8.04



Properties for FastWWS

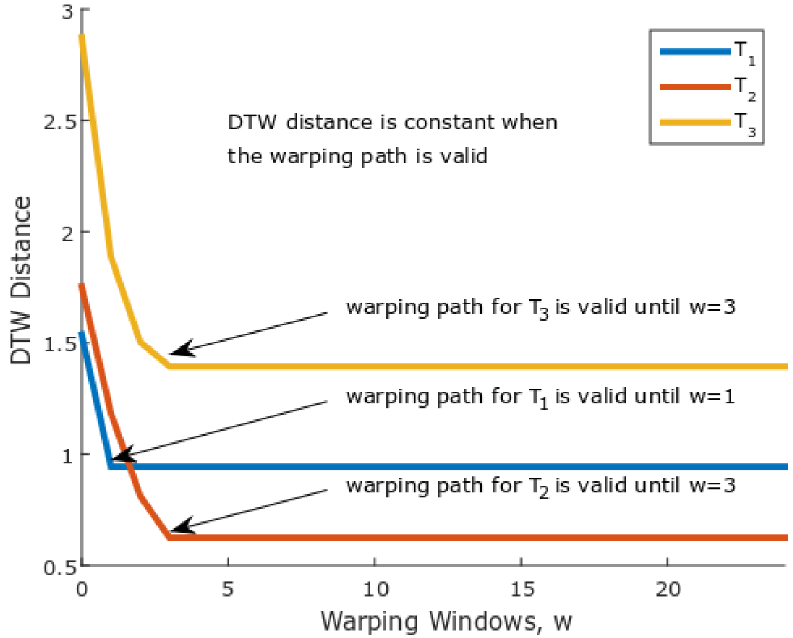
1. Warping path can be valid for several windows



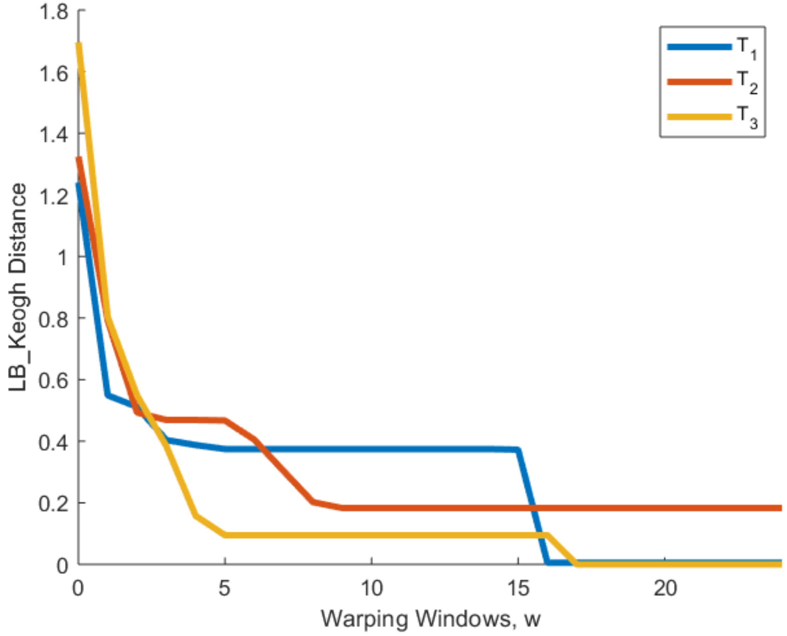
Full DTW, $w = 24$ $w = 6$ $w = 5$

Properties for FastWWS

2. DTW is monotone with warping window
   - $\mathrm{DTW}_w(Q,C) \leq \mathrm{DTW}_{w-1}(Q,C)$



3. LB Keogh is monotone with warping window
   - $\mathrm{LB\_Keogh}_w(Q,C) \leq \mathrm{LB\_Keogh}_{w-1}(Q,C)$



New Lower Bounds to prune Nearest Neighbours before computing $\mathrm{DTW}_w(Q,C)$

$$\mathrm{DTW}_w(Q,C) \geq \mathrm{DTW}_{w+1}(Q,C)$$

$$\mathrm{LB\_Keogh}_w(Q,C) \geq \mathrm{LB\_Keogh}_{w+1}(Q,C) \geq \mathrm{LB\_Kim}(Q,C)$$

FastWWS Intuition

- Efficiently fill up a NN table, giving the nearest neighbour of every time series for all windows
- Naïvely create the table using DTW, requires $\theta(N^2L^3)$ operations

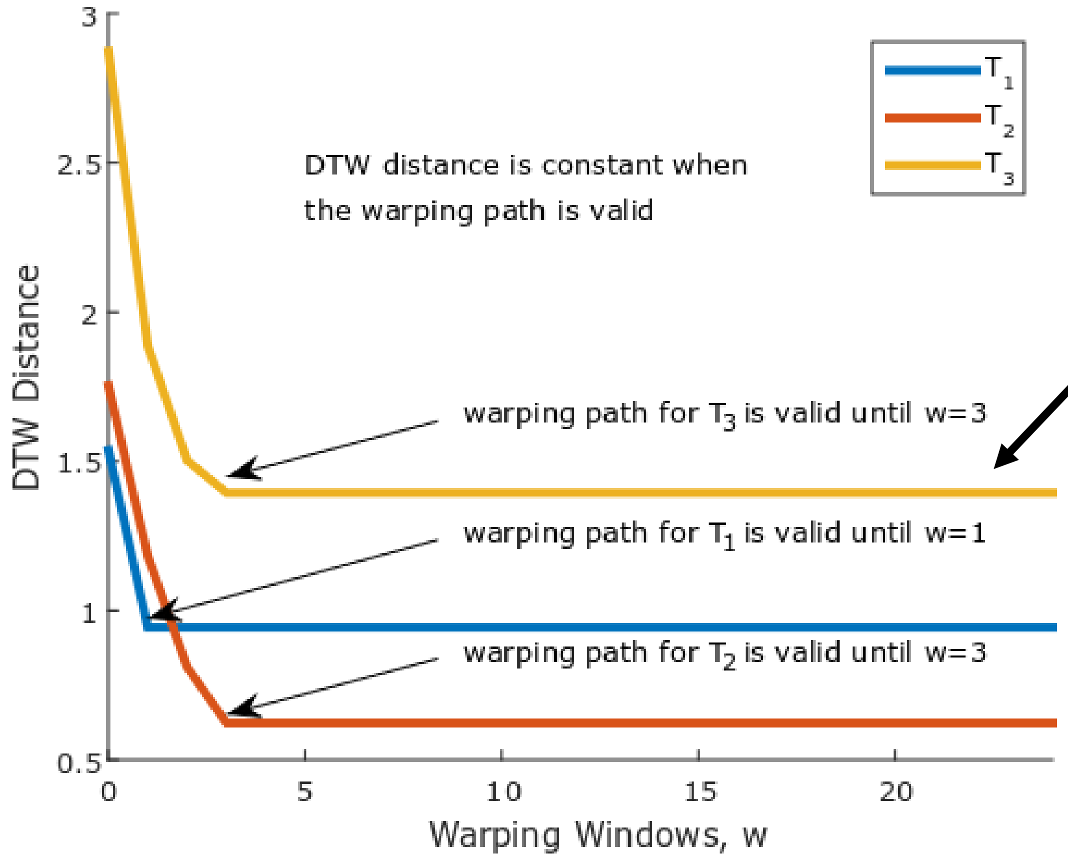
Prior approaches typically go from **smallest** to **largest** with a subset of windows

	Nearest neighbor at warping windows				
	0	1	$\cdots$	$L-2$	$L-1$
T_1	$T_{24}(2.57)$	$T_{55}(0.98)$	$\cdots$	$T_{55}(0.98)$	$T_{55}(0.98)$
$\vdots$			$\vdots$		
T_N	$T_{60}(4.04)$	$T_{47}(1.61)$	$\cdots$	$T_{47}(1.61)$	$T_{47}(1.61)$

FastWWS goes from **largest** to **smallest**, fast enough to test all windows

FastWWS Intuition

- **FastWWS** goes from largest to smallest, applies to all or a subset of windows



- Large window validity for DTW_L (Most of the time)
- No bounds are necessary
- DTW has not changed

Thus obtain DTW_{w+k} and/or $\mathrm{LB\_Keogh}_{w+k}$ for **"FREE"** as the lower bound for DTW_w

<u>Tighter bounds for pruning</u>

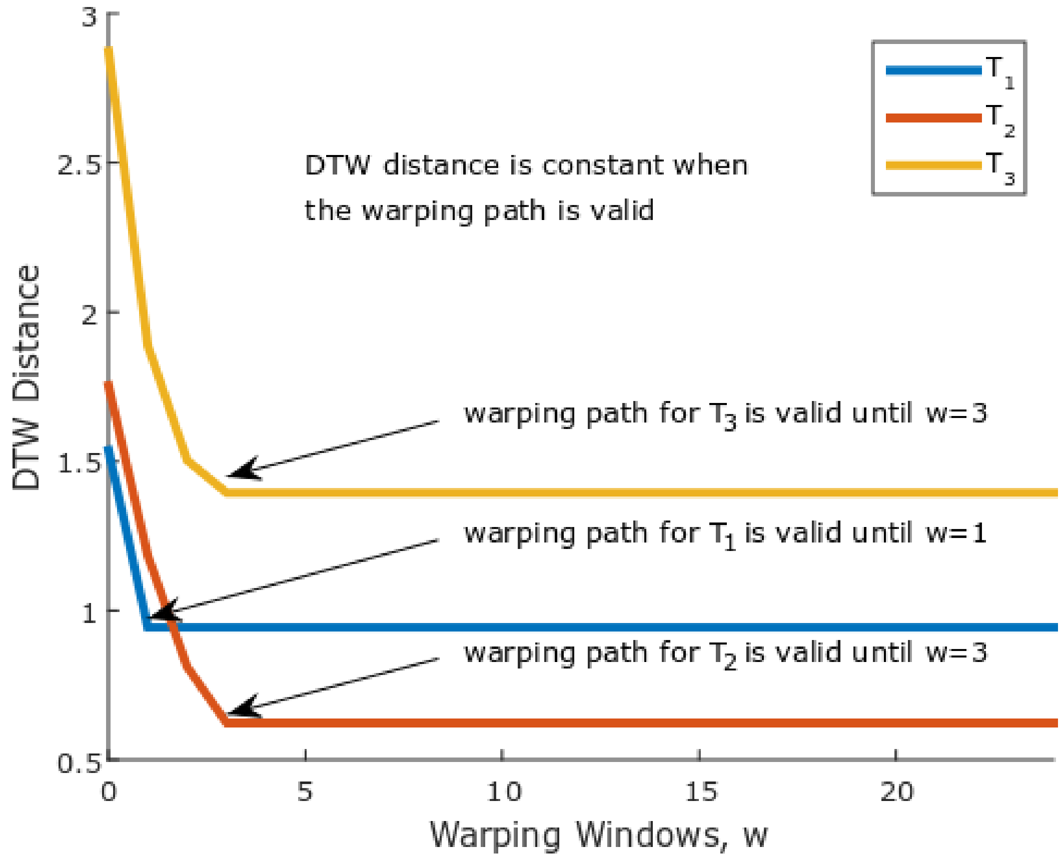
$$\mathrm{DTW}_w(Q,C) \geq \mathrm{DTW}_{w+1}(Q,C)$$

$$\mathrm{LB\_Keogh}_w(Q,C) \geq \mathrm{LB\_Keogh}_{w+1}(Q,C) \geq \mathrm{LB\_Kim}(Q,C)$$

Only use the value at $w+k$ when available, no point in computing DTW_{w+k} for DTW_w

FastWWS Intuition

- **FastWWS** goes from largest to smallest, applies to all or a subset of windows



1. If we find the nearest neighbour for a time series at window, $w = L$ and the warping path is valid to $w = 0$, then we only need to do 1 DTW computation

2. When we calculate $\mathrm{DTW}_w(Q, C)$, even if candidate C is not the nearest neighbour of Q, we do not need to recompute $\mathrm{DTW}_{w'}(Q, C)$ for all windows w' that are valid

Lazy Nearest Neighbour Assessment

- Assess if a pair of time series (Q, C), can be less than distance d for window w
- Postpones calculations for as long as possible
  1. First prune with lower bounds from larger window
  2. Try lower bounds of increasing complexity until
     a. A $\mathrm{LB}_w(Q, C) > d$
     b. Calculated $\mathrm{DTW}_w(Q, C)$
- When w decreases, any value previously calculated for a larger window becomes a lower bound for current w, stored in a Cache, $\mathcal{C}_{(Q,C)}$

$\mathbf{DTW}_{w+1}$
$\mathbf{LB\_Keogh}_{w+1}$

LB Kim
LB Keogh

LazyAssessNN Algorithm

1. First do LB Kim if hasn't been done

if $\text{cache}_{Q,C}$ is empty **do** $\text{cache}_{Q,C} = \boxed{\text{LB\_Kim}(Q,C)}$
if $\text{cache}_{Q,C}.\text{stoppedAt} == \text{DTW}_{w+k}$ **and** w is valid **then**
  if $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByDTW **else return** $\text{cache}_{Q,C}.\text{value}$
if $\text{cache}_{Q,C}.\text{stoppedAt} == \text{LB\_Kim}$ **or** LB_Keogh_{w+k} **then**
  if $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByLB

$\text{cache}_{Q,C} = \text{LB\_Keogh}_w(Q,C)$ **if** $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByLB
$\text{cache}_{Q,C} = \text{LB\_Keogh}_w(C,Q)$ **if** $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByLB
$\text{cache}_{Q,C} = \text{DTW}_w(C,Q)$ **if** $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByDTW
return $\text{cache}_{Q,C}.\text{value}$

LazyAssessNN Algorithm

2. Check lower bounds from previous window

DTW and LB Keogh from larger window (property 2 & 3)

if $\text{cache}_{Q,C}$ is empty **do** $\text{cache}_{Q,C} = \text{LB\_Kim}(Q, C)$
if $\text{cache}_{Q,C}.\text{stoppedAt} == \boxed{\text{DTW}_{w+k}}$ **and** w is valid **then**
  if $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByDTW **else return** $\text{cache}_{Q,C}.\text{value}$
if $\text{cache}_{Q,C}.\text{stoppedAt} == \text{LB\_Kim}$ **or** $\boxed{\text{LB\_Keogh}_{w+k}}$ **then**
  if $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByLB

$\text{cache}_{Q,C} = \text{LB\_Keogh}_w(Q, C)$ **if** $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByLB
$\text{cache}_{Q,C} = \text{LB\_Keogh}_w(C, Q)$ **if** $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByLB
$\text{cache}_{Q,C} = \text{DTW}_w(C, Q)$ **if** $\text{cache}_{Q,C}.\text{value} \geq d$ **return** prunedByDTW
return $\text{cache}_{Q,C}.\text{value}$

LazyAssessNN Algorithm

4. If current window w is not valid

3. Use DTW from previous window $(w + k)$ if current window w still valid (property 1)

```
if cache_{Q,C} is empty do cache_{Q,C} = LB_Kim(Q,C)
if cache_{Q,C}.stoppedAt == DTW_{w+k} and w is valid then
   if cache_{Q,C}.value ≥ d return prunedByDTW else return cache_{Q,C}.value
if cache_{Q,C}.stoppedAt == LB_Kim or LB_Keogh_{w+k} then
   if cache_{Q,C}.value ≥ d return prunedByLB

cache_{Q,C} = LB_Keogh_w(Q,C) if cache_{Q,C}.value ≥ d return prunedByLB
cache_{Q,C} = LB_Keogh_w(C,Q) if cache_{Q,C}.value ≥ d return prunedByLB
cache_{Q,C} = DTW_w(C,Q) if cache_{Q,C}.value ≥ d return prunedByDTW
return cache_{Q,C}.value
```

- Next call to LazyAssessNN will be with a smaller w
- Possible to use Early Abandon on LB_Keogh and LB_Improved [1]

[1] Lemire, D. (2009). Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern recognition, 42(9), 2169-2180.

Fast Fill the Nearest Neighbour Table

$\text{NN.fillAll}(\_,\infty)\ \forall\{w,N\}$ ← Initialise NN table with ∞ NN distance
for $s \leftarrow 2$ to N **do** ← Start with second series
  for $w \leftarrow L-1$ down to 0 **do** ← Start from largest window
    if $\text{NN}_w^{T_s} \neq \emptyset$ **then** ← a. Check if NN for T_s exist at this window
      for $t \leftarrow 1$ to $s-1$ **do** ← a. Update NN for all previous series
        $\text{res} = \textbf{LazyAssessNN}(T_s, T_t, w, \text{NN}_w^{T_s})$ **if** res not pruned **then** $\text{NN}_w^{T_s} = (T_t, \text{res})$
    else
      for $t \leftarrow 1$ to $s-1$ **do**
        $\text{res} = \textbf{LazyAssessNN}(T_s, T_t, w, \text{NN}_w^{T_s})$ **if** res not pruned **then** $\text{NN}_w^{T_s} = (T_t, \text{res})$
        $\text{res} = \textbf{LazyAssessNN}(T_s, T_t, w, \text{NN}_w^{T_t})$ **if** res not pruned **then** $\text{NN}_w^{T_t} = (T_s, \text{res})$
    for $w' \in \text{NN}_w^{T_s}.\text{valid}$ **do** $\text{NN}_{w'}^{T_s} = \text{NN}_w^{T_s}$ ← d. Propagate NN for all valid windows

b. Find NN for current series
c. Check if current series T_s is NN for previous series

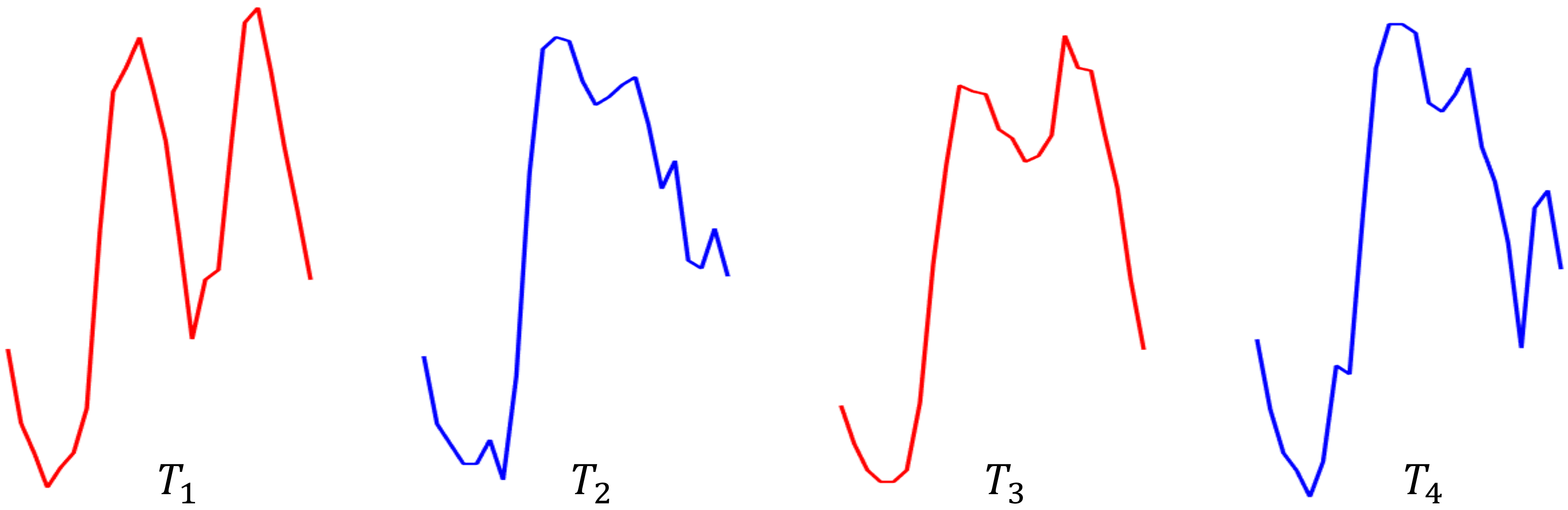
Fast Fill the Nearest Neighbour Table

- Build table for a subset $T' \subseteq T$ of increasing size until $T' = T$

1. Start with 2 time series $T' = \{T_1, T_2\}$ and fill the table as if T' is the entire dataset, starting from $w = L - 1$ to $\mathrm{w} = 0$
   - T_2 is the nearest neighbour of T_1 and vice versa
2. Add a third time series T_3 from $T \backslash \mathrm{T}'$ to T', $T' = \{T_1, T_2, T_3\}$
   a. Check if nearest neighbour exists for T_3
   b. Find the nearest neighbour of T_3 within $T' \backslash \mathrm{T}_3 = \{T_1, T_2\}$
   c. Check if T_3 is the nearest neighbour of T_1 and/or T_2
   d. Propagate nearest neighbour of T_3 for all valid windows
3. Repeat step 2 with the next time series, T_n in $T \backslash \mathrm{T}'$ until $T' = T$

FastWWS Example

- Let T be a training dataset of 4 time series, $T = \{T_1, T_2, T_3, T_4\}$
- Length of each time series is $L = 24$



Cache	StoppedAt	Value
cache_{T_1,T_2}	LB_Kim	0.040

FastWWS Example

Precompute LB Kim

Reference: $\text{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	...	22	23
T_1	∞	∞	∞	∞	∞	∞	...	∞	T_2(5, 4.254)
T_2	∞	∞	∞	∞	∞	T_1(5, 4.254)	...	T_1(5, 4.254)	T_1(5, 4.254)

Propagate $\text{NN}_w^{T_2}$ across $w = 22$ to 5

1. Initialise **Cache** & **NN Table** with ∞ NN distance, $\text{NN.fillAll}(\_,\infty)\ \ \forall\{w, N\}$
2. Start with $T' = \{T_1, T_2\}$, $w = 23$, $d_{\text{NN}} = \infty$ and Query: T_2, Candidate: T_1
   - $\textbf{LazyAssessNN}(T_1, T_2, 23, \infty)$:
     - $\text{cache}_{T_1,T_2} = \text{LB\_Kim}(T_1, T_2) = 0.040 < \infty$ **continue**
     - Compute $\text{cache}_{T_1,T_2} = \text{LB\_Keogh}_{23}(T_1, T_2) = 0.000 < \infty$ **continue**
     - Compute $\text{cache}_{T_1,T_2} = \text{LB\_Keogh}_{23}(T_2, T_1) = 0.046 < \infty$ **continue**
     - Compute $\text{cache}_{T_1,T_2} = \text{DTW}_{23}(T_1, T_2) = \{\text{validTill} = 5, 4.254\} < \infty$ **return** $\text{cache}_{T_1,T_2}.\text{value}$
   - Assign T_1 as the Nearest Neighbour for T_2 at $w = 23$ and vice versa for T_1
   - Propagate Nearest Neighbour of T_2 at $w = 23$ for $w = 22$ to 5

Cache	StoppedAt	Value
cache_{T_1,T_2}	DTW_{23}	4.254

Update cache every time we compute a distance

FastWWS Example

Reference: $\mathrm{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	...	22	23
T_1	∞	∞	∞	∞	∞	T_2(5, 4.254)	...	T_2(5, 4.254)	T_2(5, 4.254)
T_2	∞	∞	∞	∞	∞	T_1(5, 4.254)	...	T_1(5, 4.254)	T_1(5, 4.254)

3. Continue with $w = 22, d_{\mathrm{NN}} = 4.254$ and Query: T_2, Candidate: T_1
   - Since we have NN for T_2 at $w = 22$, we have to check if T_2 is NN of T_1
   - $\mathbf{LazyAssessNN}(T_1, T_2, 22, \infty)$:
     - $\mathrm{cache}_{T_1,T_2}.\mathrm{stoppedAt} == \mathrm{DTW}_{23}$ **and** $w = 22$ is valid
     - $\mathrm{cache}_{T_1,T_2}.\mathrm{value} = 4.254 < \infty$ **return** $\mathrm{cache}_{T_1,T_2}.\mathrm{value}$
   - Assign T_2 as the Nearest Neighbour for T_1 at $w = 22$
4. Repeat step 3 for all windows, $w \in \{21, \dots, 5\}$

$w = 22$ is still valid
$\therefore \mathrm{DTW}_{22}(T_1, T_2)$
$= \mathrm{DTW}_{23}(T_1, T_2)$
$= 4.254$

Cache	StoppedAt	Value
cache_{T_1,T_2}	DTW_5	4.254

FastWWS Example

Reference: $\text{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	...	22	23
T_1	T_2(0, 11.89)	T_2(1, 8.972)	T_2(2, 7.341)	T_2(3, 6.243)	T_2(4, 4.814)	T_2(5, 4.254)	...	T_2(5, 4.254)	T_2(5, 4.254)
T_2	T_1(0, 11.89)	T_1(1, 8.972)	T_1(2, 7.341)	T_1(3, 6.243)	T_1(4, 4.814)	T_1(5, 4.254)	...	T_1(5, 4.254)	T_1(5, 4.254)

$w = 4$ is not valid, recompute DTW if necessary

Cannot propagate NN as window is only valid for $w = 4$

5. Continue with $w = 4, d_{\text{NN}} = \infty$ and Query: T_2, Candidate: T_1
   - $\textbf{LazyAssessNN}(T_1, T_2, 4, \infty)$:
     - $\text{cache}_{T_1,T_2}.\text{stoppedAt} == \text{DTW}_5$ **and** $w = 4$ is not valid
     - $\text{cache}_{T_1,T_2}.\text{value} = 4.254 < \infty$ **continue**
     - Compute $\text{cache}_{T_1,T_2} = \text{LB\_Keogh}_4(T_1, T_2) = 0.000 < \infty$ **continue**
     - Compute $\text{cache}_{T_1,T_2} = \text{LB\_Keogh}_4(T_2, T_1) = 2.076 < \infty$ **continue**
     - Compute $\text{cache}_{T_1,T_2} = \text{DTW}_4(T_1, T_2) = \{\text{validTill} = 4, 4.814\} < \infty$ **return** $\text{cache}_{T_1,T_2}.\text{value}$
   - Assign T_1 as the Nearest Neighbour for T_2 at $w = 4$ and vice versa for T_1
6. Repeat step 5 for all windows, $w \in \{3, \dots, 0\}$

Cache	StoppedAt	Value
cache_{T_1,T_2}	DTW_0	11.89
cache_{T_1,T_3}	LB_Kim	0.361
cache_{T_2,T_3}	LB_Kim	0.317

FastWWS Example

Reference: $\text{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	…	22	23
T_1	T_2(0, 11.89)	T_2(1, 8.972)	T_2(2, 7.341)	T_2(3, 6.243)	T_2(4, 4.814)	T_2(5, 4.254)	…	T_2(5, 4.254)	T_2(5, 4.254)
T_2	T_1(0, 11.89)	T_1(1, 8.972)	T_1(2, 7.341)	T_1(3, 6.243)	T_1(4, 4.814)	T_1(5, 4.254)	…	T_1(5, 4.254)	T_1(5, 4.254)
T_3	∞	∞	∞	∞	∞	∞	…	∞	∞

7. Add T_3, $T' = \{T_1, T_2, T_3\}$
   - $\text{cache}_{T_1,T_3} = \text{LB\_Kim}(T_1, T_3) = 0.361 < \infty$
   - $\text{cache}_{T_2,T_3} = \text{LB\_Kim}(T_2, T_3) = 0.317 < \infty$
   - Since $\text{LB\_Kim}(T_2, T_3) < \text{LB\_Kim}(T_1, T_3)$, start with (T_2, T_3) pair

When adding a new series, initialise the row to ∞ - meaning no NN candidate yet

Cache	StoppedAt	Value
cache_{T_1,T_2}	DTW_0	11.89
cache_{T_1,T_3}	LB_Kim	0.361
cache_{T_2,T_3}	LB_Kim	0.317

FastWWS Example

$\mathrm{DTW}_{23}(T_2, T_3) = 1.612 < 4.254$

Update $\mathrm{NN}_{23}^{T_2} = T_3$

Reference: $\mathrm{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	…	22	23
T_1	T_2(0, 11.89)	T_2(1, 8.972)	T_2(2, 7.341)	T_2(3, 6.243)	T_2(4, 4.814)	T_2(5, 4.254)	…	T_2(5, 4.254)	T_2(5, 4.254)
T_2	T_1(0, 11.89)	T_1(1, 8.972)	T_1(2, 7.341)	T_1(3, 6.243)	T_1(4, 4.814)	T_1(5, 4.254)	…	T_1(5, 4.254)	T_3(4, 1.612)
T_3	∞	∞	∞	∞	∞	∞	…	∞	T_2(4, 1.612)

Nearest Neighbour for T_3 is T_2

8. For $T_2, T_3, w = 23, d_{\mathrm{NN}} = \infty$ and Query: T_3, Candidate: T_2
   - **LazyAssessNN**$(T_2, T_3, 23, \infty)$:
     - $\mathrm{cache}_{T_2,T_3}.\mathrm{value} = 0.317 < \infty$ **continue**
     - Compute $\mathrm{cache}_{T_2,T_3} = \mathrm{LB\_Keogh}_{23}(T_2, T_3) = 0.000 < \infty$ **continue**
     - Compute $\mathrm{cache}_{T_2,T_3} = \mathrm{LB\_Keogh}_{23}(T_2, T_3) = 0.000 < \infty$ **continue**
     - Compute $\mathrm{cache}_{T_2,T_3} = \mathrm{DTW}_{23}(T_2, T_3) = \{\mathrm{validTill} = 4, 1.612\} < \infty$ **return** $\mathrm{cache.value}$
   - Assign T_2 as the Nearest Neighbour for T_3 at $w = 23$
   - Since $\mathrm{DTW}_{23}(T_2, T_3) = 1.612 < \mathrm{DTW}_{23}(T_1, T_2) = 4.254$, Update T_3 as the Nearest Neighbour for T_2 at $w = 23$

FastWWS Example

Cache	StoppedAt	Value
cache_{T_1,T_2}	DTW_0	11.89
cache_{T_1,T_3}	LB_Kim	0.361
cache_{T_2,T_3}	DTW_{23}	1.612

$\text{DTW}_{23}(T_1, T_3) = 3.326 < 4.254$
Update $\text{NN}_{23}^{T_1} = T_3$

Reference: $\text{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	…	22	23
T_1	T_2(0, 11.89)	T_2(1, 8.972)	T_2(2, 7.341)	T_2(3, 6.243)	T_2(4, 4.814)	T_2(5, 4.254)	…	T_2(5, 4.254)	T_3(2, 3.326)
T_2	T_1(0, 11.89)	T_1(1, 8.972)	T_1(2, 7.341)	T_1(3, 6.243)	T_1(4, 4.814)	T_1(5, 4.254)	…	T_1(5, 4.254)	T_3(4, 1.612)
T_3	∞	∞	∞	∞	∞	∞	…	∞	T_2(4, 1.612)

$\text{DTW}_{23}(T_1, T_3) = 3.326 \geq 1.612$
No change to $\text{NN}_{23}^{T_3}$

9. For T_1, T_3 ,$d_{\text{NN}} = 1.612$, $\text{DTW}_{23}(T_1, T_2) = 4.254$ and Query: T_3, Candidate: T_1
   - **LazyAssessNN**$(T_1, T_3, 23, 1.612)$:
     - $\text{cache}_{T_1,T_3}.\text{value} = 0.361 < 1.612$ **continue**
     - Compute $\text{cache}_{T_1,T_3} = \text{LB\_Keogh}_{23}(T_1, T_3) = 0.000 < 1.612$ **continue**
     - Compute $\text{cache}_{T_1,T_3} = \text{LB\_Keogh}_{23}(T_1, T_3) = 0.039 < 1.612$ **continue**
     - Compute $\text{cache}_{T_1,T_3} = \text{DTW}_{23}(T_1, T_3) = \{\text{validTill} = 2, 3.326\} \geq 1.612$ **return** prunedByDTW
   - No change to Nearest Neighbour for T_3 at $w = 23$
   - Since $\text{DTW}_{23}(T_1, T_3) = 3.326 < \text{DTW}_{23}(T_1, T_2) = 4.254$, Update T_3 as the Nearest Neighbour for T_1 at $w = 23$

Cache	StoppedAt	Value
cache_{T_1,T_2}	DTW_0	11.89
cache_{T_1,T_3}	DTW_{23}	3.326
cache_{T_2,T_3}	DTW_{23}	1.612

FastWWS Example

Reference: $\text{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	...	22	23
T_1	T_2(0, 11.89)	T_2(1, 8.972)	T_2(2, 7.341)	T_2(3, 6.243)	T_3(2, 3.326)	T_3(2, 3.326)	...	T_3(2, 3.326)	T_3(2, 3.326)
T_2	T_1(0, 11.89)	T_1(1, 8.972)	T_1(2, 7.341)	T_1(3, 6.243)	T_3(4, 1.612)	T_3(4, 1.612)	...	T_3(4, 1.612)	T_3(4, 1.612)
T_3	∞	∞	∞	∞	T_2(4, 1.612)	T_2(4, 1.612)	...	T_2(4, 1.612)	T_2(4, 1.612)

10. Now we are sure about $\text{NN}_{23}^{T_1}$, $\text{NN}_{23}^{T_2}$ and $\text{NN}_{23}^{T_3}$
    - We can update NN for T_1, T_2, T_3 for $w = 22$ to 4 since $\text{NN}_{23}^{T_3}$ is valid until $w = 4$
    - $\text{NN}_{23}^{T_1}$ is valid until $w = 2$ and will be updated later when we move on to $w = 2$
    - Since $\text{DTW}_{23}(T_2, T_3) = 1.612 < \text{DTW}_{23}(T_1, T_3) = 3.326$, start with (T_2, T_3) pair for $w = 3$
    - $\text{DTW}_4(T_1, T_3) = \text{DTW}_{23}(T_1, T_3)$
    - $\text{DTW}_4(T_2, T_3) = \text{DTW}_{23}(T_2, T_3)$

Propagate $\text{NN}_w^{T_3}$ and update $\text{NN}_w^{T_1}$, $\text{NN}_w^{T_2}$ across $w = 22$ to 4

Cache	StoppedAt	Value
cache_{T_1,T_2}	DTW_0	11.89
cache_{T_1,T_3}	DTW_4	3.326
cache_{T_2,T_3}	DTW_4	1.612

FastWWS Example

Reference: $\text{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	...	22	23
T_1	T_3(0, 4.911)	T_3(1, 3.486)	T_3(2, 3.326)	T_3(2, 3.326)	T_3(2, 3.326)	T_3(2, 3.326)	...	T_3(2, 3.326)	T_3(2, 3.326)
T_2	T_3(0, 4.395)	T_3(1, 2.598)	T_3(2, 1.882)	T_3(3, 1.614)	T_3(4, 1.612)	T_3(4, 1.612)	...	T_3(4, 1.612)	T_3(4, 1.612)
T_3	T_2(0, 4.395)	T_2(1, 2.598)	T_2(2, 1.882)	T_2(3, 1.614)	T_2(4, 1.612)	T_2(4, 1.612)	...	T_2(4, 1.612)	T_2(4, 1.612)

$w = 3$ is not valid, recompute DTW if necessary

11. For T_2, T_3 continue with $w = 3, d_{\text{NN}} = \infty$ and Query: T_3, Candidates: T_2
    - $\textbf{LazyAssessNN}(T_2, T_3, 3, \infty)$:
        - $\text{cache}_{T_2,T_3}.\text{stoppedAt} == \text{DTW}_4$ **and** $w = 3$ is not valid
        - $\text{cache}_{T_2,T_3} = 1.612 < \infty$ **continue**
        - Compute $\text{cache}_{T_2,T_3} = \text{LB\_Keogh}_3(T_2, T_3) = 0.421 < \infty$ **continue**
        - Compute $\text{cache}_{T_2,T_3} = \text{LB\_Keogh}_3(T_3, T_2) = 0.328 < \infty$ **continue**
        - Compute $\text{cache}_{T_2,T_3} = \text{DTW}_3(T_2, T_3) = \{\text{validTill} = 3, 1.614\} < \infty$ **return** cache.value
    - Assign T_2 as the Nearest Neighbour for T_3 at $w = 3$
    - Since $\text{DTW}_3(T_2, T_3) = 1.614 < \text{DTW}_3(T_1, T_2) = 6.243$, Update T_3 as the Nearest Neighbour for T_2 at $w = 3$
12. Repeat the algorithm for all windows, $w \in \{2, \dots, 0\}$

FastWWS Example

Reference: $\mathrm{NN}_w^{T_s}$(window validity, d_{NN})

w	0	1	2	3	4	5	...	22	23
T_1	T_3(0, 4.911)	T_3(1, 3.486)	T_3(2, 3.326)	T_3(2, 3.326)	T_3(2, 3.326)	T_3(2, 3.326)	...	T_3(2, 3.326)	T_3(2, 3.326)
T_2	T_4(0, 1.658)	T_4(1, 0.632)	T_4(2, 0.620)	T_4(3, 0.599)	T_4(3, 0.599)	T_4(3, 0.599)	...	T_4(3, 0.599)	T_4(3, 0.599)
T_3	T_2(0, 4.395)	T_2(1, 2.598)	T_2(2, 1.882)	T_2(3, 1.614)	T_2(4, 1.612)	T_2(4, 1.612)	...	T_2(4, 1.612)	T_2(4, 1.612)
T_4	T_2(0, 1.658)	T_2(1, 0.632)	T_2(2, 0.620)	T_2(3, 0.599)	T_2(3, 0.599)	T_2(3, 0.599)	...	T_2(3, 0.599)	T_2(3, 0.599)

13. Continue adding T_4 to T' and repeat previous steps until $T' = T = \{T_1, T_2, T_3, T_4\}$

FastWWS Example

w	0	1	2	3	4	5	...	22	23
T_1	T_3	T_3	T_3	T_3	T_3	T_3	...	T_3	T_3
T_2	T_4	T_4	T_4	T_4	T_4	T_4	...	T_4	T_4
T_3	T_2	T_2	T_2	T_2	T_2	T_2	...	T_2	T_2
T_4	T_2	T_2	T_2	T_2	T_2	T_2	...	T_2	T_2
Acc	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**...**	**0.75**	**0.75**

14. Classify every instance for each window in one pass of the table
    - Yields the best window at $\boldsymbol{w = 0}$ with LOO-CV accuracy of **0.75**

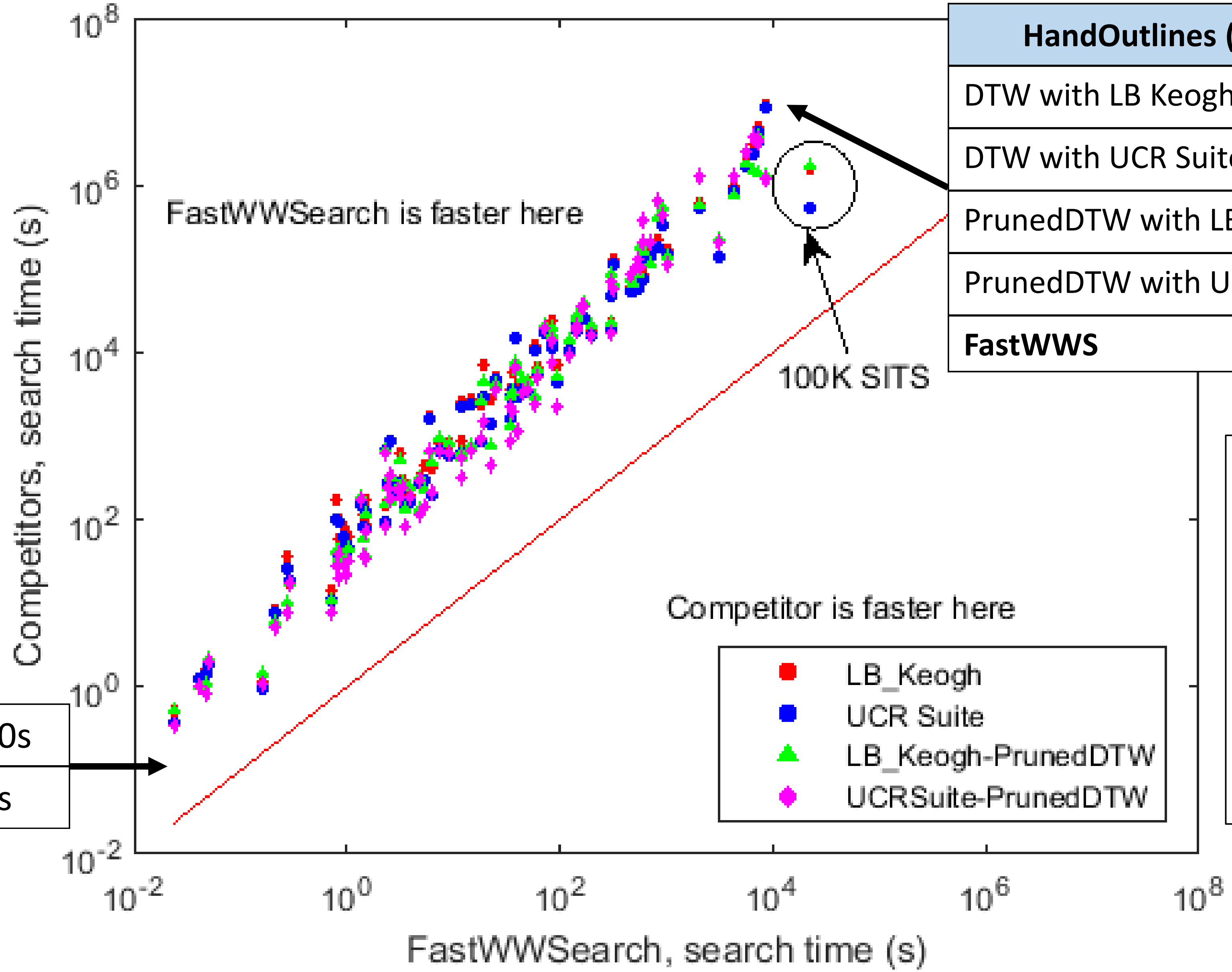
Experimental Evaluation

- Evaluate the efficiency of FastWWS
  - LOO-CV with NN Search
    1. DTW with LB Keogh (Baseline)
    2. UCR Suite
    3. Pruned DTW with LB Keogh
    4. UCR Suite with Pruned DTW
  - LOO-CV with FastWWS
- Exhaustive search on all methods
- Average results over 10 runs for different reshuffling of T
- 85 benchmark time series datasets http://www.cs.ucr.edu/~eamonn/time_series_data/

for $w = 0 \text{ to } L$ **do**
    $error = 0$
    for each $s \text{ in } T$ **do**
        $nn_s = \text{nn\_search}(s, T \backslash \text{s}, w)$
        if $nn_s.class \neq s.class$ **then** $error{+}{+}$

    if $error < bestError$ **then**
        $bestWW = w$
        $bestError = error$

FastWWS is FASTER and more EFFICIENT than all known methods!



HandOutlines ($L = 2709$) - 1000x	
DTW with LB Keogh	105 days
DTW with UCR Suite	100 days
PrunedDTW with LB Keogh	15 days
PrunedDTW with UCR Suite	14 days
FastWWS	**2.5 hours**

State of the arts:	10s
FastWWS:	1s

Average speed up

On the N^2 Term

$N \leq 200 \Rightarrow 106\text{x}$

$\boldsymbol{N > 200 \Rightarrow 184\text{x}}$

On the L^3 Term

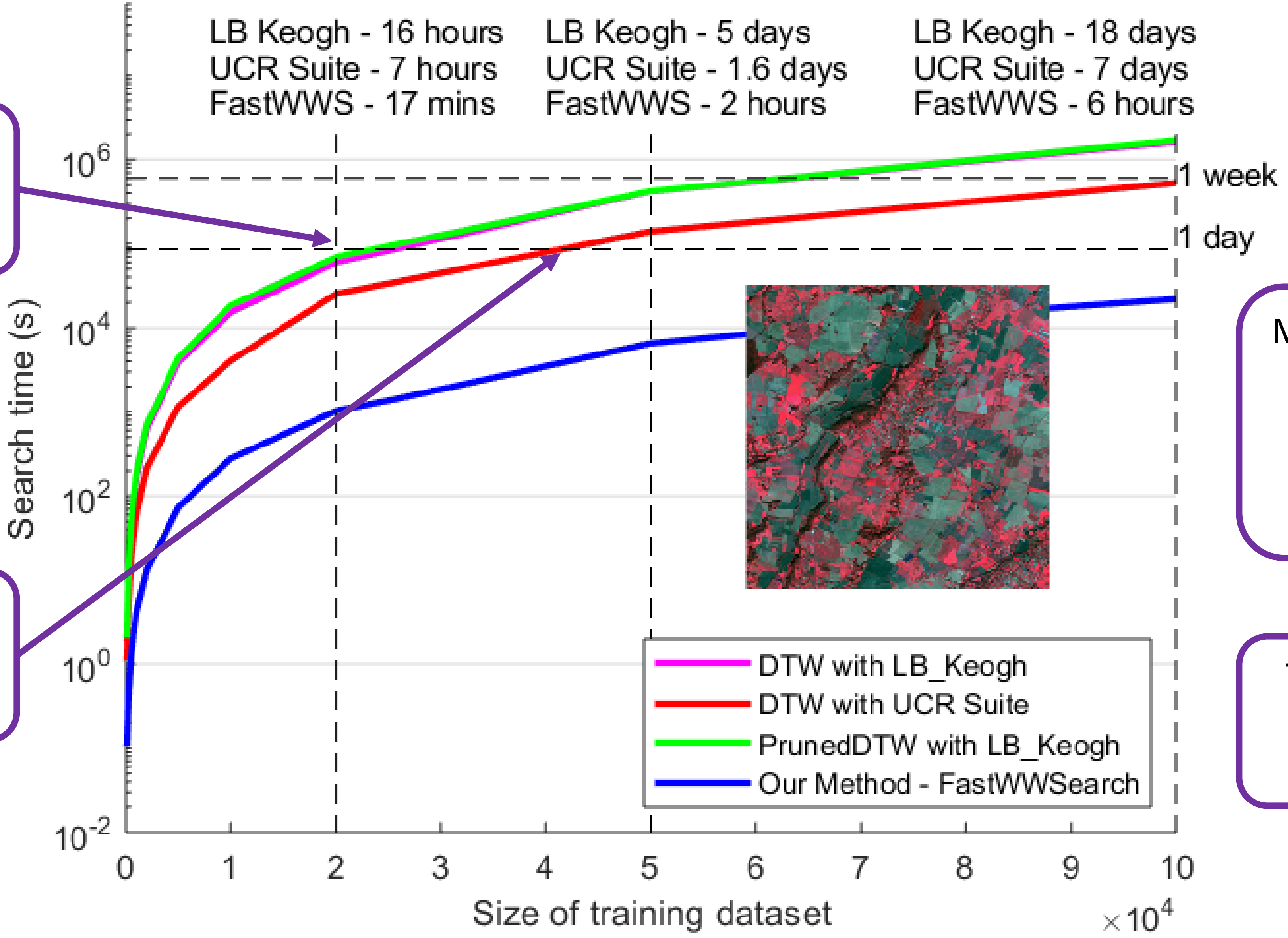
$L \leq 300 \Rightarrow 67\text{x}$

$\boldsymbol{L > 300 \Rightarrow 250\text{x}}$

FastWWS can SCALE too!

At just above **20k**, LB Keogh takes more than a day

At around **45k**, UCR Suite takes more than a day

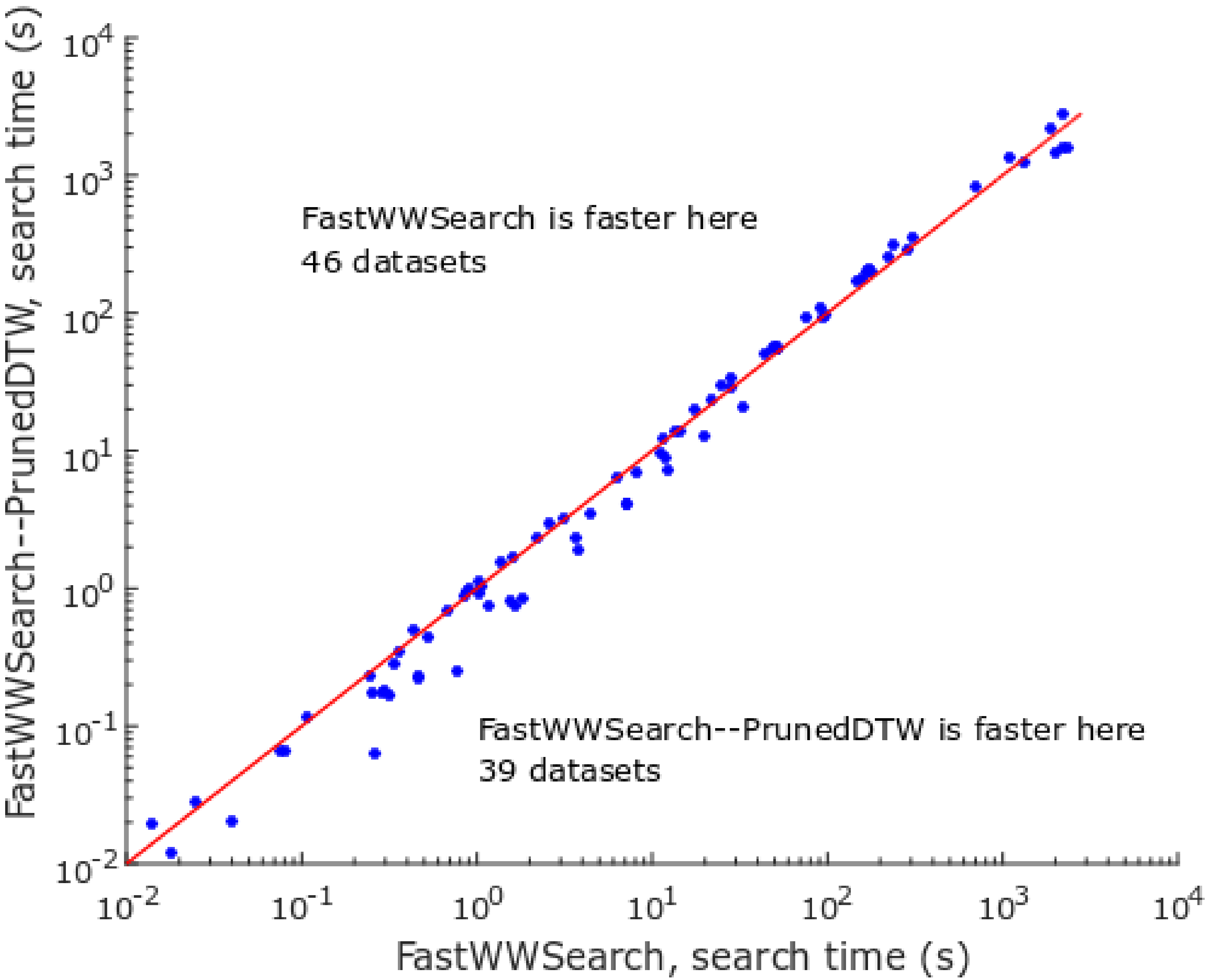


More than a week at **100k**

FastWWS takes only 6 hours

The short length ($L = 24$) affects PrunedDTW

FastWWS with PrunedDTW

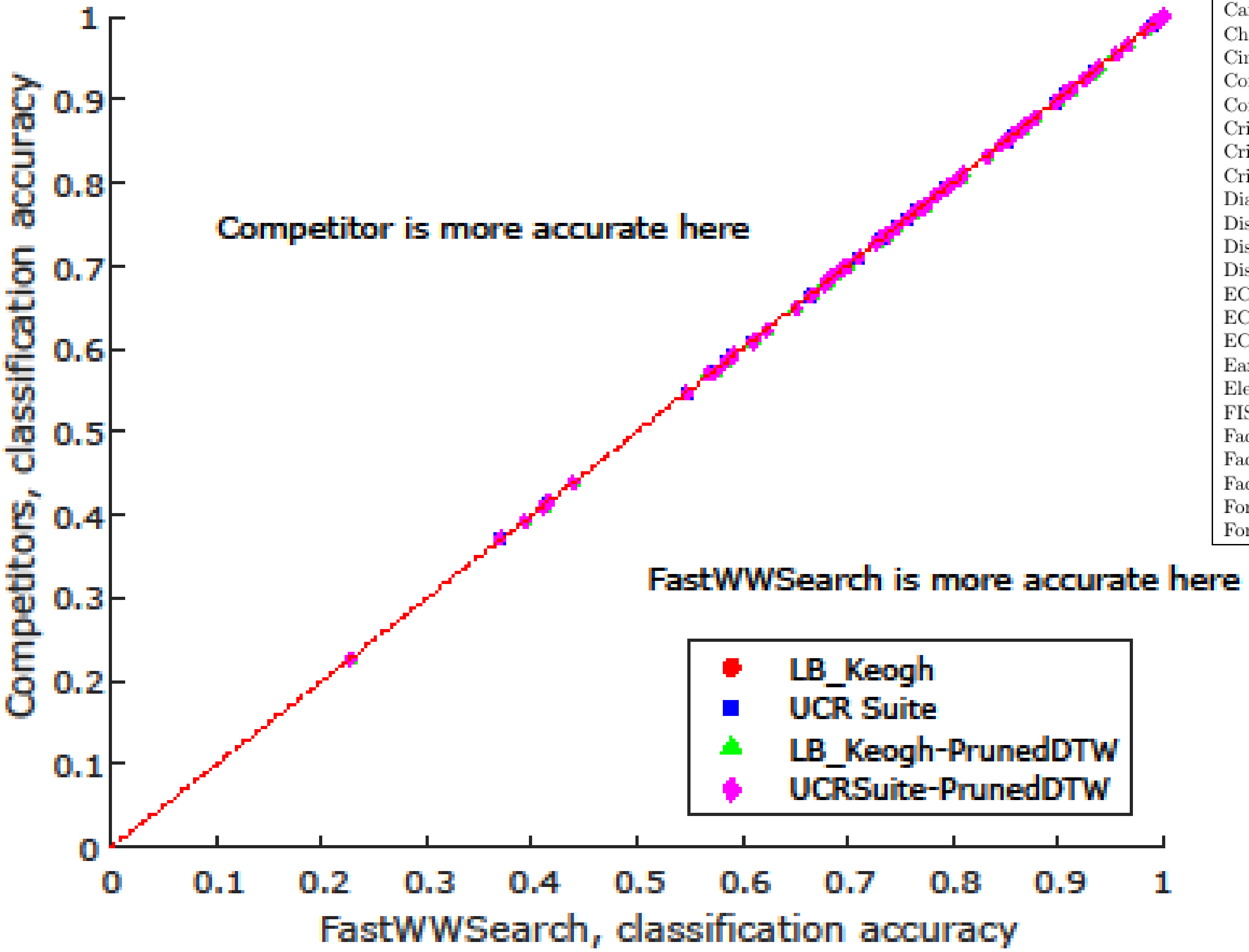


FastWWS-PrunedDTW

1. Compute Euclidean Distance ($w = 0$)
2. Use it as upper bound to prune DTW at larger window

- Not necessary faster
- **FastWWS** is faster on 55% of the Benchmark datasets
- Due to overhead in **PrunedDTW** in checking the upper bounds

Classification Accuracy



	Best warping window learnt by the following methods				
Datasets	LB_Keogh	UCR Suite	LB_Keogh–PrunedDTW	UCR Suite–PrunedDTW	**FastWWSearch**
50words	24	24	24	24	24
Adiac	6	6	6	6	6
ArrowHead	0	0	0	0	0
Beef	0	0	0	0	0
BeetleFly	36	36	36	36	36
BirdChicken	33	33	33	33	33
CBF	14	14	14	14	14
Car	9	9	9	9	9
ChlorineConcentration	0	0	0	0	0
CinC_ECG_torso	10	10	10	10	10
Coffee	0	0	0	0	0
Computers	74	74	74	74	74
Cricket_X	31	31	31	31	31
Cricket_Y	47	47	47	47	47
Cricket_Z	15	15	15	15	15
DiatomSizeReduction	0	0	0	0	0
DistalPhalanxOutlineAgeGroup	1	1	1	1	1
DistalPhalanxOutlineCorrect	2	2	2	2	2
DistalPhalanxTW	0	0	0	0	0
ECG200	0	0	0	0	0
ECG5000	1	1	1	1	1
ECGFiveDays	0	0	0	0	0
Earthquakes	17	17	17	17	17
ElectricDevices	13	13	13	13	13
FISH	19	19	19	19	19
FaceAll	4	4	4	4	4
FaceFour	6	6	6	6	6
FacesUCR	16	16	16	16	16
FordA	2	2	2	2	2
FordB	6	6	6	6	6

Accuracy should be the same as the window found is the same and **FastWWS** is **EXACT**

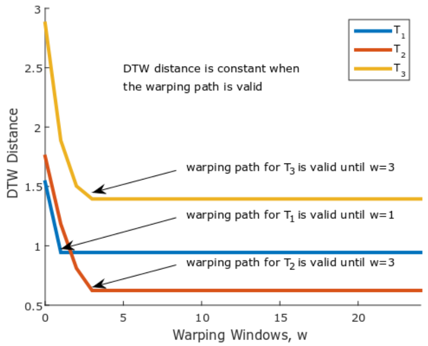
Conclusions

- A novel and exact algorithm to speedup the search for the best parameter (warping window) for DTW
  - **FastWWS** is more **EFFICIENT** and **FASTER**
  - **FastWWS** can **SCALE**

- Our results, datasets and source code are online at
  - https://bit.ly/SDM18
  - https://github.com/ChangWeiTan/FastWWSearch
  - Slides: http://changweitan.com/research/SDM18-slides.pdf

Future Work

- Search for the best parameter for other TS similarity functions
  - LCSS (δ, ε), MSM (c), ERP (g, λ) etc.
  - Satisfies the three properties:
    1. Its **distance** stays **valid** for some parameters
    2. Its **distance** is **monotone** with its parameters
    3. Its **lower bound** is **monotone** with its parameters



- Scaling up the State of the Arts in Time Series Classification
  - Elastic Ensembles (EE) [1]
  - Collective of Transformation-Based Ensembles (COTE) [2]

[1] Lines, J., & Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery, 29(3), 565-592.
[2] Bagnall, A., Lines, J., Hills, J., & Bostrom, A. (2015). Time-series classification with COTE: the collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering, 27(9), 2522-2535.

MONASH University
Information Technology

modélisation intelligence processus systèmes mips

Australian Government
Australian Research Council

Thank you!
Questions and Answers


CW. Tan


M. Herrmann


G. Forestier


G.I. Webb


F. Petitjean

This work was supported by the Australian Research Council under grant DE170100037. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-16-1-4023

chang.tan@monash.edu

github.com/ChangWeiTan/FastWWSearch

bit.ly/SDM18