FTW: Fast Similarity Search under the Time Warping Distance

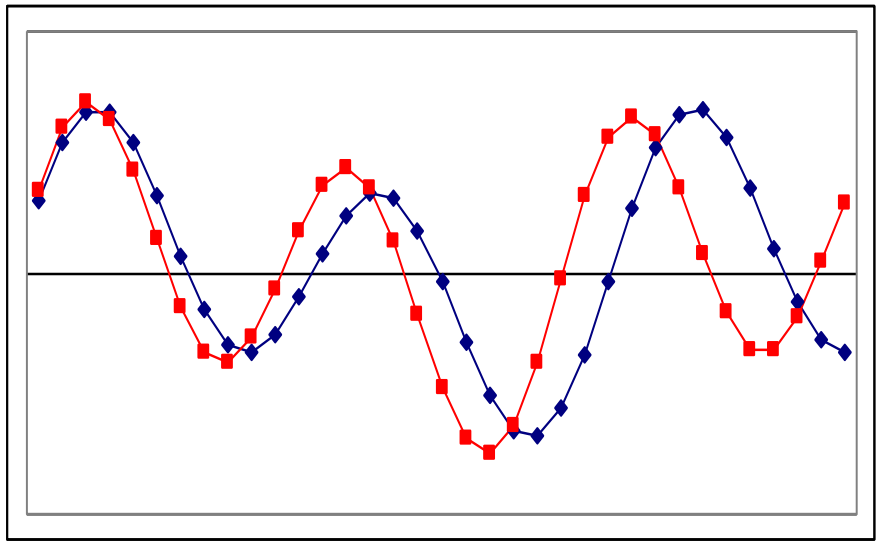
Yasushi Sakurai (NTT Cyber Space Labs)
Masatoshi Yoshikawa (Nagoya Univ.)
Christos Faloutsos (Carnegie Mellon Univ.)

Motivation

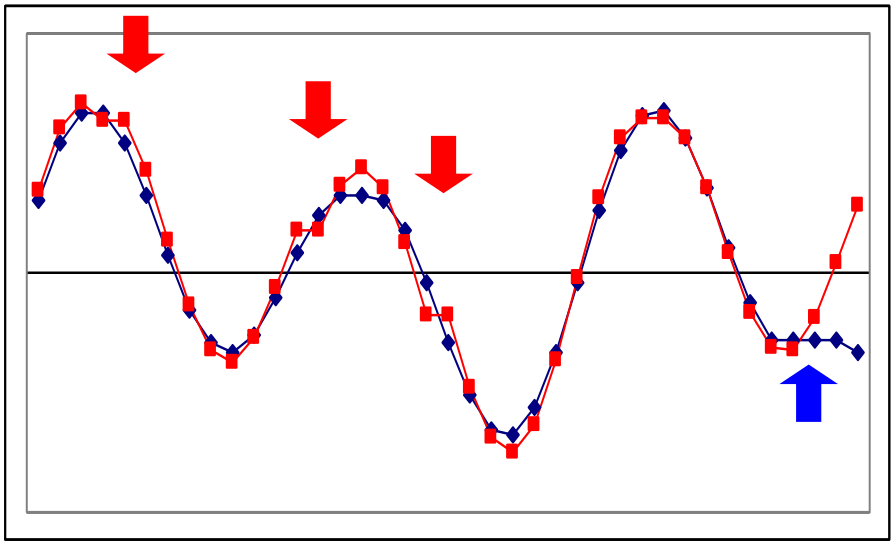
- Time-series data
  - many applications
    - computational biology, astrophysics, geology, meteorology, multimedia, economics
- Similarity search
  - Euclidean distance
  - DTW (Dynamic Time Warping)
    - Useful for different sequence lengths
    - Different sampling rates
    - scaling along the time axis

Mini-introduction to DTW

- DTW allows sequences to be stretched along the time axis
  - Minimize the distance of sequences
  - Insert ‘stutters’ into a sequence
  - THEN compute the (Euclidean) distance



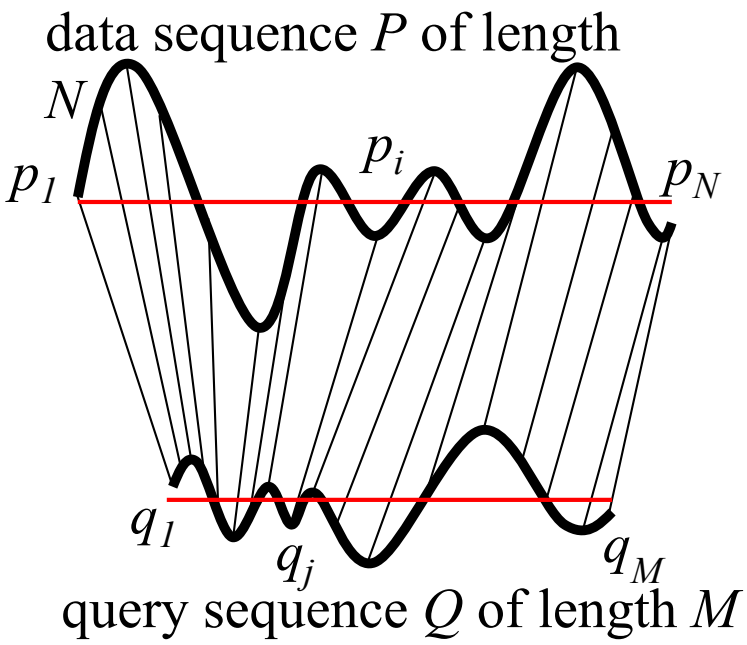
original

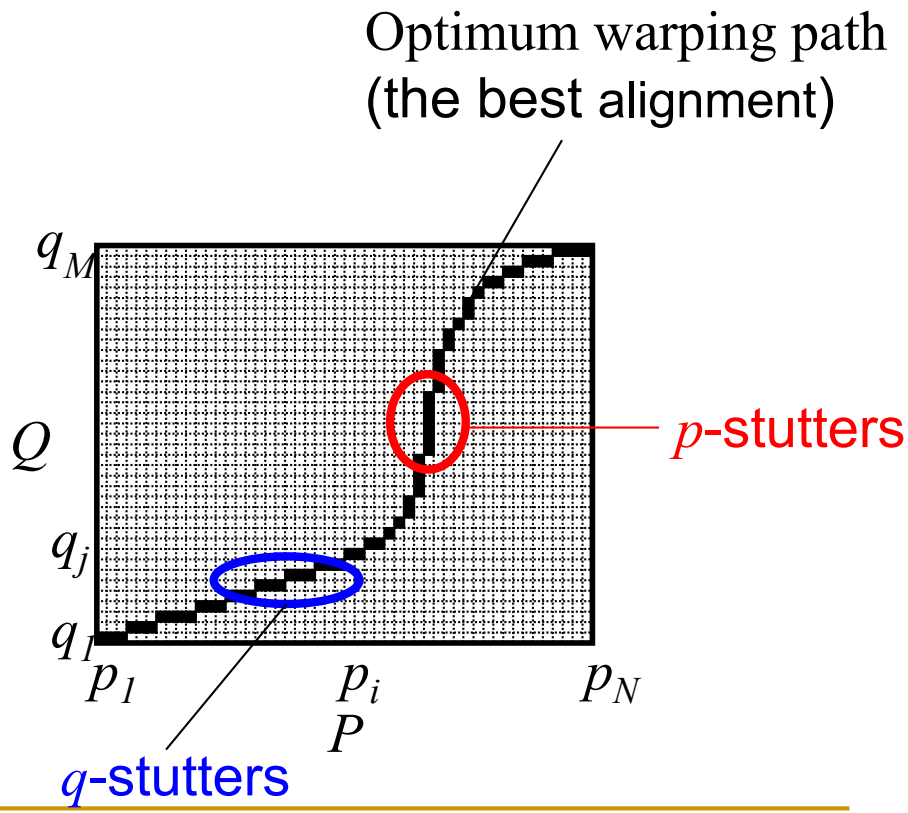


‘stutters’:

Mini-introduction to DTW

- DTW is computed by dynamic programming
  - Warping path: set of grid cells in the time warping matrix





Mini-introduction to DTW

- DTW is computed by dynamic programming

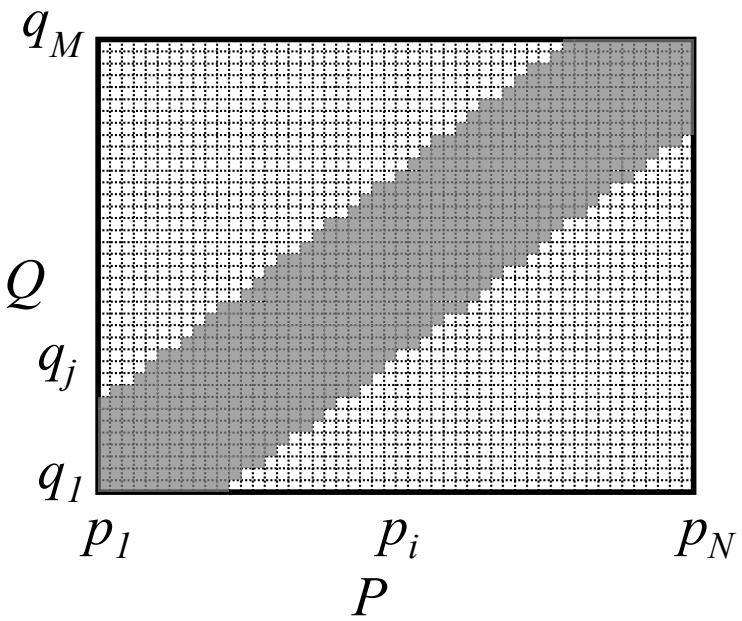
$$p_1, p_2, \ldots, p_{i,;} \qquad q_1, q_2, \ldots, q_j$$

$$D_{dtw}(P,Q) = f(N,M)$$

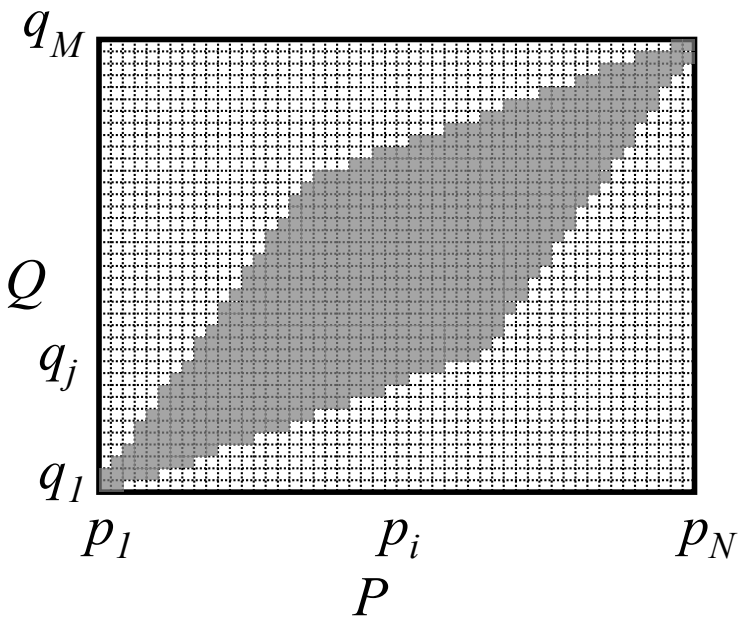
$$f(i,j) = \left\| p_i - q_j \right\| + \min \begin{cases} f(i,j-1) & p\text{-stutter} \\ f(i-1,j) & q\text{-stutter} \\ f(i-1,j-1) & \text{no stutter} \end{cases}$$

Mini-introduction to DTW

- Global constraints limit the warping scope
  - Warping scope: area that the warping path is allowed to visit



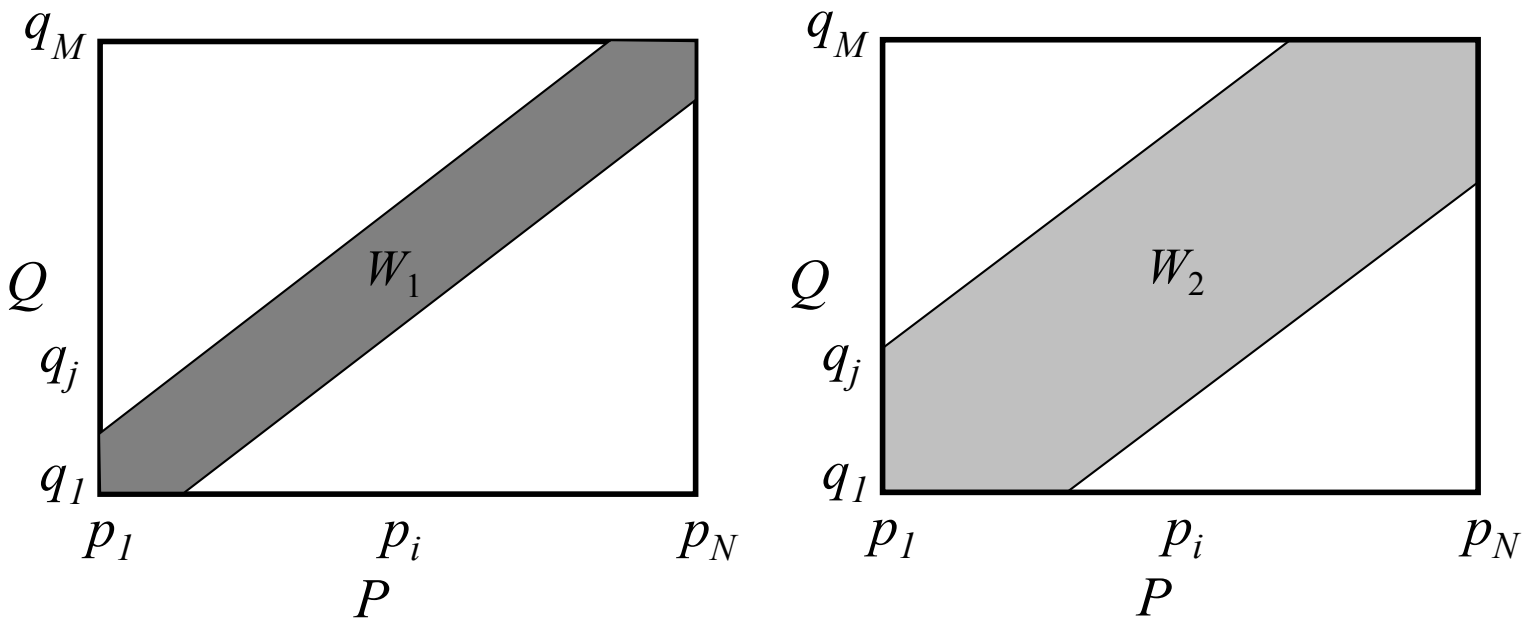
Sakoe-Chiba Band



Itakura Parallelogram

Mini-introduction to DTW

- Width of the warping scope W is user-defined



Sakoe-Chiba Band

Motivation

- Similarity search for time-series data
  - DTW (Dynamic Time Warping)
    - scaling along the time axis

    But…
    - High search cost $O(NM)$
    - prohibitive for long sequences

Our Solution, FTW

- Requirements:
  1. Fast
  2. No false dismissals
  3. No restriction on the sequence length
     - It should handle data sequences of different lengths
  4. Support for any, as well as for no restriction on "warping scope"

Problem Definition

- Given
  - S time-series data sequences of unequal lengths $\{P_1, P_2, \ldots, P_S\}$,
  - a query sequence Q,
  - an integer k,
  - (optionally) a warping scope W,
- Find the k-nearest neighbors of Q from the data sequence set by using DTW with W

Overview

- Introduction
- Related work
- Main ideas
- Experimental results
- Conclusions

Related Work

- Sequence indexing
  - Agrawal et al. (FODO 1998)
  - Keogh et al. (SIGMOD 2001)
  - …
- Subsequence matching
  - Faloutsos et al. (SIGMOD 1994)
  - Moon et al. (SIGMOD 2002)
  - …

Related Work

- Fast sequence matching for DTW
  - Yi et al. (ICDE 1998)
  - Kim et al. (ICDE 2001)
  - Chu et al. (SDM 2002)
  - Keogh (VLDB 2002)
  - Zhu et al. (SIGMOD 2003)
  - …
- None of the existing methods for DTW fulfills all the requirements

Overview

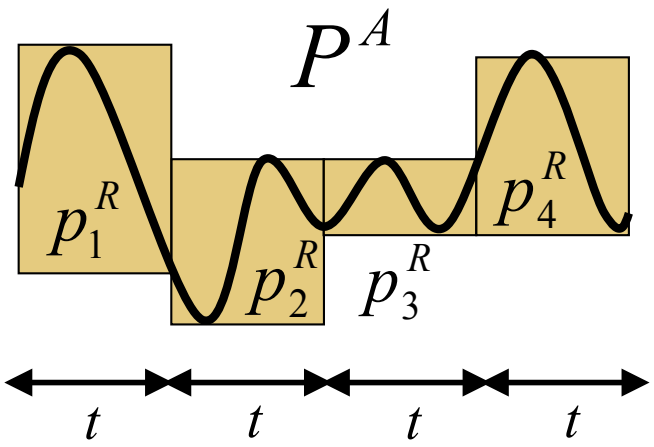
- Introduction
- Related work
- Main ideas
- Experimental results
- Conclusions

Main Idea (1) – LBS

- LBS (Lower Bounding distance measure with Segmentation)
- P^A : Approximate sequences
  - p_i^R: segment range
  - p_i^U : upper value
  - p_i^L: lower value

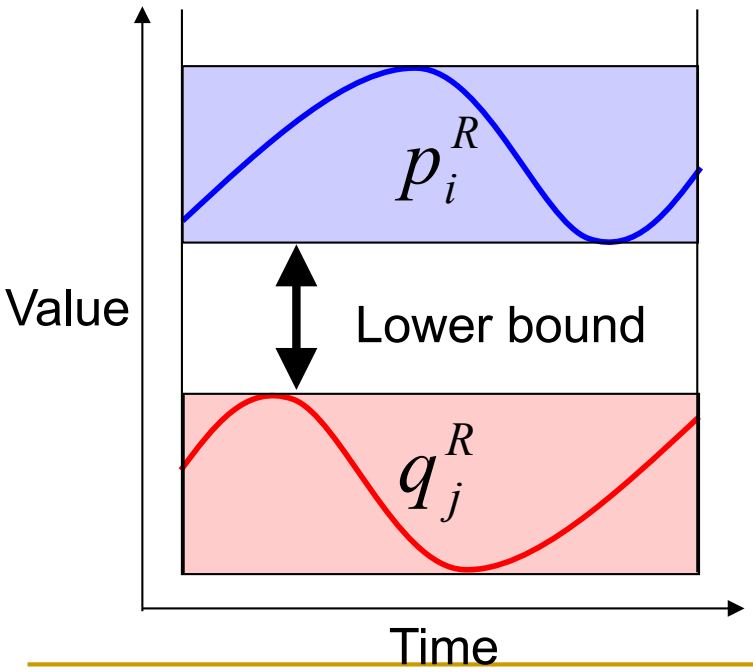
$$p_i^R = (p_i^L : p_i^U)$$

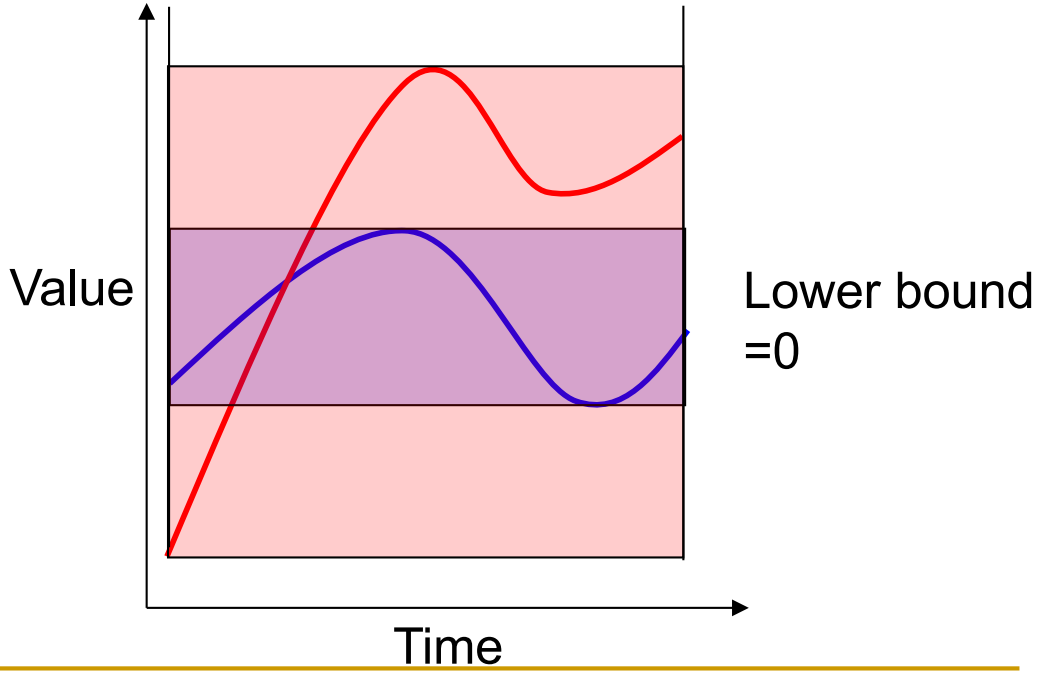
  - t: length of time intervals*



Main Idea (1) – LBS

- Compute lower bounding distance
  - Distance of the two ranges p_i^R and q_j^R : distance of their two closest points





Main Idea (1) – LBS

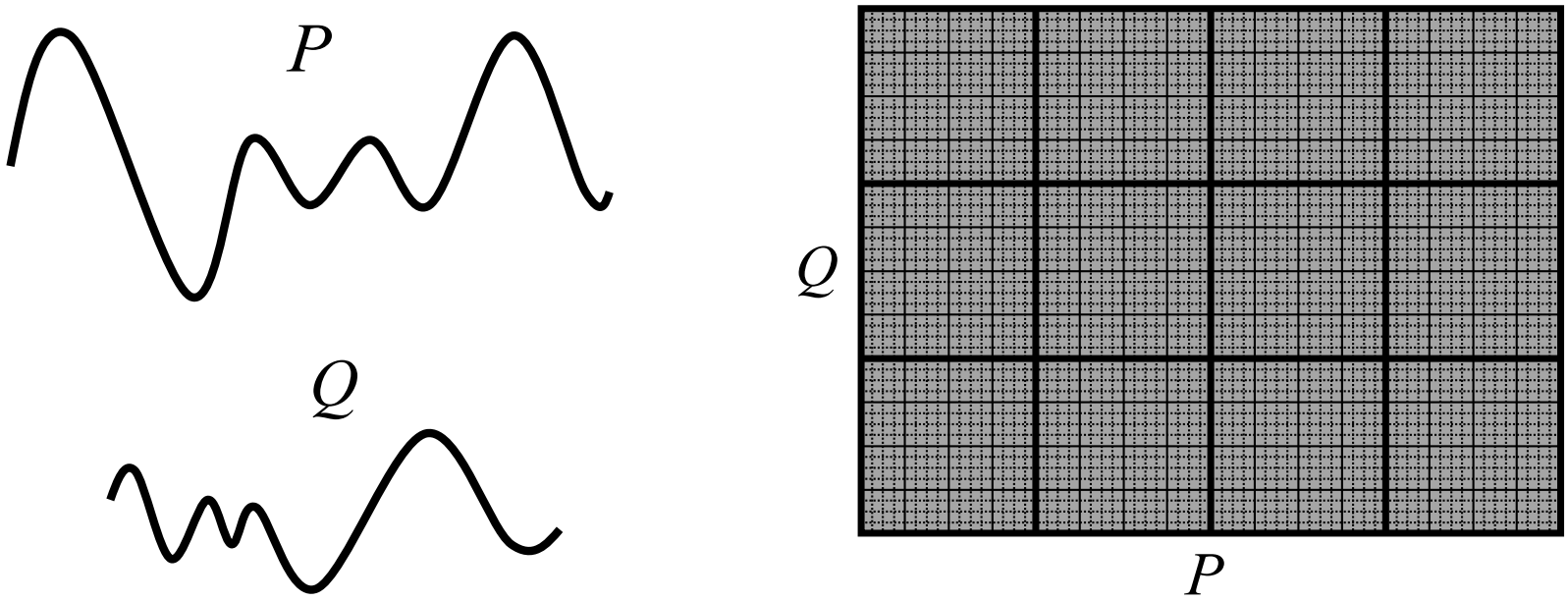
details

- Compute lower bounding distance
  - Distance of the two ranges p_i^R and q_j^R : distance of their two closest points

$$D_{seg}(p_i^R, q_j^R) = \begin{cases} \left\| p_i^L - q_j^U \right\| & (p_i^L > q_j^U) \\ \left\| q_j^L - p_i^U \right\| & (q_j^L > p_i^U) \\ 0 & (otherwise) \end{cases}$$

Main Idea (1) – LBS

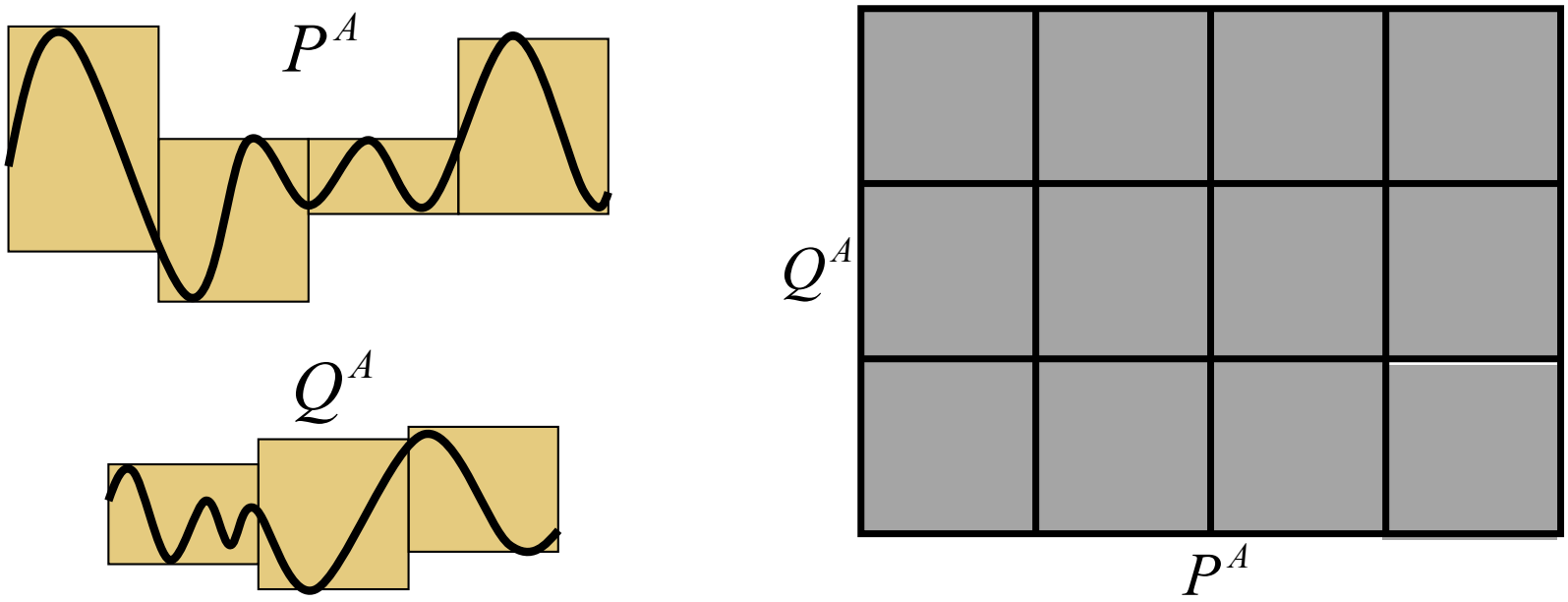
- Exact DTW distance



Main Idea (1) – LBS

- Compute lower bounding distance from P^A and Q^A
- Use a dynamic programming approach

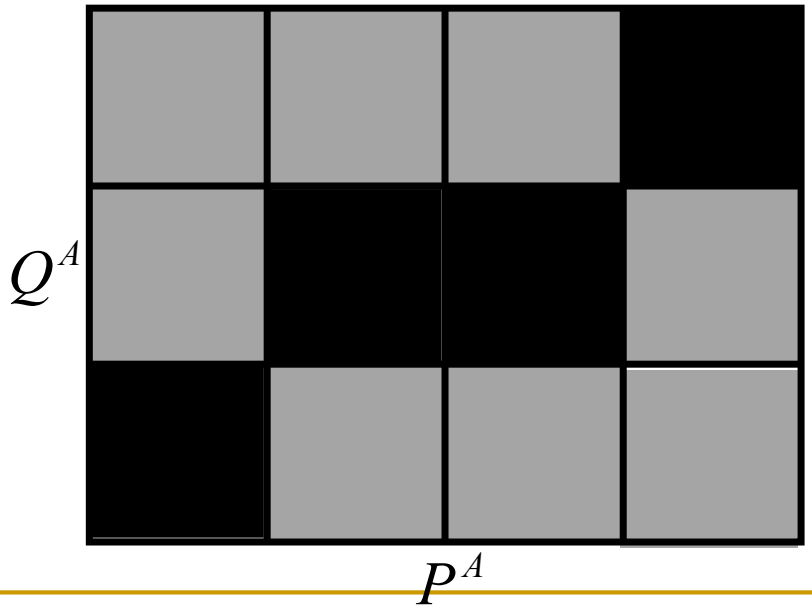
$$D_{lbs}(P^A, Q^A) \leq D_{dtw}(P, Q)$$

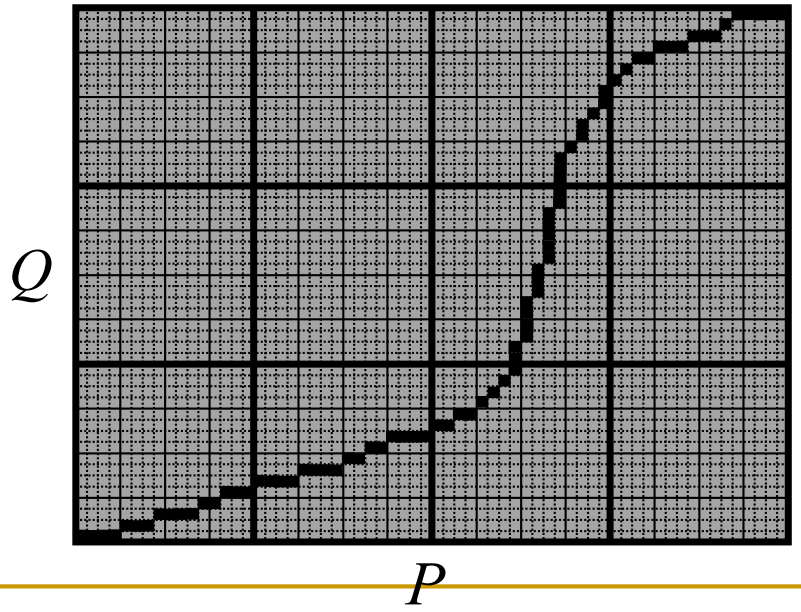


Main Idea (1) – LBS

- Compute lower bounding distance from P^A and Q^A
- Use a dynamic programming approach

$$D_{lbs}(P^A, Q^A) \leq D_{dtw}(P, Q)$$





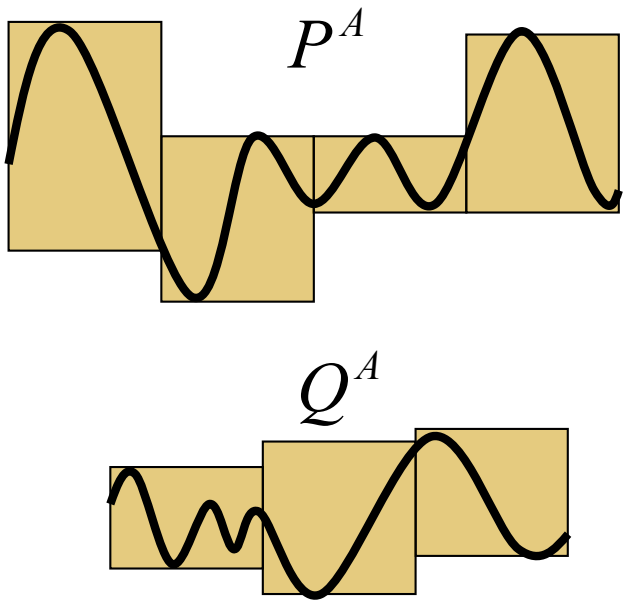
Main Idea (2) – EarlyStopping

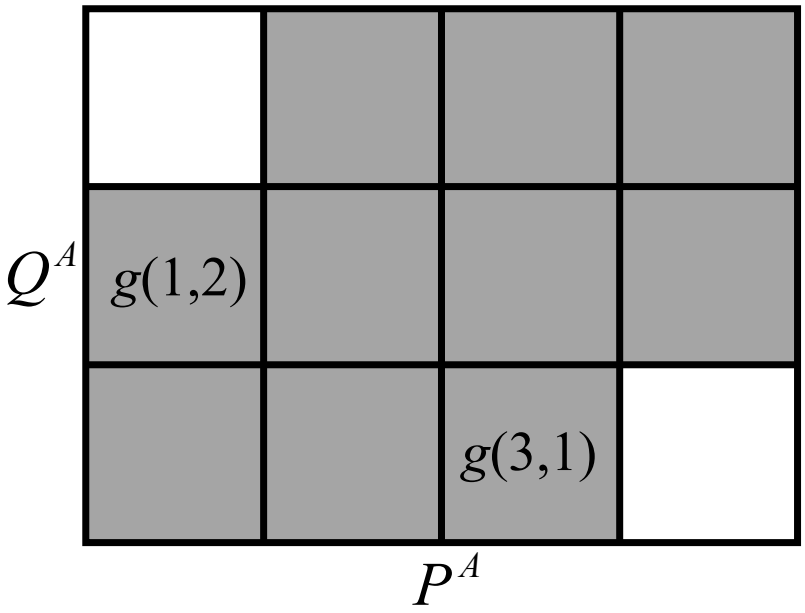
- Exploit the fact that we have found k-near neighbors at distance d_{cb}
  - d_{cb}: k-nearest neighbor distance (the Current Best)

    the exact distance of the best k candidates so far

Main Idea (2) – EarlyStopping

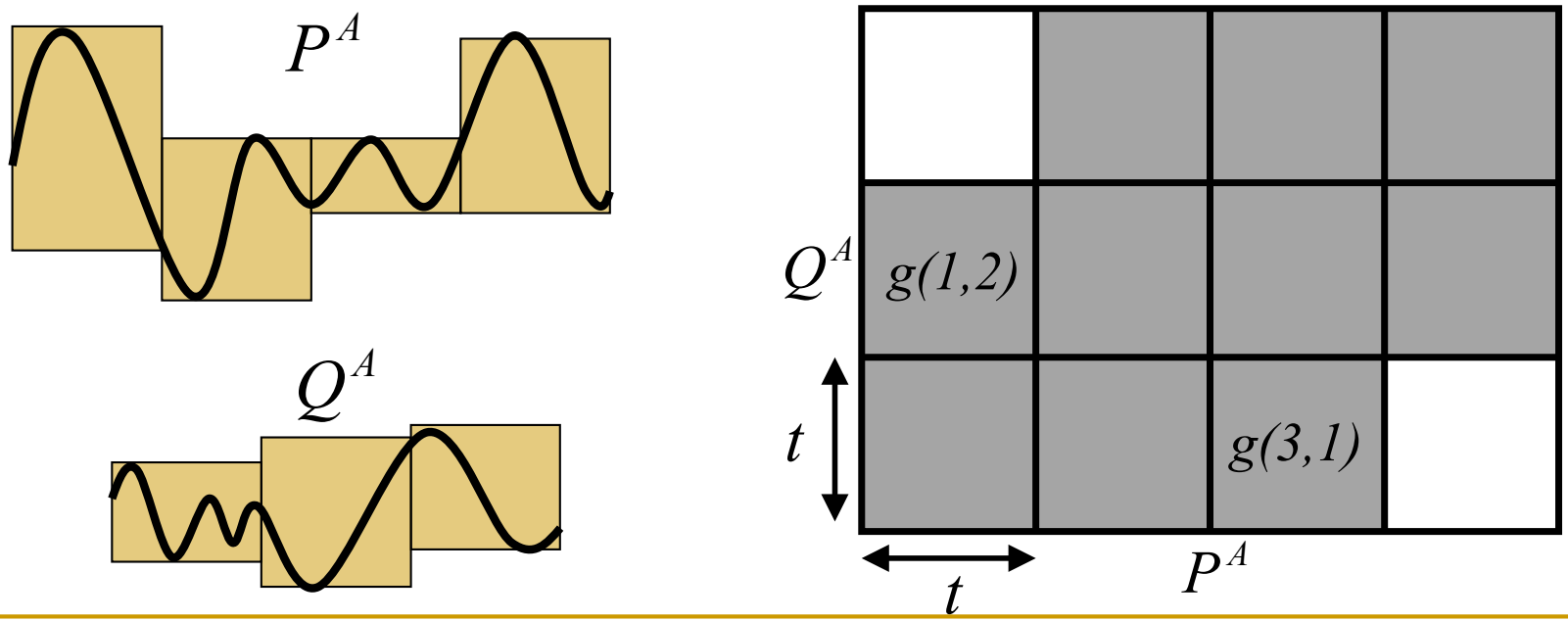
- Exclude useless warping paths by using d_{cb}
  - Omit $g(1,3)$ if $g(1,2) > d_{cb}$
  - Omit $g(4,1)$ if $g(3,1) > d_{cb}$





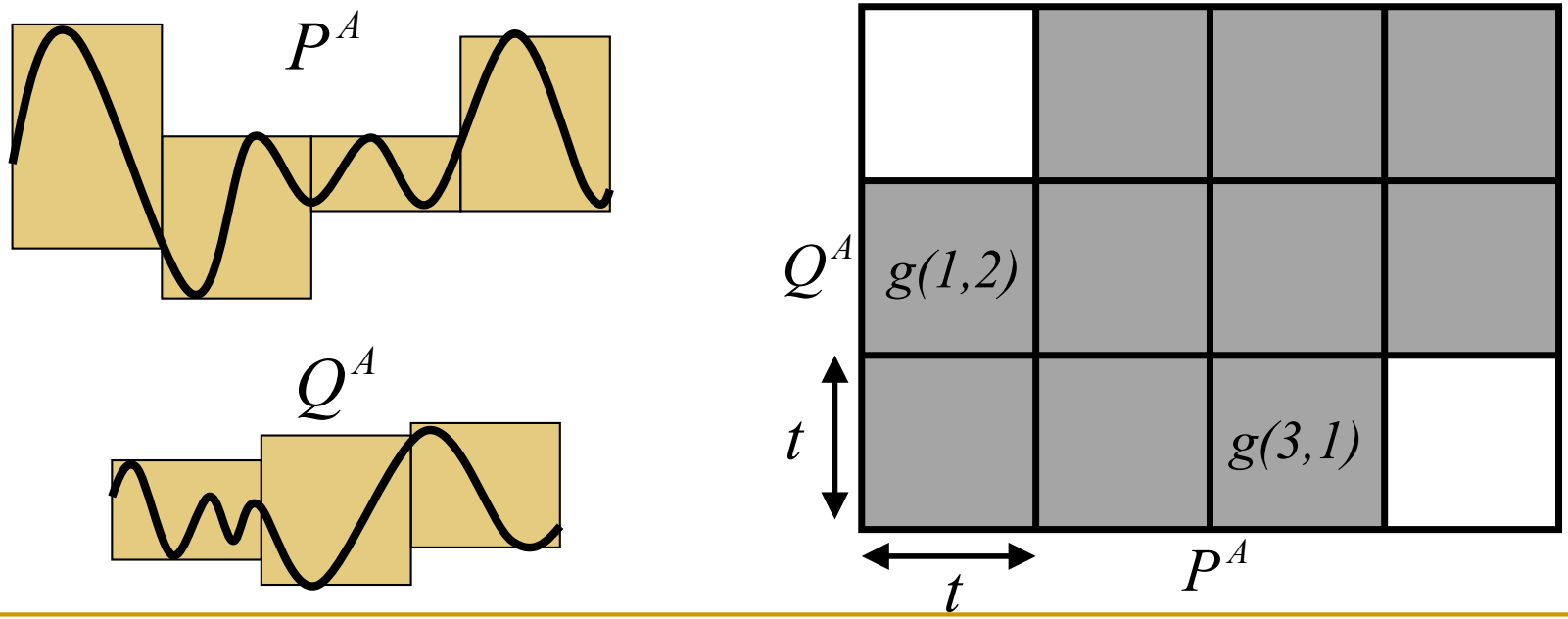
Main Idea (3) – Refinement

- Q: How to choose t (length of time intervals)?



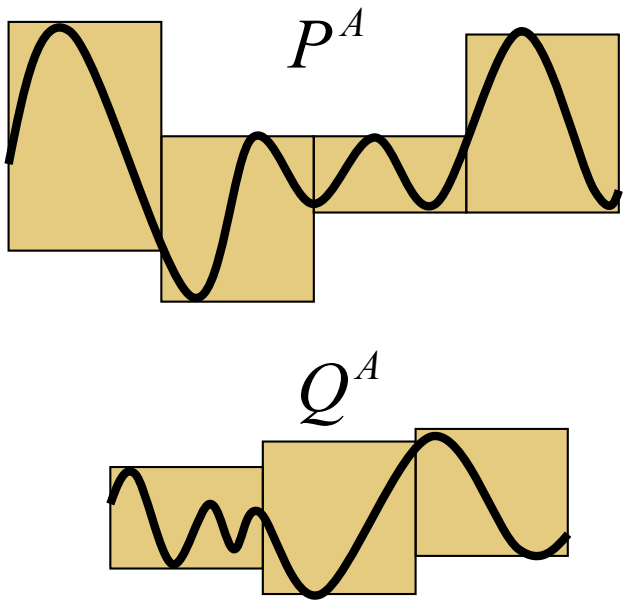
Main Idea (3) – Refinement

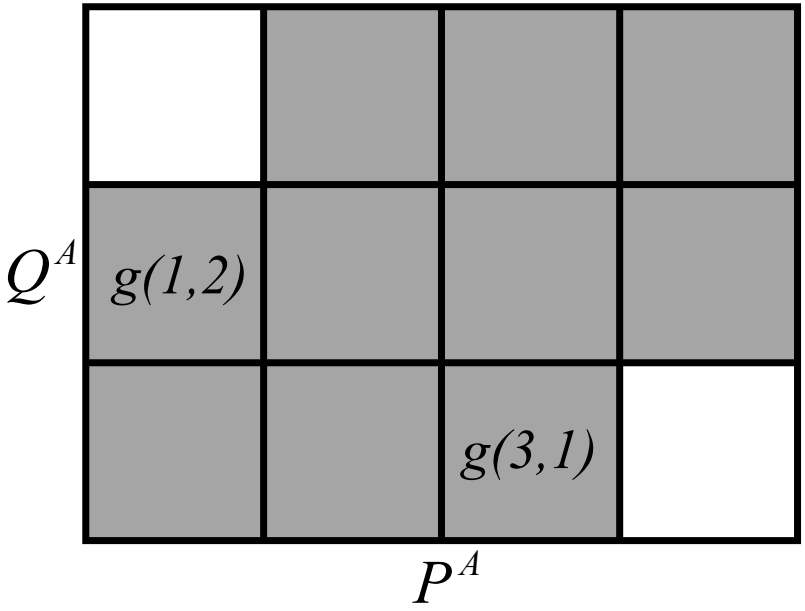
- Q: How to choose t (length of intervals)?
- A: Use multiple granularities, as follows:



Main Idea (3) – Refinement

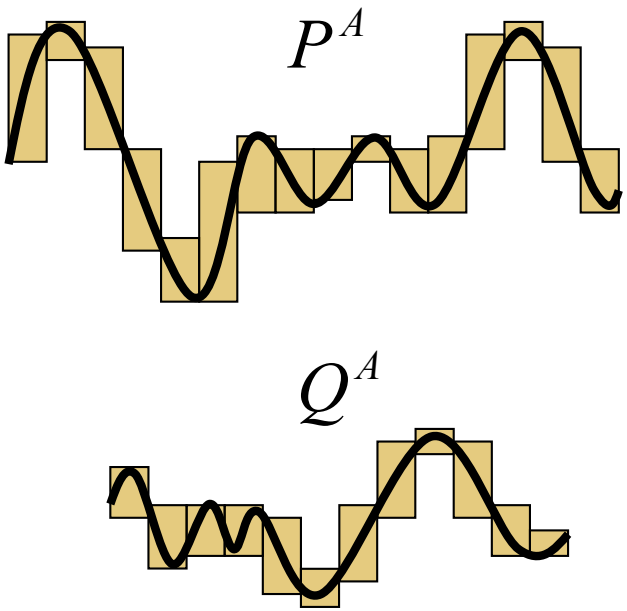
- Compute the lower bounding distance from the coarsest sequences as the first refinement step
- Ignore P if $D_{lbs}(P^A, Q^A) > d_{cb}$, otherwise:

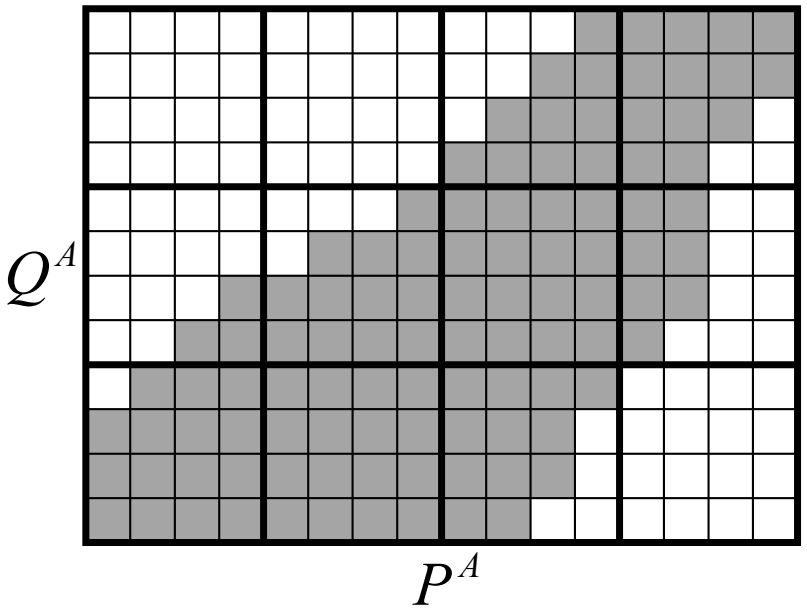




Main Idea (3) – Refinement

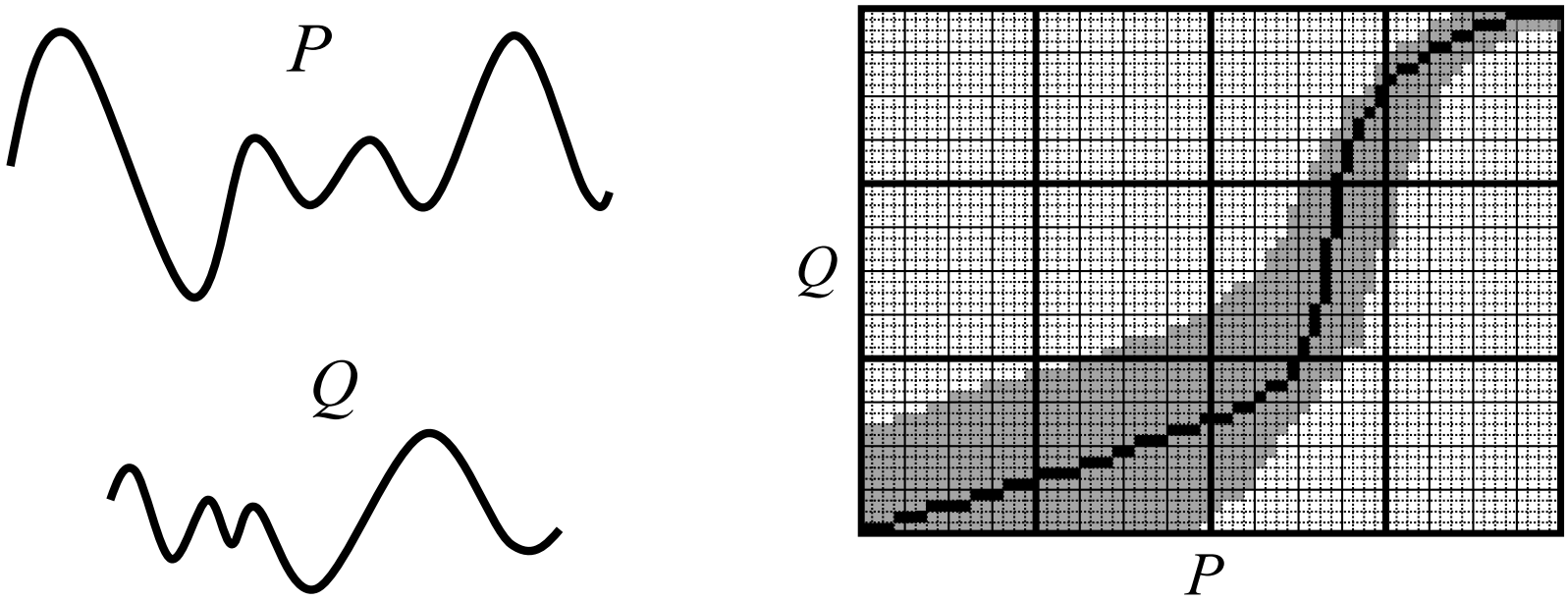
- … compute the distance from more accurate sequences as the second refinement step
- … repeat





Main Idea (3) – Refinement

- … until the finest granularity
- Update the list of k-nearest neighbors if $D_{dtw}(P,Q) \leq d_{cb}$



Overview

- Introduction
- Related work
- Main ideas
- Experimental results
- Conclusions

Experimental results

- Setup
  - Intel Xeon 2.8GHz, 1GB memory, Linux
  - Datasets:

    Temperature, Fintime, RandomWalk
  - Four different time intervals (for n=2048)

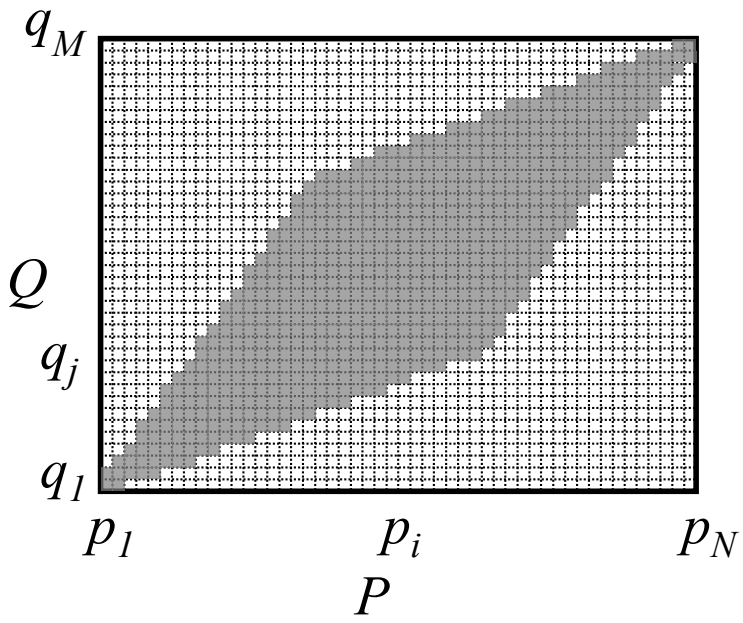
    t_1=2, t_2=8, t_3=32, t_4=128
- Evaluation
  - Compared FTW with LB_PAA (the best so far)
  - Mainly computation time

Outline of experiments

- Speed vs db size
- Speed vs warping scope W
- Effect of filtering
- Effect of varying-length data sequences

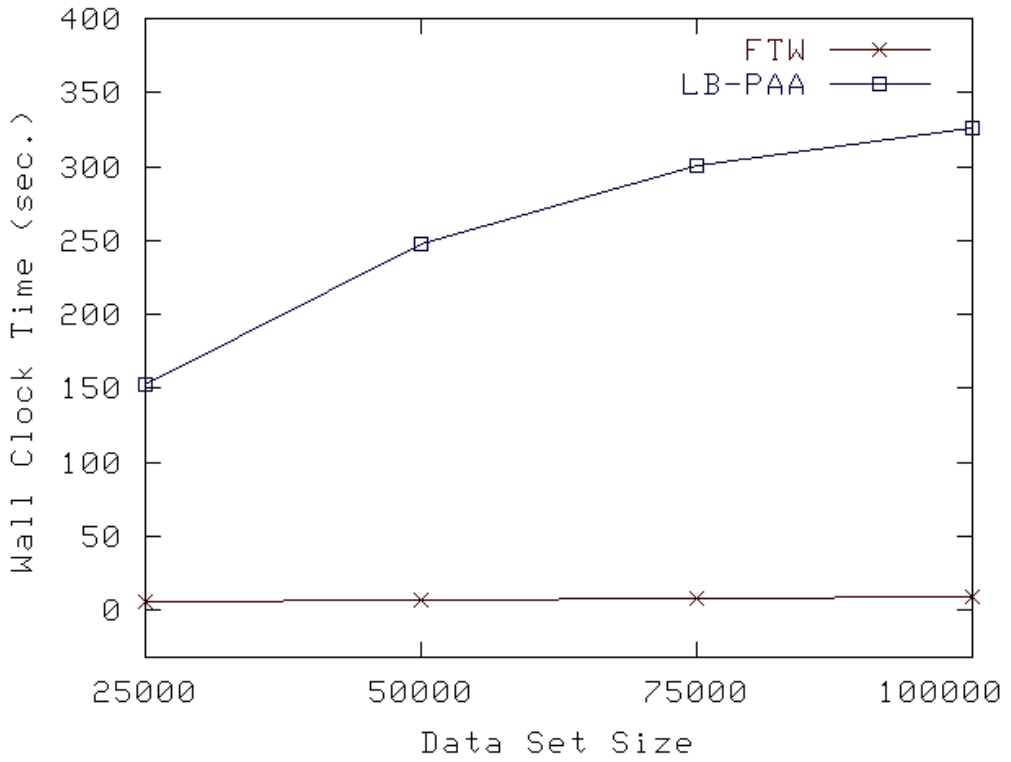
Search Performance

- Itakura Parallelogram



Search Performance

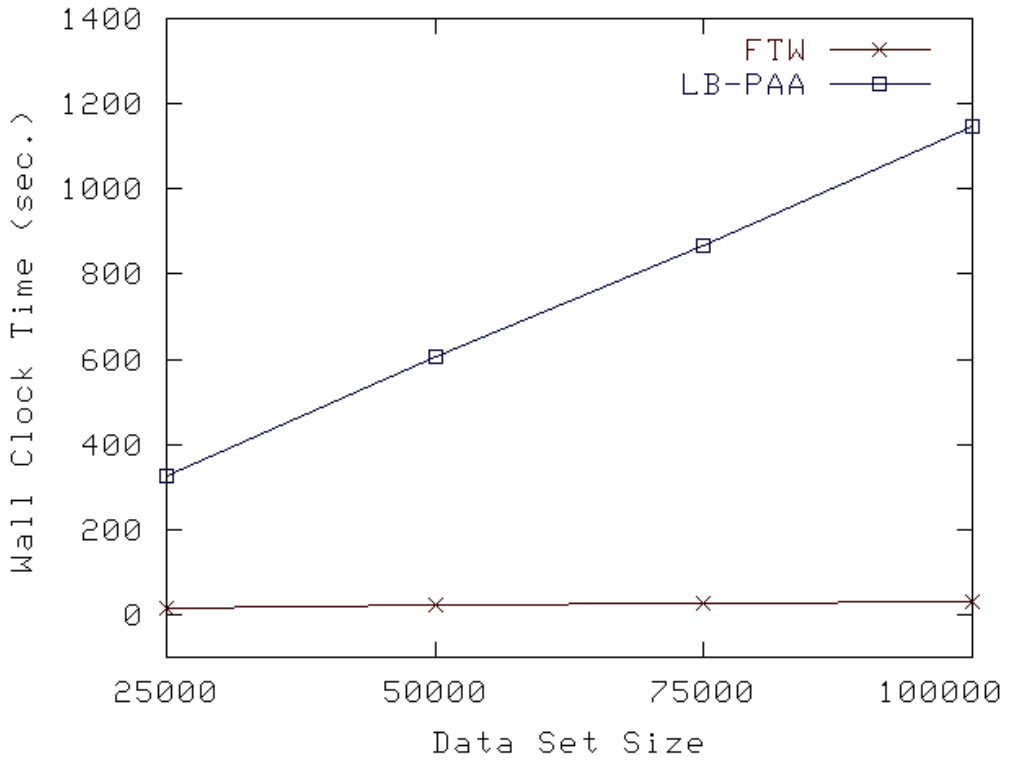
- Wall clock time as a function of data set size
- *Temperature*



FTW is up to 50 times faster!

Search Performance

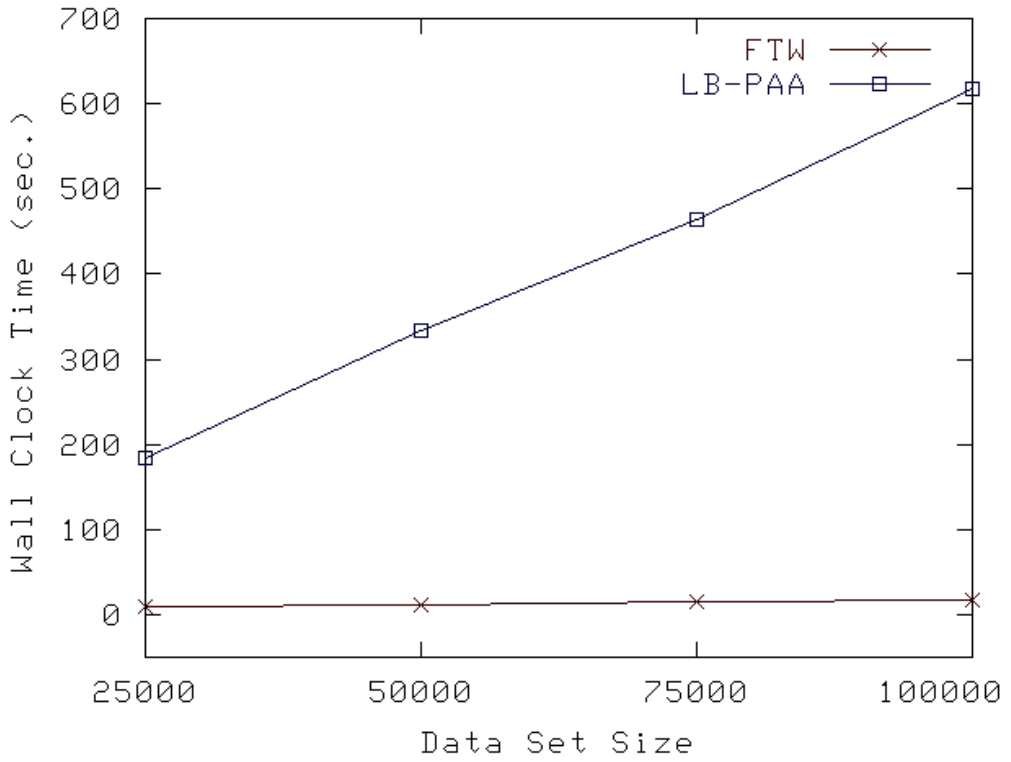
- Wall clock time as a function of data set size
- *Fintime*



FTW is up to 40 times faster!

Search Performance

- Wall clock time as a function of data set size
- *RandomWalk*



FTW is up to 40 times faster!

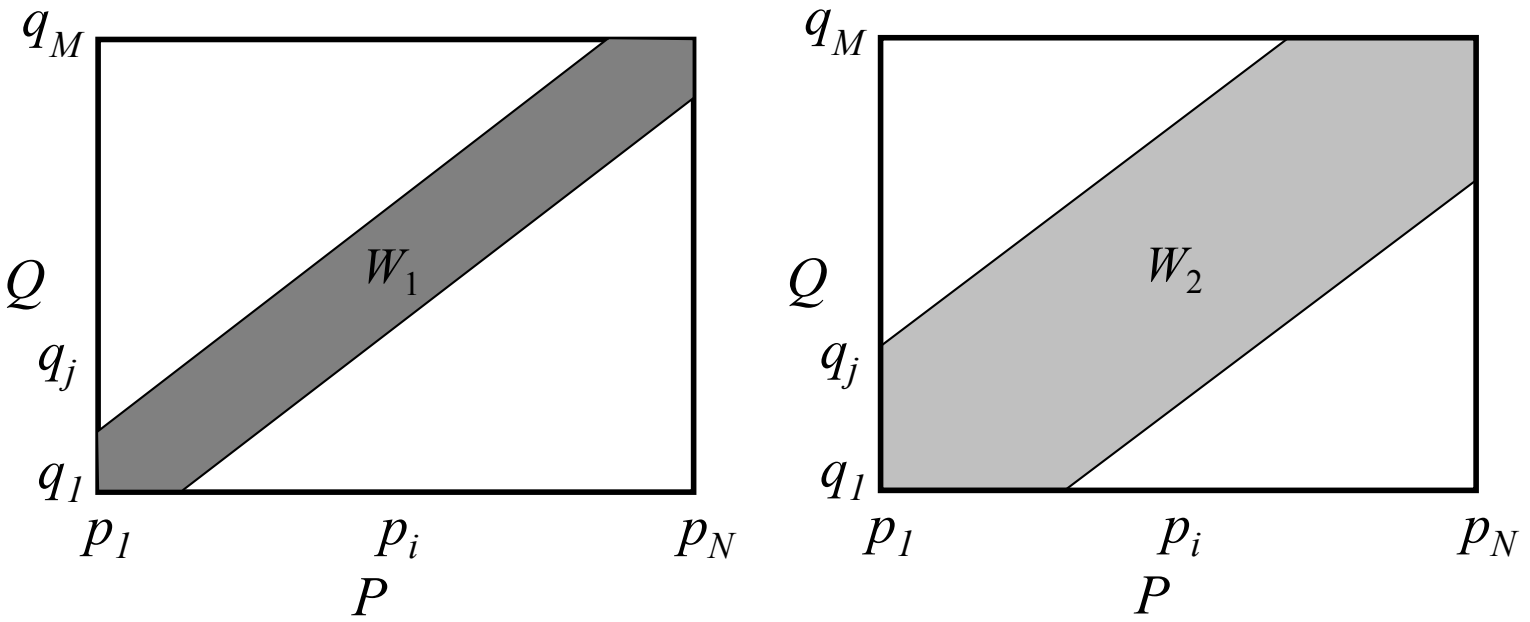
More effective as the size grows

Outline of experiments

- Speed vs db size
- Speed vs warping scope W
- Effect of filtering
- Effect of varying-length data sequences

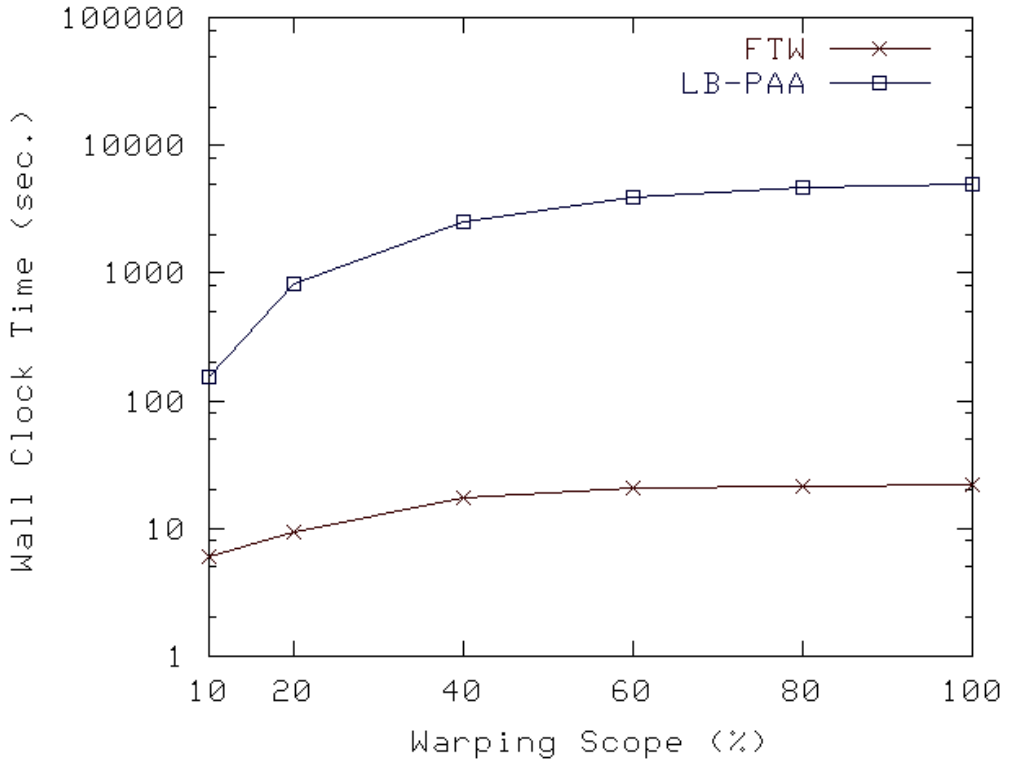
Search Performance

- Sakoe-Chiba Band



Search Performance

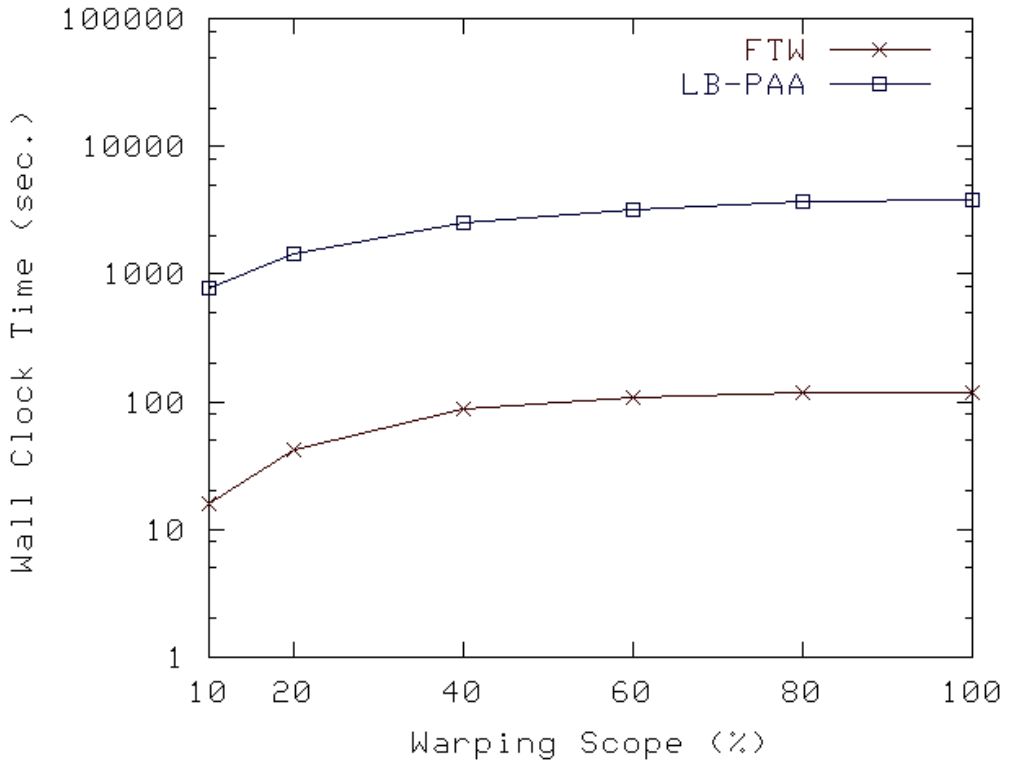
- Wall clock time as a function of warping scope
- *Temperature*



FTW is up to 220 times faster!

Search Performance

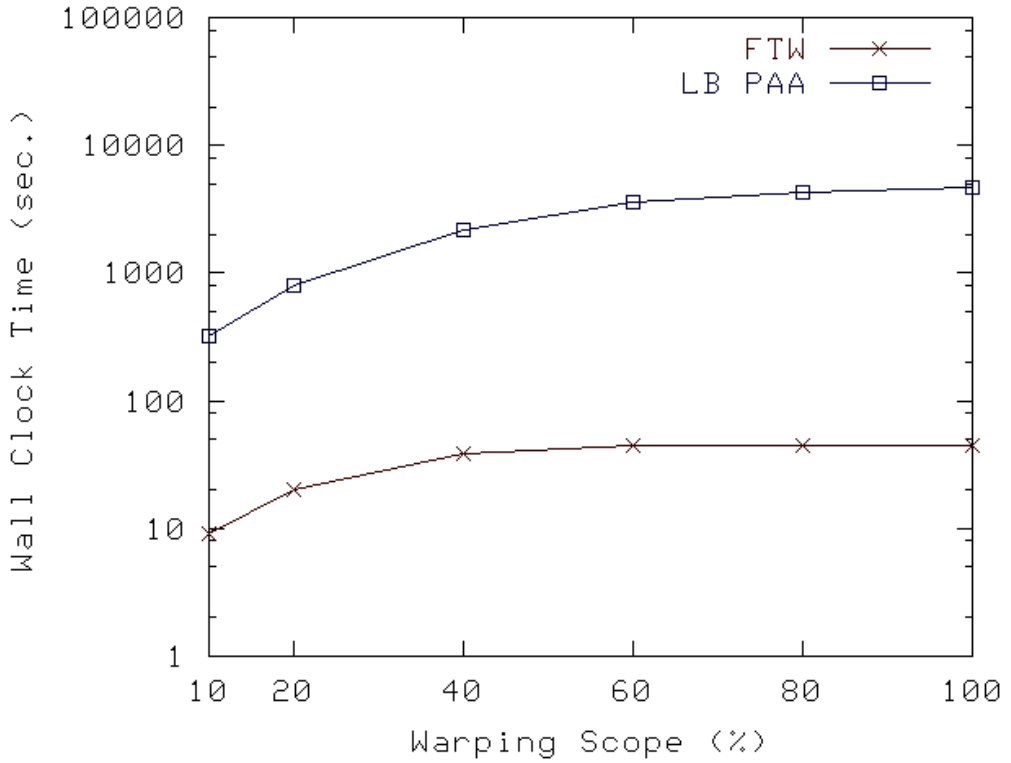
- Wall clock time as a function of warping scope
- *Fintime*



FTW is up to 70 times faster!

Search Performance

- Wall clock time as a function of warping scope
- *RandomWalk*



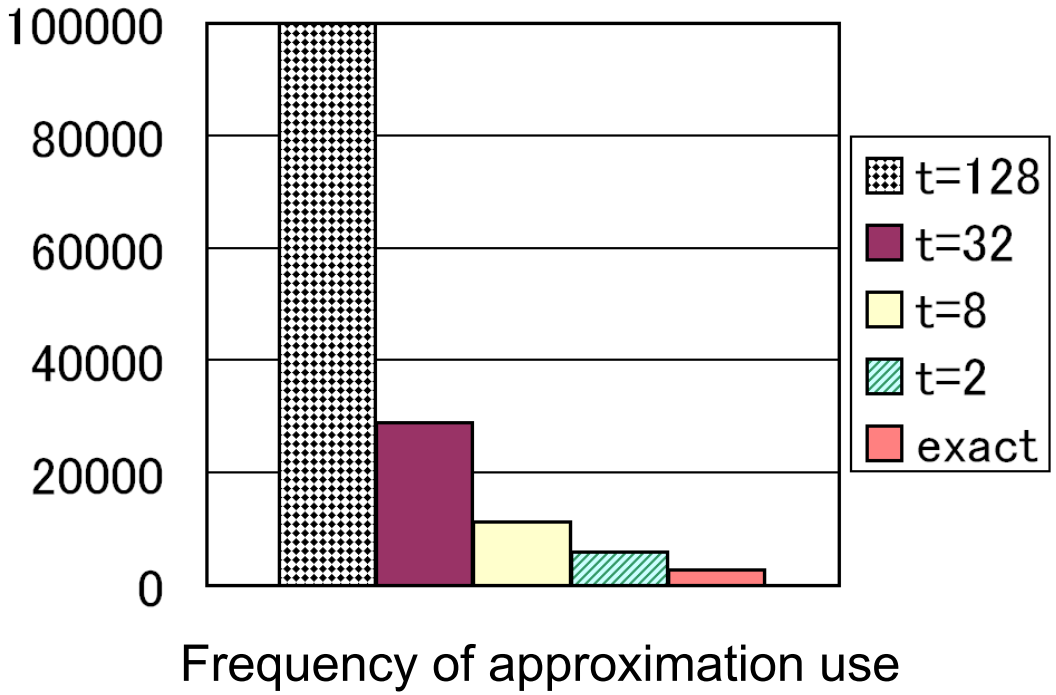
FTW is up to 100 times faster!

Outline of experiments

- Speed vs db size
- Speed vs warping scope W
- Effect of filtering
- Effect of varying-length data sequences

Effect of filtering

- Most of data sequences are excluded by coarser approximations (t_4=128 and t_3=32)
  - Using multiple granularities has significant advantages



Outline of experiments

- Speed vs db size
- Speed vs warping scope W
- Effect of filtering
- Effect of varying-length sequences

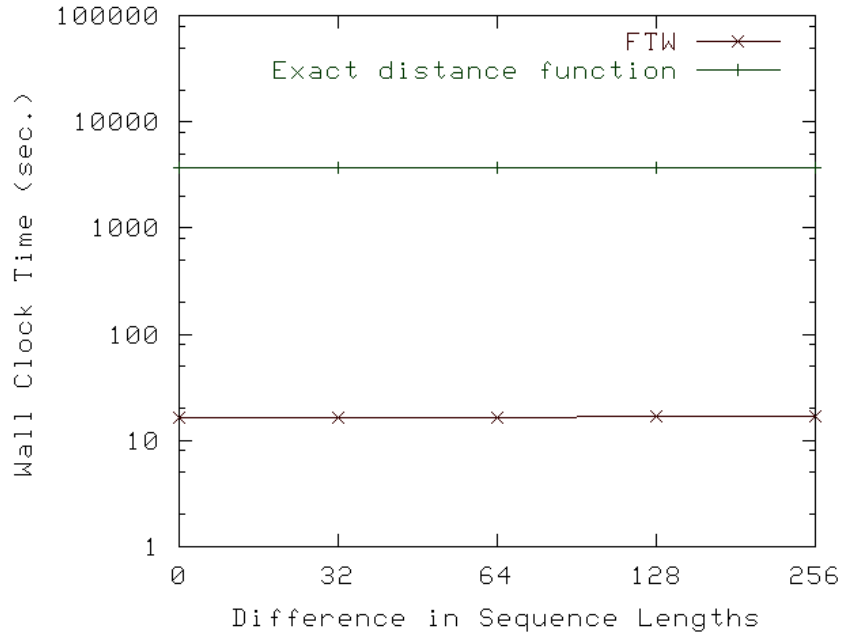
Difference in Sequence Lengths

- 5 sequence data sets

  Random(2048,0): length 2048 +/- 0

  Random(2048,32): length 2048 +/- 16

  Random(2048,64), *Random*(2048,128), *Random*(2048,256)



Outperform by 2+ orders of magnitude

LB_PAA can not handle

Overview

- Introduction
- Related work
- Main ideas
- Experimental results
- Conclusions

Conclusions

- Design goals:
  1. Fast
  2. No false dismissals
  3. No restriction on the sequence length
  4. Support for any, as well as for no restriction on “warping scope”

Conclusions

- Design goals:
  1. Fast (up to 220 times faster)
  2. No false dismissals
  3. No restriction on the sequence length
  4. Support for any, as well as for no restriction on “warping scope”

Page Accesses

details

- Sequential scan of feature data should boost performance (speed-up factors SF=5, SF=10)

$$PA_{SF} = \frac{PA_{fd}}{SF} + PA_{ds}$$

PA_{ds}: page accesses for data sequences

PA_{fd}: page accesses for feature data