

Iterative Deepening Dynamic Time Warping for Time Series

**Selina Chu, Eamonn Keogh*,
David Hart, and Michael Pazzani.**

University of California, Irvine
Information and Computer Science
Irvine, CA 92697

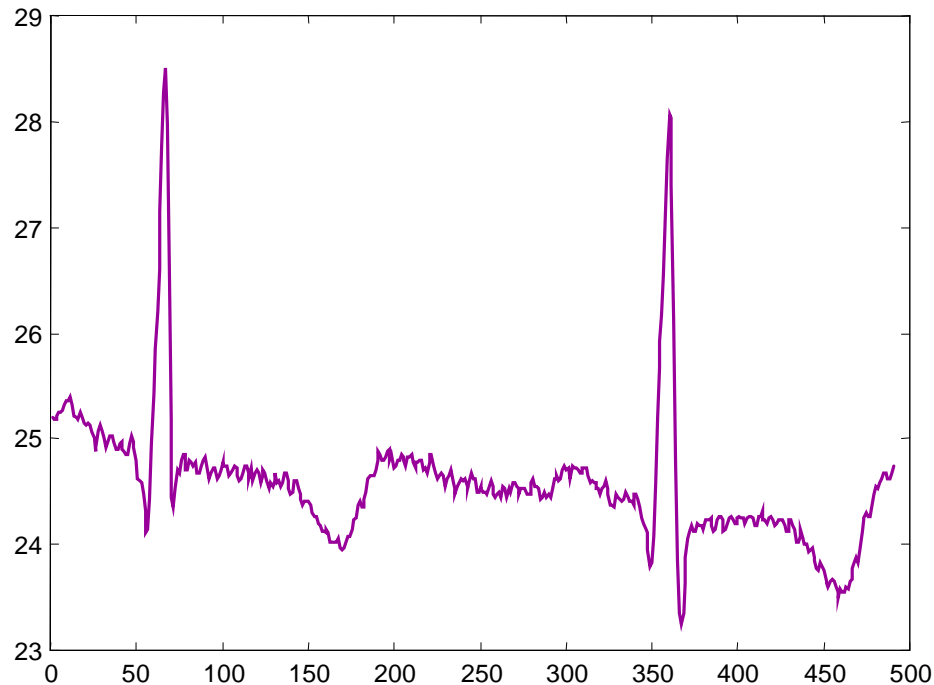
*Computer Science & Engineering Department
University of California - Riverside
Riverside, CA 92521

Outline of Talk

- What are time series?
- What is time series data mining?
- Which distance measure to use?
 - Euclidean Distance (simple but too brittle)
 - DTW (robust but too slow)
 - IDDTW (in the context of queries)
- Experimental Results
- Conclusions
- Future work

What are Time Series?

A time series is a collection of observations made sequentially in time.

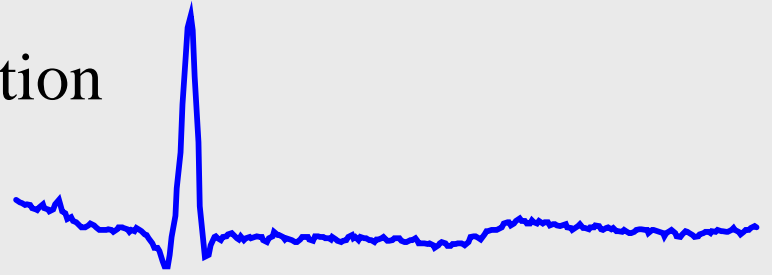


25.1750
25.2250
25.2500
25.2500
25.2750
25.3250
25.3500
25.3500
25.4000
25.4000
25.3250
25.2250
25.2000
25.1750
••
••
24.6250
24.6750
24.6750
24.6250
24.6250
24.6250
24.6750
24.7500

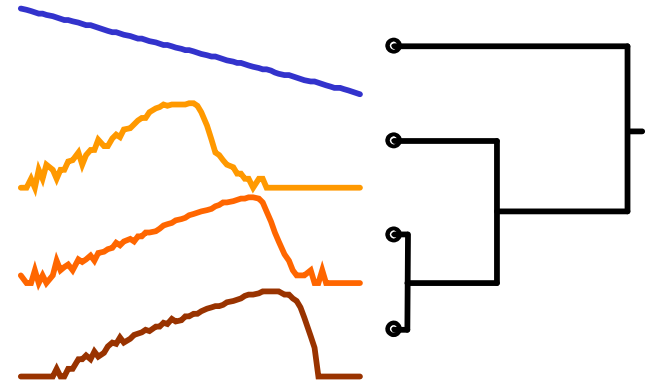
People measure things, and things (with rare exceptions) change.

Time Series Data Mining

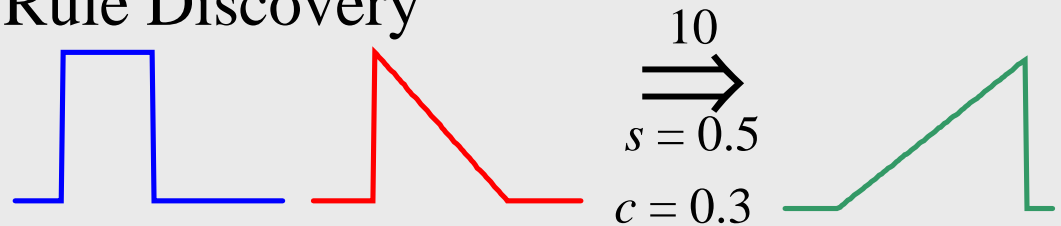
Classification



Clustering



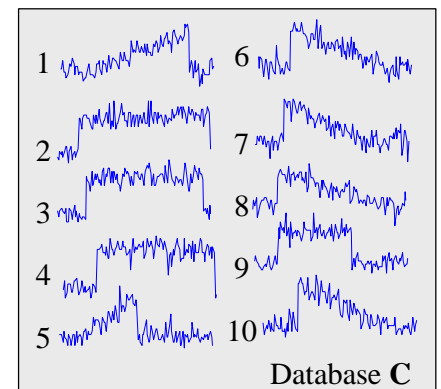
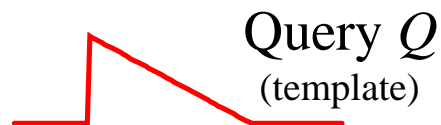
Rule Discovery



Size of Data:

- Typical Web-Log: 5 Gigabytes per week.
- Space Shuttle Database: 158 Gigabytes and growing.
- Macho Database: 2 Terabytes, updated with 3 gigabytes per day.

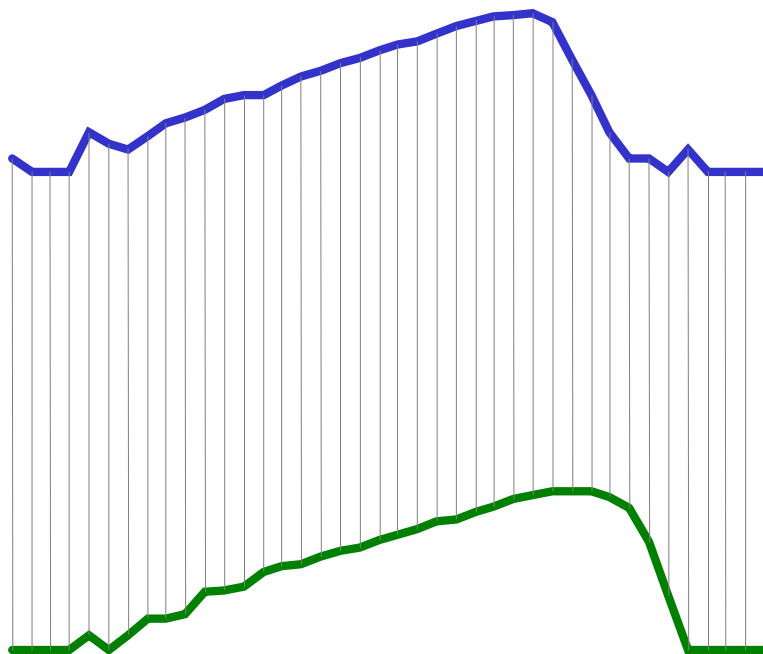
Query by Content



Most Time Series Data Mining Problems are Reduced to Measurements of Similarity

- What properties do we want from a similarity measure?
- We want a distance measure that is
 - Meaningful (captures essential features)
 - Efficient to calculate

Fixed Time Measures

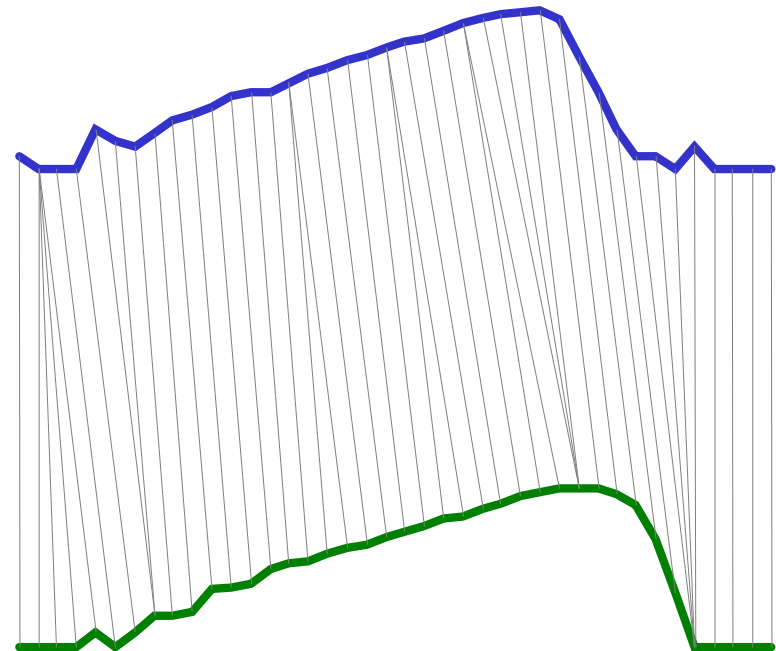


Fixed Time Axis

Sequences are aligned "one to one".

Euclidean Distance

Time Warped Measures



"Warped" Time Axis

Nonlinear alignments are possible.

Dynamic Time Warping

Cylinder-Bell-Funnel

$$c(t) = (6+\eta) \cdot X_{[a,b]}(t) + \varepsilon(t)$$

$$b(t) = (6+\eta) \cdot X_{[a,b]}(t) \cdot (t-a)/(b-a) + \varepsilon(t)$$

$$f(t) = (6+\eta) \cdot X_{[a,b]}(t) \cdot (b-a)/(b-t) + \varepsilon(t)$$

Where η and $\varepsilon(t)$ are drawn from a standard normal distribution $N(0,1)$, a is an integer drawn uniformly from the range $[16,32]$ and $(b-a)$ is an integer drawn uniformly from the range $[32, 96]$.



This dataset has been studied in a machine learning context by many researchers.

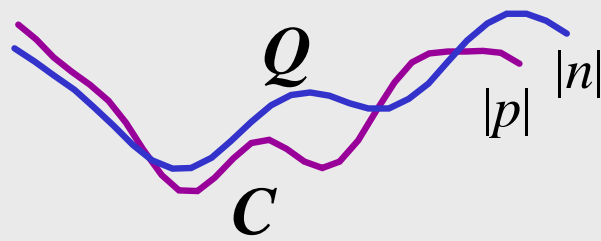
Kadous 1999; Manganaris,1997;
Saito 1994; Rodriguez 2000; Geurts
2001;

Classification experiment on Cylinder-Bell-Funnel dataset

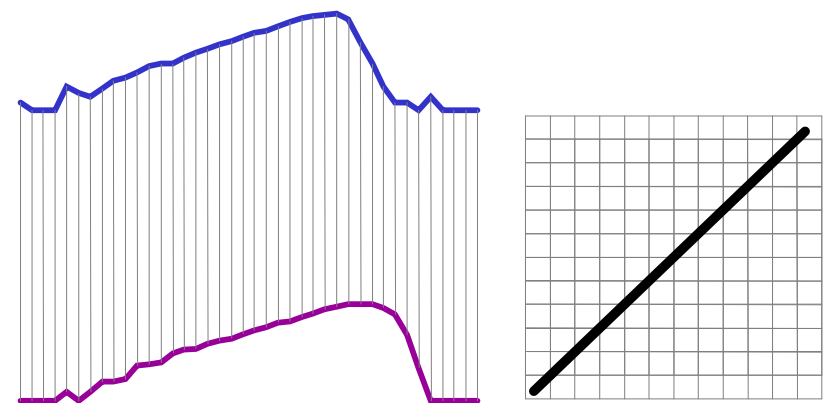
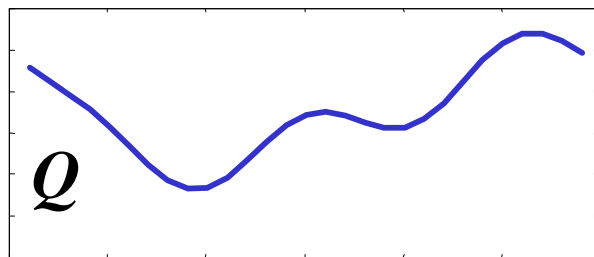
- Training data consists of 10 examples from each class.
- (One) Nearest Neighbor
- Algorithm “Leaving-one-out” evaluation, averaged over 100 runs

	Error Rate	Time
Euclidean	26.10%	1 sec
DTW	2.87%	3 hrs

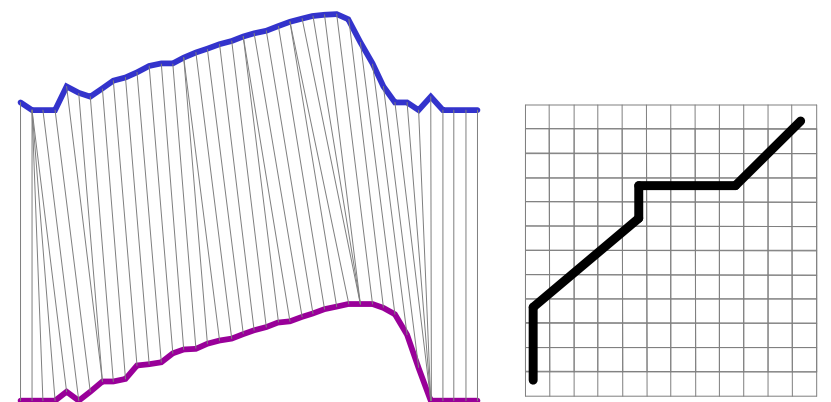
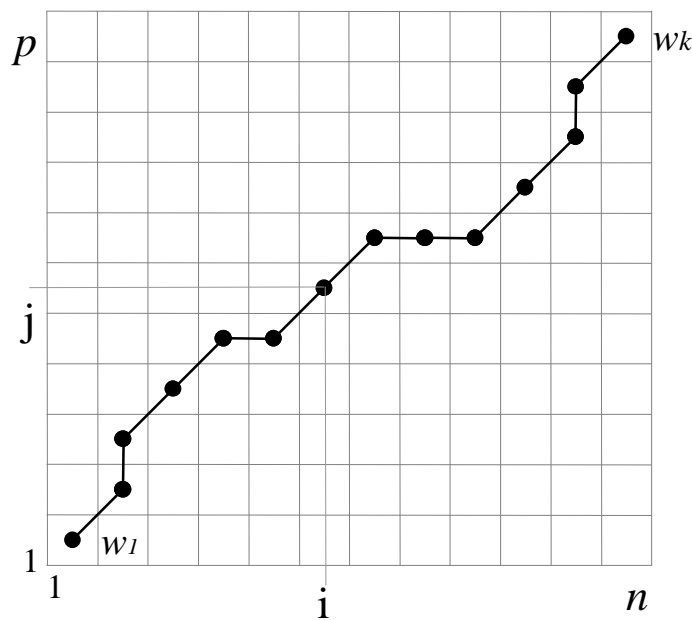
Computing the Dynamic Time Warp Distance



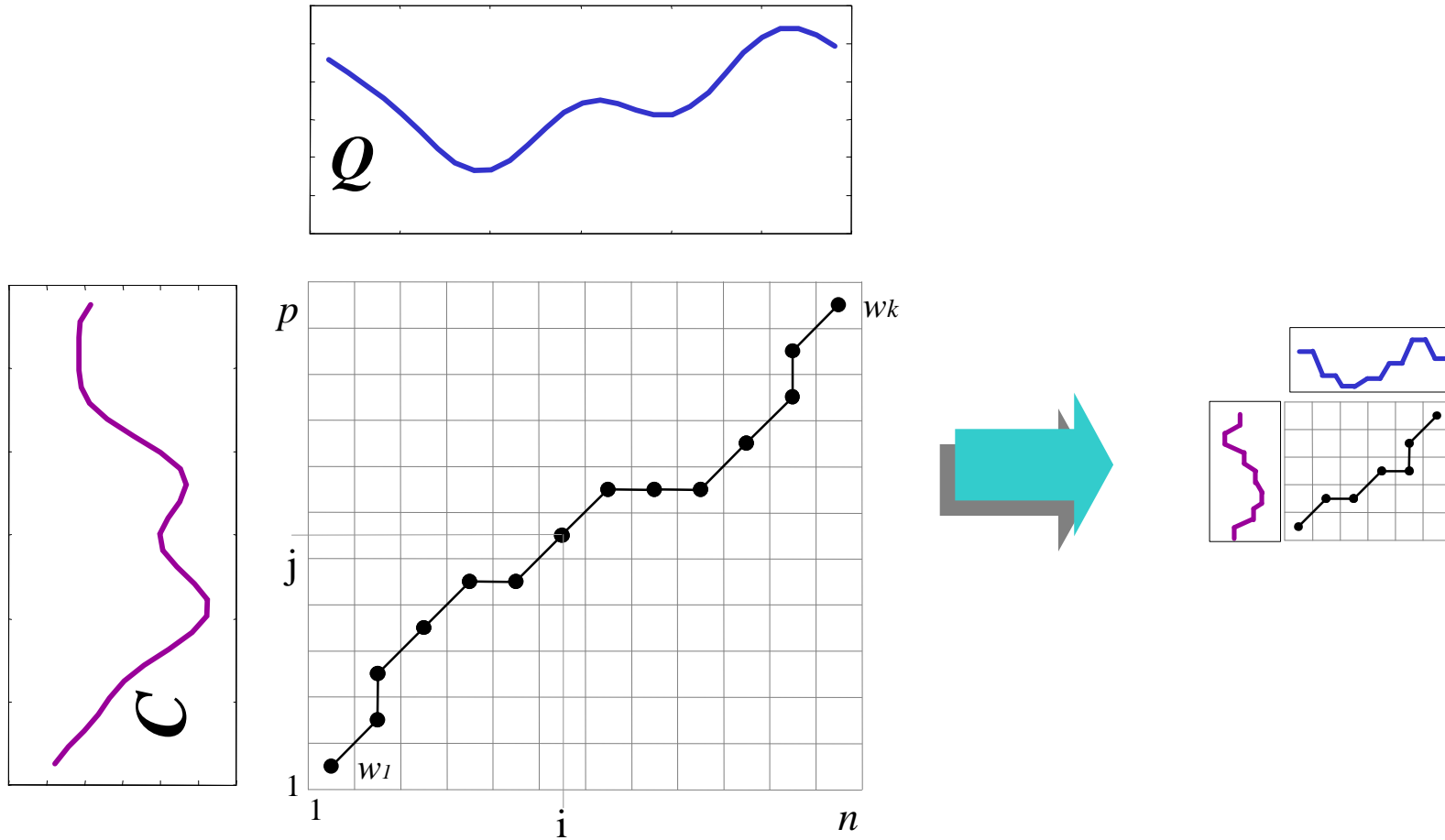
Note that the input sequences can be of different lengths



Euclidean distance is a special case of DTW

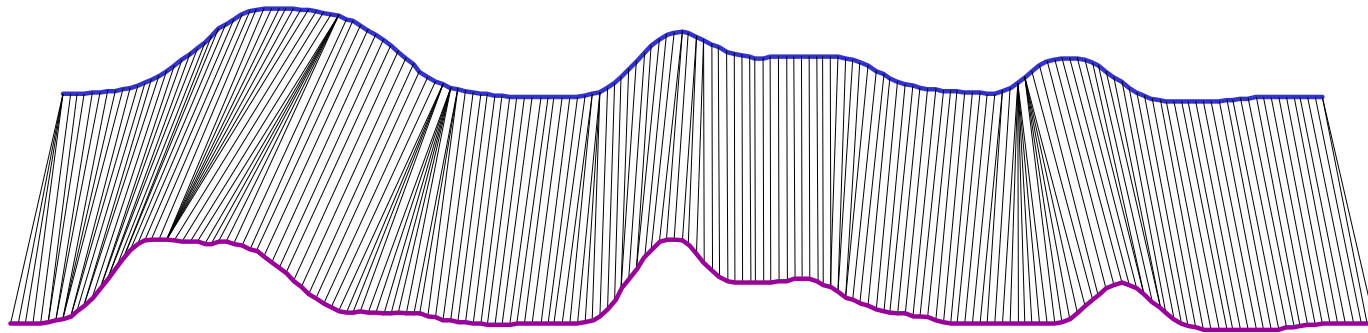


An Approximation to Dynamic Time Warp Distance I

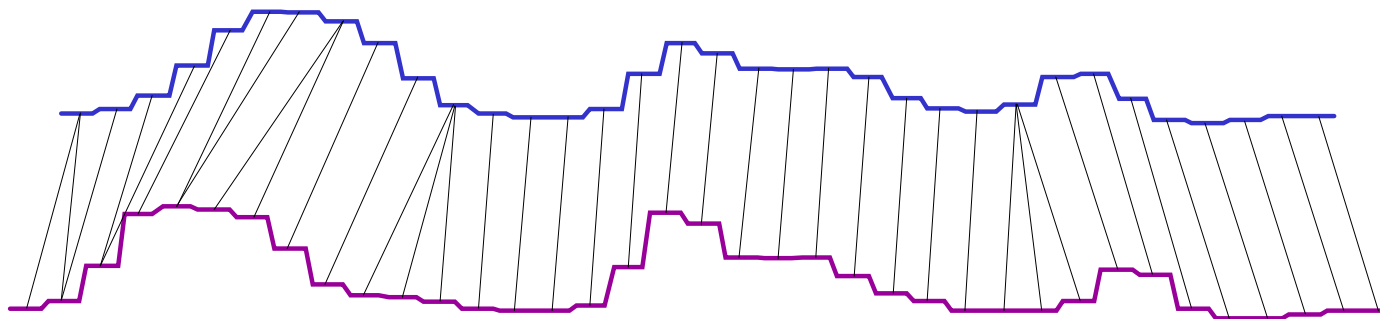


Simple Idea: Approximate the time series with some compressed or downsampled representation and perform DTW on the new representation.

An Approximation to Dynamic Time Warp Distance II



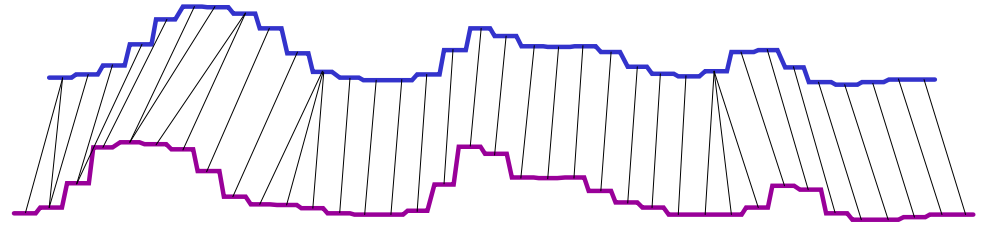
22.7 sec



1.3 sec

.. strong visual evidence to suggests it works well.

How to choose the compression rate?



- If the approximation is too fine we don't get much speedup (since the speedup is quadratic in the compression rate)
- If the approximation is too coarse, the accuracy will suffer greatly
- In general, it is up to the user to choose a compression rate that is appropriate to the task at hand, and this may be very difficult even for a domain expert.
- **Our Solution:** For some query, you begin with the coarsest approximation and do increasingly finer approximations until either you are sure that the sequence is a very poor match, or the “approximation” is so fine that it is the original data.

IDDTW

Iterative Deepening Dynamic Time Warping

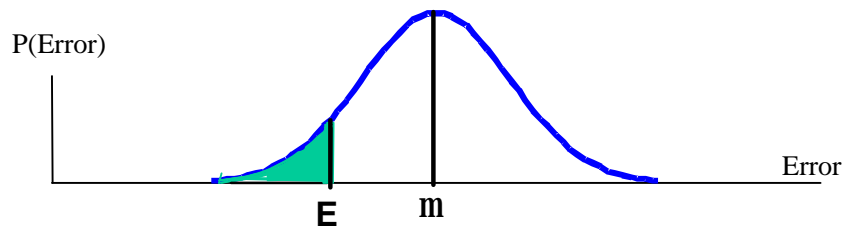
For every candidate in the database

1. Attempt DTW with a coarse approximation of the data
2. If the result is not promising, prune the candidate from consideration
3. Otherwise, attempt DTW with a finer approximation of the data, and goto 2

End

User must specify their tolerance for false dismissal, T .
(T is used to determine whether a candidate is promising)

Pruning the Candidate



$$P(\text{error} < E) = \frac{1}{s\sqrt{2\pi}} \cdot \int_{-\infty}^E e^{-\frac{(x-m)^2}{2s^2}} dx$$

$P(\text{error} < E) \leq T \Rightarrow$ go to the next level

$P(\text{error} < E) > T \Rightarrow$ go to the next candidate

Q : query sequence, C_i : candidate sequence

Assume: We have a distribution of their relative distance or error.

(Explained on the next slide)

Relative error (or distance):

$$E = \frac{DTW(\text{True } Q \text{ and BestSoFar}) - DTW(\text{Approximated } Q \text{ and } C_i)}{DTW(\text{True } Q \text{ and BestSoFar})}$$

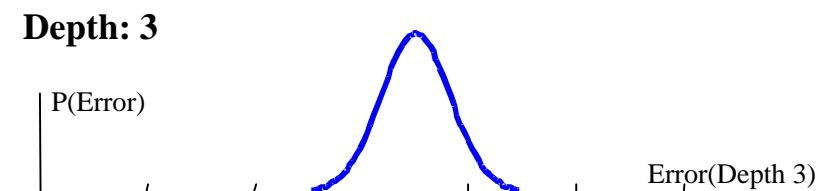
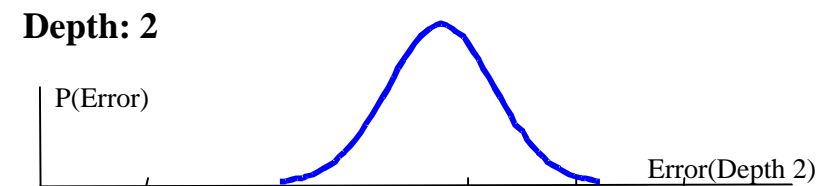
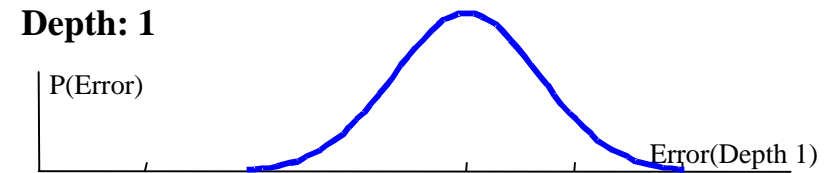
Building Models of Error Distribution

Where did the models come from?

For each level of approximation, we do random sampling to create a model.

- Randomly pick two sequences.
- Find the DTW distance between the two sequences and the DTW distance between their corresponding approximations.
- Record a distribution of their relative error.

$$Error(Depth_i) = \frac{DTW(True_Sequences) - DTW(Sequences_Approximated_at_Depth_i)}{DTW(True_Sequences)}$$



We can build a general model or a domain specific model.

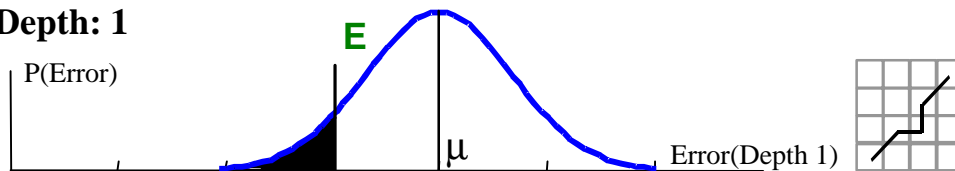
Assume we already have encountered a reasonably good match, which we call best-so-far **B** (say, $B = 30$)

The user chose 5% as the value of T

$$E = \frac{B - DTW(\text{Approximated } Q \text{ and } C_i)}{B}$$

$$P(\text{error} < E) = \frac{1}{s\sqrt{2\pi}} \cdot \int_{-\infty}^E e^{-\frac{(x-\mu)^2}{2s^2}} dx$$

Depth: 1



$$E = (B - 40) / B = -0.333$$

For every candidate in the database

1. Attempt DTW with a coarse approximation of the data
2. If the result is not promising, prune the candidate from consideration
3. Otherwise, attempt DTW with a finer approximation of the data, and goto 2

End

$$P(\text{error} < E) = 4.8\%, \quad P < T$$

Assume we already have encountered a reasonably good match, which we call best-so-far **B** (say, $B = 30$)

The user chose 5% as the value of T

$$E = \frac{B - DTW(\text{Approximated } Q \text{ and } C_i)}{B}$$

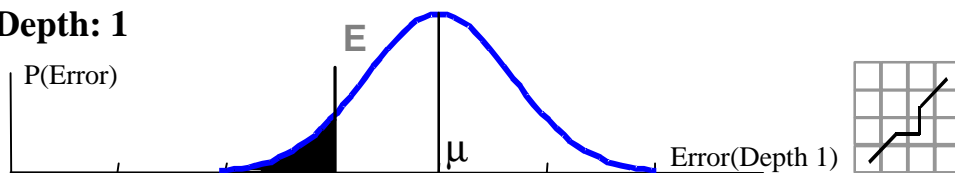
$$P(\text{error} < E) = \frac{1}{s\sqrt{2\pi}} \cdot \int_{-\infty}^E e^{-\frac{(x-\mu)^2}{2s^2}} dx$$

For every candidate in the database

1. Attempt DTW with a coarse approximation of the data
2. If the result is not promising, prune the candidate from consideration
3. Otherwise, attempt DTW with a finer approximation of the data, and goto 2

End

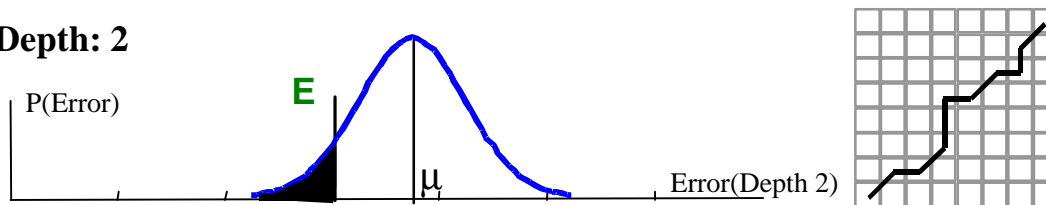
Depth: 1



$$E = (B - 40) / B = -0.333$$

$$P(\text{error} < E) = 4.8\%, \quad P < T$$

Depth: 2



$$E = (B - 37) / B = -0.233$$

$$P(\text{error} < E) = 3.5\%, \quad P < T$$

Assume we already have encountered a reasonably good match, which we call best-so-far **B** (say, $B = 30$)

The user chose 5% as the value of T

$$E = \frac{B - DTW(\text{Approximated } Q \text{ and } C_i)}{B}$$

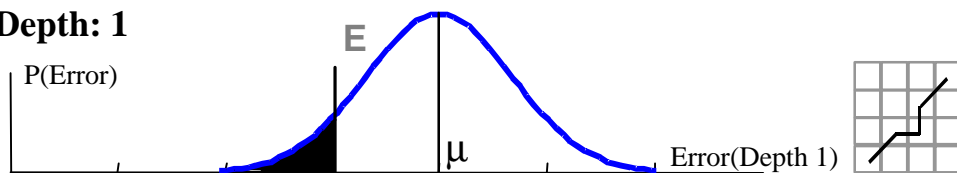
$$P(\text{error} < E) = \frac{1}{s\sqrt{2\pi}} \cdot \int_{-\infty}^E e^{-\frac{(x-\mu)^2}{2s^2}} dx$$

For every candidate in the database

1. Attempt DTW with a coarse approximation of the data
2. If the result is not promising, prune the candidate from consideration
3. Otherwise, attempt DTW with a finer approximation of the data, and goto 2

End

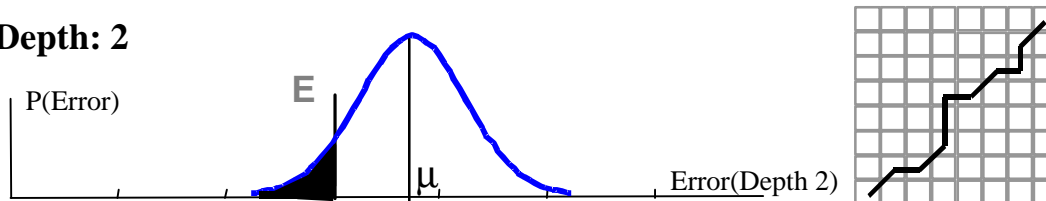
Depth: 1



$$E = (B - 40) / B = -0.333$$

$P(\text{error} < E) = 4.8\%$, $P < T$

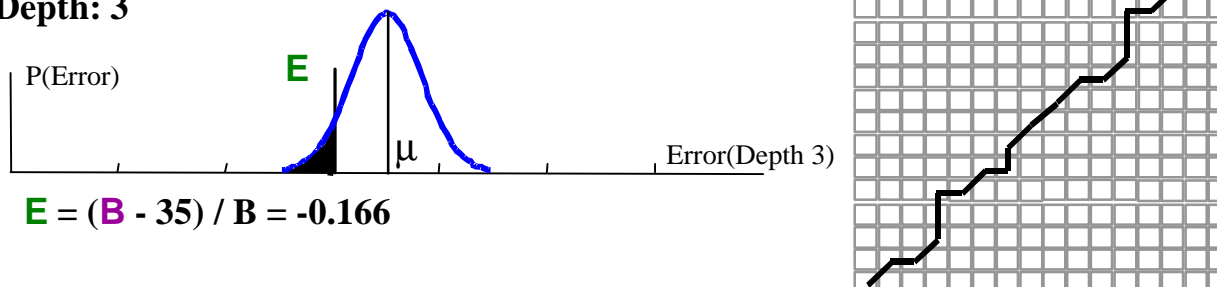
Depth: 2



$$E = (B - 37) / B = -0.233$$

$P(\text{error} < E) = 3.5\%$, $P < T$

Depth: 3



$$E = (B - 35) / B = -0.166$$

$P(\text{error} < E) = 6\%$, $P > T$

Time Complexity



$$S = N + \frac{1}{4} N + \left(\frac{1}{4}\right)^2 N + \dots + \left(\frac{1}{4}\right)^K N$$

$$S = N + \sum_{i=1}^K \left(\frac{1}{4}\right)^i N$$

$$S = \frac{4}{3} N, \text{ as } K \rightarrow \infty$$

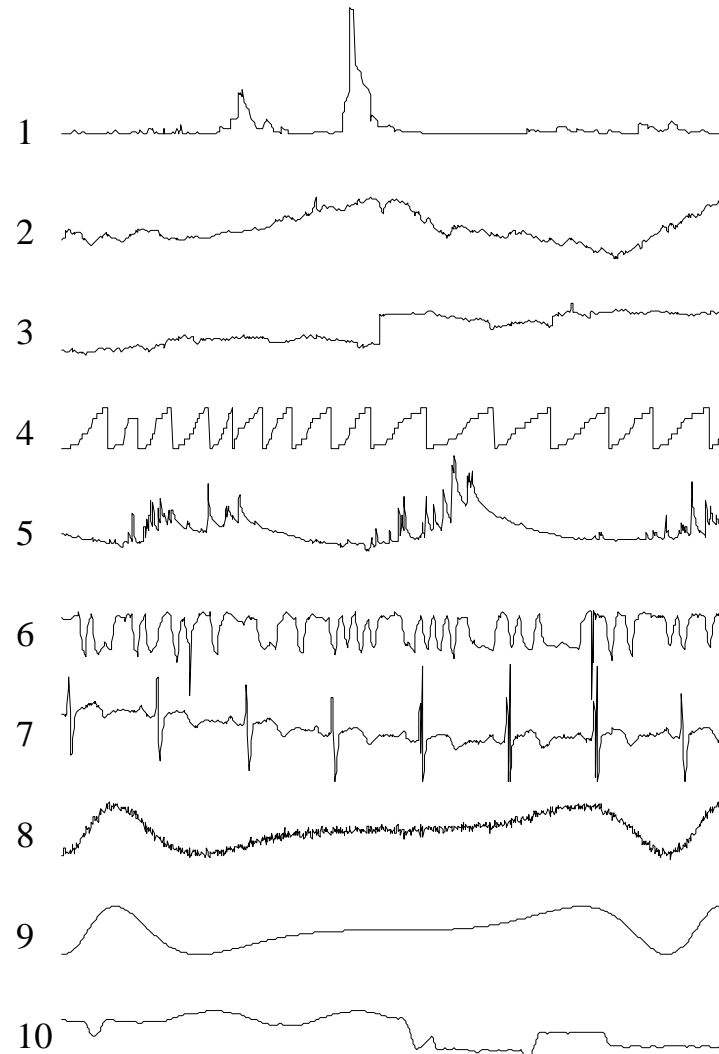
S = combined cost
 N = cost for true DTW
 K = max.depth

Computation time for Approximate DTW of the previous level is 1/4 of current

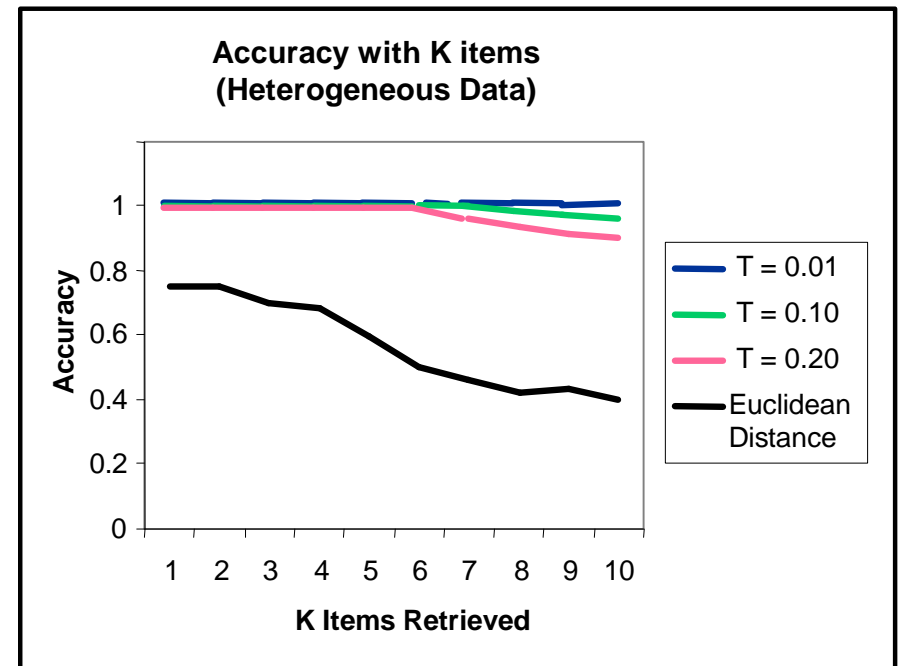
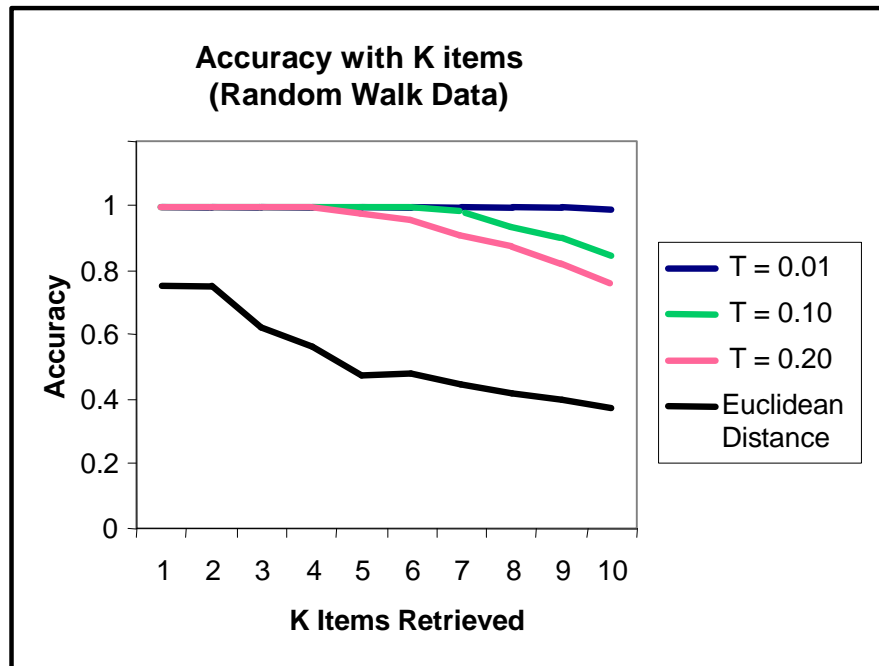
In the worse case, the amount of extra computation is at most 1/3 more than just doing DTW

Experimental Comparison

- We used two datasets
 - **Random Walk**
 - **Heterogeneous** (we combined 10 very diverse sets of time series data.)
- Compared Euclidean Distance, DTW, and IDDTW.
- 10-Nearest Neighbor search
- Measured accuracy and total computational time.



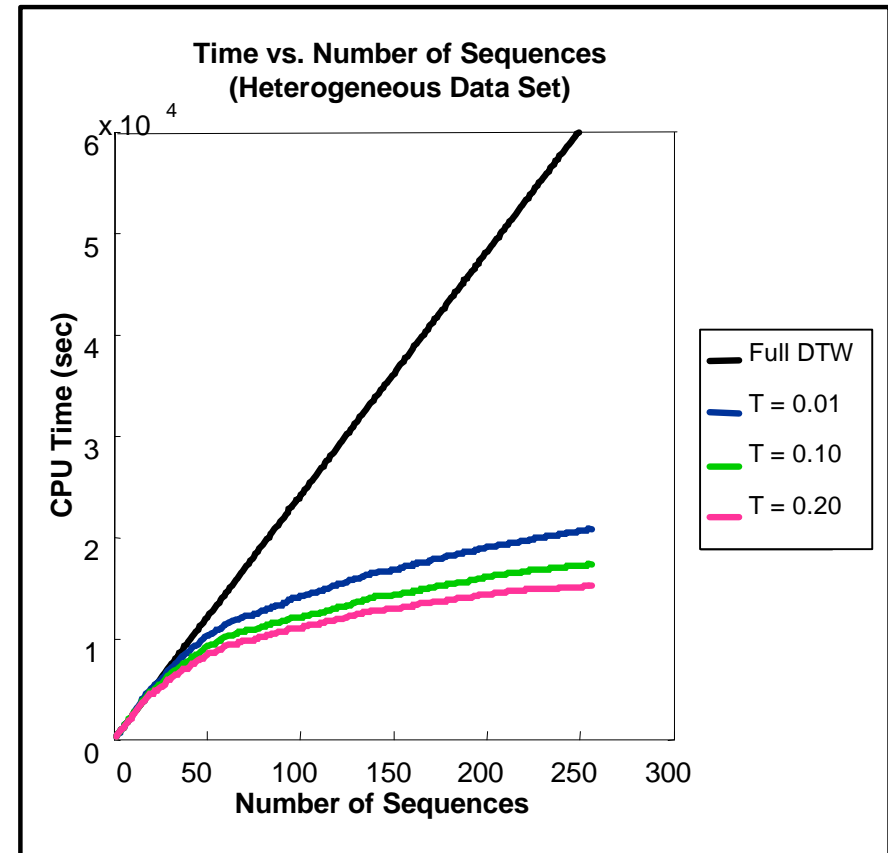
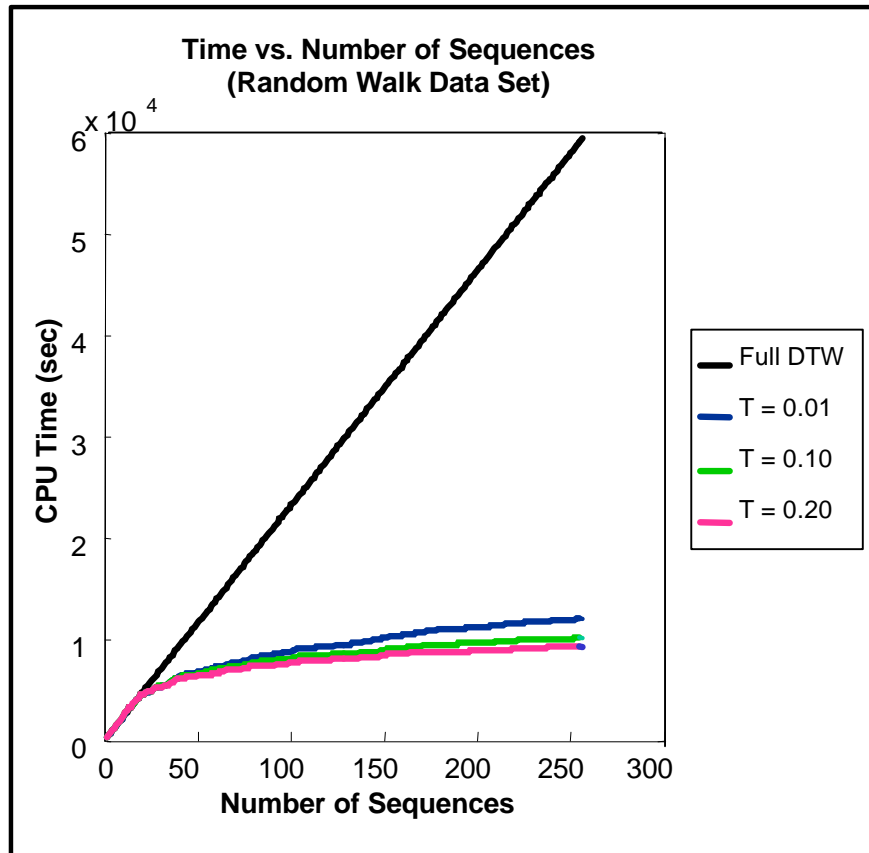
Experimental Results: Accuracy



Random Walk Data (K=10)			
Euclidean Distance	IDDTW @ T=1%	IDDTW @ T=10%	IDDTW @ T=20%
0.3750	0.9875	0.8500	0.7625

Heterogeneous Data (K=10)			
Euclidean Distance	IDDTW @ T=1%	IDDTW @ T=10%	IDDTW @ T=20%
0.4000	1.000	0.9625	0.9000

Experimental Results: Speedup



Random Walk Data (K=10)			
Full DTW	IDDTW @ T=1%	IDDTW @ T=10%	IDDTW @ T=20%
16h 29m	3h 20m	2h 48m	2h 33m

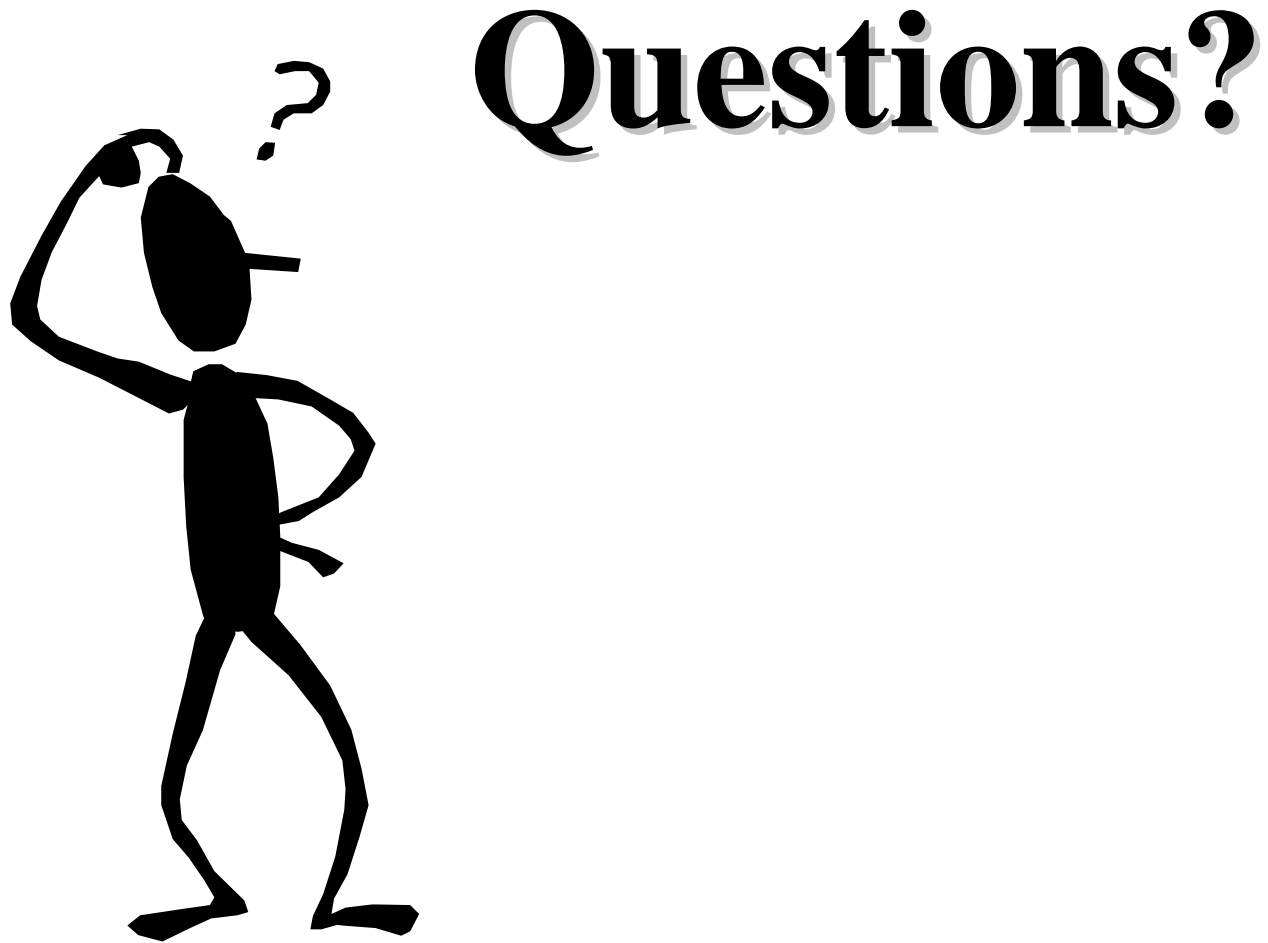
Heterogeneous Data (K=10)			
Full DTW	IDDTW @ T=1%	IDDTW @ T=10%	IDDTW @ T=20%
17h 8m	5h 46m	4h 47m	4h 12m

Conclusions

- We have seen that there is great interest in time series data mining, and most algorithms require similarity to be measured.
- We have shown that the most common method of measuring similarity - Euclidean distance - is fast to calculate, but very brittle. In contrast, DTW is very robust, but expensive to compute.
- We have introduced a technique - IDDTW - that allows the user to trade off a small amount of precision for very large gains in speed.

Future Work

- A more detailed analysis of our approach
- Extensions to other similarity search problems that feature distance measures that are expensive, but can be approximated at different levels of precision. (e.g. string edit distance)



Reproducible Results Statement: In the interests of competitive scientific inquiry, all datasets and code used in this work are available.