

Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Database

IBM Almaden Research Center

Rakesh Agrawal King-Ip Lin

Harpreet S. Sawhney Kyuseok Shim

Outline

- Problem Statement
- Model
- Data Structure and Algorithms
- Experiments
- Summary

Problem Statement

- Identify companies with similar growth pattern
- Discover stocks/mutual funds with similar price movement
- Find similar website access patterns
- Determine products with similar selling patterns

Previous Methods

- The distance measure (Euclidean distance) can be quite sensitive to few outliers
- no amplitude scaling nor offset translation
- cannot ignore unspecified portions of sequences while matching sequences

Model

- what is a **sequence**
- what is a **subsequence**
- what is the distance measure
- conditions for two sequences to be similar
- conditions for two subsequences to be similar

Sequence Similarity

- $S_i < S_k$ and $T_i < T_k$, for $1 \leq i < k < m$.
- There exists some scale C and some translation T so that for all subsequences,
 $T(C(S_i)) = T_i$
- The fraction of matching length to the total length of S and T is greater than some threshold e .

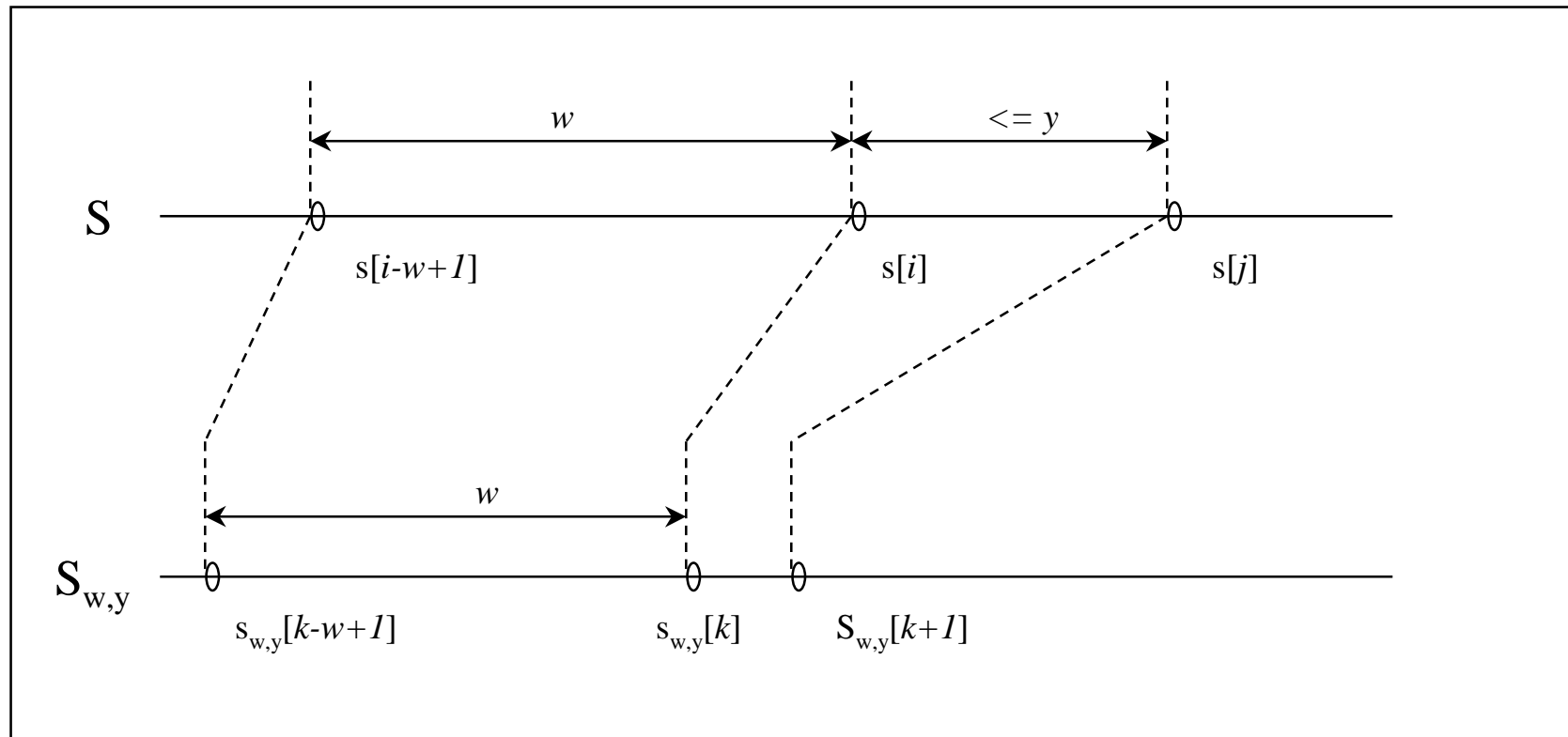
Subsequence Similarity

- Two subsequences are similar if one lies within an envelope of a specified width e
- (e, w, y) -similar: There exists some (w, y) -projections P_1, P_2 such that for all i ,
 $P_1(S[i]) - P_2(T[i]) \leq e$

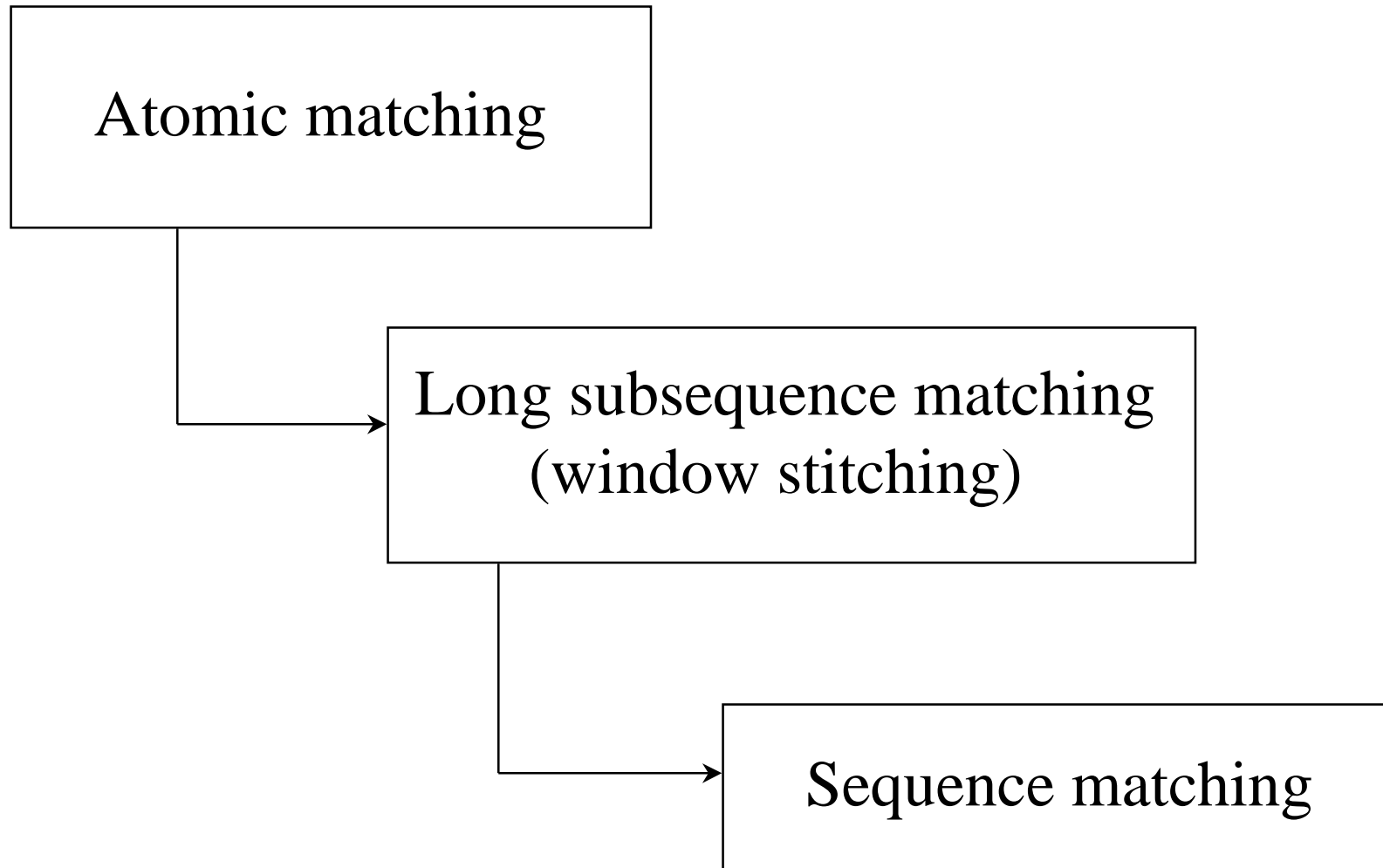
(w, y) -Similarity

- all elements in S_y are also in S , and they are in the same order
- if $S[i]$ and $S[j]$ are two elements in S corresponding to the two consecutive elements $S_y[k]$ and $S_y[k+1]$ in S_y , then $j-i \leq y$
- if $S_{w,y}[i]$ and $S_{w,y}[i+1]$ are such that their corresp. elements in S are not consecutive, then the elements of S corresp. to $S_{w,y}[i-w+1] \dots S_{w,y}[i-1]$ are consecutive

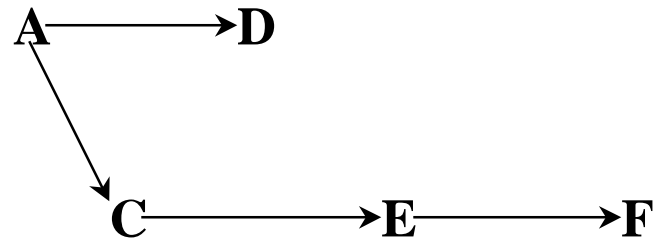
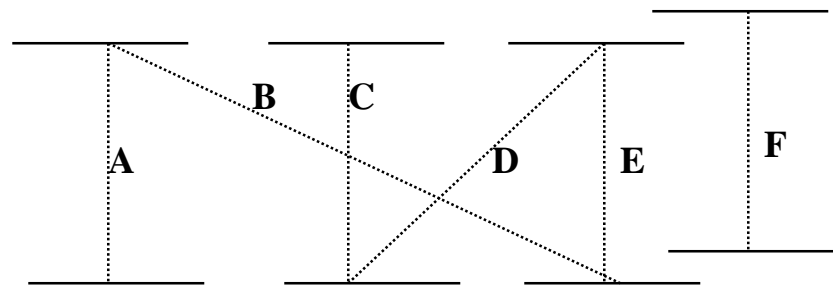
Subsequence Similarity



Flowchart



Example



Atomic Matching

- Find all pairs of gap-free subsequences of length w , called windows, in S and T
 - normalize the sequence values within each window to a range $(-1, +1)$
- Given a set of points in w -dimensional space, find all pairs of points within a distance of e from each other

Atomic Matching

- Use a multi-dimensional indexing structure to store the points and then use self-join() to retrieve all pairs of matching windows
 - normalized point
 - its coordinates
 - sequence-id of the corresp. sequence
 - starting point of the window
 - scale/translation factor (for window stitching)

Atomic Matching

- Multi-dimensional indexing Structure:
hashing, grid-file, R-tree etc
 - Dimensionality
 - Self-join()
 - Data values

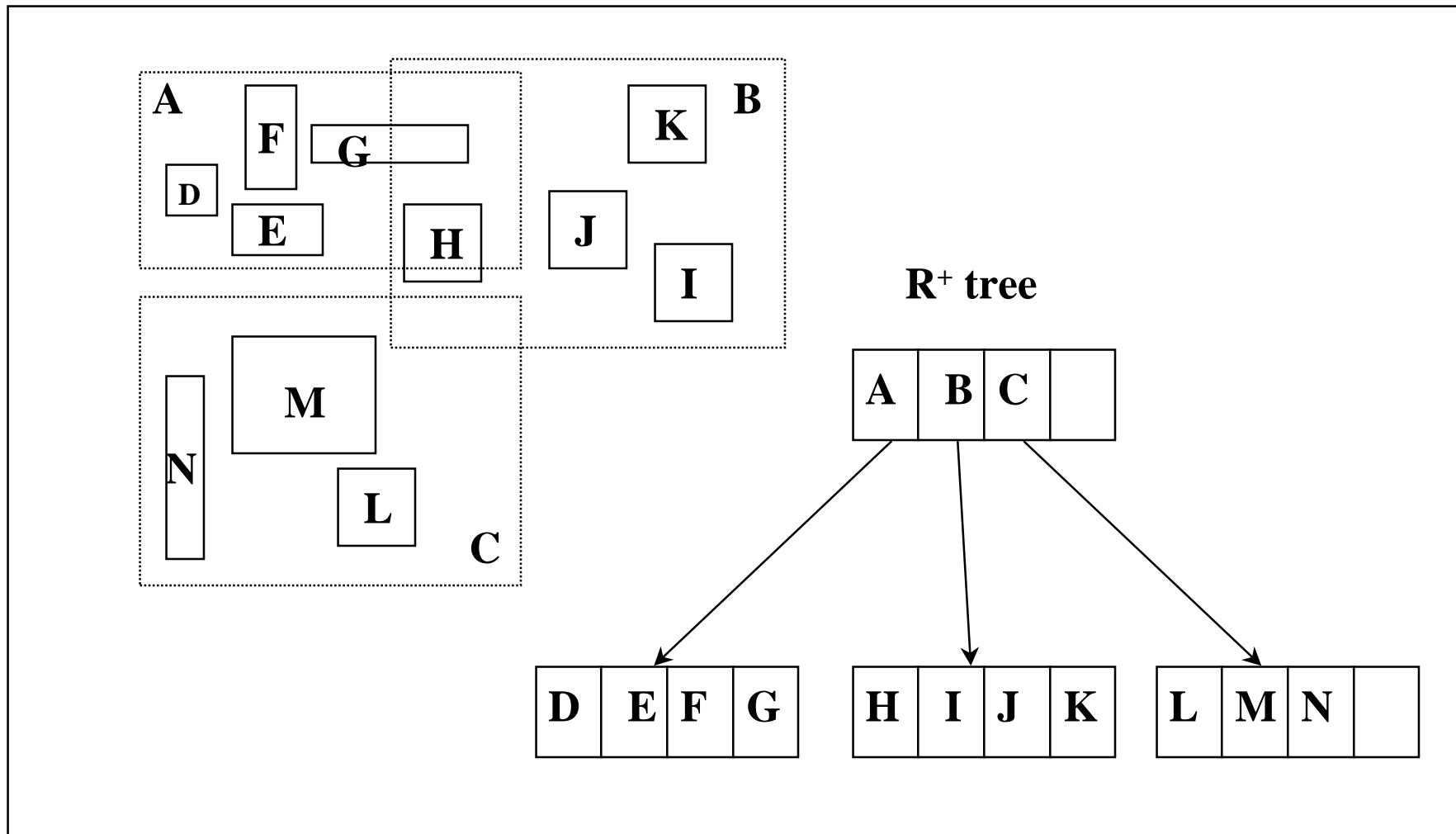
Atomic Matching

- The winner is

R-tree family

- R^* and R^+ tree

Atomic Matching



Atomic Matching

```
SelfJoin (node, path, e)
{
  if (node.type == non-leaf node)
    {
      forall (child in node.children) do
        output=SelfJoin (child, node+path, e)
    }
  else
    {
      output += Join (node, node, e)
      forall (leaf in intersect (node, path, e)) do
        output += Join (node, leaf, e);
    }
  return output
}
```

Long Subsequence Matching

(window stitching)

- find the longest path in an acyclic graph
 - vertex: each pair of matching window
$$V = (S_i, T_i)$$
 - edge: there is an edge between vertex i and j iff
 - $\text{first}(S_i) < \text{first}(S_j) \wedge \text{first}(T_i) < \text{first}(T_j)$
 - (the corresp. windows in the two matches do not overlap and the gap between them is less than y) or (the amount of overlap between S_i and S_j is the same amount of overlap between T_i and T_j in T)
 - label: $\langle L_{ij}, F_{S_i}, L_{S_j}, F_{T_i}, F_{T_j} \rangle$ $(L_{ij} = L_{S_i} - L_{S_j} + L_{T_i} - L_{T_j})$

Long Subsequence Matching

(subsequence ordering)

- Given pairs of similar subsequence, determine the maximal length match in the two sequences.
- Finding the longest path in an acyclic graph
 - arc: if the corresp. subsequence does not overlap and come later
 - weight: sum of the length of the subsequences.

Summary

- interesting idea
- great algorithms
- implementation is challenging

two sequences are similar if they have enough non-overlapping time-ordered pairs of subsequence that are similar.

if one lies within an envelope of a specified width around the other, ignoring outliers.

We allow amplitude of one of the two sequences to be scaled by any suitable amount and its offset adjusted .

normalization formula

$$w[i] = (w[i] - (wmin + wmax)/2) / ((wmax - wmin)/2)$$