PRINCIPLES OF DATA ORGANISATION

Spatial join

MOTIVATION

- Key, pointer pairs ~ index
- Join on multiple query conditions
- Spatial data
  - We focus only on intersection joins = we want to join objects that intersect
  - E.g. all pairs of rivers and cities that intersect

NON-SPATIAL JOIN

Most of the standard relational join algorithms are not suitable for spatial data, because the join condition involves multidimensional spatial attribute

- **Nested loop join**
  The only of the standard relational join algorithms applicable also to spatial data
- **Sort-merge join**
  Multi-dimensional data do not preserve proximity, so this method (used as is) is not applicable to spatial data
- **Hash join**
  Equi-joins (joins on quality) rely on grouping objects with the same value, which is not possible for spatial objects since these have an extent

SPATIAL JOIN

Given two sets of **multi-dimensional objects** in Euclidean space, a spatial join finds **all pairs** of objects **satisfying** a given **spatial relation** between the objects, such as intersection

Spatial overlay join (general spatial join)

- The data set can consist of **general spatial objects** (points, lines, polygons)
- The data sets can have **more than two dimensions**
- The relation between pairs of spatial objects can be any spatial relation
  - Nearness, enclosure, direction, …
- There can be more sets in the relation (**multiway spatial join**) or one set joined with itself (**self spatial join**)

SPATIAL JOIN

Our focus: **Simplified spatial join**

- Given two sets of rectangles R and S, find all of the pairs of intersecting rectangles between the two sets, i.e. $\{(r,s): r \cap s \neq \emptyset, r \in R, s \in S\}$

FILTER-REFINE POLICY

- The objects to be searched for can be very complex
  - Testing of the join condition (spatial predicate) itself can be highly time consuming
  - Not many objects fit into main memory
- Spatial objects are approximated using simple spatial objects
  - Like in R-trees where we used MBRs

Filter

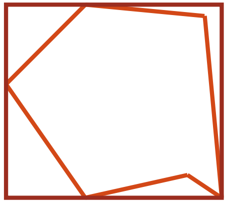
- The spatial join is conducted using the objects' approximations => candidate set

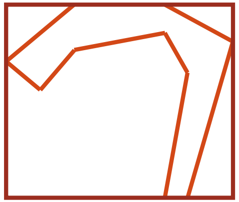
Refine

- Pairs which pass the filter are tested for the spatial predicate using their full spatial representation

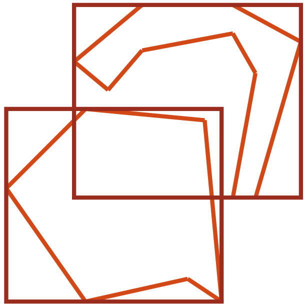
APPROXIMATIONS

- The most common approximation is minimum bounding rectangle (MBR) – the smallest rectangle fully enclosing a given object whose sides are parallel to the axes.
- Dead space is the amount of space covered by the approximation but not the approximated object
- Large dead space areas can lead to a higher false hit rate in the filtering stage, so the approximation should aim to minimize dead space





Large dead area

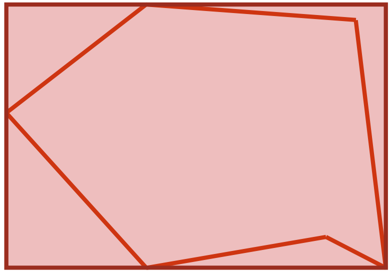


False hit

APPROXIMATIONS

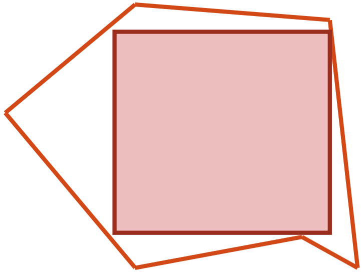
Conservative

- Every point of the object is also point of the approximation
- MBR, minimum bounding polygon, minimum bounding circle/ellipse, …
- Can have false positives



Progressive

- Every point of the approximation is also point of the object
- Maximum nested rectangles, circles, …
- Can have false negatives = we do not find all intersections



APPROXIMATIONS : OBJECT DECOMPOSITION

- Minimizes dead space by decomposing an object into disjoint fragments with their own MBRs
- After refine step and extra filtering stage is needed to remove duplicities