Closest Pair Queries in Spatial Databases

Antonio Corral, Yannis Manolopoulos,
Yannis Theodoridis (*), Michael Vassilakopoulos

(*) speaker

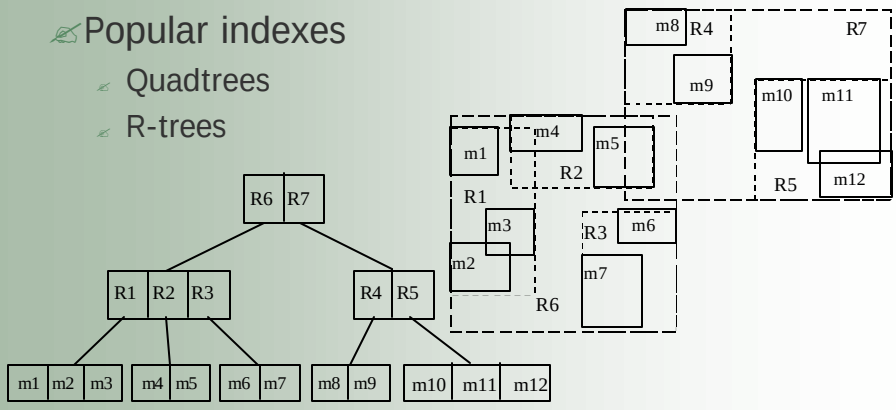
2000 ACM SIGMOD Conference
Dallas, TX, May 2000

Outline

- Spatial Data and Spatial Queries
- CP Queries
- CP Algorithms
- Performance Comparison and Guidelines
- Conclusions

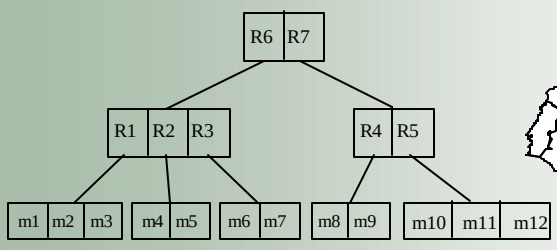
Spatial Data ...

- Maps, networks, scientific data, etc.
- Popular indexes
  - Quadtrees
  - R-trees



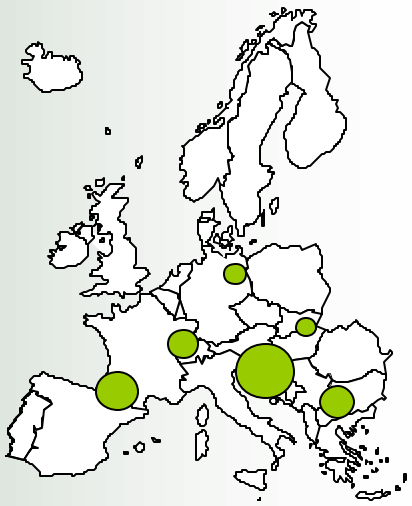
... and Spatial Queries

- Selection queries
  - point/range
  - topological/direction
  - k-nearest-neighbor



... and Spatial Queries (cont'd)

- Join queries
  - depth-first traversal (Brinkhoff et al., SIGMOD'93)
  - breadth-first traversal (Huang et al., VLDB'97)
  - multi-way joins (Papadias et al., PODS'99)
  - similarity / spatial distance joins (Koudas and Sevcik, ICDE'98; Faloutsos et al., SIGMOD'00)

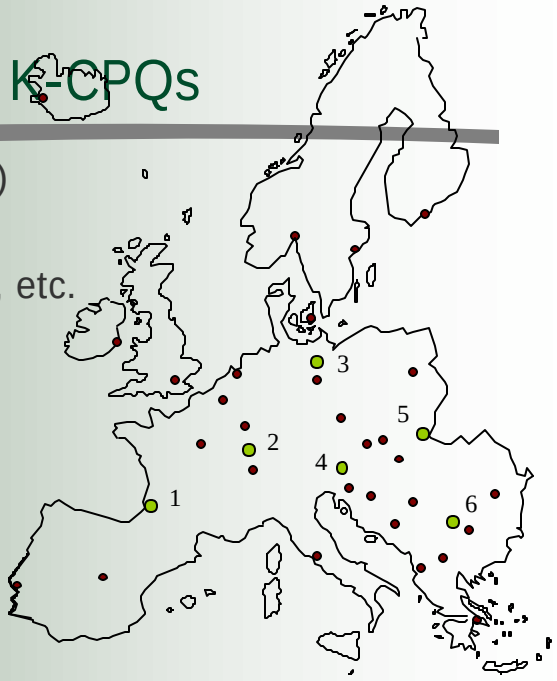


... and Spatial Queries (cont'd)

- K-closest-pair queries: a combination of NN and join queries
  - -incremental- distance join (Hjaltason and Samet, SIGMOD'98)

Example of K-CPQs

- 1-CP: (Berlin, 3)
- 2-CP: (Sofia, 6)
- 3-CP: (Bern, 2), etc.



Definition and useful metrics

Definition:

Let $P=\{p_1, \ldots, p_{NP}\}$ and $Q=\{q_1, \ldots, q_{NQ}\}$.

As 1-CP we define

(p_z,q_l): $dist(p_i,q_j) \leq dist(p_z,q_l) \forall p_i \in P \wedge \forall q_j \in Q$

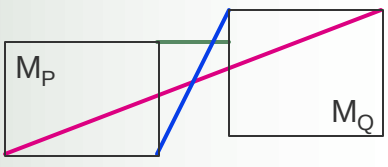
Useful Metrics (for R-tree nodes):

$\forall (p_i,q_j)$:

$MINMINDIST(M_P,M_Q) \leq dist(p_i,q_j) \leq MAXMAXDIST(M_P,M_Q)$

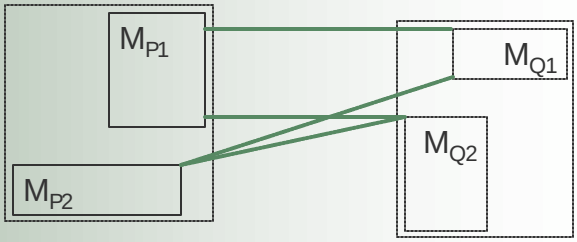
$\exists (p_i,q_j)$:

$dist(p_i,q_j) \geq MINMAXDIST(M_P,M_Q)$



5 CP-algorithms

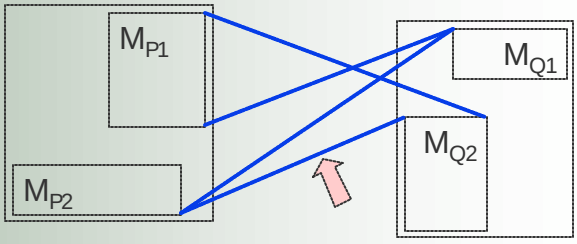
- **Naïve algorithm**
  - makes no use of useful metrics, i.e. propagate downwards all possible pairs of R-tree paths
- **Exhaustive algorithm (EXH)**
  - prune pairs of nodes with MINMINDIST > current minimum (say T)



5 CP-algorithms (cont'd)

- **Simple recursive algorithm (SIM)**
  - among candidate pairs of nodes, find the one with the minimum MINMAXDIST, if min(MINMAXDIST) ? T then update T, and propagate downwards
    i.e. prune pairs of nodes with MINMINDIST > (updated) T

dist(M_{P2} ,M_{Q2}) serves as the threshold for MINMINDIST

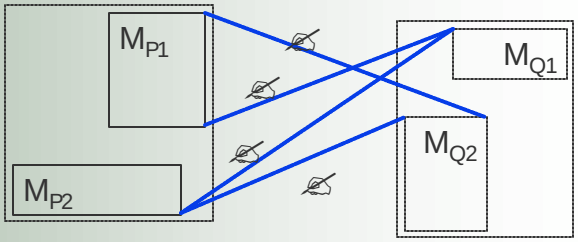


5 CP-algorithms (cont'd)

Sorted Distances recursive algorithm (STD)

- as the previous algorithm, but a priority is given to the pair with the minimum MINMAXDIST

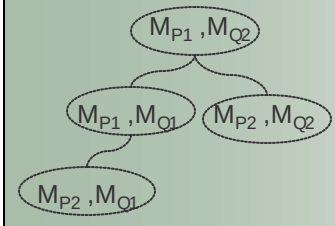
first check (M_{P2} ,M_{Q2}) paths, then (M_{P1} ,M_{Q1}) paths, and so on

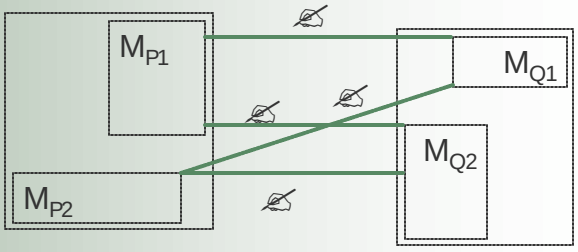


5 CP-algorithms (cont'd)

HEAP algorithm

- **non-recursive**: maintain a heap including pairs of nodes according to MINMINDIST. The pair on top of the heap is the next candidate for visiting (if MINMINDIST(pair_on_the_top) ? T then STOP)





More on CP-algorithms

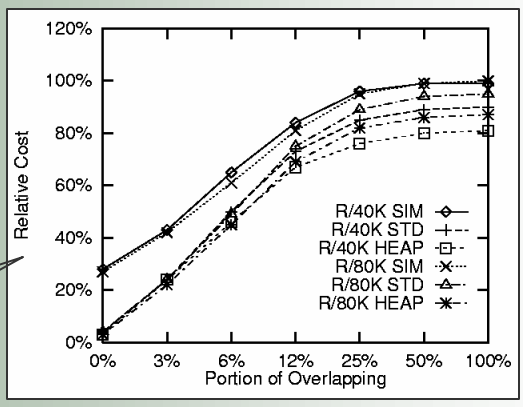
Issues addressed

- Treatment of ties on MINMINDIST values
  - the pair including the node with the largest area is a good choice
- Treatment of R-trees with different heights
  - proposal of the novel (and promising) 'fix-at-root'
- Extending to K-CPs
  - Maintain a K-heap including pairs of points according to their distance (all algorithms)
  - make use of MAXMAXDIST metric while pruning unnecessary paths (all but naïve and exhaustive)
- Point-to-point comparison with related work

Performance Comparison

- Parameters involved
  - the effect of buffer capacity
  - the effect of overlap between the two workspaces
- General guidelines
  - HEAP for overlapping workspaces and zero buffer
  - STD for buffer of reasonable size (> 4 pages)
  - Large K values do not affect the relative performance
- Comparison with related work
  - HEAP and STD improve performance up to 20% and 50%, respectively.

Some charts ...

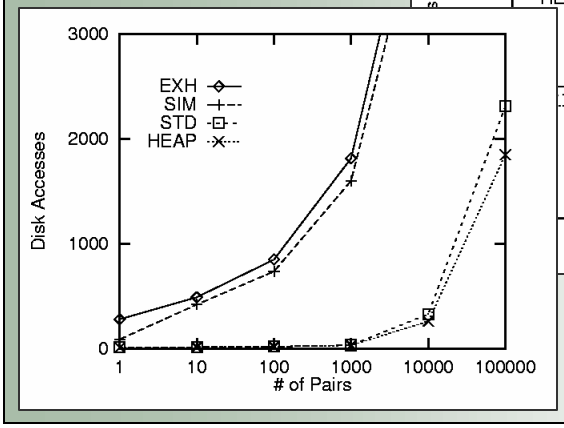


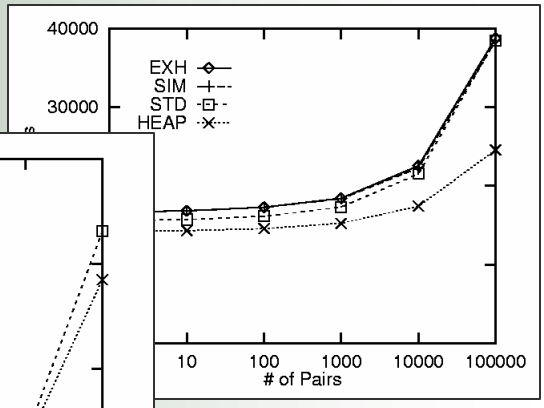
with respect to EXH

The effect of overlap cannot be neglected !

Some charts (cont'd)

0% overlap





100% overlap

Conclusions

- Although important, CPQs have not gained special attention in database literature
- Defined three useful metrics
  - MINMINDIST, MAXMAXDIST, MINMAXDIST
- Proposed and evaluated four algorithms
  - three recursive (EXH, SIM, STD)
  - one iterative (HEAP)
- First that address the effect of overlapping between the two workspaces
- Future work: ‘self-CPQ’ and ‘semi-CPQ’