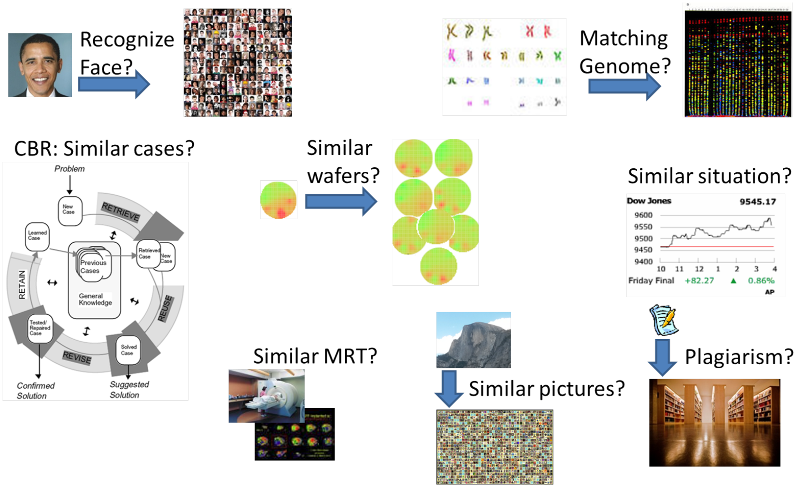
Optimizing Similarity Search in the M-Tree

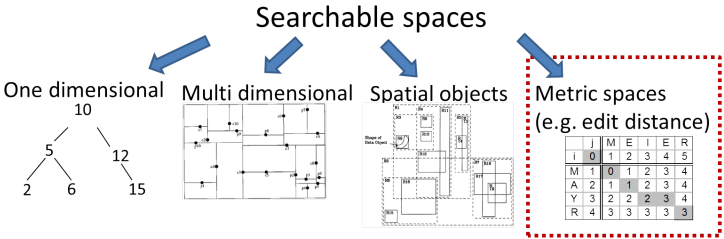
Steffen Guhlemann [steffenguhlemann@hotmail.com],
Uwe Petersohn [Uwe.Petersohn@tu-dresden.de], and
Klaus Meyer-Wegener [klaus.meyer-wegener@fau.de]

09.03.2017

Examples: Similarity search in metric spaces



Searchable spaces



Metric spaces

- No (common) structure, only distance function obeying metric axioms
  - *Positivity*: $\forall x, y \in O : x \neq y \Rightarrow d_{x,y} > 0$,
  - *Symmetry*: $\forall x, y \in O : d_{x,y} = d_{y,x}$,
  - *Triangle inequality*: $\forall x, y, z \in O : d_{x,z} \leq d_{x,y} + d_{y,z}$.
- Curse of dimensionality
- Expensive distance computation
- Single data item representation consumes much memory

State of the art – Index structures for similarity search in metric spaces

Requirements

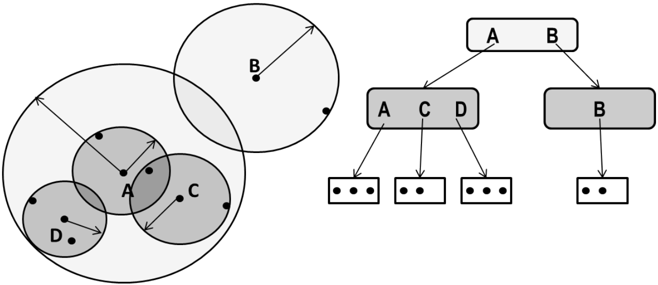
- Persistent storage of data in arbitary domains
- Linear storage complexity $O(N)$
- Efficient (sublinear) incremental changes and queries (range, kNN)
- Possibility for domain specific optimizations
- Query performance comparable to data of the intrinsic dimensionality

Existing Index structures

- Multiple existing structures
- Most have serious drawbacks, e.g.
  - BK-Tree, Fixed Query Tree and derivatives only handle discrete distance functions
  - AESA and it's derivatives have a quadratic storage complexity of $O(N^2)$
  - Vantage-Point-Tree and D-Index are static structures (no incremental inserts/deletes)
  - The Bisector Tree does not allow to minimize I/O
  - Some structures only claim to be metric access structures but actually only work in euclidian vector spaces (e.g. M^+-Tree and BM^+-Tree)
- Best baseline (fulfills most requirements): M-Tree and it's variants

The M-Tree (Ciaccia et al. 1997, Zezula et al. 2006)

Hierarchical space decomposition into hyperspherical nodes.



A **leaf node** consists of:

- Key value
- Distance to parent node
- Possibly pointer to full data set

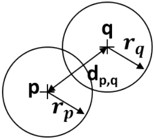
An **inner node** consists of:

- Key value
- Pointer to child nodes
- Radius of subtree
- Distance to parent node

Improved search algorithms – Existing algorithms and optimizations

Basic principle:

- Recursive tree descend – test intersection of node and query hypersphere

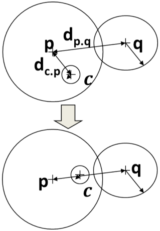


Optimization idea:

- $d_n^{\perp}$ based on (expensive) dist.calculation: $d_{n,q}$
- First try heuristic bound $d_{n,relaxed}^{\perp} \leq d_n^{\perp}$ using $\perp_n \leq d_{n,q}$
- If sufficient to exclude n, avoided calculation of $d_{n,q}$

Examples of heuristics:

- Classic M-Tree: precomputed distance to parent node
- CM-Tree (Aronovich and Spiegler 2007): precomputed bilateral child distances (nodewise AESA)
- Domain specific heuristic for Levenshtein distance:
  - Bartolini et al. 2002: Bag heuristics
  - *EM-Tree*: Domain specific Length heuristic



Range Query

(Upper Bound) Enclosure: $\top_n + r_n \leq r_q / d_{n,q} + r_n \leq r_q$

- Whole node *n* inside query hyperball
  $\Rightarrow$ All elements below *n* in result set

Upper Bound Intersection: $\top_n + r_n > r_q \geq \top_n - r_n$

- Node *n* is intersected
- Needs to be expanded (without distance computation $d_{n,q}$)
  - But missing $d_{n,q}$ can make child distance heuristic less acurate
  - can not test for enclosure based on $d_{n,q} + r_n \leq r_q$

Zero intervall: $\top_n = \perp_n$

- Determine distance without computation: $d_{n,q} := \top_n (= \perp_n)$

Combination of heuristics

- E.g. new Length heuristics for edit distance
- $\perp_n = \min_i(\perp_{n,i})$

One Child Cut: $|n| = 1$

- *n* has only one child *c* – “aerial root”
- If *n* is expanded, *c* needs to be examined
  $\Rightarrow$ Avoid examining *n*, directly examine *c*

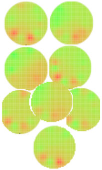
Experimental data

Metric spaces:

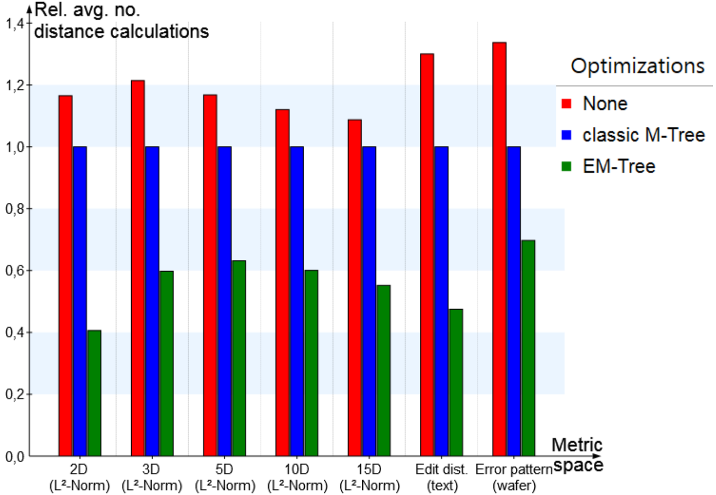
- Range of euclidian vector spaces 2D–15D (10 clusters, gaussian drawn points around cluster center)
- Levenshtein edit distance: Drawn from a pool of 270'000 lines of source code
- Wafer deformations:
  - 66'000 observed Wafer deformations in lithographic step of semiconductor processing
  - Difference-Wafer: Absolute difference of deformation on each surface point
  - Distance: Integral of Difference-Wafer

Experiments:

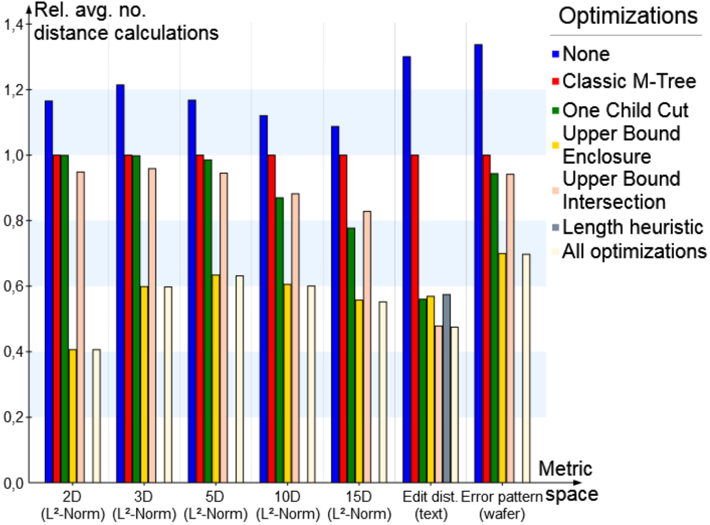
- 10'000 entries per tree
- 1'000 queries per tree
- 100 repetitions



Range Query optimizations – Experimental Results



Range Query optimizations – Experimental Results

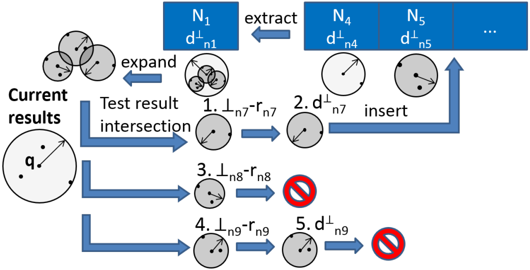


(k) Nearest Neighbor Query

- Query radius $r_q = \max_{e \in F_k}\{d_{e,q}\}$ unknown, bound shrinks during search
- Order of expansion and timing of heuristics use matters

Classic algorithm:

- Expansion priority queue sorted by $d_n^{\perp} = \max\{d_{n,q} - r_n, 0\}$

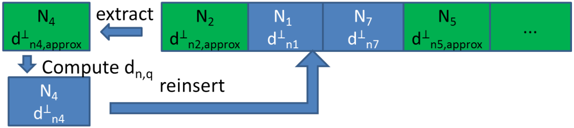


Evaluation:

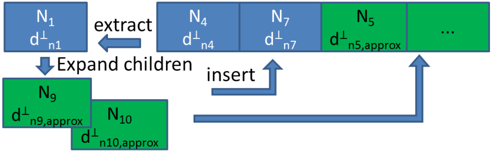
- Minimizes number of node expansions (not distance calculations)
- Highly ineffective use of distance heuristics

(k) Nearest Neighbor Query – improvement in the EM-Tree

- General optimizations (*multiple heuristics, One Child Cut, Zero interval*)
- A^*-like *two-level expansion queue*
- Insert nodes by heuristic dist.bound: $d^{\perp}_{n,approx} = \max\{\perp_n - r_n, 0\} (\leq d^{\perp}_n)$

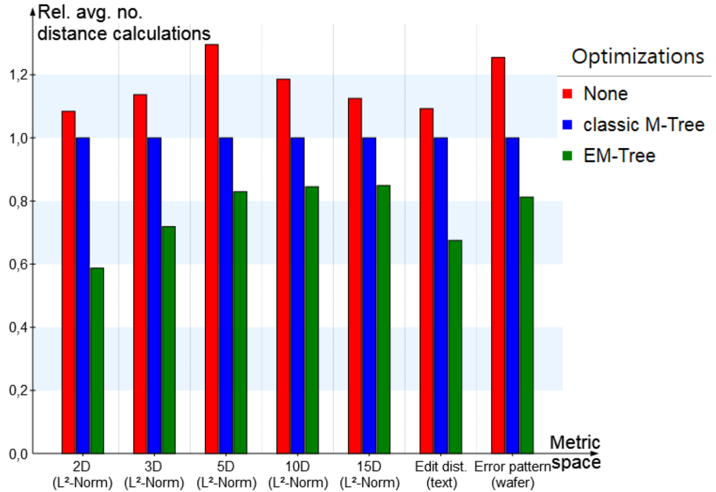


- If such node is removed off the queue, compute $d_{n,q}$ and $d^{\perp}_n$ and reinsert



$\Rightarrow$ *Minimal possible expansion effort*

(k) Nearest Neighbor Query Optimizations – Experimental Results



Summary

Contributions

- Identification of general search optimization concepts to reduce distance calculations
- Development of more efficient algorithms for
  - Range Queries
  - (k-) Nearest Neighbor Queries
- Easy extension of kNN-Query to any time algorithm

Outlook

- Analyze, measure and optimize search-I/O- and -time-effort
- Compare with approximate similarity search
- Compare with other metric index structures
- Additional index option for classic DBMS
- Optimize tree structure
  - M-Tree is very similar to B-Tree
  - But has considerable degrees of freedom when building the tree (Split is neigher complete nor free of overlap)
  - Investigate possibilities to intelligently use these degrees of freedom to create a tree that can be searched more efficiently

Thank you for your attention!