Efficient Metric Indexing for Similarity Search

Lu Chen[1#], Yunjun Gao[2#], Xinhan Li[3#], Christian S. Jensen[4†], Gang Chen[5#]
#College of Computer Science, Zhejiang University, Hangzhou, China
†Department of Computer Science, Aalborg University, Denmark
{[1]luchen, [2]gaoyj, [3]lixh, [5]cg}@cs.zju.edu.cn [4]csj@cs.aau.dk

Motivation

- Similarity search is useful in many areas, such as multimedia retrieval, pattern recognition, and computational biology.
- A generic model is desirable that is capable of accommodating not just a single data type and similarity metric, but a wide spectrum.
- Pivot-based methods outperform compact partitioning methods in terms of CPU costs, but their I/O costs are high because the data is not clustered well.

Contributions

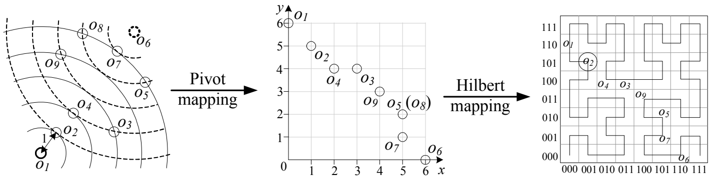
- Develop the SPB-tree, which integrates the compact partitioning with a pivot-based approach, and proposes an efficient pivot selection algorithm to identify a few but effective pivots.
- Present efficient similarity search algorithms and corresponding cost models.
- Conduct extensive experiments with both real and synthetic data sets to demonstrate the performance of the SPB-tree and our proposed algorithms.

Metric Spaces

- A metric space is a tuple (M, d), in which M is the domain of objects and d is a distance function which defines the similarity between the objects in M.
- Symmetry: $d(q, o) = d(o, q)$; non-negativity: $d(q, o) \geq 0$; identity: $d(q, o) = 0$ iff $q = o$; and triangle inequality: $d(q, o) \leq d(q, p) + d(p, o)$.

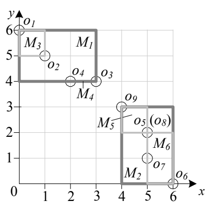
SPB-tree Construction

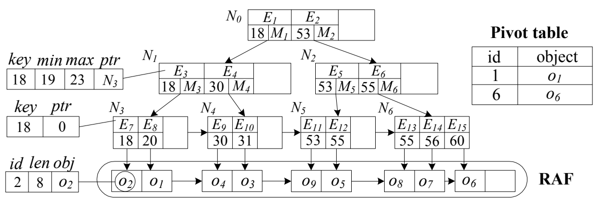
- Map the objects in a metric space to data points in a vector space using well-chosen pivots.
- Map the data points in the vector space into integers in a one-dimensional space using the SFC.
- Employ a B$^+$-tree with MBB information to index the resulting integers.



SPB-tree Structure

- A pivot table stores selected objects (e.g., o_1 and o_6) to map a metric space into a vector space.
- A B$^+$-tree is employed to index the SFC values of objects
  - The minimum SFC value *key* in its subtree
  - The pointer *ptr* to the root node of its subtree
  - The SFC values *min* and *max* to represent the MBB M_i
- A RAF to keep objects separately
  - An object identifier *id*
  - The length *len* of the object
  - The real object *obj*





Pivot table

id	object
1	o_1
6	o_6

Pivot Selection Algorithm

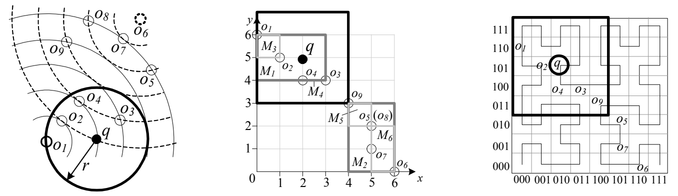
- **Precision**. Given a set *OP* of object pairs in a metric space, the quality of a pivot set *P* is evaluated as

$$precison(P) = \frac{1}{|OP|} \sum_{<o_i, o_j> \in OP} \frac{D(\phi(o_i), \phi(o_j))}{d(o_i, o_j)}$$

- HF based incremental pivot selection algorithm (HFI)
  - Employ the HF algorithm [2] to obtain outliers as candidate pivots *CP*
  - Incrementally select effective pivots *P* from *CP* to maximize *precision*

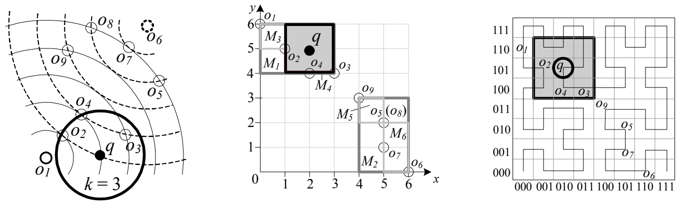
Metric Range Query Algorithm

- **Metric Range Query:** Given an object set O, a query object q, and a search radius r in M, $RQ(q, r) = \{o | o \in O \wedge d(q, o) \leq r\}$.
- **Pruning Rule:** Given a pivot set P, if an object o is enclosed in $RQ(q, r)$, then $\phi(o)$ is certainly contained in the mapped range region $RR(r)$.
- Metric Range Query Algorithm
  - Compute $RR(r)$ using a pivot set P
  - Traverse SPB-tree in a depth-first paradigm to verify objects contained in $RR(r)$



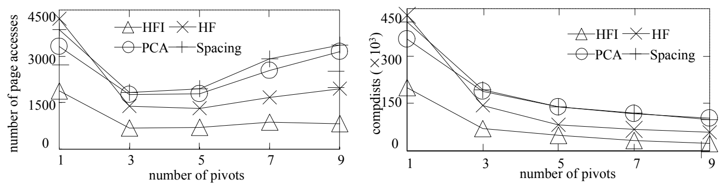
Metric *k*NN Search Algorithm

- **Metric *k*NN Search.** Given an object set O, a query object q, and an integer k in a generic metric space M, $kNN(q, k) = \{R \mid R \subseteq O \wedge |R| = k \wedge \forall r \in R, \forall o \in O - R, d(q, r) \leq d(q, o)\}$.
- **Pruning Rule.** Given a query object q and a B$^+$-tree entry E, E can be safely pruned if $MIND(q, E) \geq curND_k$.
  - $MIND(q, E)$ denotes the mapped minimum distance between q and E
  - $curND_k$ represents the distance from q to the current k-th NN
- Metric *k*NN Search Algorithm
  - Traverse SPB-tree in a best-first paradigm to verify objects not pruned



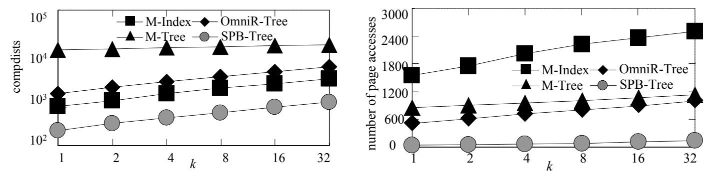
Performance

- Comparisons among Pivot Selection Algorithms (*Words*)



I/O cost vs. number of pivots CPU cost vs. number of pivots

- Comparisons among Metric Similarity Algorithms (*Color*)



I/O cost vs. number of pivots CPU cost vs. number of pivots

References

[1] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB*, 1997, pp. 426–435.

[2] C. T. Jr., R. F. S. Filho, A. J. M. Traina, M. R. Vieira, and C. Faloutsos, "The Omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient," *VLDB J.*, 16(4), pp. 483–505, 2007.

[3] D. Novak, M. Batko, and P. Zezula, "Metric Index: An efficient and scalable solution for precise and approximate similarity search," *Inf. Syst.*, 36(4), pp. 721–733, 2011.