Novel Approaches to Nearest Neighbors

Random Walks. SEARCH Class.

Yury Lifshits

http://yury.name

Steklov Institute of Mathematics at St.Petersburg
California Institute of Technology

August 2007

Outline

1. Welcome to nearest neighbors!

Outline

1. Welcome to nearest neighbors!
2. Nearest Neighbors via Random Walks

Outline

1. Welcome to nearest neighbors!
2. Nearest Neighbors via Random Walks
3. Data Structure Complexity: SEARCH Class

Chapter I

Welcome to Nearest Neighbors!

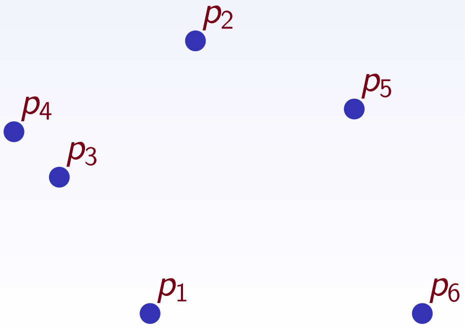
Problem Statement

Search space: object domain $\mathbb{U}$, similarity function σ

Input: database $S = \{p_1, \ldots, p_n\} \subseteq \mathbb{U}$

Query: $q \in \mathbb{U}$

Task: find $\operatorname{argmax} \sigma(p_i, q)$



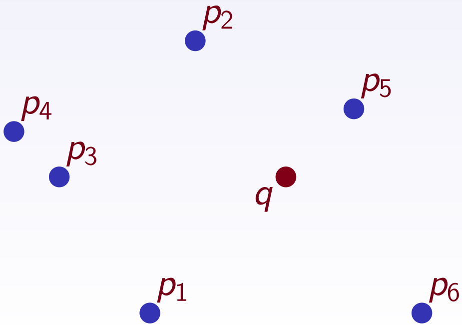
Problem Statement

Search space: object domain $\mathbb{U}$, similarity function σ

Input: database $S = \{p_1, \ldots, p_n\} \subseteq \mathbb{U}$

Query: $q \in \mathbb{U}$

Task: find $\operatorname{argmax}\, \sigma(p_i, q)$



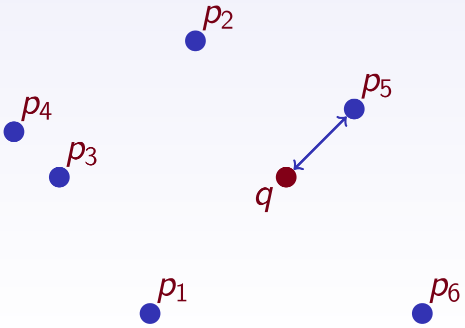
Problem Statement

Search space: object domain $\mathbb{U}$, similarity function σ

Input: database $S = \{p_1, \ldots, p_n\} \subseteq \mathbb{U}$

Query: $q \in \mathbb{U}$

Task: find $\operatorname{argmax}\ \sigma(p_i, q)$



Applications

Content-based retrieval Spelling correction Searching for similar DNA sequences Related pages web search Concept matching kNN classification rule Nearest-neighbor interpolation Near-duplicate detection Plagiarism detection Computing co-occurrence similarity Recommendation systems Personalized news aggregation Behavioral targeting Maximum likelihood decoding MPEG compression

Brief History

1908 Voronoi diagram

1967 kNN classification rule by Cover and Hart

1973 Post-office problem posed by Knuth

1997 The paper by Kleinberg, beginning of provable upper/lower bounds

2006 Similarity Search book by Zezula, Amato, Dohnal and Batko

2008 First International Workshop on Similarity Search. Consider submitting!

Some Nearest Neighbor Solutions

Sphere Rectangle Tree Orchard's Algorithm LAESA
k-d-B tree Geometric near-neighbor access tree
Excluded middle vantage point forest mvp-tree Fixed-height
fixed-queries tree AESA Vantage-point
tree R*-tree Burkhard-Keller tree BBD tree
Navigating Nets Voronoi tree Balanced aspect ratio tree Metric tree
vp^s-tree M-tree Locality-Sensitive Hashing
SS-tree R-tree Spatial approximation tree Multi-vantage
point tree Bisector tree mb-tree
Generalized hyperplane tree
Hybrid tree Slim tree Spill Tree Fixed queries tree X-tree k-d
tree Balltree Quadtree Octree Post-office tree

Part II

Disorder Inequality

This section represents joint work with Navin Goyal and Hinrich Schütze

Concept of Disorder

Sort all objects in database S by their similarity to p
Let $\text{rank}_p(s)$ be position of object s in this list

Concept of Disorder

Sort all objects in database S by their similarity to p
Let $\mathrm{rank}_p(s)$ be position of object s in this list

Disorder inequality for some constant D:

$$\forall p, r, s \in \{q\} \cup S : \qquad \mathrm{rank}_r(s) \leq D \cdot (\mathrm{rank}_p(r) + \mathrm{rank}_p(s))$$

Minimal D providing disorder inequality is called disorder constant of a given set

Concept of Disorder

Sort all objects in database S by their similarity to p
Let $\mathrm{rank}_p(s)$ be position of object s in this list

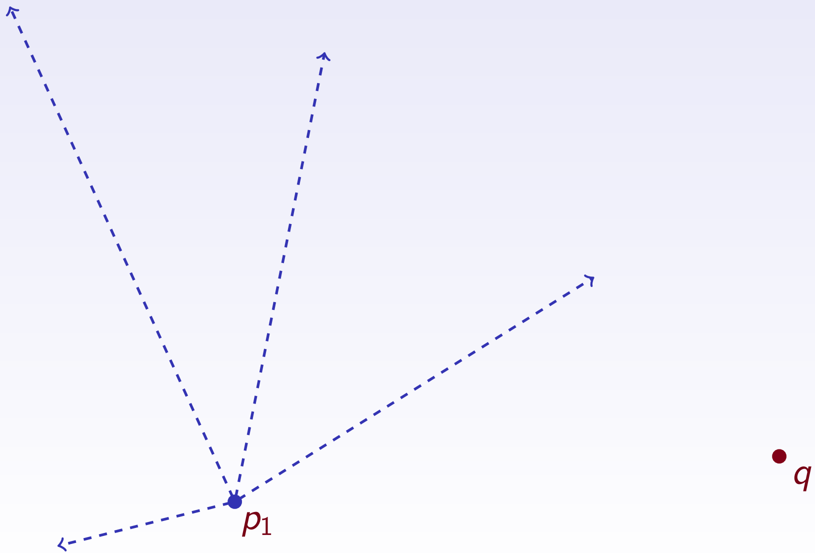
Disorder inequality for some constant D:

$$\forall p, r, s \in \{q\} \cup S : \qquad \mathrm{rank}_r(s) \leq D \cdot (\mathrm{rank}_p(r) + \mathrm{rank}_p(s))$$

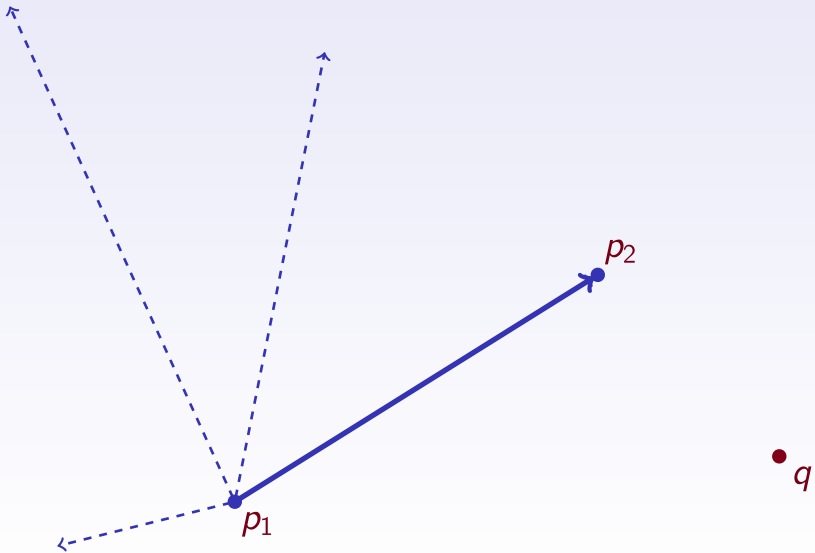
Minimal D providing disorder inequality is called disorder constant of a given set

For "regular" sets in d-dimensional Euclidean space $D \approx 2^{d-1}$

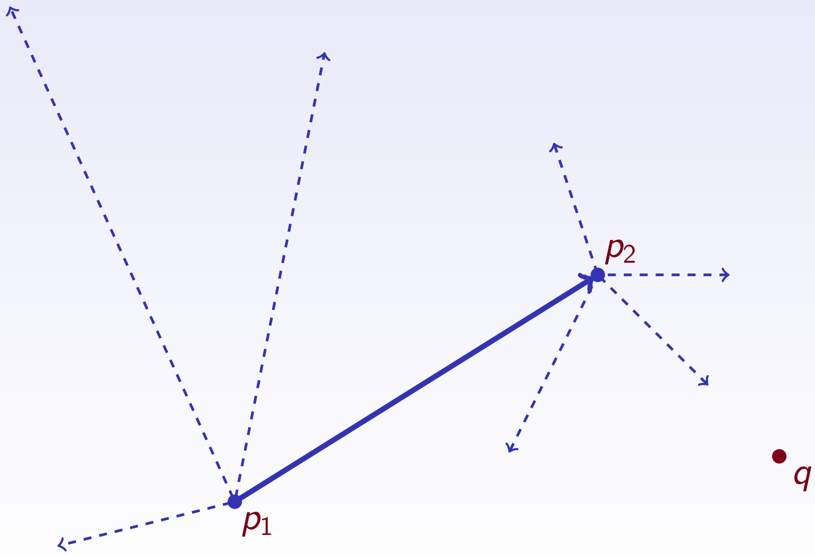
Ranwalk Informally (1/2)



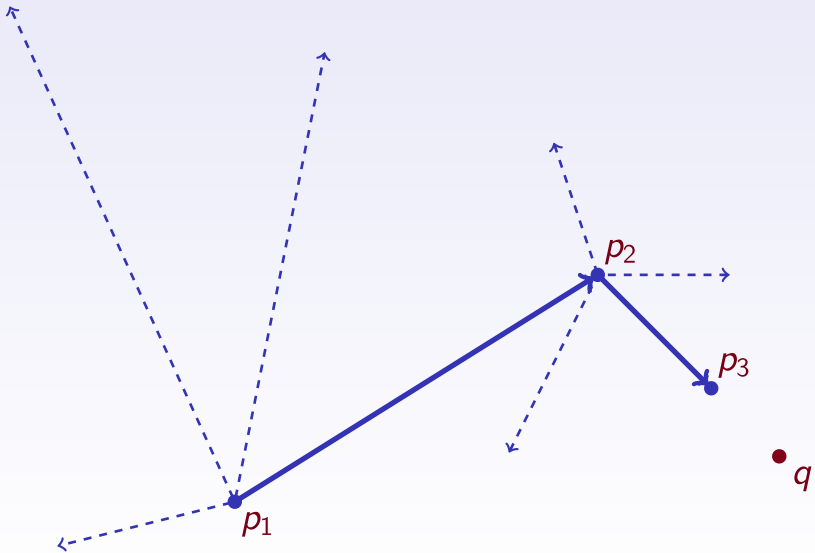
Ranwalk Informally (1/2)



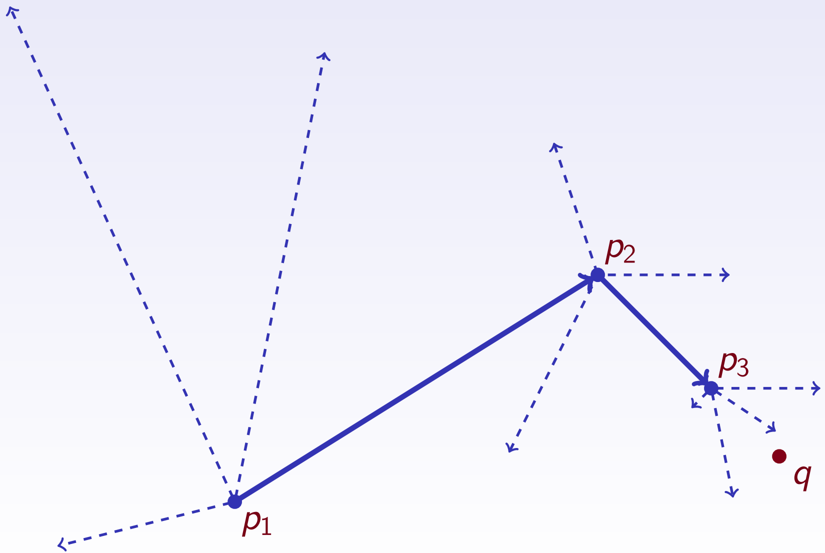
Ranwalk Informally (1/2)



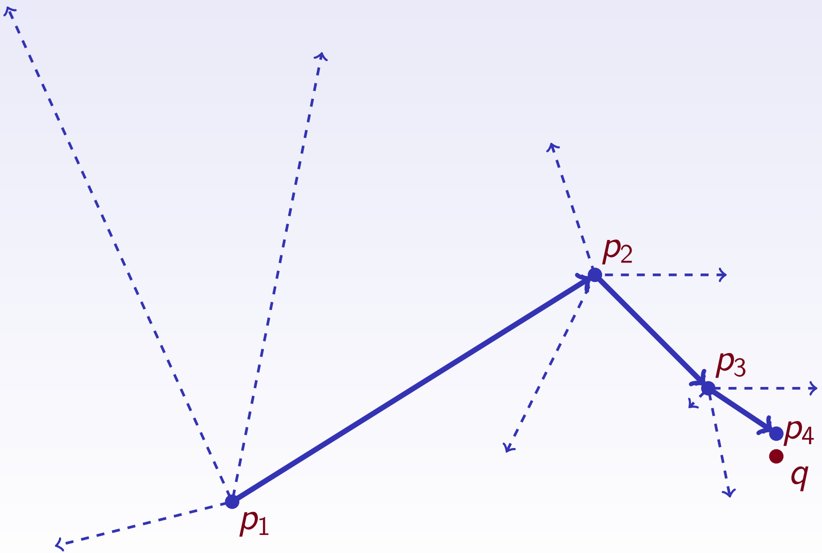
Ranwalk Informally (1/2)



Ranwalk Informally (1/2)



Ranwalk Informally (1/2)



Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)
4. Among all bus routes from p_3 choose the one going most closely to q, move there (p_4)

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)
4. Among all bus routes from p_3 choose the one going most closely to q, move there (p_4)
5. Repeat this $\log n$ times and return the final city

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)
4. Among all bus routes from p_3 choose the one going most closely to q, move there (p_4)
5. Repeat this $\log n$ times and return the final city

Transport system: for level k choose c random arcs to $\frac{n}{2^k}$ neighborhood

Ranwalk Algorithm

Preprocessing:

- For every point p in database we sort all other points by their similarity to p

  Data structure: n lists of $n-1$ points each.

Query processing:

1. Step 0: choose a random point p_0 in the database.
2. From $k=1$ to $k=\log n$ do Step k: Choose $D' := 3D(\log\log n + 1)$ random points from $\min(n, \frac{3Dn}{2^k})$-neighborhood of p_{k-1}. Compute similarities of these points w.r.t. q and set p_k to be the most similar one.
3. If $\mathrm{rank}_{p_{\log n}}(q) > D$ go to step 0, otherwise search the whole D^2-neighborhood of $p_{\log n}$ and return the point most similar to q as the final answer.

Analysis of Ranwalk

Theorem

Assume that database points together with query point $S \cup \{q\}$ *satisfy disorder inequality with constant* D*:*

$$rank_x(y) \leq D(rank_z(x) + rank_z(y)).$$

Then Ranwalk algorithm always answers nearest neighbor queries correctly. It uses the following resources:
Preprocessing space: $\mathcal{O}(n^2)$.
Preprocessing time: $\mathcal{O}(n^2 \log n)$.
Expected query time: $\mathcal{O}(D \log n \log\log n + D^2)$.

Arwalk Algorithm

Preprocessing:

- For every point p in database we sort all other points by their similarity to p. For every *level number* k from 1 to $\log n$ we store pointers to $D' = 3D(\log\log n + \log 1/\delta)$ random points within $\min(n, \frac{3Dn}{2^k})$ most similar to p points.

Query processing:

1. Step 0: choose a random point p_0 in the database.
2. From $k = 1$ to $k = \log n$ do Step k: go by p_{k-1} pointers of level k. Compute similarities of these D' points to q and set p_k to be the most similar one.
3. Return $p_{\log n}$.

Analysis of Algorithm

Theorem

Assume that database points together with query point $S \cup \{q\}$ *satisfy disorder inequality with constant* D*:*

$$rank_x(y) \leq D(rank_z(x) + rank_z(y)).$$

Then for any probability of error δ *Arwalk algorithm answers nearest neighbor query within the following constraints:*
Preprocessing space: $\mathcal{O}(nD \log n(\log \log n + \log 1/\delta))$.
Preprocessing time: $\mathcal{O}(n^2 \log n)$.
Query time: $\mathcal{O}(D \log n(\log \log n + \log 1/\delta))$.

Future of Disorder (1/2)

Average disorder. If disorder inequality does not hold for a small fraction of pairs, how should we modify our algorithm?

Future of Disorder (1/2)

Average disorder. If disorder inequality does not hold for a small fraction of pairs, how should we modify our algorithm?

Improving our algorithms. Is it possible to combine advantages of Ranwalk and Arwalk? Does there exist a deterministic algorithm with sublinear search time utilizing small disorder assumption? E.g., can we use expanders for derandomization?

Future of Disorder (2/2)

Disorder of random sets. Compute disorder values for some modelling examples. For example, consider n random points on d-dimensional sphere

Future of Disorder (2/2)

Disorder of random sets. Compute disorder values for some modelling examples. For example, consider n random points on d-dimensional sphere

Lower bounds. Is it possible to prove lower bounds on preprocessing and query complexities in some "black-box" model of computation?

Part III

Data Structure Complexity: SEARCH Class

Inclusions with Preprocessing (1/2)

Input

Family $\mathcal{F}$ of subsets of U

Query task

Given a set $f_{new} \subseteq U$ to decide
whether $\exists f \in \mathcal{F} : \quad f_{new} \subseteq f$

Constraints

Data storage after preprocessing $poly(|\mathcal{F}| + |U|)$
Time for query processing $poly(|U|)$

Inclusions with Preprocessing (1/2)

Input

Family $\mathcal{F}$ of subsets of U

Query task

Given a set $f_{new} \subseteq U$ to decide
whether $\exists f \in \mathcal{F} : \quad f_{new} \subseteq f$

Constraints

Data storage after preprocessing $poly(|\mathcal{F}| + |U|)$
Time for query processing $poly(|U|)$

Open problem: is there an algorithm satisfying given constraints?

Inclusions with Preprocessing (2/2)

Reformulation in SAT style:

Input
Formula $\mathcal{F}$ in DNF with n variables

Query task
Given an assignment x to evaluate $\mathcal{F}(x)$

Constraints
Data storage after preprocessing $poly(|\mathcal{F}|)$
Time for query processing $poly(n)$

Inclusions with Preprocessing (2/2)

Reformulation in SAT style:

Input

Formula $\mathcal{F}$ in DNF with n variables

Query task

Given an assignment x to evaluate $\mathcal{F}(x)$

Constraints

Data storage after preprocessing $poly(|\mathcal{F}|)$
Time for query processing $poly(n)$

Open problem: is there an algorithm satisfying given constraints?

"NP Analogue" for Search Problems

Every problem in **SEARCH class** is characterized by poly-time computable Turing Machine M:

Input

Strings $x_1, \dots, x_n$, $\quad |x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable problems in SEARCH

Input

Strings $x_1, \ldots, x_n$, $\quad |x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable problems in SEARCH

Input

Strings $x_1, \ldots, x_n$, $|x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable solution

Preprocessing in $poly(m, n)$ space

Query processing in $poly(m, \log n)$ time with RAM access to preprocessed database

Tractable problems in SEARCH

Input

Strings $x_1, \ldots, x_n$, $\quad |x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable solution

Preprocessing in $poly(m, n)$ space

Query processing in $poly(m, \log n)$ time with RAM access to preprocessed database

Inclusions is in SEARCH. Is it tractable?

Complete problems in SEARCH (1/2)

Program Search problem:

Input

Turing machines $P_1 \ldots, P_n$

Query task

Given string y of length m to answer
whether $\exists i : P_i(y) = yes$ after at most m steps

Complete problems in SEARCH (1/2)

Program Search problem:

Input

Turing machines $P_1 \ldots, P_n$

Query task

Given string y of length m to answer whether $\exists i : P_i(y) = yes$ after at most m steps

Open problem: is Program Search tractable?

Complete problems in SEARCH (2/2)

Parallel Run problem:

Input

$x_1 \ldots, x_n$

Query task

Given poly-time computable P to answer whether $\exists i : P(x_i) = yes$

Complete problems in SEARCH (2/2)

Parallel Run problem:

Input

$x_1 \ldots, x_n$

Query task

Given poly-time computable P to answer whether $\exists i : P(x_i) = yes$

Open problem: is Parallel Run tractable?

NN Proofs?

NN-proof system:

- Fix some family of basic statements about points in multidimensional space and some proof system
- Can we compute $poly(|S|)$ statements about points of database Ssuch that for any query q and any real nearest neighbor $p_{NN} \in S$ there is a logarithmic proof from precomputed statements that indeed p_{NN} is nearest point is S to q

NN Proofs?

NN-proof system:

- Fix some family of basic statements about points in multidimensional space and some proof system
- Can we compute $poly(|S|)$ statements about points of database Ssuch that for any query q and any real nearest neighbor $p_{NN} \in S$ there is a logarithmic proof from precomputed statements that indeed p_{NN} is nearest point is S to q

Do such an NN proof system exist?

Highlights

- Random walk provide logarithmic nearest neighbor search for bounded disorder sets

Highlights

- Random walk provide logarithmic nearest neighbor search for bounded disorder sets
- SEARCH class: is it tractable?

Highlights

- Random walk provide logarithmic nearest neighbor search for bounded disorder sets
- SEARCH class: is it tractable?
- Do NN proof systems exist?

Highlights

- Random walk provide logarithmic nearest neighbor search for bounded disorder sets
- SEARCH class: is it tractable?
- Do NN proof systems exist?

Thanks for your attention! Questions?

References

The Homepage of Nearest Neighbors and Similarity Search

http://simsearch.yury.name

N. Goyal, Y. Lifshits, H.Schütze
Disorder Inequality: A Combinatorial Approach to Nearest Neighbor Search. Submitted.
http://yury.name/papers/goyal2008disorder.pdf

B. Hoffmann, Y. Lifshits, D.Novotka
Maximal Intersection Queries in Randomized Graph Models. CSR'07.
http://yury.name/papers/hoffmann2007maximal.pdf

P. Zezula, G. Amato, V. Dohnal, M. Batko
Similarity Search: The Metric Space Approach. Springer, 2006.
http://www.nmis.isti.cnr.it/amato/similarity-search-book/

G.R. Hjaltason and H. Samet
Index-driven similarity search in metric spaces. ACM Transactions on Database Systems, 2003
http://www.cs.utexas.edu/~abhinay/ee382v/Project/Papers/ft_gateway.cfm.pdf