On Locality-sensitive Indexing in Generic Metric Spaces

David Novak, Martin Kyselak, Pavel Zezula

Masaryk University
Brno, Czech Republic

September 18, 2010
SISAP 2010, Istanbul

Outline of the Talk

1. Motivation and Objectives
2. Locality Sensitive Hashing
   - Fundamentals
   - Collision Probability
   - LSH Techniques
3. Metric LSH with M-Index
   - Principles
   - Indexing & Searching
4. Analysis of LSH Properties

Motivation and Objectives

Locality-sensitive Hashing (LSH) after a decade of development

- established theoretical fundamentals
- LSH functions for several specific data types + distance functions

Motivation and Objectives

Locality-sensitive Hashing (LSH) after a decade of development

- established theoretical fundamentals
- LSH functions for several specific data types + distance functions

Metric Index (M-Index) as an indexing & searching principle

- hash functions that preserve locality of data
- based purely on metric properties

Motivation and Objectives

Locality-sensitive Hashing (LSH) after a decade of development

- established theoretical fundamentals
- LSH functions for several specific data types + distance functions

Metric Index (M-Index) as an indexing & searching principle

- hash functions that preserve locality of data
- based purely on metric properties

Objectives of this work

- redefinition of M-Index in terms of LSH approach
- experimental verification of LSH properties of M-Index
- comparison with established LSH function
- analysis of LSH properties with respect to *kNN* search

LSH Fundamentals

Basic idea:

- construct a family of locality-sensitive hash functions
  - **close** objects tend to be hashed to the **same bucket**
  - **distant** objects tend to be hashed to **different buckets**
- access objects hashed to the same bucket as query object
- construct several hash indexes

LSH Fundamentals

Basic idea:

- construct a family of locality-sensitive hash functions
  - **close** objects tend to be hashed to the **same bucket**
  - **distant** objects tend to be hashed to **different buckets**
- access objects hashed to the same bucket as query object
- construct several hash indexes

Definition

A family $\mathcal{H}$ of functions $h : \mathcal{D} \to M$ is called (r_1, r_2, p_1, p_2)-sensitive for $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ if for any $u, v \in \mathcal{D}$:

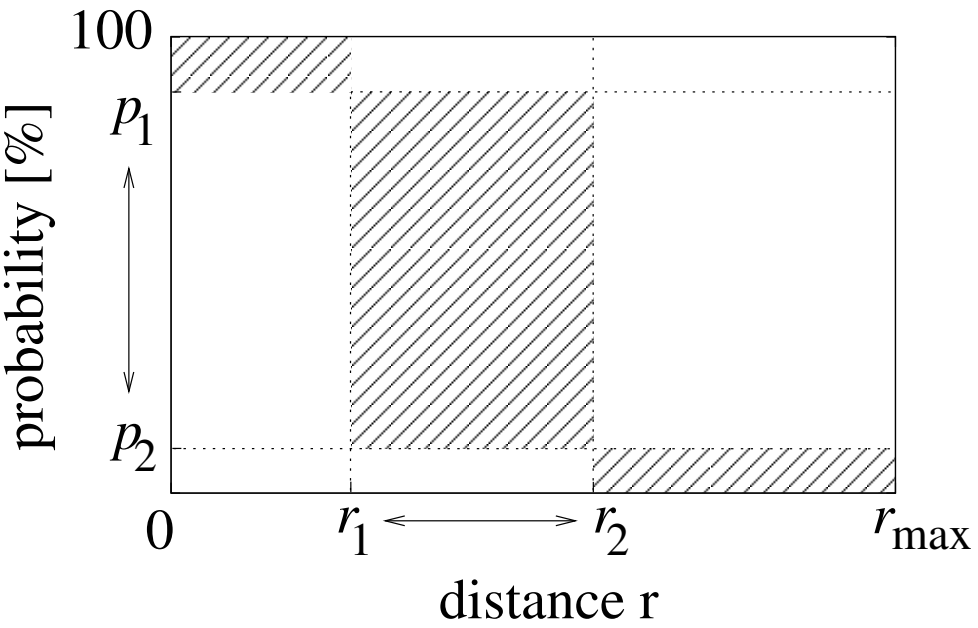
- if $d(u, v) \leq r_1$ then $\mathcal{P}_{\mathcal{H}}[h(u) = h(v)] \geq p_1$,
- if $d(u, v) > r_2$ then $\mathcal{P}_{\mathcal{H}}[h(u) = h(v)] \leq p_2$.

LSH function family should satisfy inequalities $r_1 < r_2$ and $p_1 \gg p_2$

Collision Probability

Collision probability curve:

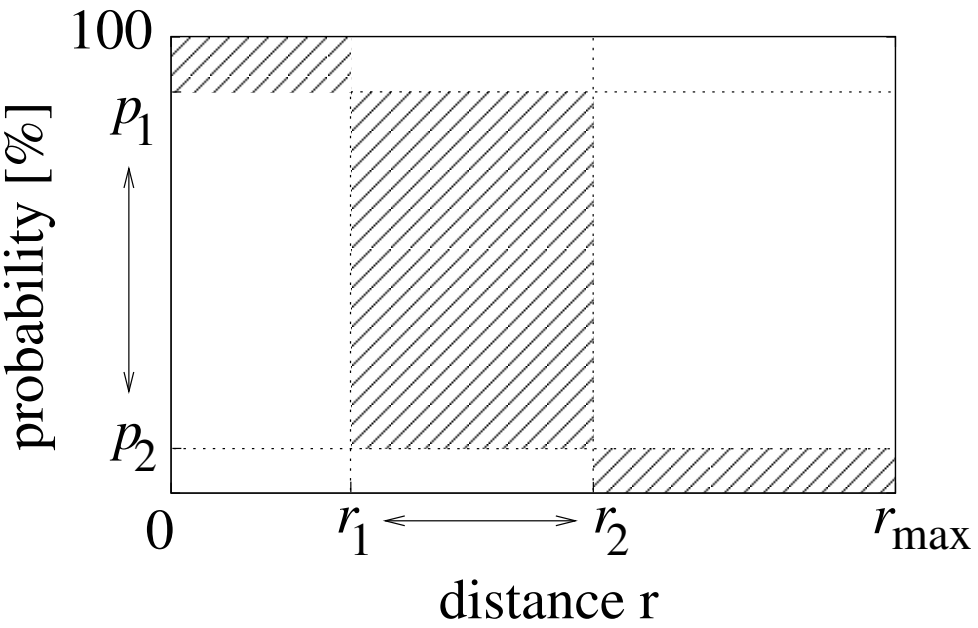
- probability of $h(u) = h(v)$ for $u, v \in \mathcal{D}$ with respect to distance $d(u, v)$



Collision Probability

Collision probability curve:

- probability of $h(u) = h(v)$ for $u, v \in \mathcal{D}$ with respect to distance $d(u, v)$



A good shape of this curve:

- p_1 as high as possible for distances r_1 used as query radii
- p_2 as low as possible for "not interesting" distances r_2
- decrease steeply between r_1 and r_2

LSH Techniques

Concatenation technique:

- family of functions is defined: $\mathcal{G} = \{g : \mathcal{D} \rightarrow M^K\}$
- where $g(u) = (h_1(u), \ldots, h_K(u))$, $h_i \in \mathcal{H}$

LSH Techniques

Concatenation technique:

- family of functions is defined: $\mathcal{G} = \{g : \mathcal{D} \rightarrow M^K\}$
- where $g(u) = (h_1(u), \ldots, h_K(u))$, $h_i \in \mathcal{H}$

Classic LSH approach is to tune parameters:

- K – number of concatenated functions/values
- L – number of hash tables

LSH Techniques

Concatenation technique:

- family of functions is defined: $\mathcal{G} = \{g : \mathcal{D} \rightarrow M^K\}$
- where $g(u) = (h_1(u), \ldots, h_K(u))$, $h_i \in \mathcal{H}$

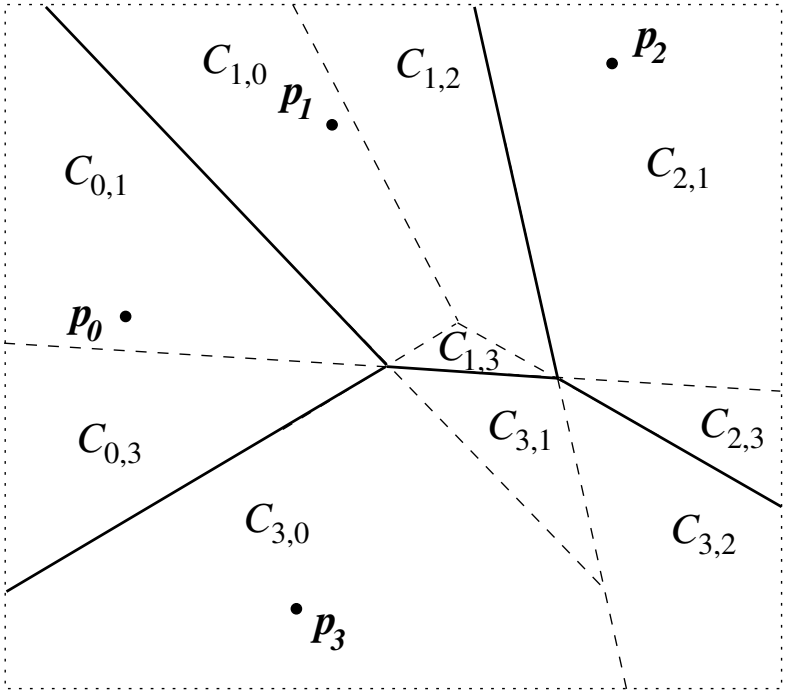
Classic LSH approach is to tune parameters:

- K – number of concatenated functions/values
- L – number of hash tables

Multi-probe approach:

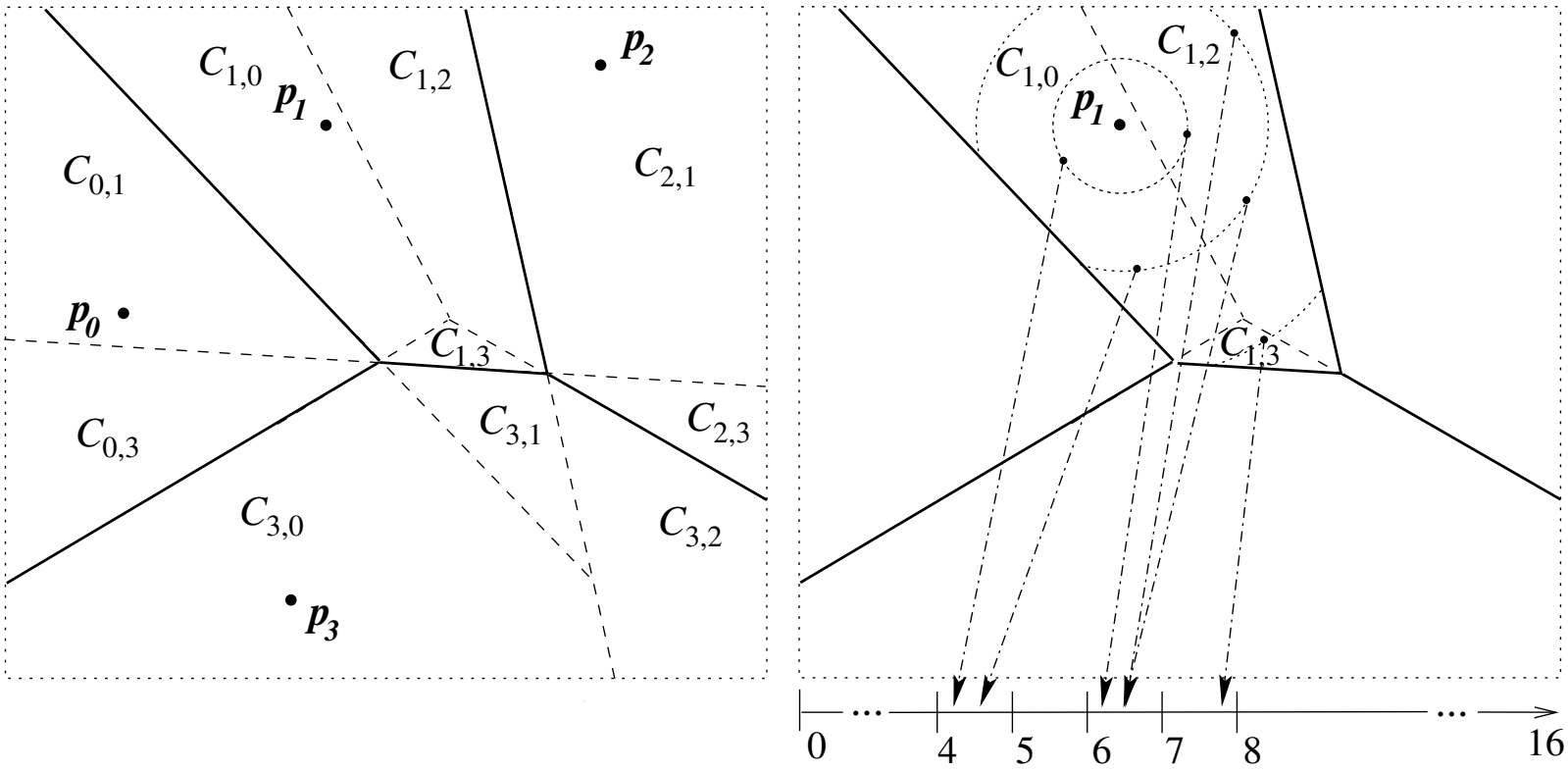
- access more buckets that could contain objects near query q
- hash keys $g(q)$ that differ by ± 1 in one or several components:
  - $(h_1(q) \pm 1, \ldots, h_K(q) \pm 1)$

M-Index Mapping



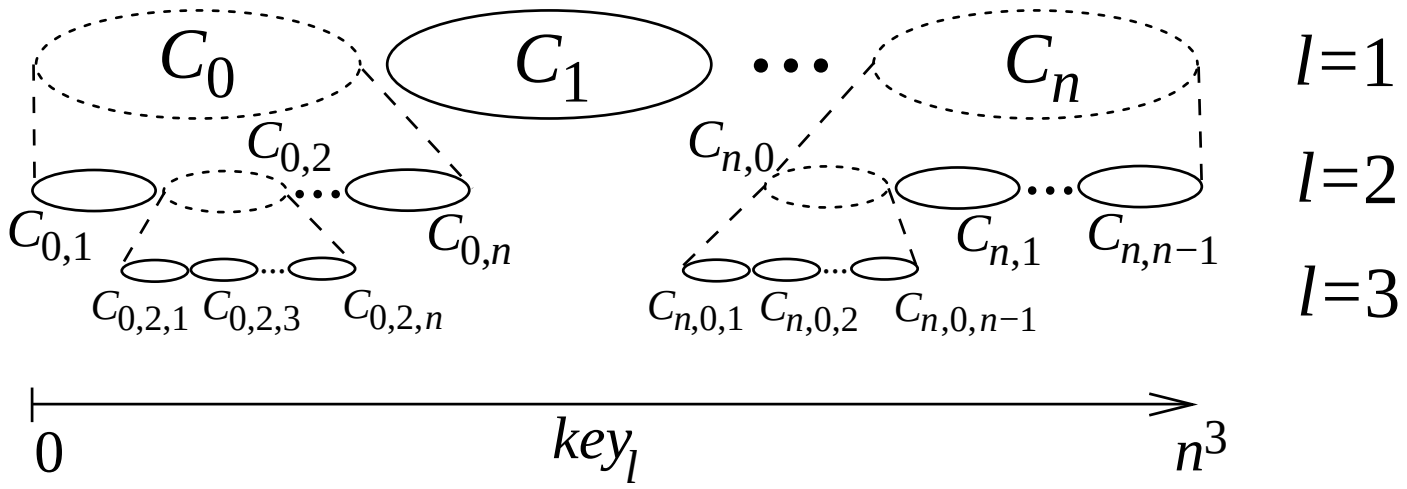
- recursive Voronoi-like partitioning using set of pivots $p_0, \ldots, p_{n-1}$

M-Index Mapping



- recursive Voronoi-like partitioning using set of pivots $p_0, \ldots, p_{n-1}$
- mapping of object o: cluster identification + distance $d(o, p_i)$

Dynamic Hashing with M-Index



- overfull clusters split further (up to a maximum level $1 \leq l_{\max} \leq n$)
- mapping (cluster numbering) analogous to extensible hashing

LSH Indexing & Searching with M-Index

M-Index defines a family of locality-sensitive functions

- parameters: n, $l_{\max}$, cluster capacity
- every instance is identified by a specific set of pivots $p_0, \ldots, p_{n-1}$

LSH Indexing & Searching with M-Index

M-Index defines a family of locality-sensitive functions

- parameters: n, $l_{\max}$, cluster capacity
- every instance is identified by a specific set of pivots $p_0, \ldots, p_{n-1}$

Approximate search with M-Index is multi-probe

- always access cluster C_q for query $kNN(q, k)$
- a given order to access other buckets
  - driven by a heuristic
- accessed clusters form virtual buckets
  - query dependent, arbitrary and precisely tunable size

Evaluation Settings

Two real-life datasets with one million objects

- Color Structure: MPEG-7 descriptor (64-dim vector + L_2 distance)
- CoPhIR: five MPEG-7 descriptors (280 dimensions + a complex aggregation function, intrinsic dimensionality: 13)

Evaluation Settings

Two real-life datasets with one million objects

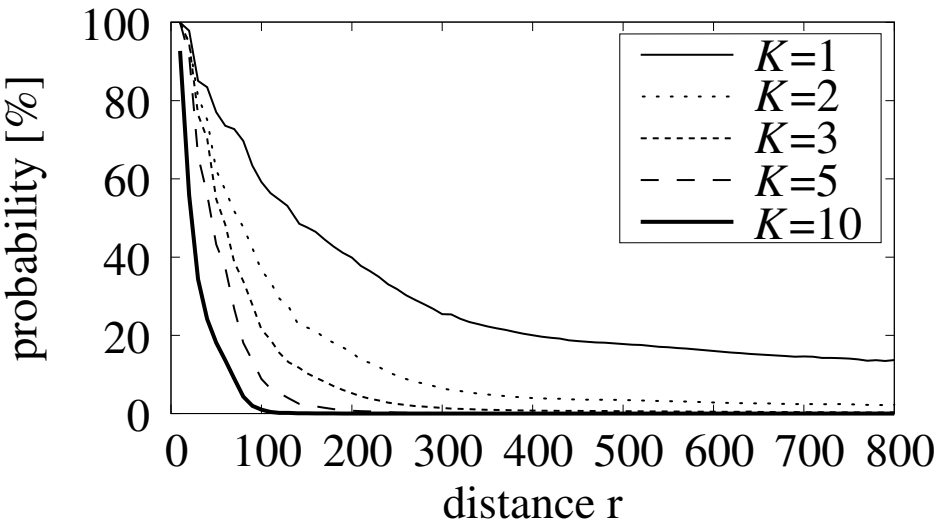
- Color Structure: MPEG-7 descriptor (64-dim vector + L_2 distance)
- CoPhIR: five MPEG-7 descriptors (280 dimensions + a complex aggregation function, intrinsic dimensionality: 13)

Standard LSH for L_2 distance on $\mathbf{v_1}, \mathbf{v_2} \in \mathbb{R}^d$:

- random vector $\mathbf{a} \in \mathbb{R}^d$ (entries chosen from a 2-stable distribution)
- scalar $b \in \mathbb{R}$ chosen from the range $[0, w)$
- hash function $h_{\mathbf{a},b} : \mathbb{R}^d \rightarrow M$, namely $h_{\mathbf{a},b}(\mathbf{v}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{w} \rfloor$

Standard LSH Collision Probability

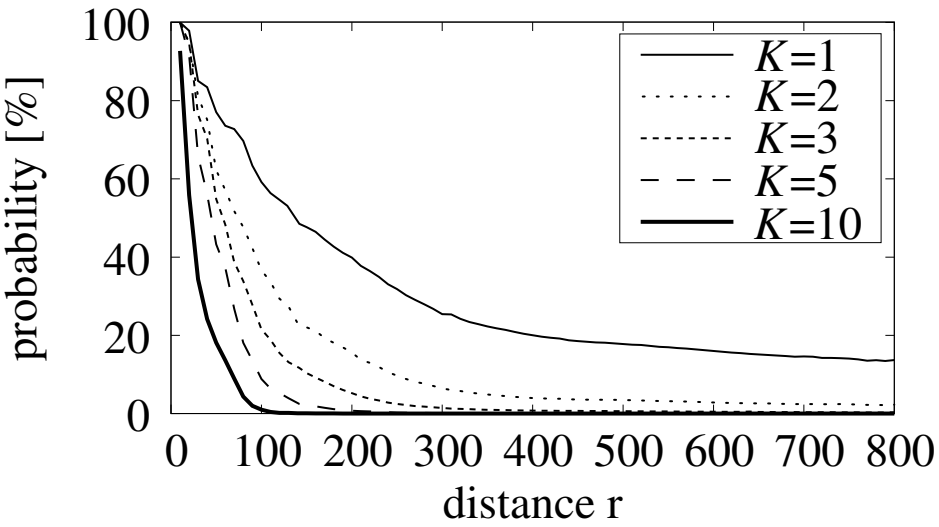
Collision probability with respect to object mutual distance



- Color Structure dataset
- standard LSH function family $h(u)$ for L_2 metric
- concatenation of K hashes $g(u) = (h_1(u), \ldots, h_K(u))$

Standard LSH Collision Probability

Collision probability with respect to object mutual distance

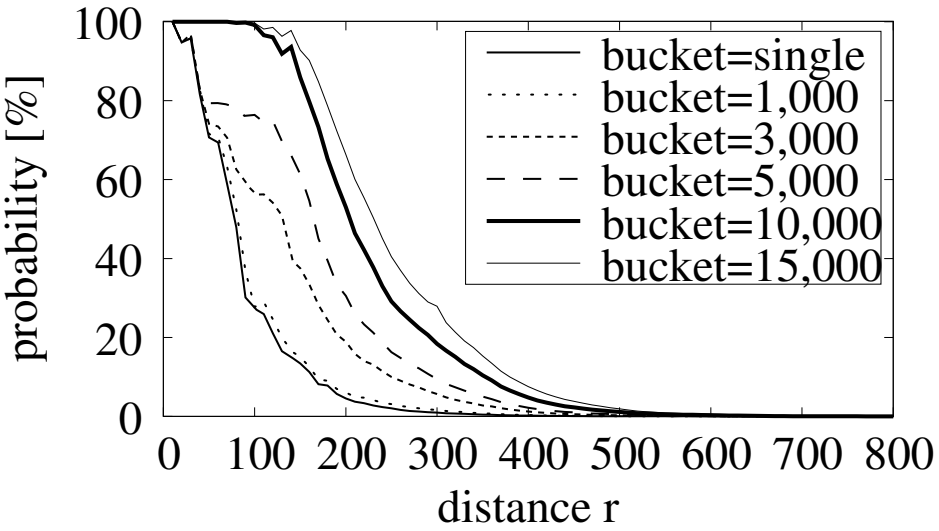


- Color Structure dataset
- standard LSH function family $h(u)$ for L_2 metric
- concatenation of K hashes $g(u) = (h_1(u), \ldots, h_K(u))$

Average bucket sizes (percentage of dataset size):

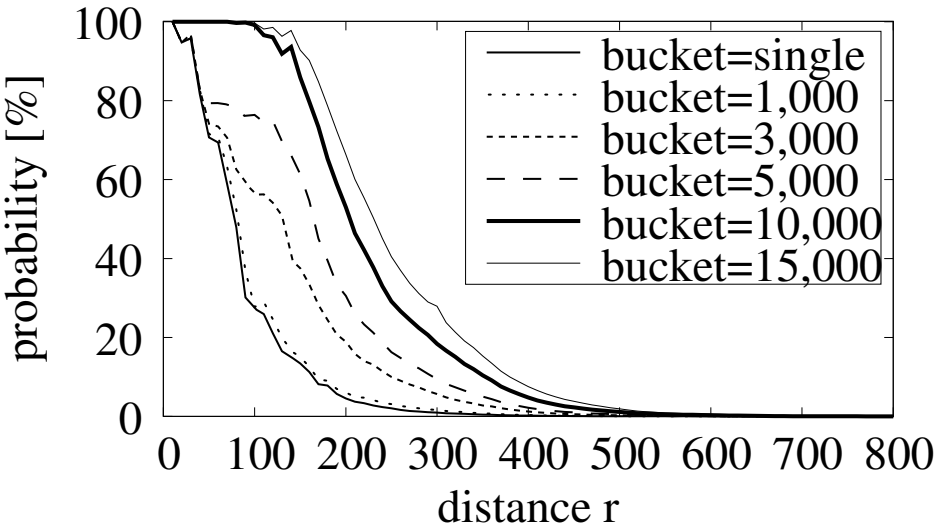
concatenation	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 10$
avg bucket size	17%	3.2%	0.6%	0.12%	0.03%	0.0007%

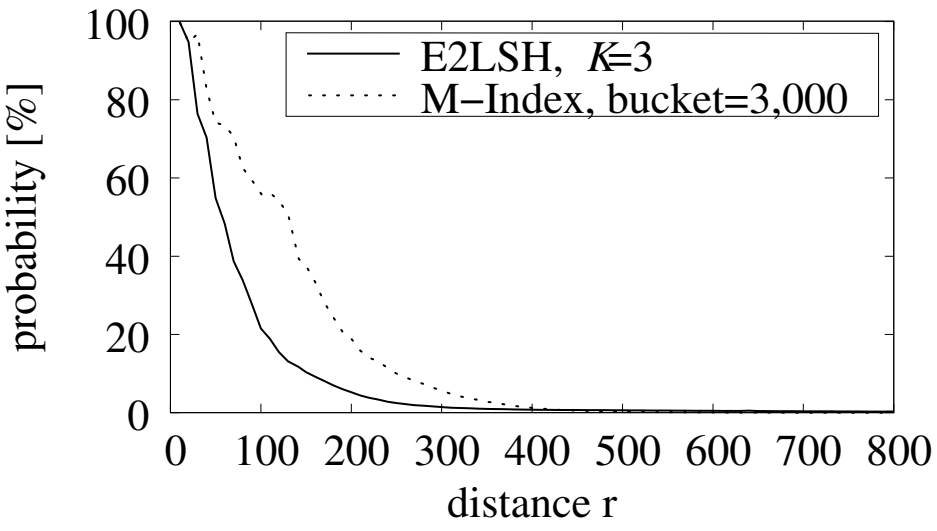
M-Index Collision Probability



- M-Index with virtual buckets
- tunable number of accessed objects

M-Index Collision Probability

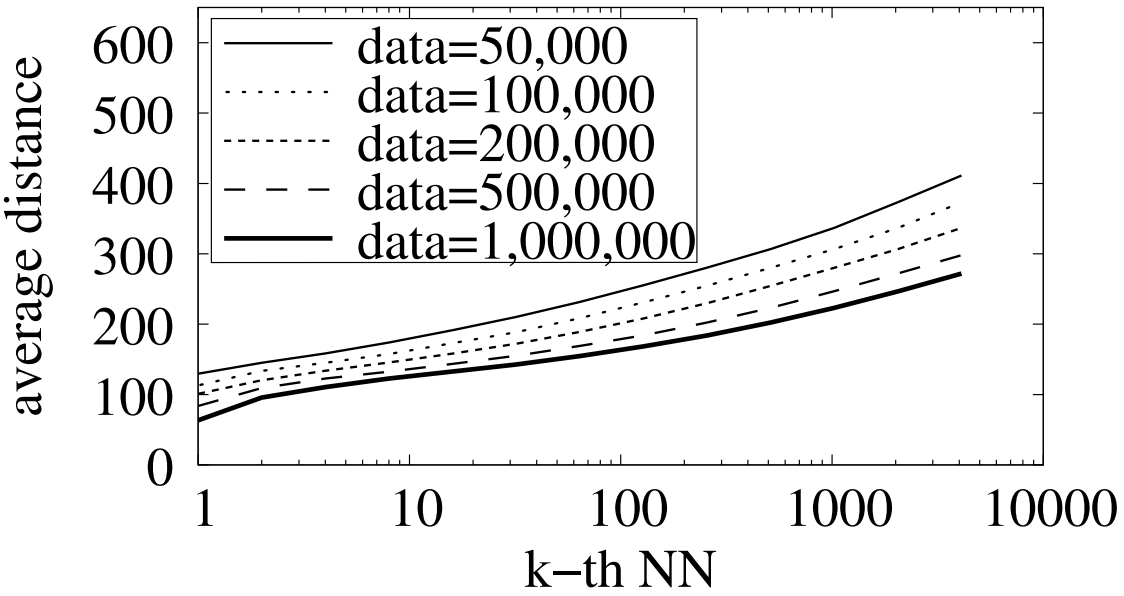




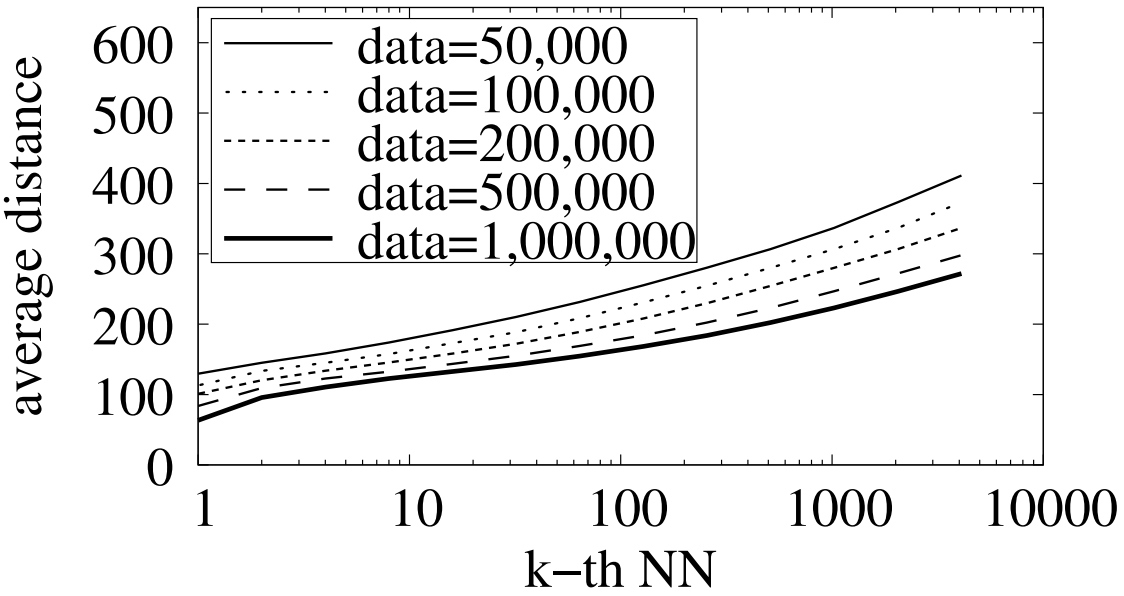
- M-Index with virtual buckets
- tunable number of accessed objects

- comparison for similar settings: bucket with 3,000 objects

k-th Nearest Neighbors



k-th Nearest Neighbors

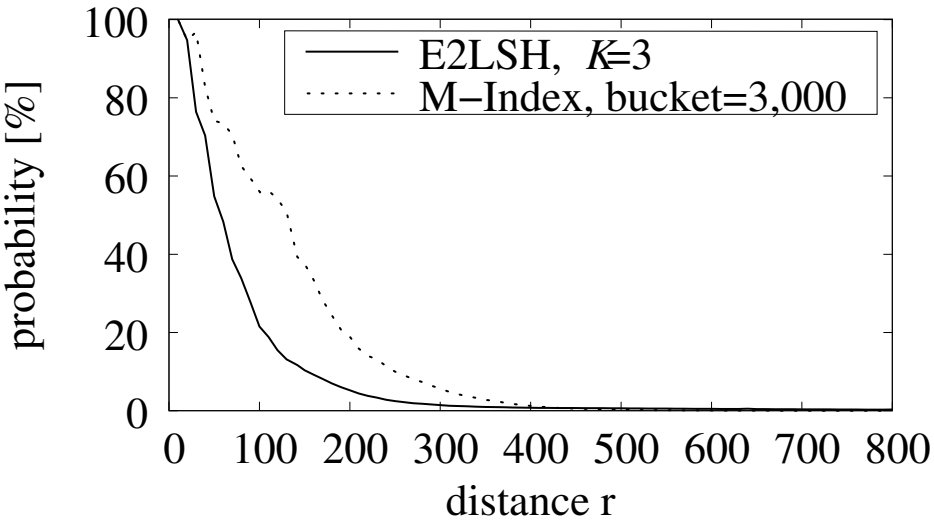


Definition

Given metric space $\mathcal{M} = (\mathcal{D}, d)$, a finite set of indexed objects $X \subseteq \mathcal{D}$ and two objects $p, q \in X$, let us denote $N_{p(q)} = \{u \in X | d(p, u) < d(p, q)\}$. We say that p, q are k-th nearest neighbors for $k \in \mathbb{N}$, iff

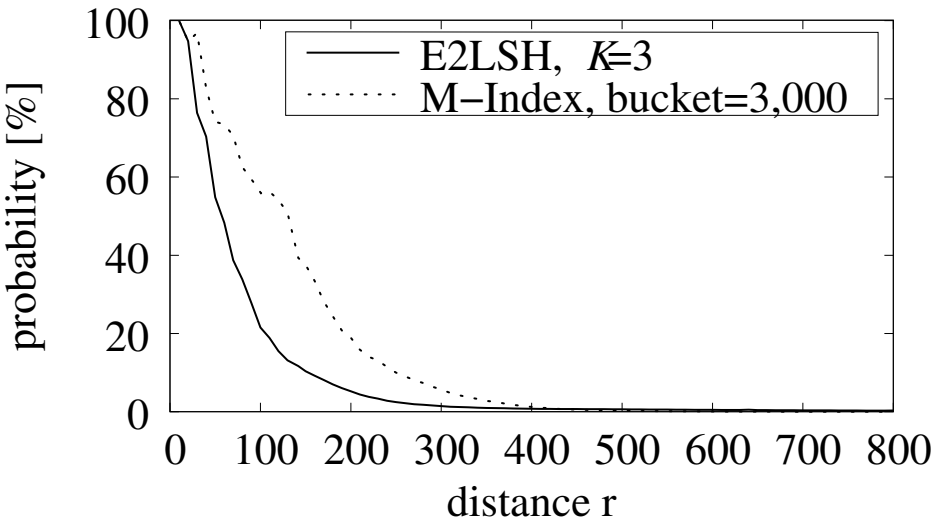
$$\min\left\{|N_{p(q)}|, |N_{q(p)}|\right\} = k - 1.$$

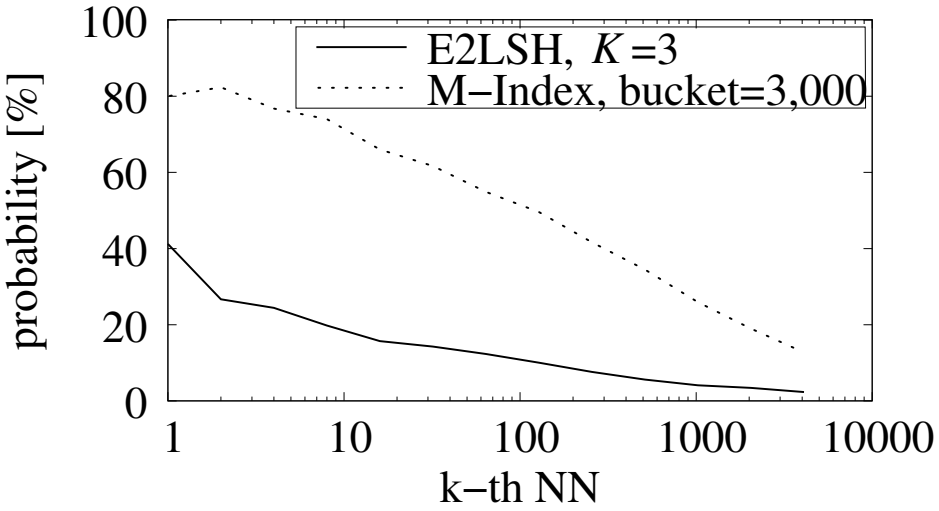
Collision Probability of kNNs



- collision probability with respect to mutual distances

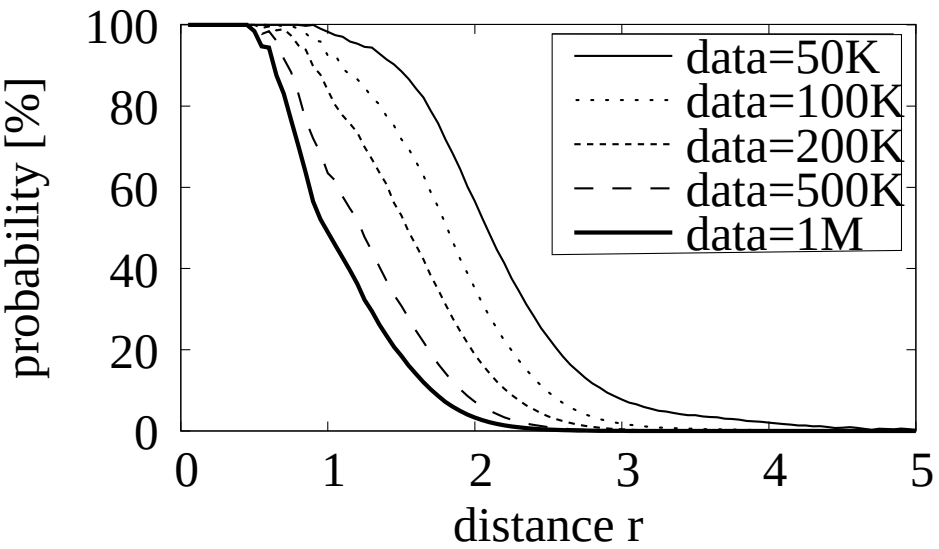
Collision Probability of kNNs





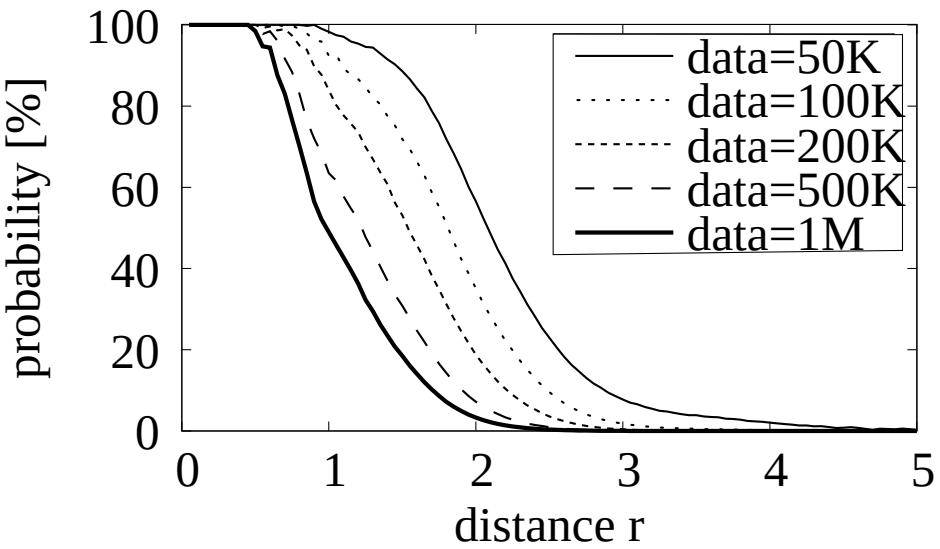
- collision probability with respect to mutual distances
- collision probability of k-NNs with respect to k

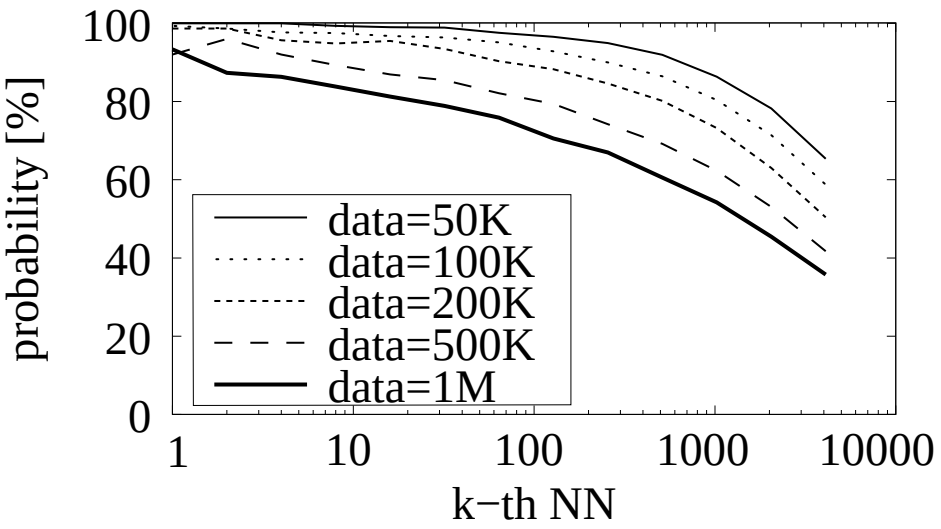
Scalability



- CoPhIR dataset
- fixed virtual-bucket size:
  10,000 objects

Scalability





- CoPhIR dataset
- fixed virtual-bucket size: 10,000 objects

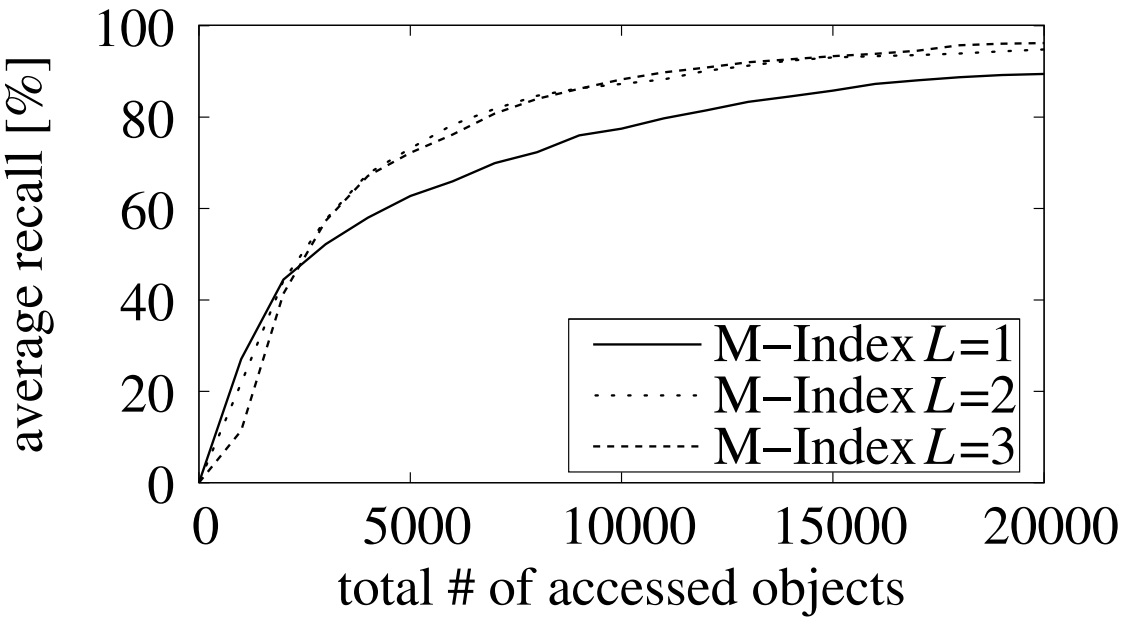
- collision probability of k-NNs with respect to k

Multiple Indexes

- use multiple indexes: $L > 1$
- measure recall with respect to total number of accessed objects

Multiple Indexes

- use multiple indexes: $L > 1$
- measure recall with respect to total number of accessed objects



- 1,000,000 CoPhIR dataset

Conclusions

- M-Index demonstrated to be locality-sensitive in terms of LSH theory
- we discussed shapes of the collision probability curves
- we propose to measure collision probability with respect to k-th NNs
- M-Index multi-probe seems more efficient than standard LSH concatenation
- M-Index allows to tune the search costs precisely

Conclusions

- M-Index demonstrated to be locality-sensitive in terms of LSH theory
- we discussed shapes of the collision probability curves
- we propose to measure collision probability with respect to k-th NNs
- M-Index multi-probe seems more efficient than standard LSH concatenation
- M-Index allows to tune the search costs precisely

Future work directions:

- study influence of multiple indexes (parameter L)
  - statistical properties of recall (variance)
- investigate fault-tolerance gained by multiple indexes
  - distributed M-Index