Metric Indexes based on Recursive Voronoi Partitioning

David Novak and Pavel Zezula

Laboratory of Data Intensive Systems and Applications (DISA)
Masaryk University, Brno, Czech Republic

Spring 2015
http://disa.fi.muni.cz/

Outline of the Talk

1. Motivation: Efficiency of Similarity Search
2. Metric Data Partitioning
   - Fundamentals
   - Voronoi Partitioning
3. M-Index
   - Principles
   - Precise and Approximate Search
   - M-Index Related Pieces of Work
4. PPP-Codes
   - Data Encoding and Searching
   - Efficiency Evaluation
   - Visual Search Demo

Motivation

- The similarity is key to human cognition, learning, memory...

  [cognitive psychology]

Motivation

- The similarity is key to human cognition, learning, memory. . .

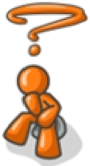
  [cognitive psychology]

- everything we can see, hear, measure, observe is in digital form

Motivation

- The similarity is key to human cognition, learning, memory...

  [cognitive psychology]
- everything we can see, hear, measure, observe is in digital form
- Computers should be able to search data based on similarity



Motivation

- The similarity is key to human cognition, learning, memory. . .
  [cognitive psychology]
- everything we can see, hear, measure, observe is in digital form
- Computers should be able to search data based on similarity

The similarity search problem has two aspects

- **effectiveness**: how to measure similarity of two "objects"
  - domain specific (photos, X-rays, voice, music, EEG, MTR. . .)

Motivation

- The similarity is key to human cognition, learning, memory...
  [cognitive psychology]
- everything we can see, hear, measure, observe is in digital form
- Computers should be able to search data based on similarity

The similarity search problem has two aspects

- **effectiveness**: how to measure similarity of two "objects"
  - domain specific (photos, X-rays, voice, music, EEG, MTR...)
- **efficiency**: how to realize similarity search fast
  - using a given data + similarity measure
  - on very large data collections

Efficiency: Motivation Example

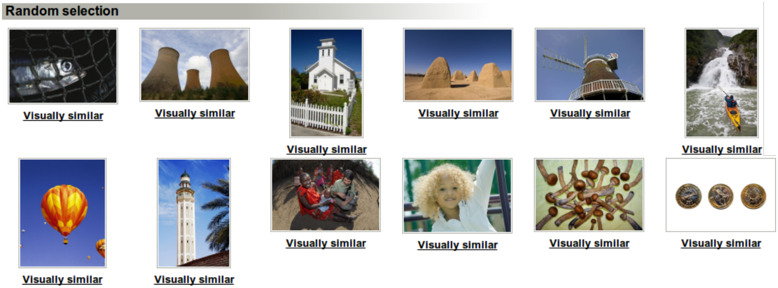
Example of data:

- general images (photos)
- every image processed by a deep convolutional neural network
  - to obtain a visual characterization of the image (descriptor)
  - compared by Euclidean distance to measure visual similarity

Efficiency: Motivation Example

Example of data:

- general images (photos)
- every image processed by a deep convolutional neural network
  - to obtain a visual characterization of the image (descriptor)
  - compared by Euclidean distance to measure visual similarity



Efficiency: Motivation Example

Example of data:

- general images (photos)
- every image processed by a deep convolutional neural network
  - to obtain a visual characterization of the image (descriptor)
  - compared by Euclidean distance to measure visual similarity

Efficiency: Motivation Example

Example of data:

- general images (photos)
- every image processed by a deep convolutional neural network
    - to obtain a visual characterization of the image (descriptor)
    - compared by Euclidean distance to measure visual similarity

Efficiency problem:

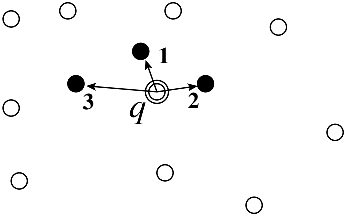
- 20 million of images with such descriptors
- each descriptor is a 4096-dimensional float vector
- ⇒ over 320 GB of data to be organized for similarity search
    - answer similarity queries online

Distance-based Similarity Search

- generic similarity search
  - applicable to many domains
- data modeled as metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is a *domain* of objects and δ is a total *distance function* $\delta : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}_0^+$ satisfying postulates of identity, symmetry, and triangle inequality

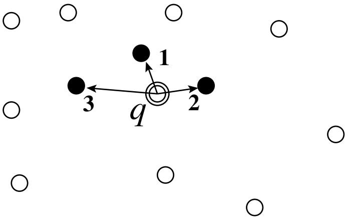
Distance-based Similarity Search

- generic similarity search
  - applicable to many domains
- data modeled as metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is a *domain* of objects and δ is a total *distance function* $\delta : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}_0^+$ satisfying postulates of identity, symmetry, and triangle inequality
- query by example: k-NN(q) returns k objects x from the dataset $\mathcal{X} \subseteq \mathcal{D}$ with the smallest $\delta(q, x)$



Distance-based Similarity Search

- generic similarity search
  - applicable to many domains
- data modeled as metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is a *domain* of objects and δ is a total *distance function* $\delta : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}_0^+$ satisfying postulates of identity, symmetry, and triangle inequality
- query by example: k-NN(q) returns k objects x from the dataset $\mathcal{X} \subseteq \mathcal{D}$ with the smallest $\delta(q, x)$
- dataset $\mathcal{X}$ may be very large
- function δ may be time consuming



Voronoi Partitioning

In metric space, there is no absolute order of the objects, no coordinates,...

Voronoi Partitioning

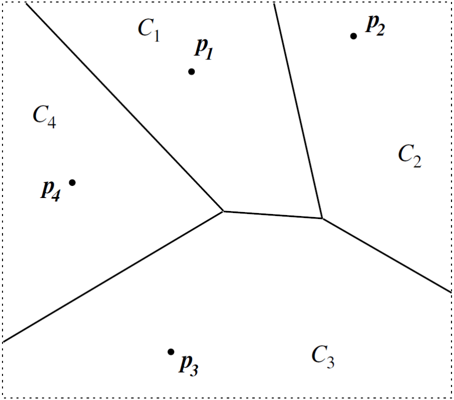
In metric space, there is no absolute order of the objects, no coordinates,...

- Dataset partitioning is done using reference objects (pivots)

Voronoi Partitioning

In metric space, there is no absolute order of the objects, no coordinates,...

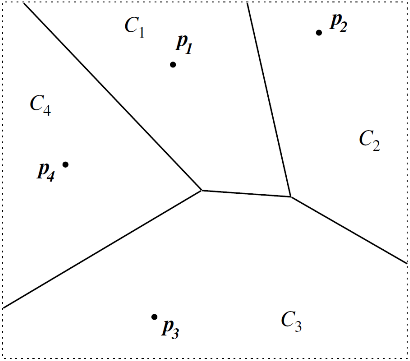
- Dataset partitioning is done using reference objects (pivots)
- Let us have a fixed set of n pivots $\{p_1, \ldots, p_n\}$



- Voronoi cell C_i = all objects for which pivot p_i is the closest

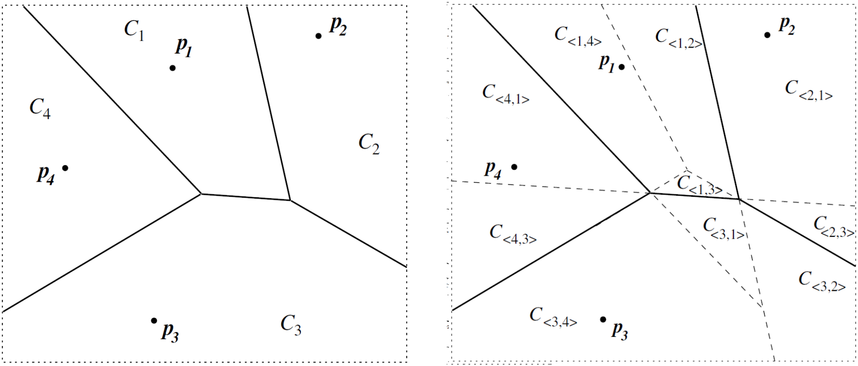
Recursive Voronoi Partitioning

- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively



Recursive Voronoi Partitioning

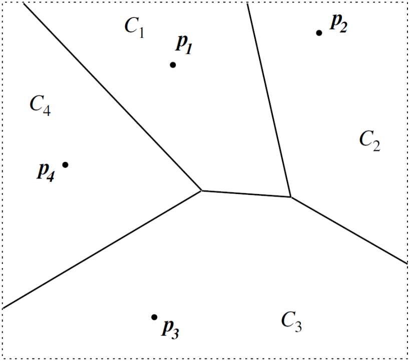
- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively

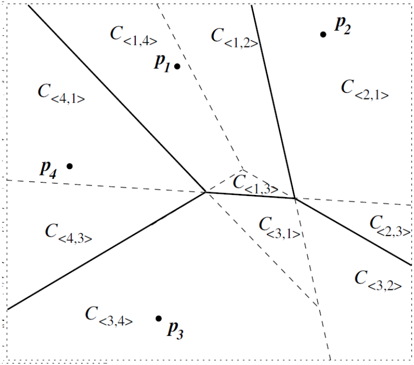


- Partition each C_i using the other pivots $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$

Recursive Voronoi Partitioning

- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively

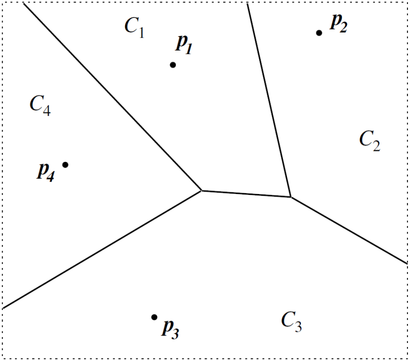


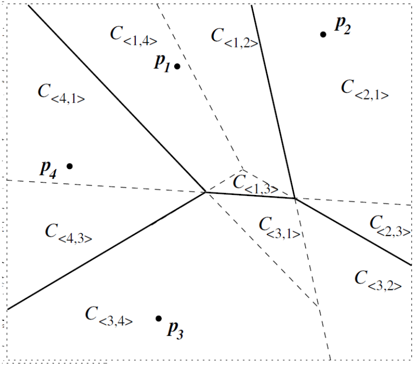


- Partition each C_i using the other pivots $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$
- $C_{i,j}$ = objects for which p_i is the closest and p_j the second closest

Recursive Voronoi Partitioning

- Let us use the same set of n pivots $p_1, \ldots, p_n$ recursively





- Partition each C_i using the other pivots $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$
- $C_{i,j}$ = objects for which p_i is the closest and p_j the second closest
    - this principle can be used l-times recursively up to level $l = n$

Pivot Permutations

A different point of view: **(prefixes of) pivot permutations**

- For each object $x \in \mathcal{X}$, order the pivots according to distances $\delta(x, p_i)$

Pivot Permutations

A different point of view: **(prefixes of) pivot permutations**

- For each object $x \in \mathcal{X}$, order the pivots according to distances $\delta(x, p_i)$
- Let Π_x be a permutation on the set of pivot indexes $\{1, \ldots, n\}$ such that $\Pi_x(j)$ is index of the j-th closest pivot from x
  - for example, $\Pi_x(1)$ is index of the closest pivot from x
  - $p_{\Pi_x(j)}$ is the j-the closest pivot from x

Pivot Permutations

A different point of view: **(prefixes of) pivot permutations**

- For each object $x \in \mathcal{X}$, order the pivots according to distances $\delta(x, p_i)$
- Let Π_x be a permutation on the set of pivot indexes $\{1, \ldots, n\}$ such that $\Pi_x(j)$ is index of the j-th closest pivot from x
  - for example, $\Pi_x(1)$ is index of the closest pivot from x
  - $p_{\Pi_x(j)}$ is the j-the closest pivot from x
- Formally: Π_x is permutation on $\{1, \ldots, n\}$ such that $\forall i : 1 \leq i < n$:

$$\delta(x, p_{\Pi_x(i)}) < \delta(x, p_{\Pi_x(i+1)})$$

Pivot Permutations

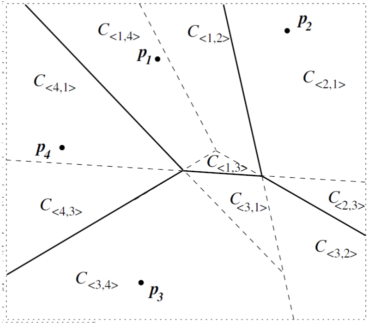
A different point of view: **(prefixes of) pivot permutations**

- For each object $x \in \mathcal{X}$, order the pivots according to distances $\delta(x, p_i)$
- Let Π_x be a permutation on the set of pivot indexes $\{1, \ldots, n\}$ such that $\Pi_x(j)$ is index of the j-th closest pivot from x
  - for example, $\Pi_x(1)$ is index of the closest pivot from x
  - $p_{\Pi_x(j)}$ is the j-the closest pivot from x
- Formally: Π_x is permutation on $\{1, \ldots, n\}$ such that $\forall i : 1 \leq i < n$:

$$\delta(x, p_{\Pi_x(i)}) < \delta(x, p_{\Pi_x(i+1)})$$

- Π_x is denoted as pivot permutation (PP) with respect to x.

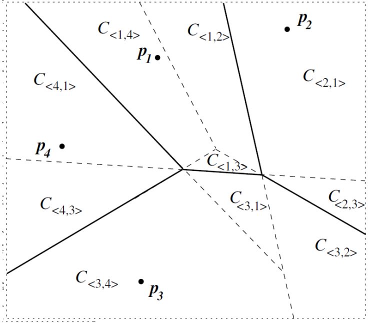
Correspondence between Voronoi partitioning and PPs



- Recursive Voronoi partitioning to level l
- Cell $C_{\langle i_1,\dots,i_l \rangle}$ contains objects x for which

$$\Pi_x(1) = i_1,\ \Pi_x(2) = i_2,\ \dots, \Pi_x(l) = i_l$$

Correspondence between Voronoi partitioning and PPs



- Recursive Voronoi partitioning to level l
- Cell $C_{\langle i_1,\ldots,i_l \rangle}$ contains objects x for which

$$\Pi_x(1) = i_1,\ \Pi_x(2) = i_2,\ \ldots, \Pi_x(l) = i_l$$

- l-tuple $\langle i_1, \ldots, i_l \rangle$ is an l-prefix of pivot permutation Π_x
  - pivot permutation prefix (PPP)
  - terms "Voronoi cell" and "PPP" correspond to each other

M-Index Indexing Structure

Our specific Voronoi indexes: M-Index, Distributed M-Index, PPP-Codes

M-Index Indexing Structure

Our specific Voronoi indexes: M-Index, Distributed M-Index, PPP-Codes

M-Index: basic properties

- uses dynamic recursive Voronoi partitioning (details later)
- it defines a (hash) mapping from the metric space to (float) numbers
- data either in memory or on disk (continuous chunks)
- both precise and approximate similarity search

M-Index Indexing Structure

Our specific Voronoi indexes: M-Index, Distributed M-Index, PPP-Codes

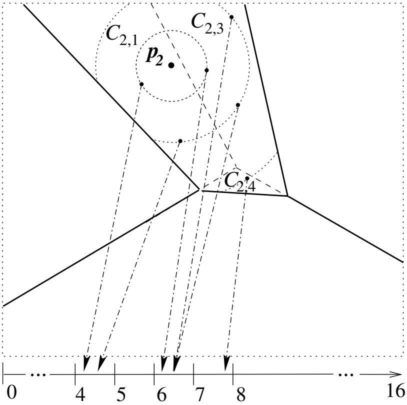
M-Index: basic properties

- uses dynamic recursive Voronoi partitioning (details later)
- it defines a (hash) mapping from the metric space to (float) numbers
- data either in memory or on disk (continuous chunks)
- both precise and approximate similarity search

Novak, D. and Batko, M. (2009). Metric Index: An Efficient and Scalable Solution for Similarity Search. In *Proceedings of SISAP 09*, (pp. 65–73). IEEE Comput. Soc. Press.

Novak, D., Batko, M. and Zezula, P. (2011). Metric Index: An Efficient and Scalable Solution for Precise and Approximate Similarity Search. *Inform. Syst.*, 36(4), 721–733.

M-Index Mapping Function



example with $n = 4$ and $l = 2$

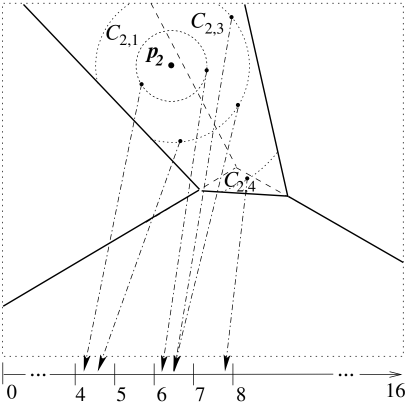
integral part of the key

- identification of the cell

fractional part of the key

- position within the cell
- distance from the closest pivot

M-Index Mapping Function



example with $n = 4$ and $l = 2$

integral part of the key

- identification of the cell

fractional part of the key

- position within the cell
- distance from the closest pivot

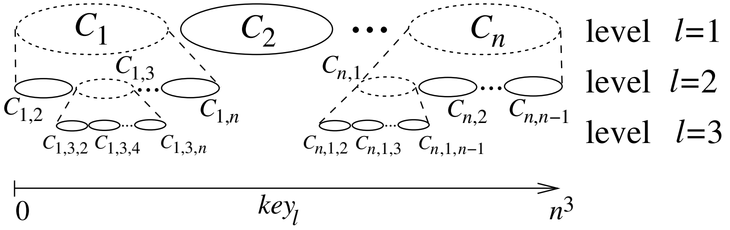
$$key_l(x) =$$

$$= \delta(p_{\Pi_x(1)}, x) + \sum_{i=1}^{l} (\Pi_x(i) - 1) \cdot n^{(l-i)}$$

domain of δ normalized to $[0, 1)$

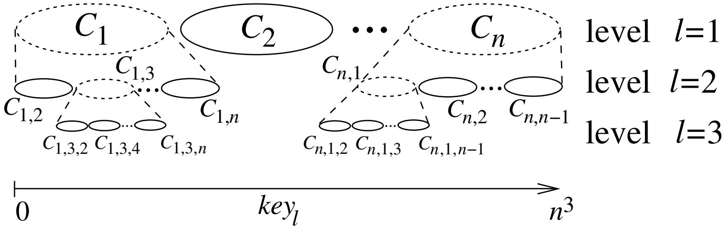
size of the key_l domain: n^l

M-Index with Dynamic Level



- partition only those cells that exceed certain capacity
- pick a maximum level $1 \leq l_{\max} \leq n$

M-Index with Dynamic Level



- partition only those cells that exceed certain capacity
- pick a maximum level $1 \leq l_{\max} \leq n$
- the key_l formula becomes:

$$key_l(x) = d(p_{\Pi_x(1)}, x) + \sum_{i=1}^{l} (\Pi_x(i) - 1) \cdot n^{(l_{\max} - i)}$$

M-Index: Precise Range Query Evaluation

Precise evaluation of range query $R(q, r)$ employs practically all known metric principles of space pruning and filtering:

M-Index: Precise Range Query Evaluation

Precise evaluation of range query $R(q, r)$ employs practically all known metric principles of space pruning and filtering:

- double-pivot distance constraint
  - skip accessing of Voronoi cell C_i if

$$\delta(q, p_i) - \delta(q, p_{\Pi_q(1)}) > 2 \cdot r$$

    - use hyperplane between pivot p_i and $q_{\Pi_q(1)}$
    - apply l-times for cell $C_{i_1,\ldots,i_l}$

M-Index: Precise Range Query Evaluation

Precise evaluation of range query $R(q, r)$ employs practically all known metric principles of space pruning and filtering:

- double-pivot distance constraint
  - skip accessing of Voronoi cell C_i if

$$\delta(q, p_i) - \delta(q, p_{\Pi_q(1)}) > 2 \cdot r$$

    - use hyperplane between pivot p_i and $q_{\Pi_q(1)}$
    - apply l-times for cell $C_{i_1,\ldots,i_l}$
- range-pivot distance constraint
  - each leaf cell $C_{i_1,\ldots,i_l}$ stores $r_{\min}$ and $r_{\max}$ as min and max of distances $\{\delta(x, p_{i_1}) | x \in C_{i_1,\ldots,i_l}\}$
  - skip accessing of cell $C_{i_1,\ldots,i_l}$ if

$$\delta(q, p_{i_1}) + r < r_{\min} \qquad \text{or} \qquad \delta(q, p_{i_1}) - r > r_{\max}$$

M-Index: Precise Range Query Evaluation (cont.)

- object-pivot distance constraint
  - the fractional part of an M-Index key is an object-pivot distance
  - for range query $R(q, r)$ identify interval of keys in cell $C_{i_1,\ldots,i_l}$

$$[\delta(q, p_{i_1}) - r, \delta(q, p_{i_1}) + r]$$

M-Index: Precise Range Query Evaluation (cont.)

- object-pivot distance constraint
  - the fractional part of an M-Index key is an object-pivot distance
  - for range query $R(q, r)$ identify interval of keys in cell $C_{i_1,\ldots,i_l}$

$$[\delta(q, p_{i_1}) - r, \delta(q, p_{i_1}) + r]$$

- pivot filtering
  - store distances $\delta(x, p_1), \ldots, \delta(x, p_n)$ together with each object x
  - skip computation of $\delta(q, x)$ at query time if

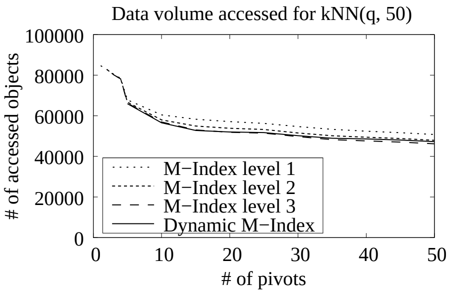
$$\max_{i \in \{1,\ldots,n\}} |\delta(q, p_i) - \delta(x, p_i)| > r$$

M-Index Precise Strategy: Brief Evaluation

- Dataset: CoPhIR (Content-based Photo Information Retrieval)
  - combination of five MPEG-7 descriptors
  - 280 dimensions altogether, weighted sum of partial distances

M-Index Precise Strategy: Brief Evaluation

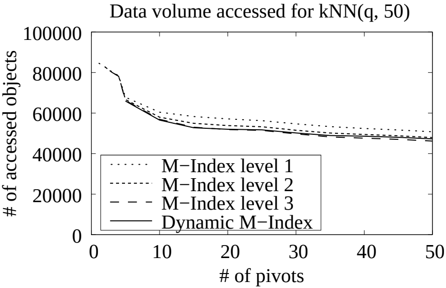
- Dataset: CoPhIR (Content-based Photo Information Retrieval)
  - combination of five MPEG-7 descriptors
  - 280 dimensions altogether, weighted sum of partial distances

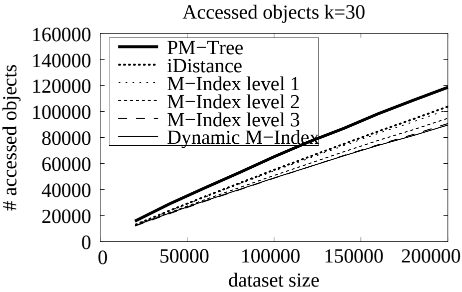


- dataset size: 100,000
- dynamic M-Index: $l_{max} = 5$

M-Index Precise Strategy: Brief Evaluation

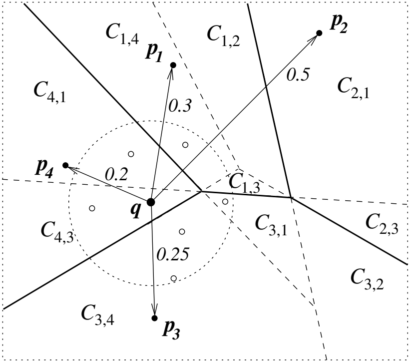
- Dataset: CoPhIR (Content-based Photo Information Retrieval)
  - combination of five MPEG-7 descriptors
  - 280 dimensions altogether, weighted sum of partial distances





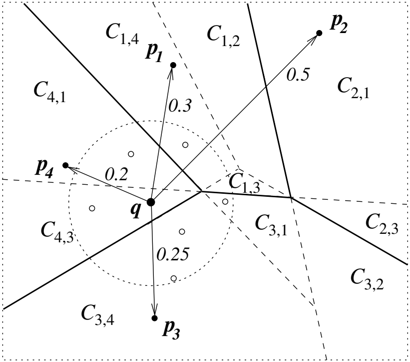
- dataset size: 100,000
- dynamic M-Index: $l_{\max} = 5$
- 20 pivots

Approximate Strategy for M-Index



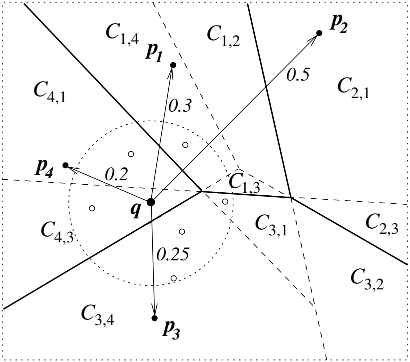
- Determine order in which to visit individual cells
- Estimate "distances" between the query and the Voronoi cells (PPPs)

Approximate Strategy for M-Index



- Determine order in which to visit individual cells
- Estimate "distances" between the query and the Voronoi cells (PPPs)
- query q is represented by distances $\delta(q, p_1), \delta(q, p_2), \dots, \delta(q, p_n)$

Approximate Strategy for M-Index



- Determine order in which to visit individual cells
- Estimate “distances” between the query and the Voronoi cells (PPPs)
- query q is represented by distances $\delta(q,p_1),\delta(q,p_2),\ldots,\delta(q,p_n)$

- each Voronoi cell is assigned a “penalty” with respect to q

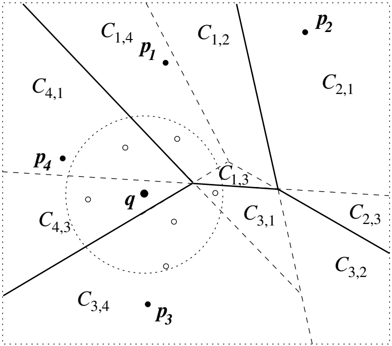
$$penalty(C_{i_1,\ldots,i_l}) = \sum_{j=1}^{l} \max\left\{\delta(p_{i_j},q) - \delta(p_{\Pi_q(j)},q), 0\right\}$$

Approximate Strategy: Other Options

- another natural option is to represent the query by its Voronoi cell
  - and to estimate "distances" between the Voronoi cells (PPPs)

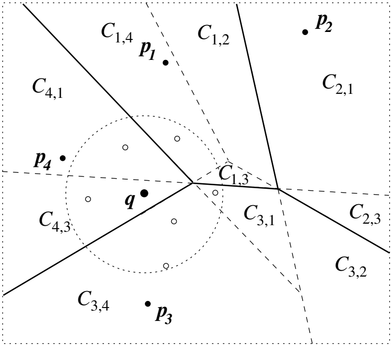
Approximate Strategy: Other Options

- another natural option is to represent the query by its Voronoi cell
  - and to estimate "distances" between the Voronoi cells (PPPs)



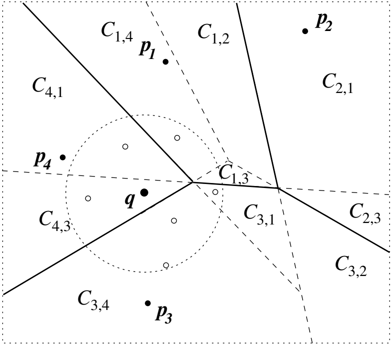
Approximate Strategy: Other Options

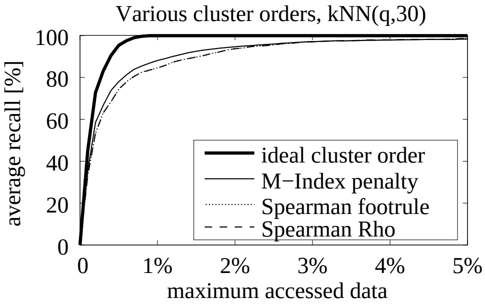
- another natural option is to represent the query by its Voronoi cell
  - and to estimate "distances" between the Voronoi cells (PPPs)
- Kendall Tau, Spearman Footrule distance, Spearman Rho, ...



Approximate Strategy: Other Options

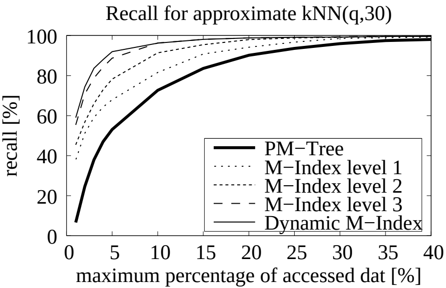
- another natural option is to represent the query by its Voronoi cell
  - and to estimate "distances" between the Voronoi cells (PPPs)
- Kendall Tau, Spearman Footrule distance, Spearman Rho, ...





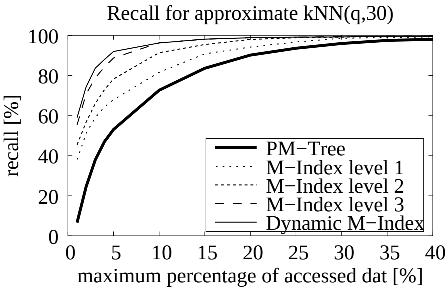
- using richer information for the query than for data is worth

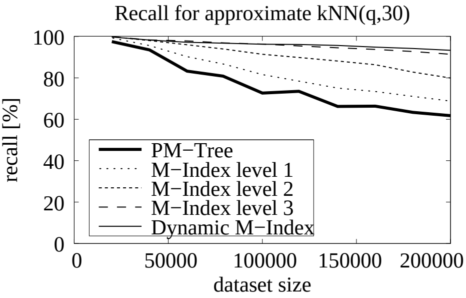
M-Index Approximate Strategy: Brief Evaluation



- dataset of 100,000 objects

M-Index Approximate Strategy: Brief Evaluation





- dataset of 100,000 objects
- algorithm accesses 10,000 objects

M-Index Related Pieces of Work

- Distributed indexes: M-Chord (preliminary), distributed M-Index

Novak, D., and Zezula, P. (2006). M-Chord: A Scalable Distributed Similarity Search Structure. In Proceedings InfoScale 06 (pp. 1–10). ACM Press.

Novak, D., Batko, M., and Zezula, P. (2012). Large-scale similarity data management with distributed Metric Index. Information Processing & Management, 48(5), 855–872.

M-Index Related Pieces of Work

- Distributed indexes: M-Chord (preliminary), distributed M-Index

  Novak, D., and Zezula, P. (2006). M-Chord: A Scalable Distributed Similarity Search Structure. In Proceedings InfoScale 06 (pp. 1–10). ACM Press.

  Novak, D., Batko, M., and Zezula, P. (2012). Large-scale similarity data management with distributed Metric Index. Information Processing & Management, 48(5), 855–872.

- M-Index defines a locality-sensitive hashing function for metric spaces

  Novak, D., Kyselak, M., and Zezula, P. (2010). On locality-sensitive indexing in generic metric spaces. In Processing of SISAP 10 (pp. 59–66). ACM Press.

M-Index Related Pieces of Work

- Distributed indexes: M-Chord (preliminary), distributed M-Index

  Novak, D., and Zezula, P. (2006). M-Chord: A Scalable Distributed Similarity Search Structure. In Proceedings InfoScale 06 (pp. 1–10). ACM Press.

  Novak, D., Batko, M., and Zezula, P. (2012). Large-scale similarity data management with distributed Metric Index. Information Processing & Management, 48(5), 855–872.

- M-Index defines a locality-sensitive hashing function for metric spaces

  Novak, D., Kyselak, M., and Zezula, P. (2010). On locality-sensitive indexing in generic metric spaces. In Processing of SISAP 10 (pp. 59–66). ACM Press.

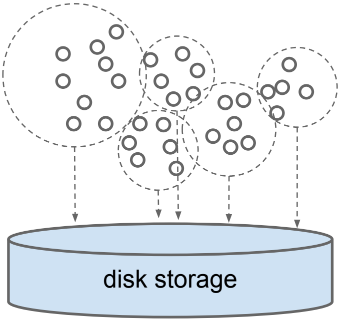
- Multiple independent M-Indexes to improve the search (LSH style)

  Novak, D., and Zezula, P. (2014). Performance Study of Independent Anchor Spaces for Similarity Searching. The Computer Journal, 57(11), 1741–1755.

Standard Similarity Search Approach

standard approach to large-scale approximate search (e.g. M-Index):

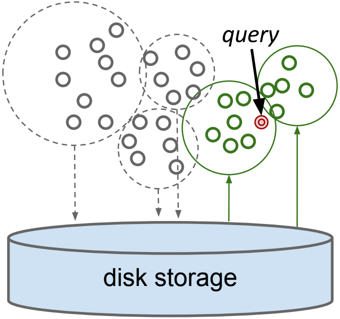
- dataset $\mathcal{X}$ is partitioned and stored



Standard Similarity Search Approach

standard approach to large-scale approximate search (e.g. M-Index):

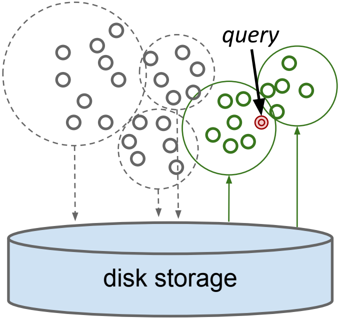
- dataset $\mathcal{X}$ is partitioned and stored
- given query q, the "most-promising" partitions form the candidate set



Standard Similarity Search Approach

standard approach to large-scale approximate search (e.g. M-Index):

- dataset $\mathcal{X}$ is partitioned and stored
- given query q, the "most-promising" partitions form the candidate set
- the candidate set S_C is refined by calculating $\delta(q, x)$, $\forall x \in S_C$



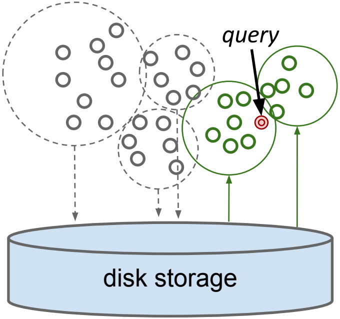
Standard Similarity Search Approach

standard approach to large-scale approximate search (e.g. M-Index):

- dataset $\mathcal{X}$ is partitioned and stored
- given query q, the "most-promising" partitions form the candidate set
- the candidate set S_C is refined by calculating $\delta(q, x)$, $\forall x \in S_C$

majority of the search costs:

- reading and refinement of S_C



Standard Similarity Search Approach

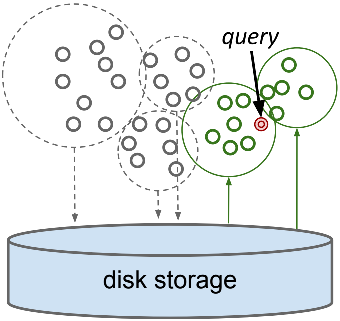
standard approach to large-scale approximate search (e.g. M-Index):

- dataset $\mathcal{X}$ is partitioned and stored
- given query q, the "most-promising" partitions form the candidate set
- the candidate set S_C is refined by calculating $\delta(q, x)$, $\forall x \in S_C$

majority of the search costs:

- reading and refinement of S_C

$\Longrightarrow$ accuracy of the candidate set is key



PPP-Codes in a Nutshell

1. data space is partitioned multiple-times independently
    - each partitioning is defined by one pivot space (recursive Voronoi)

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space (recursive Voronoi)
2. based on these partitionings, objects are mapped onto memory codes

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space (recursive Voronoi)
2. based on these partitionings, objects are mapped onto memory codes
3. given query q, multiple ranked candidate sets are generated

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space (recursive Voronoi)
2. based on these partitionings, objects are mapped onto memory codes
3. given query q, multiple ranked candidate sets are generated
4. these candidate rankings are effectively merged
   - the merged candidate set is smaller and more accurate

PPP-Codes in a Nutshell

1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space (recursive Voronoi)
2. based on these partitionings, objects are mapped onto memory codes
3. given query q, multiple ranked candidate sets are generated
4. these candidate rankings are effectively merged
   - the merged candidate set is smaller and more accurate
5. the final candidate set is retrieved and refined

PPP-Codes in a Nutshell

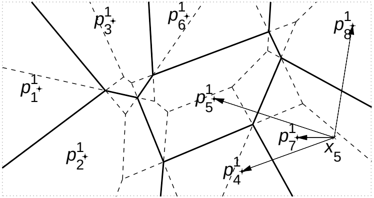
1. data space is partitioned multiple-times independently
   - each partitioning is defined by one pivot space (recursive Voronoi)
2. based on these partitionings, objects are mapped onto memory codes
3. given query q, multiple ranked candidate sets are generated
4. these candidate rankings are effectively merged
   - the merged candidate set is smaller and more accurate
5. the final candidate set is retrieved and refined

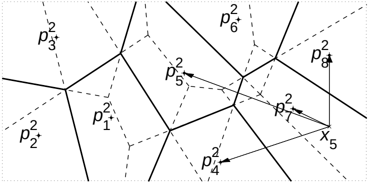
Novak, D. and Zezula, P. (2014). Rank Aggregation of Candidate Sets for Efficient Similarity Search. In DEXA 2014 Proceedings, Part II (Vol. 8645, pp. 42–58). Springer.

Best paper of DEXA 2014 Award.

Space Partitioning and Data Encoding

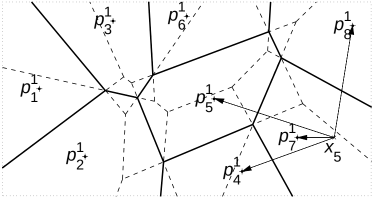
PPP-Codes define λ independent recursive Voronoi-like space partitionings

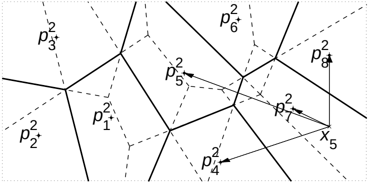




Space Partitioning and Data Encoding

PPP-Codes define λ independent recursive Voronoi-like space partitionings





each data object $x \in \mathcal{X}$ is encoded by position in these diagrams

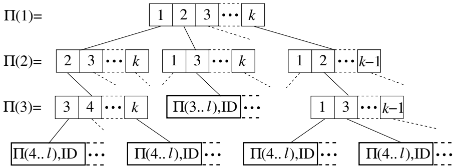
$$PPP\text{-}Code_l^{1..\lambda}(x) = \langle \Pi_x^1(1..l), \ldots, \Pi_x^\lambda(1..l) \rangle.$$

where $\Pi_x^j(1..l)$ is the l-prefix of the j-th pivot permutation of object x

PPP-Code Index

We build a trie-like structure for each pivot space

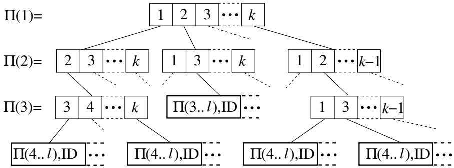
- the memory trie contains only the PPP-Codes and object IDs
- with a focus is on memory optimization



PPP-Code Index

We build a trie-like structure for each pivot space

- the memory trie contains only the PPP-Codes and object IDs
- with a focus is on memory optimization



Given query q, Voronoi cells from each partitioning (trie) are ranked

- in a similar way as for M-Index
- result: λ independent candidate rankings of object IDs

Candidate set Identification

- λ rankings ψ_q^j of IDs are aggregated into the final ranking

Candidate set Identification

- λ rankings ψ_q^j of IDs are aggregated into the final ranking

$q \in \mathcal{D}$

$$\psi_q^1\colon\ \overbrace{\{\boldsymbol{x}\ y_1\, y_2\}}^{\text{objects with the rank '1'}}\ \overbrace{\{y_3\, y_4\, y_5\}}^{\text{rank '2'}}\ \overbrace{\{y_6\}}^{\text{rank '3'}}\ \ldots$$

$$\psi_q^2\colon\ \{y_3\, y_2\}\ \{y_1\, y_4\, y_6\, y_7\}\ \{\boldsymbol{x}\ y_8\}\ \ldots$$

$$\psi_q^3\colon\ \{\boldsymbol{x}\}\ \{y_3\, y_4\, y_5\}\ \{y_2\, y_6\}\ \ldots$$

$$\psi_q^4\colon\ \{y_1\, y_2\}\ \{y_3\, y_4\, y_5\}\ \{y_8\}\ \{y_6\}\ \ldots$$

$$\psi_q^5\colon\ \{y_1\, y_2\}\ \{y_4\, y_5\}\ \{y_3\}\ \{\boldsymbol{x}\ y_7\}\ \ldots$$

Candidate set Identification

- λ rankings ψ_q^j of IDs are aggregated into the final ranking
- ranking of object x is **p**-percentile (e.g. median) of its λ ranks

$$\Psi_{\mathbf{p}}(q, x) = percentile_{\mathbf{p}}(\psi_q^1(x), \psi_q^2(x), \dots, \psi_q^\lambda(x))$$

$q \in \mathcal{D}$

objects with the rank '1'
rank '2'
rank '3'

ψ_q^1: $\{\mathbf{x}\ y_1\ y_2\}$ $\{y_3\ y_4\ y_5\}$ $\{y_6\}$ ...
ψ_q^2: $\{y_3\ y_2\}$ $\{y_1\ y_4\ y_6\ y_7\}$ $\{\mathbf{x}\ y_8\}$ ...
ψ_q^3: $\{\mathbf{x}\}$ $\{y_3\ y_4\ y_5\}$ $\{y_2\ y_6\}$ ...
ψ_q^4: $\{y_1\ y_2\}$ $\{y_3\ y_4\ y_5\}$ $\{y_8\}$ $\{y_6\}$ ...
ψ_q^5: $\{y_1\ y_2\}$ $\{y_4\ y_5\}$ $\{y_3\}$ $\{\mathbf{x}\ y_7\}$ ...

$$\Psi_{0.5}(q, x) = percentile_{0.5}\{1, 1, \mathbf{3}, 4, ?\} = 3$$

Idea Behind the Rank Aggregation

- the Voronoi cells span large areas of the space

Idea Behind the Rank Aggregation

- the Voronoi cells span large areas of the space
- given a query, the “close” cells contain also distant data objects
    - actually, far more distant objects than close ones

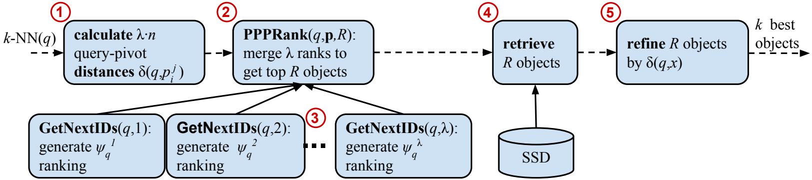
Idea Behind the Rank Aggregation

- the Voronoi cells span large areas of the space
- given a query, the "close" cells contain also distant data objects
    - actually, far more distant objects than close ones
- having several "orthogonal" partitionings
    - the query-relevant objects are at top positions of "all" partitionings
    - the distant objects at top positions vary

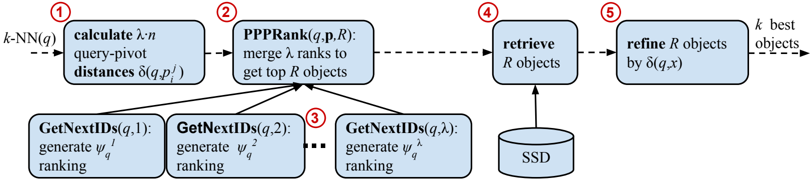
Idea Behind the Rank Aggregation

- the Voronoi cells span large areas of the space
- given a query, the "close" cells contain also distant data objects
  - actually, far more distant objects than close ones
- having several "orthogonal" partitionings
  - the query-relevant objects are at top positions of "all" partitionings
  - the distant objects at top positions vary
- the percentile-based aggregation increases probability that query-relevant objects are ranked higher than the distant ones

Overall schema of the PPP-Codes search algorithm



Overall schema of the PPP-Codes search algorithm



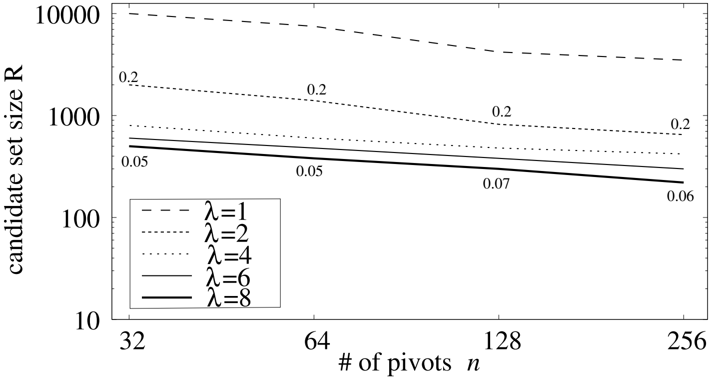
- individual steps run in separate threads
- requires a fast ID-object storage (SSD or distributed)

Evaluation 1: Accuracy of the Candidate Set

How many candidate objects are needed to achieve certain recall level

Evaluation 1: Accuracy of the Candidate Set

How many candidate objects are needed to achieve certain recall level



Candidate set size R necessary to achieve 80% of 1-NN recall

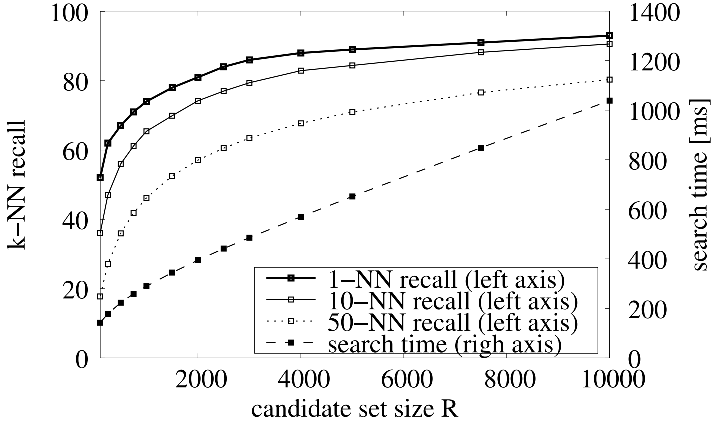
Settings: 1M CoPhIR dataset, $l = 8$ and $\mathbf{p} = 0.75$

Evaluation 2: Overall Efficiency

candidate set size R vs. recall and time

Evaluation 2: Overall Efficiency

candidate set size R vs. recall and time



Recall and search time while increasing candidate set size R.

Settings: 100M CoPhIR dataset, $n = 512$, $l = 8$, $\lambda = 5$, $\mathbf{p} = 0.5$ (3rd rank out of 5)

PPP-Codes Conclusions

The results of the PPP-Codes evaluation show that

- even two pivot spaces help, more than five do not help much
- the candidate set is reduced by one–two orders of magnitude
- the rank & merge algorithm is complex but usually worth
  - for larger data objects and complex distance function

PPP-Codes Conclusions

The results of the PPP-Codes evaluation show that

- even two pivot spaces help, more than five do not help much
- the candidate set is reduced by one–two orders of magnitude
- the rank & merge algorithm is complex but usually worth
  - for larger data objects and complex distance function

Demonstration of image visual search

- 20 million images
- powerful visual descriptors from deep convolutional neural networks
  - 4096-dimensional vectors with L_2 distance
- PPP-Codes index (1 GB in memory, 124 GB on the SSD disk)
- *To be presented at SIGIR 2015*

http://disa.fi.muni.cz/demos/profiset-decaf/