Metric Index: An Efficient and Scalable Solution for Similarity Search

David Novak and Michal Batko

Masaryk University
Brno, Czech Republic

August 29, 2009
SISAP 2009, Prague
Outline of the Talk

1 Motivation and Objectives
   • Similarity Indexing for Metric Spaces

2 Metric Index
   • One-level M-Index
   • Multi-level M-Index
   • Dynamic M-Index
   • M-Index Search Principles
   • Approximate Strategy for M-Index

3 Experimental Evaluation
Efficient and scalable metric-based index is still an issue

15 years of research:
- **principles** of space partitioning, search space pruning, filtering
- **fundamental** memory-based structures
- **advanced** index and search solutions
- **approximate** similarity search
- **distributed** index structures
Intentions of M-Index:

- synergically employ practically all known metric principles of space pruning and filtering
- fixed building costs (static set of reference points)
- use well-established efficient structures to factually store data
  - indexing based on mapping to real domain (use B⁺-tree)
- efficient precise and approximation similarity search
- straightforward way to distribute the structure

Metric Index

Notation and assumptions:

Metric space $\mathcal{M}$ is a pair $\mathcal{M} = (\mathcal{D}, d)$, where $\mathcal{D}$ is a set and $d$ is a total function $d: \mathcal{D} \times \mathcal{D} \rightarrow [0, 1)$ satisfying standard metric space conditions.
Metric Index

Intentions of M-Index:

- synergically employ practically all known metric principles of space pruning and filtering
- fixed building costs (static set of reference points)
- use well-established efficient structures to factually store data
  - indexing based on mapping to real domain (use $B^+$-tree)
- efficient precise and approximation similarity search
- straightforward way to distribute the structure

Notation and assumptions:

- Metric space $\mathcal{M}$ is a pair $\mathcal{M} = (\mathcal{D}, d)$, where $\mathcal{D}$ is a set and $d$ is a total function $d : \mathcal{D} \times \mathcal{D} \rightarrow [0, 1)$ satisfying standard metric space conditions.
Preliminaries: iDistance

\[ iDist(o) = d(p_i, o) + i \cdot c \]

- indexing technique for vector spaces
- application of object-pivot distance constraint
- index based purely on **metric** principles
- Voronoi partitioning
- **double-pivot** distance constraint for search-space pruning
Having a fixed set of $n$ pivots $\{p_0, p_1, \ldots, p_{n-1}\}$ and an object $o \in D$, let
\[
(\cdot)_o : \{0, 1, \ldots, n-1\} \rightarrow \{0, 1, \ldots, n-1\}
\]
be a permutation of indexes such that
\[
d(p_{(0)_o}, o) \leq d(p_{(1)_o}, o) \leq \cdots \leq d(p_{(n-1)_o}, o).
\]

$l$-level M-Index uses $l$-prefix of the pivot permutation, $1 \leq l \leq n$. 

in *l*-level M-Index, $\forall o \in D : o \in C_{(0)}o, \ldots, C_{(l)}o$

repetitive application of double-pivot distance constraint
key\_1(o) =

\[ = d(p_{(0)\circ}, o) + \sum_{i=0}^{l-1} (i) \cdot n^{(l-1-i)} \]

integral part of the key
- identification of the cluster

fractional part of the key
- position within the cluster

size of the key\_1 domain: \( n^l \)
M-Index with Dynamic Level

- establish a **maximum level** $1 \leq l_{\text{max}} \leq n$
- slightly modify the $key_l$ formula
  - analogous to *extensible hashing* + object-pivot distance
establish a maximum level $1 \leq l_{\text{max}} \leq n$

- slightly modify the $key_l$ formula
  - analogous to extensible hashing + object-pivot distance

- physically store the data according to the key
  - in a $B^+$-tree or similar structure
Search space **pruning principles** for $R(q, r)$ query

- **double-pivot** distance constraint
  - skip accessing of cluster $C_i$ if
    
    $$d(p_i, q) - d(p_{(0)}q, q) > 2 \cdot r$$

- due to recursive Voronoi partitioning
- apply $l$-times for $C_{i_0, \ldots, i_{l-1}}$
M-Index Search Principles

Search space pruning principles for \( R(q, r) \) query

- **double-pivot** distance constraint
  - skip accessing of cluster \( C_i \) if
    \[
    d(p_i, q) - d(p_{(0)}q, q) > 2 \cdot r
    \]
  - due to recursive Voronoi partitioning
  - apply \( l \)-times for \( C_{i_0, \ldots, i_{l-1}} \)

- **range-pivot** distance constraint
  - for leaf-level clusters \( C_{p,*} \) store min. and max. distance
    \[
    r_{\text{max}} = \max\{d(p, o)|o \in C_{p,*}\}
    \]
  - skip accessing of cluster \( C_{p,*} \) if
    \[
    d(p, q) + r < r_{\text{min}} \quad \text{or} \quad d(p, q) - r > r_{\text{max}}
    \]
M-Index Search Principles (cont.)

- **object-pivot distance constraint**
  - the M-Index key contains the object-pivot distance
  - identify interval of keys in cluster $C_{i_0, \ldots, i_l-1}$

$$[d(p_{i_0}, q) - r, d(p_{i_0}, q) + r]$$
M-Index Search Principles (cont.)

- **object-pivot distance constraint**
  - the M-Index key contains the object-pivot distance
  - identify interval of keys in cluster $C_{i_0,...,i_{l-1}}$

\[
[d(p_{i_0}, q) - r, d(p_{i_0}, q) + r]
\]

- **pivot filtering**
  - store distances $d(p_0, o), \ldots, d(p_{n-1}, o)$ together with object $o$
  - skip computation of $d(q, o)$ at query time if

\[
\max_{i \in \{0,...,n-1\}} |d(p_i, q) - d(p_i, o)| > r.
\]
Approximate Strategy for M-Index

Determine the order in which to visit individual clusters

- priority queue of clusters
- heuristic which analyzes distances $d(p_0, q), d(p_1, q), \ldots, d(p_{n-1}, q)$
- each cluster is assigned a penalty
  - distance of the cluster from $q$

\[
\text{penalty}(C_{i_0, \ldots, i_{l-1}}) = \sum_{j=0}^{l-1} \max \{ d(p_{i_j}, q) - d(p_{(j)q}, q), 0 \}
\]
Experimental Evaluation

- 200,000 objects from CoPhIR dataset
  - combination of five MPEG-7 descriptors
  - altogether 280 dimensions, intrinsic dimensionality: 13
- measure I/O costs (page reads and objects accessed), computational costs, response times
- compare with iDistance, PM-tree (same implementation platform)

Distance computations performed during index construction \((n = 20)\)

<table>
<thead>
<tr>
<th>dataset size</th>
<th>20,000</th>
<th>80,000</th>
<th>140,000</th>
<th>200,000</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M-Index</strong></td>
<td>400,000</td>
<td>1,600,000</td>
<td>2,800,000</td>
<td>4,000,000</td>
</tr>
<tr>
<td><strong>PM-Tree</strong></td>
<td>1,205,538</td>
<td>6,299,207</td>
<td>11,627,729</td>
<td>16,897,996</td>
</tr>
</tbody>
</table>
Precise Strategy: Number of objects accessed

- dataset size: 100,000
- dynamic M-Index: $l_{\text{max}} = 5$
Precise Strategy: Number of objects accessed

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

- 20 pivots
Precise Strategy: I/O costs, response times

- higher fragmentation with more M-Index levels
Precise Strategy: I/O costs, response times

- higher fragmentation with more M-Index levels
- lower computational costs for more M-Index levels
Approximate Strategy

Recall for approximate kNN(q,30)

- **PM-Tree**
- M-Index level 1
- M-Index level 2
- M-Index level 3
- Dynamic M-Index

- **dataset of 100,000 objects**
Approximate Strategy

Recall for approximate kNN(q,30)

- dataset of 100,000 objects
- algorithm accesses 10,000 objects
Comparison with Purely Approximate Approaches

- dataset size: 1,000,000
- 32 pivots ($n = 32$)
- dynamic M-Index with $l_{\text{max}} = 6$

**Recall for $k\text{NN}(q, 30)$**

- **ideal cluster order**
- **M-Index**
- **Spearman footrule**
- **Spearman Rho**
- **PP-Index, $z = 1,000$**

**Graphical Data**

Dataset size: 1,000,000

32 pivots ($n = 32$)

Dynamic M-Index with $l_{\text{max}} = 6$
Comparison with Purely Approximate Approaches

- dataset size: 1,000,000
- 32 pivots \((n = 32)\)
- dynamic M-Index with \(l_{\text{max}} = 6\)

**Spearman Footrule** modified to use permutation prefixes
- Induced Footrule Distance \([\text{Amato, Savino: Approximate Similarity Search in Metric Spaces using Inverted Files}]\)

**Spearman Rho** used in \([\text{Chavez, Figueroa, Navarro: Effective Proximity Retrieval by Ordering Permutations}]\)

**PP-Index** has a similar structure as M-Index
- access subtrees with at least \(z = 1,000\) objects