The M-Tree

Introduction

Update and Search

What is an M-tree?
- disk-based index structure for metric distances
- reduces search space for similarity query

Features of M-trees:
- dynamic (insertion and deletion of data objects)
- balanced tree (structure does not degenerate)
- supports range and k-nearest neighbor queries

Literature:
- M-trees were introduced by Ciaccia et al. [CPZ97] in 1997
- Textbook by Zezula et al. [ZADB06] covers M-trees
**The M-Tree**

**Introduction**

**M-Tree: Illustration**

- Internal nodes: prune irrelevant subtrees
  - Internal node: tuple of \( m \) entries \( (e_1, e_2, \ldots, e_m) \)
  - \( e_i = (p_i, r_{ci}, d(p_i, pp), ptr_i) \)
  - \( p_i \): pivot (some data object)
  - \( r_{ci} \): covering radius around \( p_i \)
  - \( d(p_i, pp) \): distance between \( p_i \) and the parent pivot \( pp \) of \( p_i \)
  - \( ptr_i \): pointer to a child node

- Guarantee: all objects in subtree \( p_i \) are at most at distance \( r_{ci} \) from \( p_i \)

- Leaf nodes: store data objects
  - Leaf node: tuple of \( m \) entries \( (f_1, f_2, \ldots, f_m) \)
  - \( f_i = (o_i, d(o_i, pp)) \)
  - \( o_i \): data object
  - \( d(o_i, pp) \): distance between \( o_i \) and the pivot in the parent node

**Update and Search**

**New object** \( o_N \) **is inserted as a leaf node**

**At each internal node** (starting with the root node):

1. find set \( E \) of entries that can store \( o_N \) without increasing covering radius (i.e., \( d(o_N, pp) < r_{ci} \))
2. if \( E = \emptyset \) traverse into subtree of element \( e \in E \) with minimum distance \( d(o_N, pp) \)
3. if \( E = \emptyset \) increase covering radius of element that requires minimum increase and traverse into respective subtree

**At leaf node**:

1. compute distance between \( o_N \) and parent pivot \( d(o_N, pp) \)
2. try to store new entry \( (o_N, d(o_N, pp)) \) in leaf node
3. if node is full (overflow), then split node
The M-Tree
Update and Search

Splitting a Node in the M-Tree

- If a node \( N \) overflows, it must be split
  
  **Node split**
  
  1. create new node \( N' \) at the same level
  2. select two new pivots (for \( N \) and \( N' \))
  3. redistribute the \( m+1 \) objects to \( N \) and \( N' \)
  4. substitute old pivot by the two new pivots
  5. if parent node overflows:
     a. if parent node is non-root: split parent node
     b. if parent node is root: create new root node (tree grows by one level)

**How to choose new pivots?**

- try to keep covering radii as small as possible to avoid overlaps
- criterion: \( p_i \) and \( p_j \) are used as new pivots if \( \max(r_{ci}, r_{cj}) \) is minimal

---

Range Query

**Definition (Range Query)**

Given a set of objects \( X \subseteq D \) from a domain \( D \) and a query object \( q \in D \) with a query radius \( r \). The **range query**, \( R(q, r) \) retrieves all objects in \( X \) within distance \( r \) from \( q \):

\[
R(q, r) = \{ o \in X \mid d(o, q) \leq r \}
\]

---

Search Algorithm

1. **Start at root node**
2. For each entry \((p, r^c, d(p, p^p), \text{ptr})\) in an internal node:
   - if \( |d(q, p^p) - d(p, p^p)| - r^c > r \) (criterion A), then the subtree of \( p \) can be safely ignored (pruned)
   - if \( |d(q, p^p) - d(p, p^p)| - r^c \leq r \), then compute \( d(q, p) \); if \( d(q, p) - r^c > r \) (criterion B), then prune subtree, otherwise traverse subtree pointed to by \( \text{ptr} \)
3. For each entry \((o, d(o, p^p))\) in a leaf node:
   - if \( |d(q, o^p) - d(o, p^p)| > r \), then ignore object \( o \)
   - otherwise compute \( d(q, o) \); if \( d(q, o) \leq r \), then \( o \) is reported as an answer
Search Algorithm

- Pruning criterion B: \( d(q, p) - r^c > r \)
  - the objects in the subtree of \( p \) are within radius \( r^c \) from \( p \)
  - the range query looks for objects within radius \( r \) from \( q \)
  - if \( d(q, p) > r^c + r \), then the spheres defined by \((p, r^c)\) and \((q, r)\) are too small and do not overlap, thus \( p \) is pruned
- \( d(q, p) \) is only computed if criterion A does not hold
- Pruning criterion A is applied instead: \(|d(q, p^p) - d(p, p^p)| - r^c > r\)
  - both \( d(q, p^p) \) and \( d(p, p^p) \) are known
  - \(|d(q, p^p) - d(p, p^p)| \leq d(q, p)\) follows from the triangle inequality
  - criterion A \( \Rightarrow \) criterion B \( \Rightarrow \) subtree of \( p \) can be pruned

Paolo Ciaccia, Marco Patella, and Pavel Zezula.
M-tree: An efficient access method for similarity search in metric spaces.

Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko.