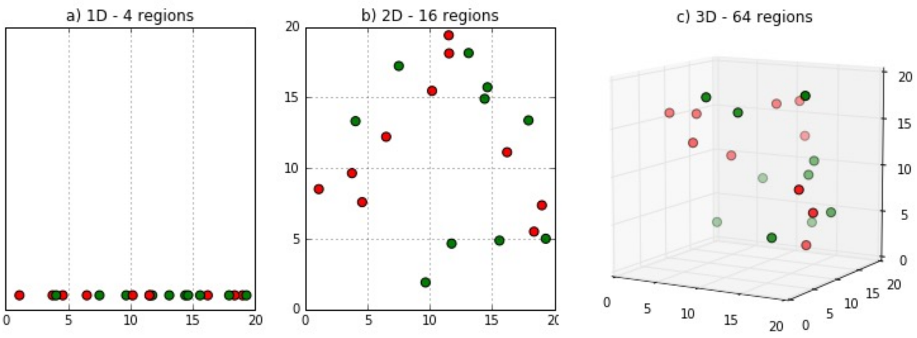
CSEG601 & CSE5601: Spatial Data Management & Application:

Distanced-Based Spatial Indexing Method: M-tree

Big Data Processing & Database Lab.
Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Tel: +82-2-705-8930
Email : jungsung@sogang.ac.kr

Curse of Dimensionality

- When the dimensionality increases, the volume of the space increases so fast that the available data become sparse
- Organizing and searching data often relies on detecting areas where objects form groups with similar properties
- In high dimensional data, all objects appear to be sparse and dissimilar in many ways, which prevents common data organization strategies from being efficient



The M-tree

- Inherently dynamic structure
- Disk-oriented (fixed-size nodes)
- Built in a bottom-up fashion
  - Inspired by R-trees and B-trees
- All data in *leaf nodes*
- *Internal nodes*: pointers to subtrees and additional information
- Objects are stored in leaves.

Metric Indexing

Feature vectors are indexed according to distances between each other.

As a dissimilarity measure, a distance function $d(O_i,O_j)$ is specified such that the metric axioms are satisfied:

$d(O_i,O_i) = 0$	reflexivity
$d(O_i,O_j) > 0$	positivity
$d(O_i,O_j) = d(O_j,O_i)$	symmetry
$d(O_i,O_k) + d(O_k,O_j) \geq d(O_i,O_j)$	triangular inequality

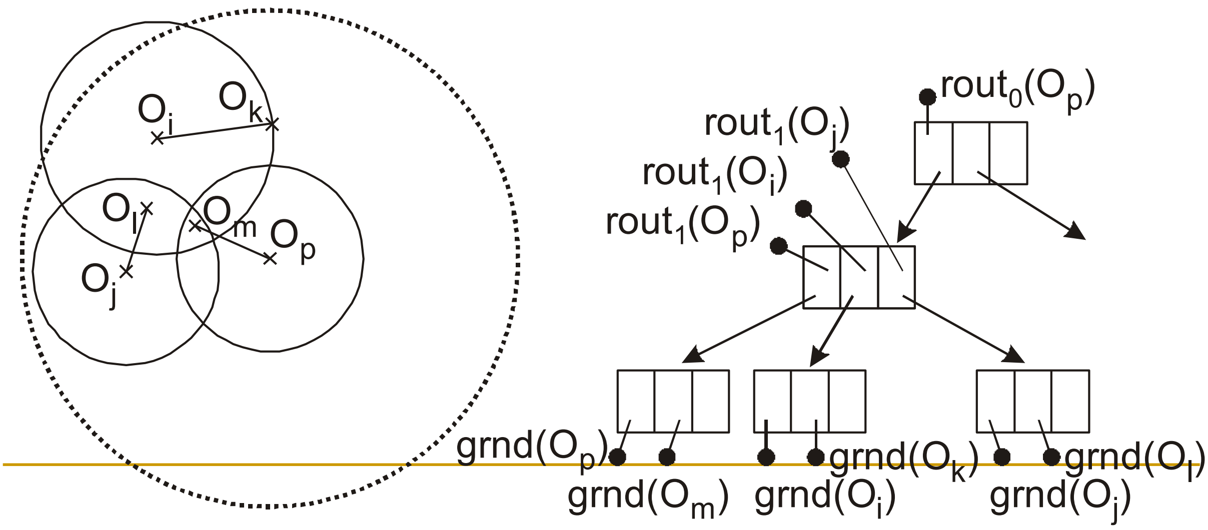
M-tree at a glance

- Indexing objects of a general metric space (not only vector spaces)
- Doesn't directly use dimensions, just the distances between objects
- The correct M-tree hierarchy is guaranteed due to the triangular inequality axiom of *d*. The hierarchy consists of nested metric regions.
- Better resists to "the Curse of Dimensionality"
- The hierarchy of nodes allows to natively implement the similarity queries

Structure of the M-tree

The M-tree nodes contain items of two types:

- ground objects in leafs, representing the data objects
- routing objects in inner nodes, representing the metric regions



M-tree: Internal Node

- Internal node consists of an entry for each subtree
- Each entry consists of:
  - Pivot: *p*
  - Covering *radius* of the sub-tree: r^c
  - Distance from *p* to *parent* pivot p^p: $d(p,p^p)$
  - Pointer to sub-tree: *ptr*

$$\boxed{\boxed{\langle p_1, r_1^c, d(p_1,p^P), ptr_1\rangle}\boxed{\langle p_2, r_2^c, d(p_2,p^P), ptr_2\rangle}\cdots\boxed{\langle p_m, r_m^c, d(p_m,p^P), ptr_m\rangle}}$$

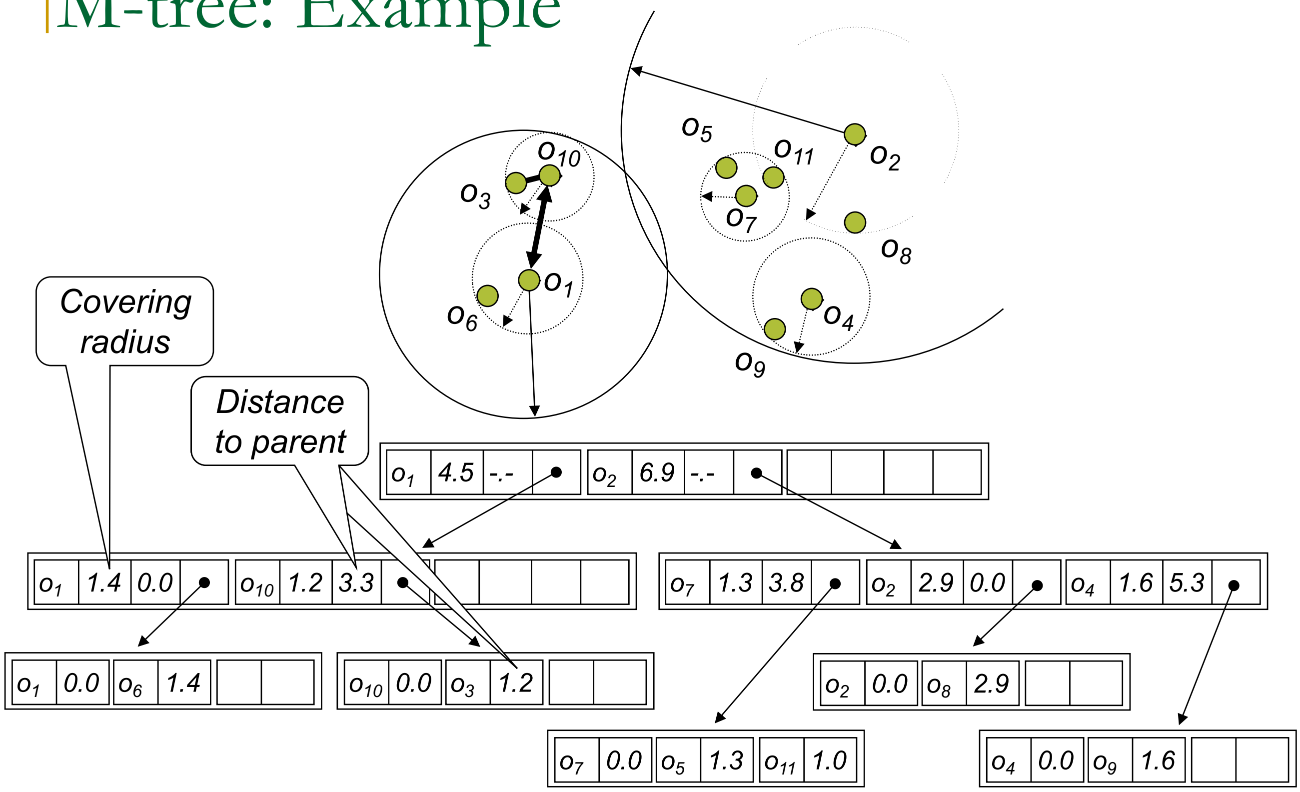
  - All objects in subtree *ptr* are within the distance r^c from *p.*

M-tree: Leaf Node

- leaf node contains **data entries**
- each entry consists of pairs:
  - object (its identifier): *o*
  - distance between *o* and its parent pivot: $d(o,o^p)$

$$\boxed{\boxed{\langle o_1, d(o_1,o^P)\rangle}\boxed{\langle o_2, d(o_2,o^P)\rangle}\cdots\boxed{\langle o_m, d(o_m,o^P)\rangle}}$$

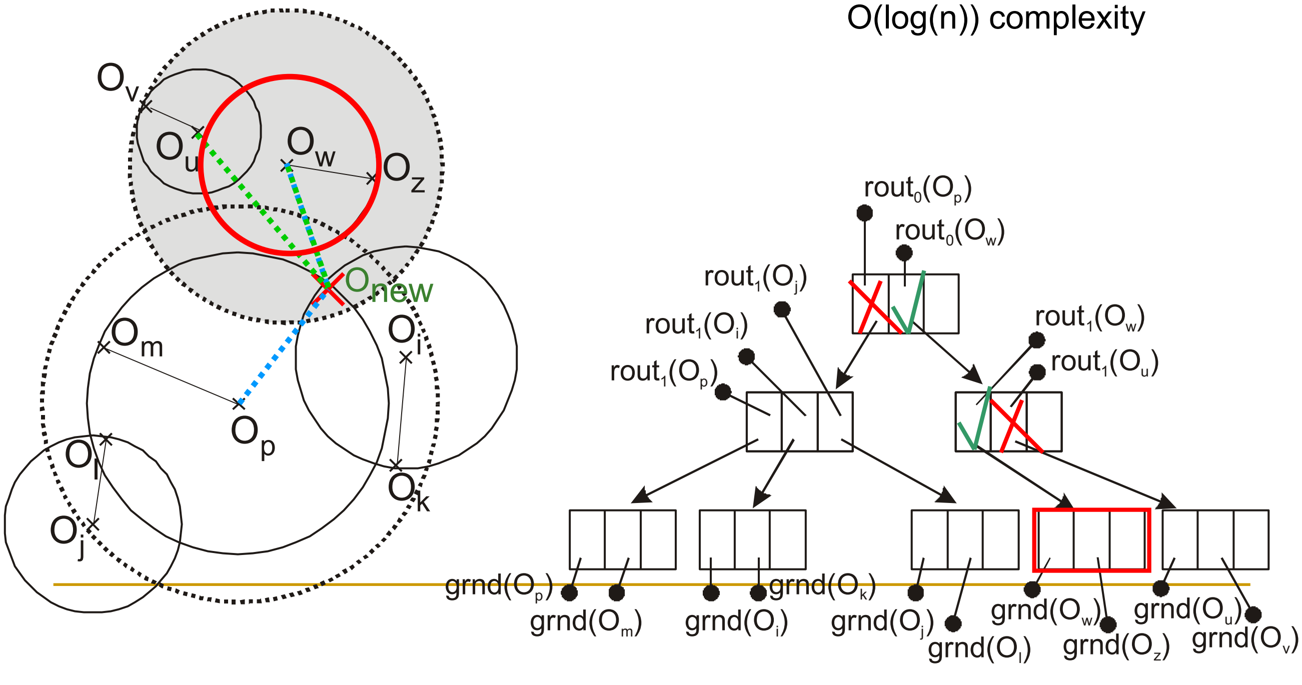
M-tree: Example



M-tree, single-way insertion

During an object insertion, only single sub-tree is further processed on a current level of M-tree. A heuristic criterion: a node is chosen, that spatially contains the inserted object and/or whose routing object is the nearest

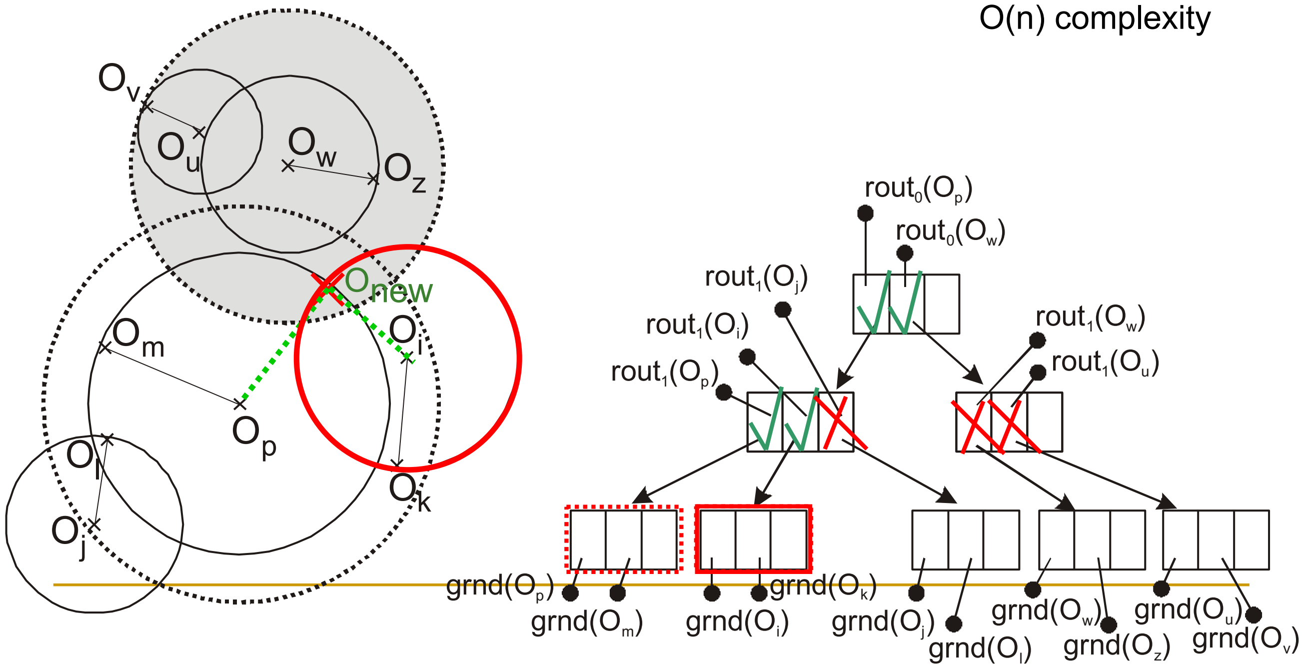
O(log(n)) complexity



M-tree, multi-way insertion

During an object insertion, a point query for the inserted object is executed and routing objects of all relevant non-full leafs are checked. The nearest one is chosen. If such leaf doesn't exist, the single-way insertion is performed.

O(n) complexity



M-tree: Insert

- Insert a new object o_N:
- recursively descend the tree to locate the *most suitable leaf* for o_N
- in each step enter the subtree with pivot p for which:
  - no enlargement of radius r^c needed, i.e., $d(o_N,p) \le r^c$
    - in case of ties, choose one with p nearest to o_N
  - minimize the enlargement of r^c

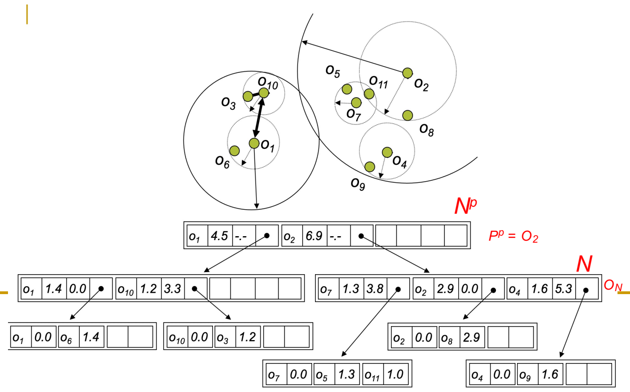
M-tree: Insert (cont.)

- when reaching leaf node N then:
  - if N is not full then store o_N in N
  - else **Split**(N,o_N).

M-tree: Split

Split(N,o_N):

- Let S be the set containing all entries of N and o_N
- Select pivots p_1 and p_2 from S
- Partition S to S_1 and S_2 according to p_1 and p_2
- Store S_1 in N and S_2 in a new allocated node N'
- If N is root
  - Allocate a new root and store entries for p_1, p_2 there
- else (*let N^p and p^p be the parent node and parent pivot of N*)
  - Replace entry p^p with p_1
  - If N^p is full, then **Split**(N^p,p_2)
  - else store p_2 in node N^p

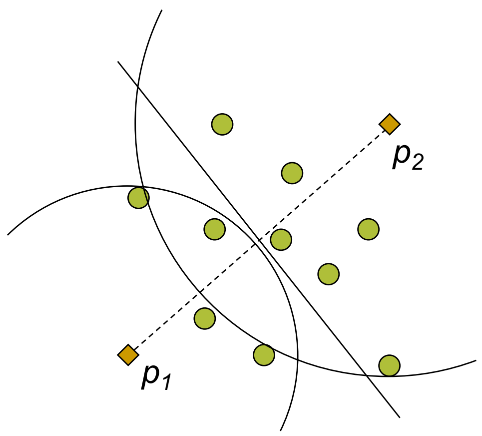


M-tree: Pivot Selection

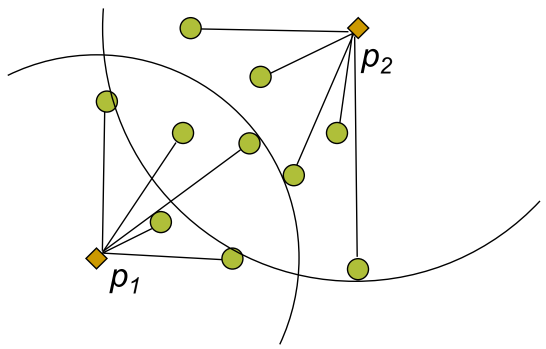
- Several pivots selection policies
  - ***RANDOM*** – select pivots p_1, p_2 randomly
  - ***m_RAD*** – select p_1, p_2 with minimum $(r_1^c + r_2^c)$
  - ***mM_RAD*** – select p_1, p_2 with minimum $max(r_1^c, r_2^c)$
  - ***M_LB_DIST*** – let $p_1 = p^p$ and $p_2 = o_i \mid \max_i \{d(o_i, p^p)\}$
    - Uses the pre-computed distances only
- Two versions (for most of the policies):
  - **Confirmed** – reuse the original pivot p^p and select only one
  - **Unconfirmed** – select two pivots (notation: ***RANDOM_2***)
- In the following, the ***mM_RAD_2*** policy is used.

M-tree: Split Policy

- Partition S to S_1 and S_2 according to p_1 and p_2
- Unbalanced
  - Generalized hyperplane



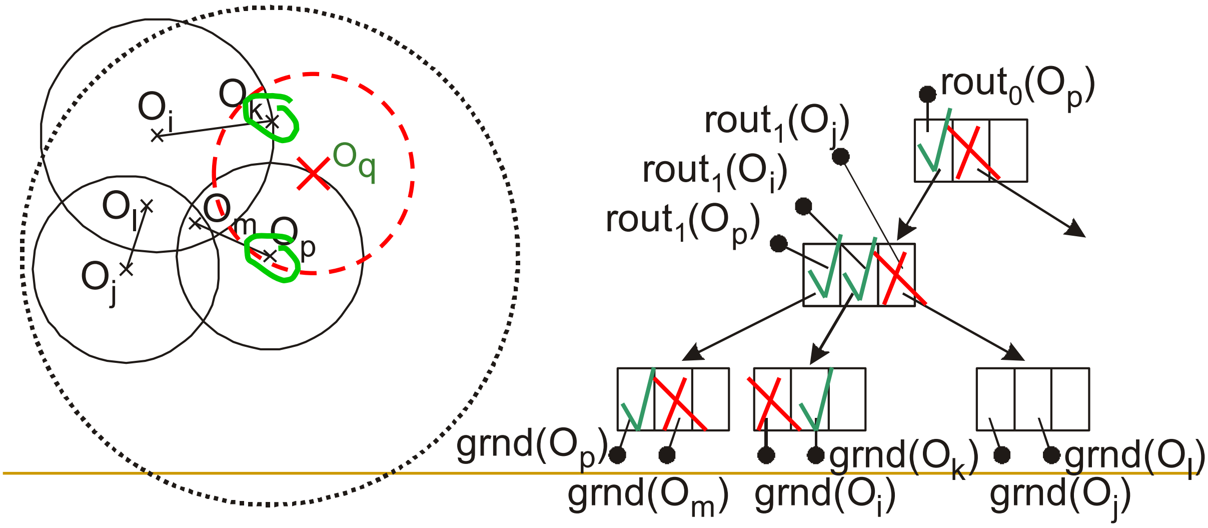
- Balanced
  - Larger covering radii
  - Worse than unbalanced one



Similarity queries in the M-tree

A range query is specified by a query object O_q and a query radius r_q.
A k-NN query is based on a modified range query (using dynamic radius) and a priority queue.

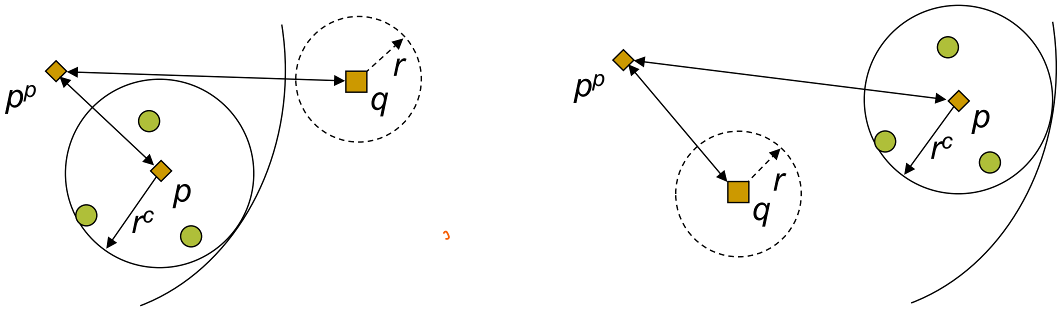
During the range query evaluation, the M-tree is LIFO-passed and only the relevant (i.e. intersecting) metric regions (their nodes resp.) are further processed.



M-tree: Range Search

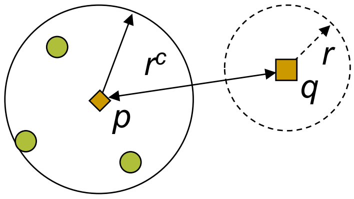
Given $R(q,r)$:

- Traverse the tree in a depth-first manner
- In an internal node, for each entry $\langle p, r^c, d(p,p^p), ptr \rangle$
  - Prune the subtree if $|d(q,p^p) - d(p,p^p)| - r^c > r$
  - Application of the pivot-pivot constraint



M-tree: Range Search (cont.)

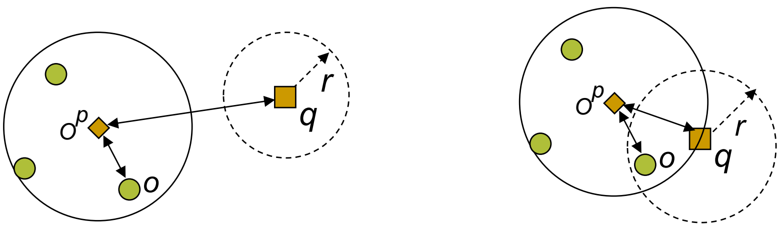
- If not discarded, compute $d(q,p)$ and
  - Prune the subtree if $d(q,p) - r^c > r$
  - Application of the range-pivot constraint



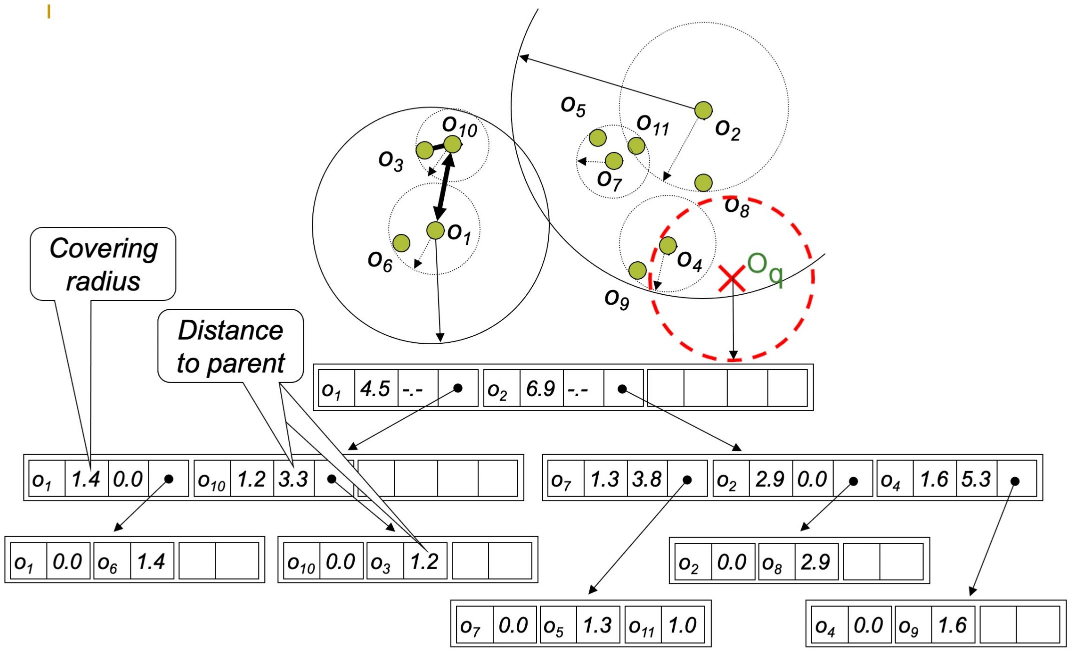
- All non-pruned entries are searched recursively.

M-tree: Range Search in Leaf Nodes

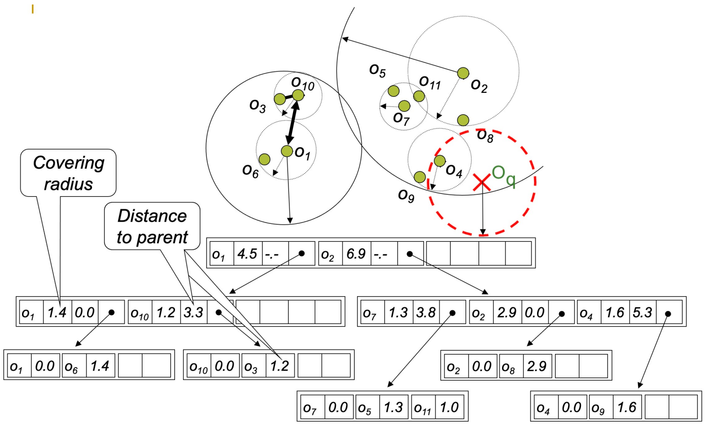
- In a leaf node, for each entry $\langle o, d(o,o^p)\rangle$
  - Ignore entry if $|d(q,o^p) - d(o,o^p)| > r$
  - else compute $d(q,o)$ and check $d(q,o) \le r$
  - Application of the object-pivot constraint



- Example: We are going to perform a range search $R(O_q, 3.0)$ (i.e., a *red dotted circle centered at O_q with radius 3.0*) over the following M-tree. Given $d(O_q,O_1) = 11.5$, $d(O_q,O_2) = 9.7$, $d(O_q,O_4) = 2.8$, and $d(O_q,O_9) = 3.1$, show how the range search are pruned over the nodes of the following M-Tree.



- Example: We are going to perform a range search $R(O_q, 3.0)$ (i.e., a *red dotted circle centered at O_q with radius 3.0*) over the following M-tree. Given $d(O_q,O_1) = 11.5$, $d(O_q,O_2) = 9.7$, $d(O_q,O_4) = 2.8$, and $d(O_q,O_9) = 3.1$, show how the range search are pruned over the nodes of the following M-Tree.



Root Node of the M-tree: Application of the range-pivot constraint

O_1 entry:Prune O_1 entry since $d(O_q,O_1) - 4.5 = 11.5 - 4.5 = 7.0 > 3.0$

O_2 entry: Don't Prune O_2 entry since $d(O_q,O_2) - 6.9 = 9.7 - 6.9 = 2.8 < 3.0$

O_2 Node: Application of the pivot-pivot constraint

O_7 entry:Prune O_7 entry since $|d(O_q,O_2)-d(O_7,O_2)|-1.3=|9.7-3.8| -1.3=5.9-1.3=4.6> 3.0$

O_2 entry:Prune O_2 entry since $|d(O_q,O_2)-d(O_2,O_2)|-2.9=|9.7-0.0| - 2.9=9.7-2.9=6.8> 3.0$

O_4 entry:Search O_4 entry since $|d(O_q,O_2)-d(O_4,O_2)|-1.6=|9.7-5.3|-1.6=4.4 -1.6=2.8 < 3.0$

$d(O_q,O_4) - 1.6 = 2.8 - 1.6 = 1.2 < 3.0$

O_4 Node: Application of the object-pivot constraint

O_4 entry: Cannot ignore O_4 entry since $|d(O_q,O_4)-d(O_4,O_4)|=|2.8-0.0|=2.8 < 3.0$

$d(O_q,O_4) = 2.8 < 3.0$ → O_4 is in the range

O_9 entry: Cannot ignore O_9 entry since $|d(O_q,O_4)-d(O_9,O_4)| =|2.8-1.6| =1.2 < 3.0$

$d(O_q,O_9) = 3.1 > 3.0$ → O_9 is not in the range.

M-tree: *k*-*NN* Search

Given k-$NN(q)$:

- Based on a *priority queue* and the pruning mechanisms applied in the range search.
- Priority queue:
  - Stores pointers to sub-trees where qualifying objects can be found.
  - Considering an entry $E=\langle p,r^c,d(p,p^p),ptr\rangle$, the pair $\langle ptr,d_{min}(E)\rangle$ is stored.
  - $d_{min}(E)=max\ \{\ d(p,q) - r^c,\ 0\ \}$
- Range pruning: instead of fixed radius r, use the distance to the k-th current nearest neighbor.