Organizing Similarity Spaces using Metric Hulls

Miriama Jánošová[1], David Procházka[1], Vlastislav Dohnal[1]

[1]Faculty of Informatics, Masaryk University, Brno, Czech Republic

Introduction

A novel concept of a metric hull has recently been introduced to encompass a set of objects. The **hull representation** of a group C [1] is defined as a set of boundary objects selected from C, such that every other object from C is covered by them. Such boundary objects are referred to as *hull objects*. In Fig. 3, the hull objects are highlighted.
Following a metric-hull computation method that generates a hierarchy of metric hulls, we propose a metric index structure for unstructured and complex data, a Metric Hull Tree (MH-tree). We provide a bulk-loading procedure for MH-tree construction. With respect to the design of the tree, we provide an implementation of an approximate kNN search operation. Finally, we utilize the Profimedia dataset to evaluate MH-tree's various building and ranking strategies and compare the results to M-tree.

MH-Tree Structure

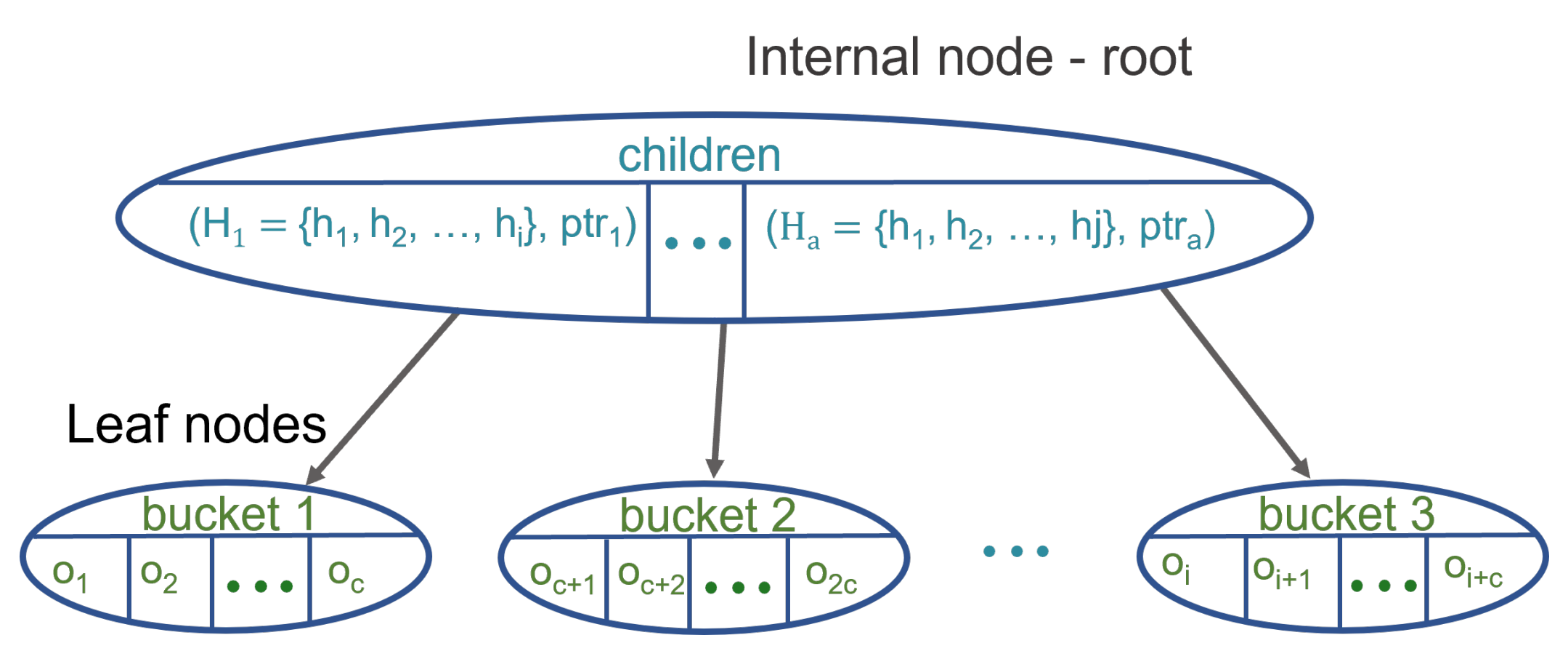


Fig. 1 MH-Tree Structure

Metric Hull Tree (MH-Tree) [2] is a hierarchical n-ary tree index structure. It is composed of two types of nodes, specifically *leaf* and *internal* nodes.

- *Leaf node* encapsulates a bucket – a storage of c objects.
- *Internal node* contains a list of a pointers (ptr_i) to children nodes and their hull representations (H_i)

Bulk-Loading

We propose creating the MH tree by a bulk loading procedure in a bottom-up manner. Firstly, we group all the database objects into leaf nodes:

- Select the furthest object o_f from the database - new cluster's nucleus
- Repeat until the size of cluster is equal to c:
  - Execute 1-NN query for each cluster object to identify the set of closest yet-not-clustered objects – $candidates$
  - Select o_c from $candidates$, such that the sum of distances between o_c and cluster objects is the smallest and add it to the cluster
- Compute the hull representation of a node

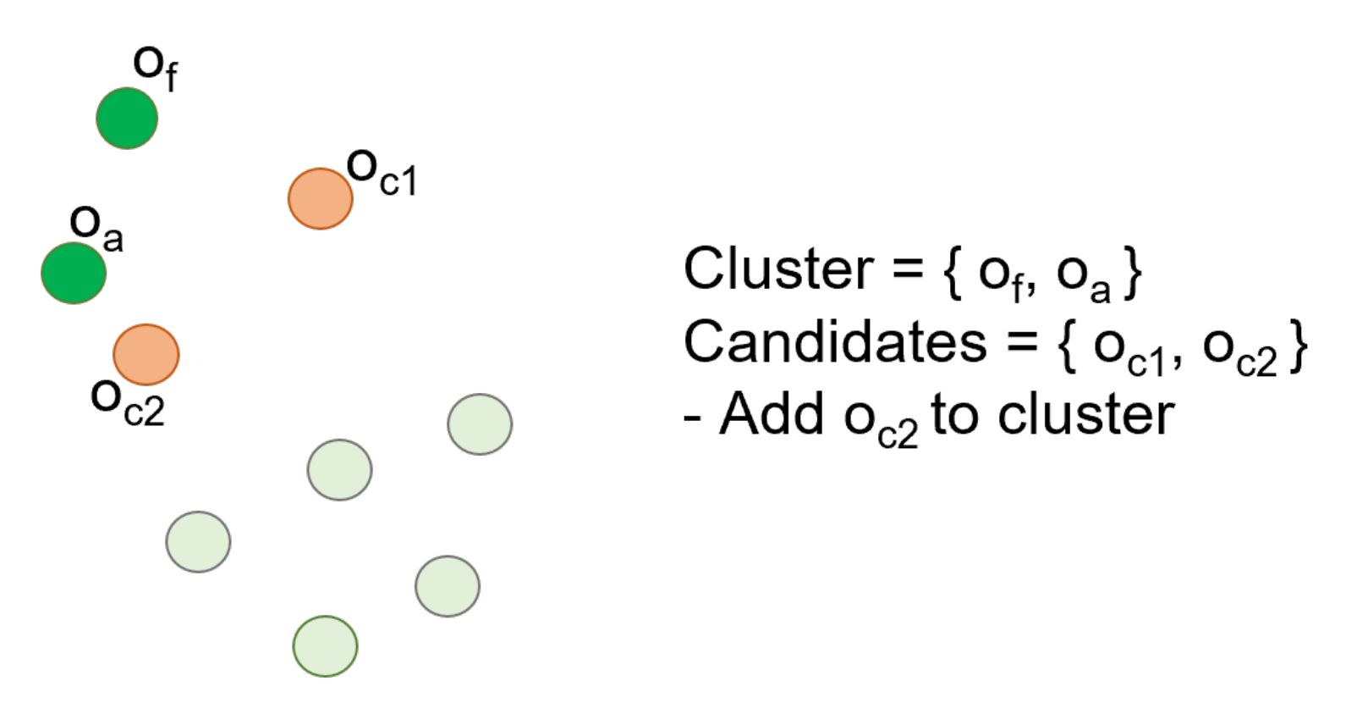

o_f
o_{c1}
o_a
o_{c2}
Cluster = { o_f, o_a }
Candidates = { o_{c1}, o_{c2} }
- Add o_{c2} to cluster

Fig. 2 Select the next bucket object

Next, we apply the merging procedure to collapse all the leaves into the first layer of internal nodes, such that each node has a children. This merging procedure is repeated until only a single node is obtained – the root of the tree.

- Merging procedure:
  - Select the furthest node n_f and execute a-NN query to obtain a nearest nodes to n_f to form an internal node
  - Compute the hull representations of a new internal node from the hull representations of children
  - The proximity of nodes is measured by the distance between their hulls:

$$d(\mathcal{H}_1, \mathcal{H}_2) = \min_{\forall h_1 \in \mathcal{H}_1, \forall h_2 \in \mathcal{H}_2} d(h_1, h_2).$$

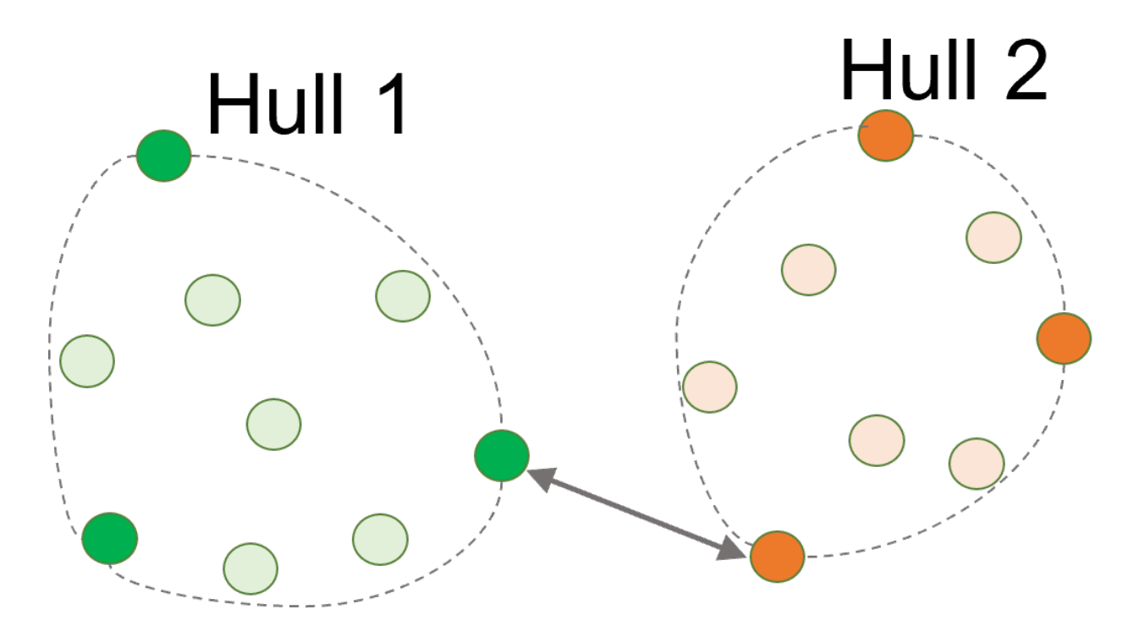


Fig. 3 Distance between hulls

Searching in MH-Tree

Searching in the MH-Tree for nearest neighbors of a *query* employs a priority queue PQ of nodes to be processed. However, the algorithm here provides only approximate results since we assume a *limit* on the number of visited objects during search and also, the manner in which the MH-Tree is built introduces some impressions. During the traversal, nodes are visited based on the likelihood of containing relevant answers. The probability is determined by RANK, which is defined exclusively based on the distances to the hull objects.

- The tree traversal starts in the *root* node, and nodes are visited with respect to PQ
- When an internal node is visited, add its child nodes to PQ
- When a leaf node is visited, take objects from the bucket and update the *answer*
- Terminate the traversal after visiting the limited amount of nodes

Ranking Nodes During Search

For leaf nodes, we define the ordering of nodes as

$$\text{RANK}_{LEAF}(q, \mathcal{H}) = \min_{\forall h \in \mathcal{H}} d(q, h).$$

For internal nodes, the $\text{RANK}_{INT}(q, \mathcal{H}, k)$ function is defined as follows:

$$\begin{cases} -\max_{\forall h \in \mathcal{H}} d(q, h) & \text{q } \textit{is covered} \text{ by } \mathcal{H}, k = 1 \\ \max_{\forall h \in \mathcal{H}} d(q, h) & \text{q } \textit{is not covered} \text{ by } \mathcal{H}, k = 1 \\ -\min_{\forall h \in \mathcal{H}} d(q, h) & \text{q } \textit{is covered} \text{ by } \mathcal{H}, k > 1 \\ \min_{\forall h \in \mathcal{H}} d(q, h) & \text{q } \textit{is not covered} \text{ by } \mathcal{H}, k > 1 \end{cases}$$

Experimental Evaluation

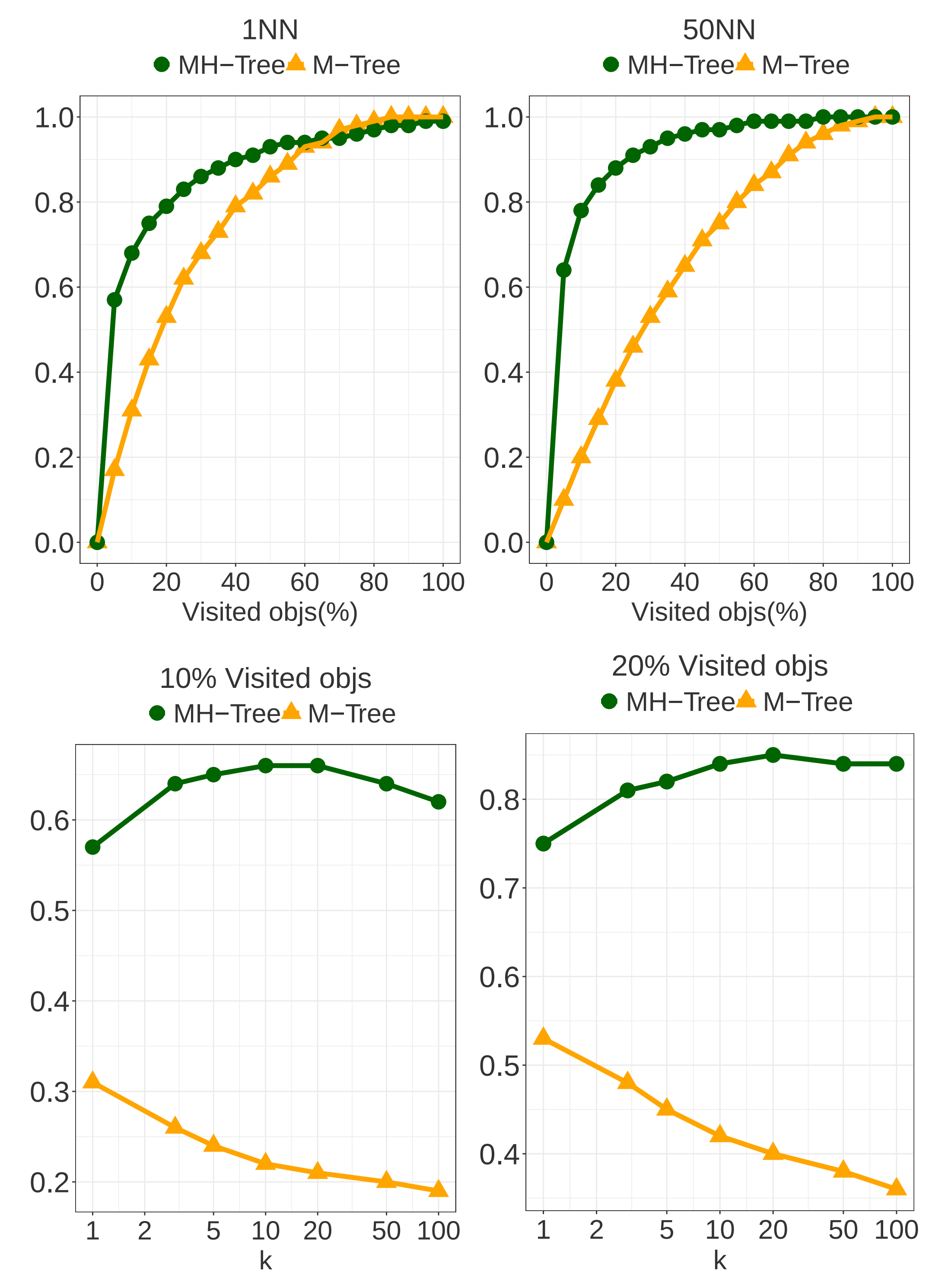
We performed the experiments over the Profimedia dataset [3]. It is a collection of 4096-D vectors extracted from Photo-stock images by Convolutional Neural networks. We used two different sizes of data, specifically 10k and 100k images. To measure the similarity between the data, we employ Euclidean distance.

- Methodology:
  - Employ approximate k-NN search with termination after 5% up to 100% visited objects
  - Measure the trade-off between accuracy and efficiency as

$$recall = \frac{| S \cap S_a |}{| S |},$$

where set S is a result of precise k-NN search and S_a a result of approximate search

  - Compare the results with M-Tree optimized by Slim-down insertion
- Measured recall :



Conclusion

We introduced a novel index structure MH-Tree build upon principles of the concept of metric hulls. The fat factor of MH-tree suggests the metric hull representation of the data is rather compact.
We compared the best-performing setup of MH-tree with the M-tree optimized by the Slim-down algorithm. The results showcase MH-tree outperforms M-tree significantly – fewer nodes are visited for the same recall. Specifically, the performance of MH-tree was higher by 30-40% on average depending on the number of extracted neighbors.

References

[1] Matej Antol, Miriama Janosova, and Vlastislav Dohnal. Metric hull as similarity-aware operator for representing unstructured data. *Pattern Recognition Letters*, pages 1–8, 2021. doi: 10.1016/j.patrec.2021.05.011.

[2] David Procházka. Indexing structure based on metric hulls. Bachelor thesis, Masaryk University, FI, 2021. URL `https://is.muni.cz/th/jk21s/`.

[3] David Novak, Michal Batko, and Pavel Zezula. Large-scale image retrieval using neural net descriptors. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1039–1040. ACM, 2015. ISBN 978-1-4503-3621-5.

Supported by the ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No.CZ.02.1.01/0.0/0.0/16_019/0000822)