MUNI
FI

On Dynamicity of Metric Hull Trees

Josef Podaný

Faculty of Informatics, Masaryk University

April 11, 2022

Outline

Motivation

- 95% of big data is unstructured [3] – images, documents, video, audio, webpages…
- Challenge: **manage complex data efficiently** and evaluate similarity queries faster than by sequential scans
- Traditional data retrieval methods are lacking in this direction
- $\Longrightarrow$ Utilize **metric spaces** to build **efficient indexing structures**

Metric Space

- Metric space $\mathcal{M} = (\mathcal{D}, d)$
- Domain of valid objects $\mathcal{D}$
- Distance function $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}_0^+$, satisfying the following metric postulates for all objects $x, y, z \in \mathcal{D}$:

$$d(x, y) \geq 0, \tag{1}$$

$$x = y \iff d(x, y) = 0, \tag{2}$$

$$d(x, y) = d(y, x), \tag{3}$$

$$d(x, z) \leq d(x, y) + d(y, z). \tag{4}$$

Metric Hull Representation

- A **Hull Representation** [1] of a group C is defined as $\mathcal{H}(C) = \{p_i \mid p_i \in C\}$ and any other object $o \in C$ is **covered** by hull. Each p_i corresponds to a boundary object of C referred to as *hull object*.

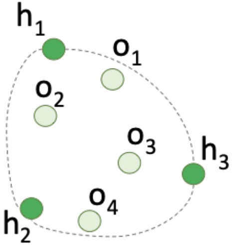
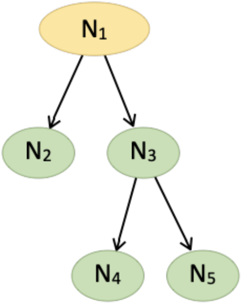


Figure: The hull representation $\mathcal{H} = \{h_1, h_2, h_3\}$ covering objects $h_1, h_2, h_3, o_1, o_2, o_3, o_4$

Metric Hull Tree

- Hierarchical *n*-ary tree index structure [4]
- Consists of
  - *Internal nodes* – contain a list of pointers to children nodes and their hull representations
  - *Leaf nodes* – contain a bucket of stored objects and its hull representation
- Parametrized by bucket capacity *c*, tree arity *a*
- Supports:
  - Bulk-loading from a set of objects
  - Exact-match query
  - Approximate kNN search
  - Inserting new objects



Methodology of Experimental Evaluation

- Measuring the recall of the $50NN$ queries as $R = \frac{|S \cap S_a|}{|S|}$
- 10.000 DeCAF descriptors [5]
- Benchmarking trees with arity 6, bucket capacity 8

Recall Degradation After Inserting

- *"If the structure becomes highly unbalanced, it should be rebuilt from scratch."* [4]

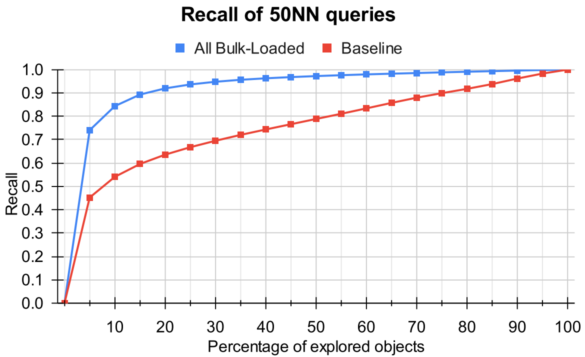


Figure: $a = 6$, $c = 8$. *All Bulk-Loaded* bulk-loaded with 10.000 objects, *Baseline* bulk-loaded with 5.000 and 5.000 objects inserted.

Keeping the Tree Balanced

- Ensure the tree has depth of $\mathcal{O}(\log n)$, where n is the number of nodes
- Idea: **follow the insertion technique used in B+ trees and M-trees**, growing the trees at the root

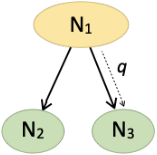


Figure: (1): $a = 2$. Store q in the bucket of N_3.

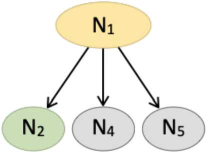


Figure: (2): $a = 2$. Split N_3 into N_4, N_5.

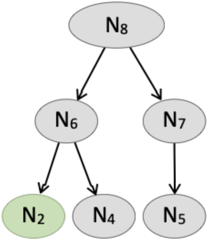


Figure: (3): $a = 2$. Repair N_1 by splitting.

Splitting Strategies

- Set of objects $\mathcal{A} = \{o_1, o_2, \ldots, o_n\}$
- Distance function $d : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}_0^+$
- Function *split*$(\mathcal{A}, d)$ splits the set into two halves $\mathcal{B}, \mathcal{C}$
- When splitting buckets:
  - $\mathcal{A}$ is the set of the stored objects, d is the Euclidean distance, $n = 2c + 1$
- When splitting nodes:
  - $\mathcal{A}$ is the set of children nodes, d is the *node-to-node* distance function $d_n(n_i, n_j) = d_h(n_i.hull, n_j.hull)$, $n = a + 1$

Splitting Strategies 2

- Selection of an outlier o_f in $\mathcal{A}$:

$$o_f = \underset{p \in \mathcal{A}}{\operatorname{argmax}} \sum_{q \in \mathcal{A}} d(p,q) \tag{5}$$

- Selection of the nearest neighbor $o_{nn} \in \mathcal{A}$ with regards to $\mathcal{B}$:

$$o_{nn} = \underset{p \in \mathcal{A} \setminus \mathcal{B}}{\operatorname{argmin}} \sum_{q \in \mathcal{B}} d(p,q) \tag{6}$$

Greedy Splitting

- Idea: carve out an outlier object (5) along with it's $\lceil \frac{n-1}{2} \rceil$ closest neighbors (6)

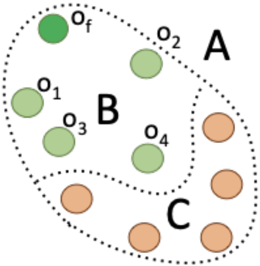


Figure: Partitioning of $\mathcal{A}$ after greedy splitting.

Fair Splitting

- Idea: Introduce fairness into the splitting by splitting the set in turns
- Select an outlier (5), its most distant object, and redistribute the nearest neighbors (6) iteratively

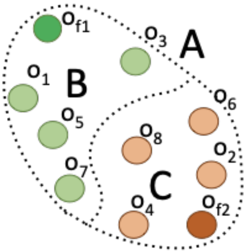


Figure: Partitioning of $\mathcal{A}$ after fair splitting.

Greedy Splitting

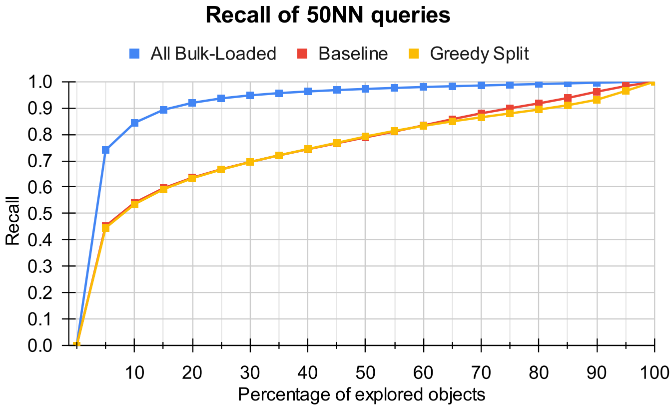


Figure: $a = 6$, $c = 8$. *All Bulk-Loaded* bulk-loaded with 10.000 objects, *Baseline* and *Greedy Split* bulk-loaded with 5.000 objects and 5.000 objects inserted.

Fair Splitting

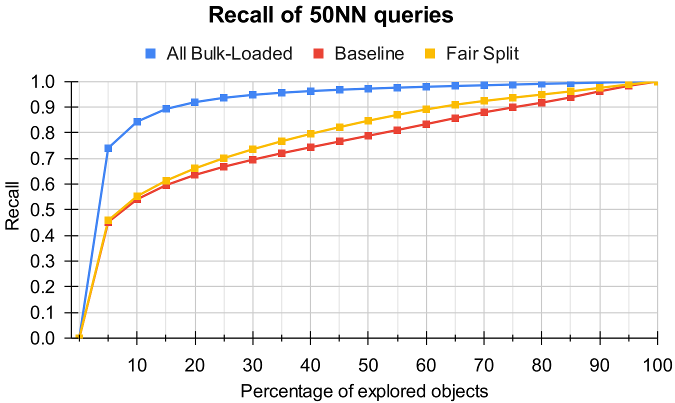


Figure: $a = 6$, $c = 8$. *All Bulk-Loaded* bulk-loaded with 10.000 objects, *Baseline* and *Fair Split* bulk-loaded with 5.000 objects and 5.000 objects inserted.

Candidate Hull Objects – Motivation

- Concept aiming to provide additional information about the objects stored under a hull
- In some cases, some underlying objects may not be covered by a hull when inserting new objects due to the recomputation of a hull

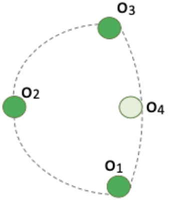


Figure: Hull $\mathcal{H} = \{o_1, o_2, o_3\}$ covering o_4

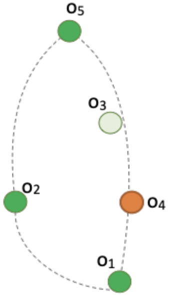


Figure: Hull $\mathcal{H} = \{o_1, o_2, o_5\}$ not covering o_4

Candidate Hull Objects

- Candidates are additional objects stored alongside hulls in internal nodes
- Candidates are added during the insertion of an object along the path of insertion
- Utilized during computation of new hulls during node splitting
- Each internal node is restricted to *m* candidates

Candidate Hull Objects

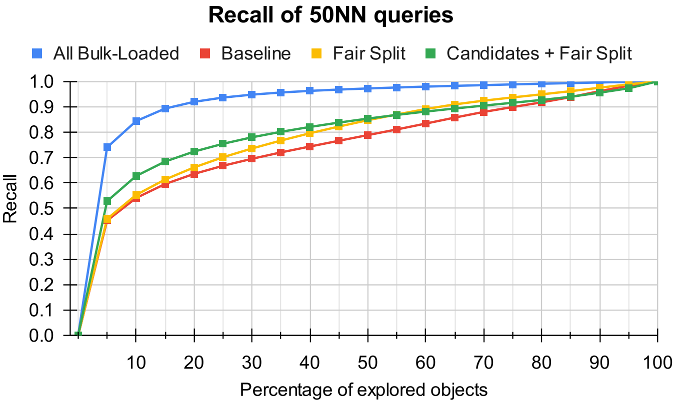


Figure: $a = 6$, $c = 8$, $m = 5$. *All Bulk-Loaded* bulk-loaded with 10.000 objects, *Baseline*, *Fair Split*, and *Candidates* bulk-loaded with 5.000 objects and 5.000 objects inserted.

Conclusion

- Introduced repairing procedure **making Metric Hull Trees balanced**
- Introduced fair and greedy splitting strategies, candidate hull objects
- Improved 50*NN* query recall by up to **5 percentage points** by splitting buckets and nodes fairly
- Improved of 50*NN* query recall by up to **8.7 percentage points** by utilizing candidate hull objects combined with fair splitting

Bibliography I

[1] Matej Antol, Miriama Jánošová, and Vlastislav Dohnal. "Metric hull as similarity-aware operator for representing unstructured data". In: *Pattern Recognition Letters* 149 (2021), pp. 91–98. ISSN: 0167-8655.

[2] Adam BODNÁR. *Operace pro metrické obaly a jejich vizualizace [online]*. Bachelor's thesis. 2021 [cit. 2022-02-02]. URL: `https://is.muni.cz/th/lsc45/`.

Bibliography II

[3] Amir Gandomi and Murtaza Haider. "Beyond the hype: Big data concepts, methods, and analytics". In: *International Journal of Information Management* 35.2 (2015), pp. 137–144. ISSN: 0268-4012. DOI: https://doi.org/10.1016/j.ijinfomgt.2014.10.007. URL: https://www.sciencedirect.com/science/article/pii/S0268401214001066.

[4] Miriama Jánošová, David Procházka, and Vlastislav Dohnal. "Organizing Similarity Spaces Using Metric Hulls". In: *International Conference on Similarity Search and Applications*. Springer. 2021, pp. 3–16.

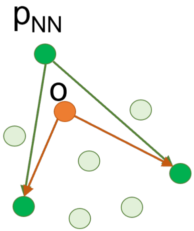
Bibliography III

[5] 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*. SIGIR ’15. Santiago, Chile: Association for Computing Machinery, 2015, pp. 1039–1040. ISBN: 9781450336215. DOI: 10.1145/2766462.2767868. URL: https://doi.org/10.1145/2766462.2767868.

Coverage Property

- Let $\mathcal{H}$ be a hull representation $\mathcal{H} = \{p_1, \ldots, p_h\}$ and an object $o \in$. Assume p_{NN} to be the nearest hull object of $\mathcal{H}$ to o, i.e., $NN = \text{argmin}_{i=1..h}(d(o, p_i))$. We say the object o is **covered** by $\mathcal{H}$ if and only if

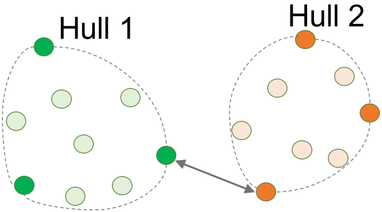
$$\sum_{i=1..h, i \neq NN} d(p_i, o) \leq \sum_{i=1..h} d(p_i, p_{NN}).$$



Distances Among Hulls

- Proximity of two hull representations $\mathcal{H}_1$ and $\mathcal{H}_2$:

$$d_h(\mathcal{H}_1, \mathcal{H}_2) = \min_{\forall h_i \in \mathcal{H}_1, \forall h_j \in \mathcal{H}_2} d(h_i, h_j).$$



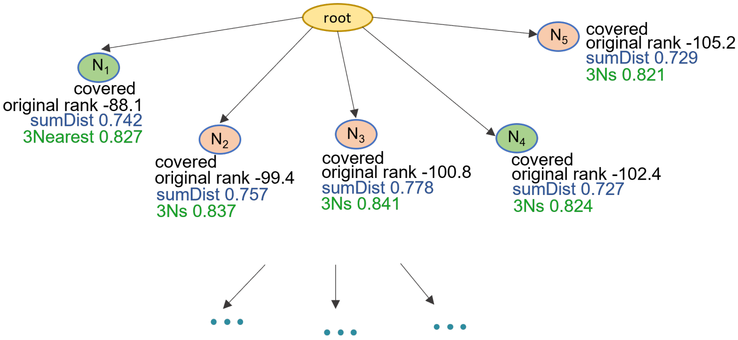
Building MH-Tree by Bulk-Loading

- Builds a balanced MH-Tree statically from a set of objects
- Better average recall than M-Tree can be achieved [4]

1. Group objects into leaf nodes, each containing at least *c* objects
2. Merge *a* closest leaves, creating a level of *internal leaves*
3. Repeat merging of *a* closest nodes until one node is obtained – a *root node*

Ranking Functions

- Formally, *rank* : $\mathcal{X} \times \mathcal{H} \times \mathbb{N} \to \mathbb{R}$
- Defines relevance of an object to a given hull
- Different ranking functions defined in [4, 2]: $rank_{original}$, $rank_{SumDist}$, $rank_{3Nearest}$, $rank_{MaxDist}$, $rank_{MaxDistInv}$, $rank_{Furthest}$



MUNI
FACULTY
OF INFORMATICS