# COL-Trees: Efficient Hierarchical Object Search in Road Networks (Extended Abstract)

**Tenindra Abeywickrama**[1], **Muhammad Aamir Cheema**[2], **Sabine Storandt**[3]
[1]RIKEN Center for Computational Science
[2]Faculty of Information Technology, Monash University
[3]Department of Computer and Information Science, University of Konstanz
tenindra.abeywickrama@riken.jp, aamir.cheema@monash.edu, sabine.storandt@uni-konstanz.de

**Abstract**

COL-Trees are a state-of-the-art data structure for hierarchical object search in road network graphs. While initially developed with multi-agent scenarios in mind, we show COL-Trees are highly suitable for single-agent queries as well, in particular, supporting upper-bound based $k$ Farthest Neighbor search for the first time. Along with reducing pre-processing costs, we strengthen the case that COL-Trees are a highly versatile data structure with potential application to other problems by the combinatorial search community. To support such exploration, we also present a highly-optimized COL-Tree implementation as open-source for the first time.

## Introduction and Background

Object search is a critical component in many real-world heuristic search and planning applications, such as finding relevant points of interest (POIs) in location-based services. Given a road network graph $G{=}(V, E)$, POIs are represented as a set of object vertices $P \subseteq V$. Search queries then retrieve POIs by network distances in $G$. This is advantageous compared to Euclidean distance as network distance can be more accurate and versatile by representing various metrics, such as travel-time. However, network distance is far more expensive to compute, necessitating effective heuristic search techniques to avoid retrieving false positive POIs.

A $k$ Nearest Neighbor ($k$NN) query returns the $k$ closest POIs from an agent's location. Another common query, Aggregate $k$ Nearest Neighbors (A$k$NN), involves *multiple agents* with POIs retrieved by distances from each agent aggregated by some function. For example, an A$k$NN query may minimize the *sum* of distances to restaurants for a group of users who want to meet. However, A$k$NN search in road networks largely borrowed heuristics from $k$NN search (Abeywickrama, Cheema, and Storandt 2020), which are not suitable when multiple agents are considered. For example, one technique utilized a recurrence rule that stated the $k$-th nearest POI must be adjacent to the $k-1$ nearest POIs (Zhu et al. 2010). While efficient for $k$NN search, the rule is not true when multiple agents are involved (actual result POIs may not be near any one agent).

A hierarchical or branch-and-bound style search naturally lends itself to the multi-agent scenario. However, existing hierarchical approaches for A$k$NN search (Yiu, Mamoulis, and Papadias 2005) rely on Euclidean heuristics, which can only provide a very loose lower-bound on network distance for important metrics such as travel-time. To address this, our previous work (Abeywickrama, Cheema, and Storandt 2020) proposed Compacted-Object Landmark Trees (COL-Trees) to enable efficient hierarchical POI search via more accurate landmark lower-bounds ("differential heuristics"), significantly speeding up A$k$NN search. Here we present the findings of an extended version (Abeywickrama, Cheema, and Storandt 2026) that demonstrates COL-Trees are effective even in single-agent scenarios where no efficient methods exist or they were not thought to be effective.



## Efficient $k$FN & Range Search

A $k$ Farthest Neighbor ($k$FN) query involves retrieving the $k$ POIs farthest from a single agent's location. $k$FN queries find important real-world applications, such as finding spatial outliers, e.g., a healthcare facility identifying elderly residents that are farthest by travel time for potential remote monitoring, or a delivery company finding the farthest customers from their warehouse to understand the worst-case delivery times. $k$FN search is also frequently used as a subroutine in solutions to many network-based search and planning problems such as vehicle routing and metric $k$-center computation. In addition to $k$NN queries, *range* queries are another proximity-based search query, which involves retrieving all POIs within some network distance radius $r$, e.g., to find all restaurants within 10-minutes drive of a user. The weaknesses of existing POI search heuristics extend beyond A$k$NN queries to $k$FN and range queries as well.

Euclidean $k$FN search can be easily performed by adapting Euclidean $k$NN search in data structures such as R-trees to use the maximum Euclidean distance. However, $k$FN search by network distance cannot use Euclidean distance as a heuristic as it is still only a lower bound on network distance. Being a maximization problem, $k$FN search requires candidates to be retrieved by upper-bounds to terminate optimally. Moreover, incremental candidate retrieval approaches such as the previously described recurrence rule, necessarily involve retrieving *all* POIs as candidates before the furthest can be determined. While such incremental approaches suit $k$NN search where the goal is to find the $k$ results and terminate, range queries do not have the benefit of stopping after

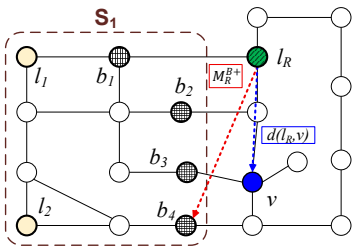


Figure 1: Lower-Bound Distance to Subgraph Borders

a fixed number of results. We show that COL-Trees are able to remedy both of these scenarios. For $k$FN queries, COL-Trees allow retrieval of $k$FN candidates by upper-bounds on network distance for the first time. For range queries, COL-Trees enable retrieving entire groups of POIs as results without having to individually evaluate each candidate POI.

## Data Structure Improvements

**Subgraph Restricted Dijkstra:** Pre-processing costs are dominated by the Dijkstra search from each landmark to all other vertices in its subgraph. While the subgraph is connected and disjoint, it is possible that shortest paths leave and re-enter, requiring Dijkstra to search outside the subgraph to ensure correctness. We propose a *subgraph restricted* variant of Dijkstra to limit this inefficiency by using the fact that any shortest path must enter and leave through one of the subgraph border vertices. Using pre-computed min and max distances from root landmarks to subgraph borders, we can prune the Dijkstra search outside the subgraph in an A*-like manner by computing lower-bounds back to the subgraph. E.g., in Figure 1, a lower-bound can be computed from $v$ to subgraph $S_1$ by $M_R^{B+} - d(l_R, v)$ (both available essentially for free) using the triangle inequality. The extended version details this and other improvements.

## Experimental Results

We evaluate the query time of our techniques on a large-scale road network for the USA with real-world POIs obtained from OpenStreetMap as well as synthetic POIs. In the extended version, we conduct extensive sensitivity analysis and evaluate machine-independent heuristic performance in terms of candidates retrieved and network distances computed. We have released all relevant code, including the first open-source implementation of COL-Trees, via GitHub[1].

Figure 2 shows the $k$FN query time and candidates retrieved with increasing object density $d$. The suffix indicates that Pruned Highway Labeling (PHL) (Akiba et al. 2014) was used for network distance computation. Note that competing A$k$NN techniques cannot be applied to $k$FN: the Euclidean heuristic cannot be used to answer $k$FN at all and the Network Voronoi Diagram (NVD) heuristic evaluates all POIs by default. As a result, we create an optimized brute-force algorithm, AUB, that evaluates all POIs but filters candidates by upper-bounds to reduce network distance computation. From Figure 2, COLT-PHL significantly outperforms AUB-PHL by up to four orders of magnitude. The advantage is greater as the POI density increases because AUB-PHL must evaluate more POIs. On the other hand, COLT-PHL query time essentially remains constant because, irrespective of the number of POIs, its pruning ability is highly effective at avoiding non-result POIs, as illustrated by the constant number of candidates retrieved. Surprisingly, COLT-PHL even outperforms the state-of-the-art heuristics on range queries, despite being less intuitive for proximity-based search, as detailed in the extended version.

[1]https://github.com/tenindra/rn-obj-search

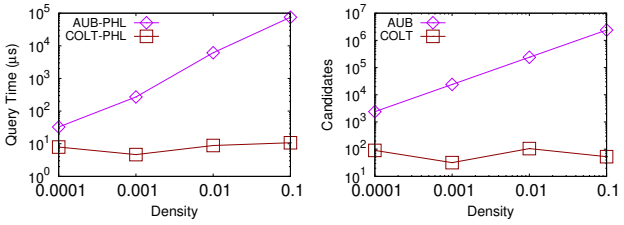


Figure 2: $k$FN Query Time & Candidates Retrieved

## Conclusion & Future Directions

We show that COL-Trees are highly suited to other types of object search problems, including those in single-agent settings, such as $k$FN and range queries. Many location-based services (LBS) rely on Euclidean distance due to lower pre-processing costs, despite lower accuracy. By also significantly closing the gap on pre-processing cost, we demonstrate that COL-Trees are a versatile data structure that may make network distance-based approaches possible for other heuristic search problems in LBS. Moreover, further improvement of COL-Trees may be possible, e.g., by using compressed differential heuristics (Goldenberg et al. 2011).

## Acknowledgments

Tenindra Abeywickrama is supported by JSPS KAKENHI Grant Number 25K24399. Muhammad Aamir Cheema is supported by Australian Research Council DP230100081.

## References

Abeywickrama, T.; Cheema, M. A.; and Storandt, S. 2020. Hierarchical Graph Traversal for Aggregate k Nearest Neighbors Search in Road Networks. In *ICAPS*, 2–10.

Abeywickrama, T.; Cheema, M. A.; and Storandt, S. 2026. COL-Trees: Efficient Hierarchical Object Search in Road Networks. arXiv:2601.22183.

Akiba, T.; Iwata, Y.; Kawarabayashi, K.-i.; and Kawata, Y. 2014. Fast Shortest-path Distance Queries on Road Networks by Pruned Highway Labeling. In *ALENEX*, 147–154.

Goldenberg, M.; Sturtevant, N. R.; Felner, A.; and Schaeffer, J. 2011. The Compressed Differential Heuristic. In *AAAI*, 24–29.

Yiu, M. L.; Mamoulis, N.; and Papadias, D. 2005. Aggregate Nearest Neighbor Queries in Road Networks. *IEEE Trans. Knowl. Data Eng.*, 17(6): 820–833.

Zhu, L.; Jing, Y.; Sun, W.; Mao, D.; and Liu, P. 2010. Voronoi-based Aggregate Nearest Neighbor Query Processing in Road Networks. In *GIS*, 518–521.