k-Nearest Neighbors on Road Networks: Euclidean Heuristic Revisited[*]

Tenindra Abeywickrama, Muhammad Aamir Cheema, David Taniar
Faculty of Information Technology, Monash University, Australia
{tenindra.abeywickrama,aamir.cheema,david.taniar}@monash.edu

Abstract

In the age of smartphones, finding the nearest points of interest (POIs) is a highly relevant problem. A popular way to solve this is to use a k Nearest Neighbor (kNN) query to retrieve POIs by their road network distances from a query location. However, we find that existing kNN methods have not been carefully compared. We present a detailed and fair experimental study of the state-of-the-art, documenting the many insights gleaned along the way. Notably, a long overlooked Euclidean distance heuristic is often the best performing method by a wide margin. We have also released all code as open-source for readers to reproduce experiments and easily add methods or queries to the testbed for new studies.[1]

1 Problem Definition

Let us consider a road network as an undirected graph $G = (V, E)$, with V being the set of vertices and E the edges between them. Each edge in E possesses a weight (e.g., travel distance) and the network distance between any two vertices is the minimum sum of weights connecting the vertices.

Now given a query vertex q and a set of object vertices O, a k Nearest Neighbor (kNN) query retrieves the k closest objects by network distance from q. For example, if O is the set of restaurants, a query might retrieve the 10 closest restaurants to the issuer's location by network distance.

2 Motivation

Finding kNNs is a challenging problem in graphs. For example, it is not practical to simply compute shortest paths to all objects in O to find the closest. However, kNN queries afford greater accuracy and flexibility, e.g., edge weights can be any metric like travel time. As a result the kNN problem is extremely popular and has been extensively studied.

But we observe several irregularities in existing work, including discrepancies in the relative performance of techniques and several overlooked comparisons. Chief among them is a neglected competitor using a simple Euclidean distance heuristic that was always hampered by an uneven playing field. Motivated by these anomalies, we present a careful and in-depth experimental study to settle these issues.

3 The Euclidean Distance Heuristic

Incremental Euclidean Restriction (IER) (Papadias et al. 2003) was an early kNN technique using a simple Euclidean distance lower-bounding heuristic reminiscent of the A* shortest path algorithm. Given a query vertex q, IER first retrieves the k closest objects to q by Euclidean distance as a *candidate set* R. Now the network distance to each candidate is computed using another technique (e.g., Dijkstra). These may not be the kNNs but the k-th furthest candidate gives an upper-bound D_k on the real k-th nearest neighbor.

IER improves R by iteratively retrieving the next nearest object c by Euclidean distance $d_e(q, c)$ from q. Euclidean distance is a lower-bound on the network distance when edge weights are physical distance. Thus if $d_e(q, c) \geq D_k$, R can no longer be improved and IER terminates. Otherwise, IER computes the network distance to c, updating R and D_k if necessary. IER was consistently reported to have inferior experimental performance (Papadias et al. 2003; Samet, Sankaranarayanan, and Alborzi 2008; Lee et al. 2012) and has since been dropped (Zhong et al. 2015).

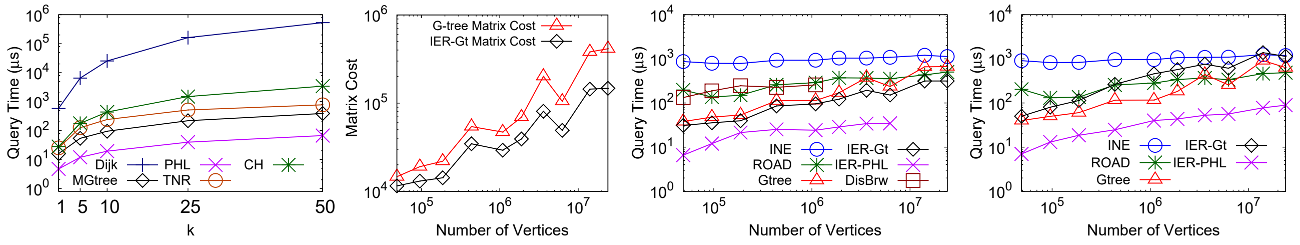
4 Revamping An Overlooked Heuristic

Curiously, Dijkstra's algorithm has always been used by IER to compute network distance. This is necessarily no faster than simply using Dijkstra's algorithm to incrementally settle vertices until k object vertices are found. In fact, it comes with the extra overhead of retrieving Euclidean candidates, making it slower! Considering the plethora of shortest path methods developed since Dijkstra, the true utility of the Euclidean heuristic has never been properly investigated.

We compare IER with several modern network distance methods (Wu et al. 2012) in Figure 1(a). Each technique comes with different trade-offs, e.g., Pruned Highway Labeling (PHL) (Akiba et al. 2014) provides the fastest query times at the expense of index size. The largest relative cost in IER is the network distance computation, and unsurprising IER-PHL is the clear winner given its faster queries. Nonetheless, even the slowest variant is orders of magnitude

[*]Extended Abstract, longer version appeared in PVLDB 2016 entitled "k-Nearest Neighbors on Road Networks: A Journey in Experimentation and In-Memory Implementation"

[1]https://github.com/tenindra/RN-kNN-Exp



(a) IER Variants on Travel Dist. (b) Matrix Cost on Travel Dist. (c) Query Time on Travel Dist. (d) Query Time on Travel Time

Figure 1: kNN on real road networks for uniformly random query and object locations with density d=0.001 × $|V|$ and k=10 (a) IER variants on NW-US (b) heuristic efficiency via G-tree matrix cost (c) travel distance queries (d) travel time queries

faster than Dijkstra, confirming our belief that IER's past experimental performance did not truly reflect its potential.

5 Experimental Results

We implemented state-of-the-art kNN techniques ourselves in C++ and made it available as open source[1]. To ensure each technique was as efficient as possible we (a) implemented carefully by benchmarking choices (b) made algorithmic improvements and (c) compared with author's code when provided. We test kNN queries on real road networks[2] for both synthetic and real POIs. Further details can be found in the full paper (Abeywickrama, Cheema, and Taniar 2016).

One of the major goals of our experimental investigation is to compare how a revitalized IER compares to the state-of-the-art. We use IER-PHL as it delivers the best performance at a higher memory cost. We also test IER utilizing the G-tree road network index (Zhong et al. 2015) (denoted as IER-GT). G-tree is also capable of answering kNN queries, using a different search heuristic based on subgraph border distances (Zhong et al. 2015), thus making IER-GT and G-tree an "apples-to-apples" comparison of different heuristics.

Figure 1(b) shows the matrix cost of kNN queries using the G-tree index for travel distance. Matrix cost represents a machine independent metric for how often the distance matrices of the G-tree index are utilized. The lower cost for IER-GT suggests that IER is using the G-tree index more efficiently than G-tree's own kNN algorithm. In other words, IER's simple Euclidean heuristic outperforms the complex heuristic used by G-tree's algorithm on travel distance.

We compare IER's query time performance against several state-of-the-art techniques such as *INE* (Papadias et al. 2003), *Distance Browsing* (Samet, Sankaranarayanan, and Alborzi 2008), *ROAD* (Lee et al. 2012) and *G-tree* (Zhong et al. 2015). Figure 1(c) verifies IER-GT's superior heuristic performance does translate to faster queries over G-tree with varying road network size for travel distance. This is despite IER-GT having the additional overhead of retrieving Euclidean NNs using an R-tree. However, IER-GT's performance degrades in Figure 1(d) as Euclidean distance can only provide a loose lower-bound for travel times. IER-PHL on the other hand consistently outperforms all techniques on both travel distance *and* travel time, by a significant margin.

[2]http://www.dis.uniroma1.it/%7Echallenge9/

6 Analysis & Conclusions

Two factors influence the surprising performance of IER. First, the heuristic is more accurate than expected, especially for travel distance. If a simple heuristic like Euclidean distance can outperform state-of-the-art heuristics, as in Figure 1(b), then further thought is required to design more effective search heuristics. Second, IER benefits from decoupling the search heuristic from the underlying road network index. This makes it orthogonal to research on shortest path computation, allowing IER to easily leverage new developments. This is exemplified by IER-PHL's query results on travel times, where the loss in heuristic performance is more than made up for by the efficiency of the network distance computation. Developing such decoupled heuristics may be a more promising direction for kNN search, e.g., by using tighter lower-bounds for travel time (Abeywickrama and Cheema 2017). We refer the reader to the full paper for further details and other experimental insights.

References

Abeywickrama, T., and Cheema, M. A. 2017. Efficient Landmark-Based Candidate Generation for kNN Queries on Road Networks. In *DASFAA*, 425–440.

Abeywickrama, T.; Cheema, M. A.; and Taniar, D. 2016. K-nearest neighbors on road networks: A journey in experimentation and in-memory implementation. *PVLDB* 9(6):492–503.

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.

Lee, K.; Lee, W.-C.; Zheng, B.; and Tian, Y. 2012. Road: A new spatial object search framework for road networks. *TKDE* 24(3):547–560.

Papadias, D.; Zhang, J.; Mamoulis, N.; and Tao, Y. 2003. Query processing in spatial network databases. In *VLDB*, 802–813.

Samet, H.; Sankaranarayanan, J.; and Alborzi, H. 2008. Scalable network distance browsing in spatial databases. In *SIGMOD*, 43–54.

Wu, L.; Xiao, X.; Deng, D.; Cong, G.; Zhu, A. D.; and Zhou, S. 2012. Shortest path and distance queries on road networks: An experimental evaluation. *PVLDB* 5(5):406–417.

Zhong, R.; Li, G.; Tan, K.; Zhou, L.; and Gong, Z. 2015. G-tree: An efficient and scalable index for spatial search on road networks. *TKDE* 27(8):2175–2189.