Article can be accessed online at http://www.publishingindia.com

Transport Optimized Nearest Neighbor Query for Location Based Services

Debajyoti Ghosh*, Prosenjit Gupta**

Abstract

Location Based Services are increasingly becoming popular due to increased usage of mobile devices by citizens seeking information on points-of-interest, travel routes, traffic conditions etc. We consider practical variants of the nearest neighbor problem on road networks, wherein the goal is to find the nearest point-of-interest from a query location. Here the notion of proximity is determined by the ease of reaching the point of interest via public transport. Using graphs modeling the road network and transport connectivity, efficient algorithms are presented.

Keywords: Nearest Neighbor Search, Computational Geometry, Road Networks, Shortest Path Problem, Location Based Services

Introduction

Location Based Services (LBS) [4][54][55] are increasingly becoming popular due to increased usage of mobile devices by citizens seeking information on points-of-interest, travel routes, traffic conditions etc. LBS aim at providing customized information keeping location as an important factor. Location-based Services include emergency services, navigation and information services, advertising services and tracking services to name a few. Due to rapid urbanization and smart city projects in various countries including India, interest in LBS has further increased. According to market research, the global LBS market is poised to grow significantly in the next few years. This has created several challenges leading to research in new technologies, architectures and algorithms [4].

Nearest Neighbor Search [41][43][48][52] and its variants have been well studied in areas as diverse as Computational Geometry, Pattern Recognition, Interpolation, Probability Theory, Data Mining, Machine Learning and Geographical Information Systems. The basic problem is defined as follows: Given a set of point S and a query point q, the goal is to find a point in S nearest to q. In Computational Geometry [1], this is usually studied as a repetitive mode query problem wherein the goal is to preprocess S into a data structure so that given q, we can find p ε S nearest to q efficiently. Knuth [2] christened this as the "Post Office Problem" in reference to the age-old problem of locating the nearest post-office to a query user. In the 2-dimensional Euclidean plane, the solution to the post-office problem is the Voronoi diagram along with an accompanying planar point location data structure [3].

In this paper, we consider a variant of the nearest neighbor problem which is relevant in the context of Location Based Services and smart cities. We consider a road network, serviced by public transport (e.g. bus, train, metro, auto-rickshaw etc.). There are several points-of-interest (POI's) spread on the road network. There are also bus stops (or train stations) located at various points on the road network. We are given a set of bus routes serving the road network, each route being an ordered sequence of bus stops. A mobile user, currently at a query location q on this road network wishes to travel to the nearest POI. The user is interested in using public transport (say buses) to reach his/her destination. Given that bus transfers are often inconvenient, lead to delays or increased cost of travel, the user wishes to minimize the number of transfers. However since number of paths to reach a POI using the minimum number of bus transfers may often not be unique, the user may wish to reach the POI using that route that is the shortest among all routes using the minimum number of transfers. We assume that the user will walk to the nearest bus stop from q and also

* NIIT University, Neemrana, Rajasthan, India. Email: debajyoti.ghosh@niituniversity.in
** NIIT University, Neemrana, Rajasthan, India. Email: prosenjit_gupta@acm.org

walk from the destination bus top to the POI concerned. We also assume that such transfers are short and ignore these in our computation.

Our goal is to preprocess all information related to the road network, POI, bus stops and bus routes into a data structure so that given any query point q, we can return the nearest POI reachable from q by the minimum number of bus transfers. We denote this as the transport route optimized nearest neighbor problem (TRNN). Given q, our algorithm will return the POI which we call TRNN(q). In another variant of the problem, we wish to preprocess all information related to the road network, POI, bus stops and bus routes into a data structure so that given any query point q, we can return the nearest POI reachable from q by the minimum number of bus transfers and in case of multiple such solutions, by the shortest one. We denote this as the transport route distance optimized nearest neighbor problem (TRDNN). Given q, our algorithm will return the POI which we call TRDNN(q).

The paper is organized as follows: Section II defines some graphs we create to model our problem appropriately. Section III reviews the past literature on nearest neighbor problems. Section IV lists a few properties of these graphs which form the basis of the algorithms. Section V provides the preprocessing and query algorithms for the transport route optimized nearest neighbor problem. The concluding remarks and directions for future research are given in Section VI.

Preliminaries

Road Network Graph RNG

We model the underlying road network as a weighted and connected undirected graph, the Road Network Graph (RNG) as G (V, E, W), where V is the set of vertices denoting the intersections of the road segments, E is the set of undirected edges in the road network and W is the (non-negative) weight function defined on the edges, W(e) being the road network distance along the length of the corresponding road segment. We define the shortest path distance or the road network distance, between any two vertices u, v $\in$ V as d(u, v), the minimum sum of weights of edges connecting u and v. P is a set of POI's located on edges and vertices. V, P, q are all points but with different semantics with respect to the network. If e = (u, v) is an edge, d(u, v) = W(e). If either a or b is not on the road network, d(a, b) = ∞. We assume that all query points and POI's are on the edges (i.e. either in the interior of edges or vertices).

Transit Map Graph TMG

As part of the input, besides the RNG we are also given a list of bus routes. Each bus route is an ordered sequence of bus stops with a designated start and end stops. We define the Transit Map Graph (TMG) = G(V, E) as follows. The vertex set V is the set of bus stops. Two vertices u and v are connected by a directed edge (u, v) iff there is a bus route connecting u and v with no other bus

stops in between. Associated with each bus stop b $\in$ V, we have two inverted lists: RR(b) is the list of bus routes that include b. RP(b) is the list of POI's for which b is the nearest bus stop.

Bus Route Intersection Graph RIG

We define the Bus Route Intersection Graph (RIG) = G(V, E) as follows. The vertex set V is a set of bus routes. Two vertices u and v are connected by an undirected edge (u, v) if there is a bus stop at which routes u and v intersect.

Past Work

The classical problem shortest path computation on general graphs or over a network has well studied by lots of researchers over the years [35]. The best known algorithm to find single source shortest paths (SSSP) is the Dijkstra's algorithm. A survey paper by Zhan and Noon [36] studies the relative performance of many of the classical shortest path algorithms when applied on a road-network dataset. Wagner and Willhalm [37] suggested a geometric approach for speed-up techniques for shortest path computations in a spatial network. There are extended versions of Dijkstra's such as A* search [31] with various expansion heuristics, that also find the shortest distance path starting from a node s to a goal node d.

In recent years, many researchers have focused on efficient computation of Nearest neighbor (NN) on road networks. It is closely related to the single-source shortest path problem, which has been extensively studied. It forms the heart of all queries, where a location is given and one seeks to find the nearest object(s) (e.g., petrol pump, ATM, hotel, restaurant, shopping mall, hospital, etc.). The

nearest neighbor query is a key operation in geographic information systems (GIS), spatial network databases, and location-based services, many other disciplines of computer science including computer vision, machine learning, pattern recognition, information retrieval, etc. and also in other interdisciplinary areas like coding theory, robotics, recommendation systems, etc. Many of these algorithms are based on techniques of computational geometry. A number of approaches exist for computing nearest neighbor queries in spatial networks. Existing approaches are either based on Dijkstra's method where distances are pre-computed [41][47][48][49][50][51] or many algorithms assume an indexing structure, e.g., an R-tree, kd tree, and search in a branch-and-bound manner [42][43][45]. Kolahdouzan and Shahabi [46] proposed Voronoi diagram based Nearest Neighbor approach, which can handle sparse datasets but is inappropriate for medium and dense datasets due to its high preprocessing and storage overhead. Samet et al. [38][39] proposed Best First Search(BFS) method to compute the nearest neighbors to a query point on a spatial network. In many applications, a rough answer is sufficient, so that algorithms have been developed that return an approximate result [40][44]. The problem of finding the In-Route Nearest Neighbor (IRNN) for a query object is first proposed by Shekhar et al. [53] to search a facility instance with the smallest detour distance from the query route on the way to the destination, given route with a destination and a current location on it.

Properties

In this section, we list a few properties of the graphs defined earlier. These properties enable us to solve the transport route optimized nearest neighbor problem efficiently.

(a) Query points can assumed to be at bus stops

Because of our assumption that the user will walk to the nearest bus stop from the query point q, and that such distance walked is negligible, without loss of generality, we can assume that all query points will be bus stop points. Given any arbitrary query point q, the solution is equivalent to the one obtained by assuming q to be at the nearest bus stop instead.

(b) POI's can be mapped to nearest bus stops

Since our query user will also walk from the destination bus top to the POI concerned, effectively the problem at hand reduces to one wherein we can assume that all POI's are mapped to the nearest bus stop.

In our model, POI's are located on the edges of the RNG. Consider a POI p located on an edge e = (u, v) of the RNG. We assume that a POI p could be on either side of the 2-way road e and an incoming user interested in visiting the POI p could approach it via a bus either from the direction of u or from the direction of v. Thus during preprocessing, we can map p to two nearest bus stops, pn(u) and pn(v), assuming the user will be approaching p from the direction of u (respectively v).

(c) If TRNN(q) = x and $x \in RP(b)$, then for any $y \in RP(b)$, $x \neq y$, y is also a solution to TRNN(q). Clearly any of the POI's in RP(b) could suffice. Also the same result holds true for TRDNN as well.

(d) Let b1 be the bus stop closest to a query point q and POI $p \in RP(b2)$. Then if for $x \in RP(b1)$ and $y \in RP(b2)$, if there is a path of length δ in the RIG, the nearest POI from q is reachable in δ or fewer bus transfers.

Consider an edge e = (u, v) in the RNG. We color as red each bus stop on ewhich is either a start or end bus top for a bus route. We also color as red bus stops nearest to u and v. All other bus stops are colored blue. We repeat this for all edges. These colors are carried over to the TMG.

(e) If two bus routes intersect, they must do so at a red bus stop.

Thus it is sufficient to consider the red vertices to determine the RIG. If the input consists of a RNG with n vertices, e edges and a sequence of k bus routes, clearly the number of red vertices in the TMG is 2(e+k).

The Algorithms

Using the properties enumerated in the previous section, we can outline the algorithms for solving the TRNN and TRDNN problems.

(a) Input: RNG and list of bus route. We assume that the RNG is presented as an adjacency list of vertices. For each edge e of the RNG, we maintain a list of POIs and bus stops in the order that they occur on the edge. Each bus route is represented as an ordered list of bus stops.

(b) Generate the TMG from the list of bus routes. In the TMG, each bus stop is a vertex. For each bus route r, we go through the list of vertices in order. If a vertex

v does not exist in the TMG, we create v. For each pair of consecutive bus stops on route r, corresponding to vertices u and v created in the TMG, we add an edge (u, v). For the vertices corresponding to the first and last stops on route r, we ensure the stops are colored red. Also if a bus stop is located as the first or last stop of an edge on the RNG, we color the corresponding vertex in the TMG as red. All other vertices are colored blue. Also while processing a bus route r, if r passes through a POI, it is labeled white. All other bus routes are labeled black. White bus routes pass through bus stops near POI's. Black bus routes connect white bus routes but do not pass through bus stops near POI's.

(c)For each vertex b in the TMG, generate RR(b). This can also be ensured during creation of the TMG. While iterating through bus stops for a bus route r, the route r is added to the list RR(b) for each bus stop b on route r.

(d) For each vertex b in the TMG, generate RP(b). We walk through each edge of the RNG, mapping each POI p to the nearest bus stop b (which could be in either direction). The POI p is added to the list RP(b) for the corresponding vertex b in the TMG.

(e) Generate the RIG from the TMG. Recall that the RIG consists of vertices corresponding to bus routes. We consider RR(b) for each red vertex b in the TMG. As discussed in the previous section, this information is sufficient to construct the RIG. For any $u \in RR(b)$, $v \in RR(b)$, $u \neq v$, (u, v) is an edge in the RIG. The weight of each edge on the RIG is assumed to be unity.

(f) Computing TRNN : Given a query point q on the RNG, we wish to reach the POI which is reachable by a route with the fewest number of bus transfers. We identify the nearest bus stop b. Then we look up the list RR(b) for the node b in the TMG. To compute the TRNN(q), we run a modified version of Dijkstra's shortest path algorithm [35] on the RIG. In the initialization step instead of setting d(s) = 0 for a single source node in Djikstra's algorithm, we set d(s) = 0 for all nodes $s \varepsilon RR(b)$. We terminate the algorithm as soon as a node r is extracted from the priority queue in Dijkstra's algorithm, such that r is white. We then walk through the bus route r and return the first POI reached.

(g) Computing TRDNN: Given a query point q on the RNG, we wish to reach the POI which is reachable by a route with the fewest number of bus transfers and in case of a tie by one of those routes which is shortest. To compute TRDNN(q), we run a Depth First Branch and Bound (DFBAB) algorithm on the RIG from each node $s \varepsilon RR(b)$. The d(.) value at a node v is a 2-tuple (f, g) where f is the minimum number of bus transfers between the starting bus stop and the boarding point for the route corresponding to node v while arriving via the minimum transfer route. The value g is the actual road network distance to arrive at the boarding point for the route corresponding to node v while arriving from the starting bus stop via the minimum transfer route. The comparison operator for comparing d values is also redefined. Let $d(u) = (f_u, g_u)$ and $d(v) = (f_v, g_v)$. Then $d(u) < d(v)$ iff $(f_u<f_v)$ or $((f_u = f_v)$ and $(g_u<g_v))$. The weight of an edge (u, v) is taken as (1, w) where w is the road network distance traversed between the bus stop where a bus on bus route corresponding to node u is boarded and the bus stop where the transfer to the bus on route v takes place. Suppose during the execution of the algorithm we reach a white node corresponding to bus route r, we note the edge (u, r) via we reach r. Let b be the bus stop at which routes u and r intersect. We then walk through the bus route r and find the first POI p reached from the bus stop b along route r. The distance between b and p must be added to the second component of the d value of node r. If the d value at any node exceeds the d value of the best solution found so far, we do not explore ahead of that node.

Conclusion

We have considered a pragmatic variant of the nearest neighbor problem on road networks which is motivated by applications in Advanced Traveler Information Systems and Location Based Services wherein the querying user travels only using public transport. We have given two algorithms for the problem in two cases where the proximity is defined in different ways. In future, we will consider other practical definitions of proximity and also test the algorithm on real data sets.

References

[1] Preparata, F. P., & Shamos, M. (1993). Computational Geometry: An Introduction, (1st ed). Springer.

[2] Knuth, D. E. (1998). The Art of Computer Programming, (2nd ed.), Vol. 3. Addison Wesley.

[3] Berg, de. M., Cheong, O., Kreveld, van. M., & Overmars, M. (2008). Computational Geometry: Algorithms and Applications. Springer.

[4] Ilarri, S., Mena, E., & Illarramendi, A. (2010). Location- Dependent query processing: Where we are and where we are heading. *ACM Computing Surveys*, March, 42(3), 1-67.

[5] Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numeriche Mathematik*, 1, 269-271.

[6] Ahuja, R. K., Mehlhorn, K., Orlin, J. B., & Tarjan, R. E. (1990). Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2), 213-223.

[7] Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM,* July, 34(3), 596-615.

[8] Goldberg, A. V. (2001). *A simple shortest path algorithm with linear average time*. In 9th European Symposium on Algorithms (ESA), vol. 2161 of LNCS, (pp.230-241), Springer.

[9] Holzer, M., Schulz, F., & Willhalm, T. (2004). *Combining speed-up techniques for shortest-path computations.* In 3rd International Workshop on Experimental and Efficient Algorithms (WEA), vol. 3059 of LNCS, (pp. 269-284). Springer.

[10] Klein, P., Rao, S., Rauch, M., & Subramanian, S. (1994). Faster shortest-path algorithms for planar graphs. I*n 26th ACM Symposium on Theory of Computing*, (pp. 27-37).

[11] Wagner, D., & Willhalm, T. (2005). Drawing graphs to speed up shortest-path computations. In Workshop on Algorithm Engineering and Experiments (ALENEX).

[12] Denardo, E. V., & Fox, B. L. (1979). Shortest-Route Methods: Reaching, Pruning, and Buckets. *Operation Research*, 27, 161-186.

[13] Schultes, D. (2008). Route Planning in Road Networks. PhD thesis, Universit at Karlsruhe (TH), Fakultat fur Informatik (2008).

[14] Gallo, G., & Pallottino, S. (1984). Shortest path methods: Complexity, interrelations and new propositions. *Networks*, 14(2), 257-267.

[15] Glover, F., Glover, R., & Klingman, D. (1984). Computational Study of an Improved Shortest Path Algorithm. Networks, 14, 25-36.

[16] Zhan, F. B., & Noon, C. E. (1998). Shortest Path Algorithms: An Evaluation using Real Road Networks. *Transportation Science*, 32, 65-73.

[17] Knuth, D. E. (1977). A Generalization of Dijkstra's Algorithm. *Information Processing Letters*, 6(1), 1-5.

[18] Yen, J. Y. (1970). An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, July, 27(4), 526-530.

[19] Lee, K. C. K., & Zheng, B. (2009). Fast object search on road networks. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, (pp. 1018-1029), ACM New York.

[20] Wu, L., Xiao, X., Deng, D., Cong, G., Zhu, A. D., & Zhou, S. (2012). Shortest path and distance queries on road networks: An experimental evaluation. *Proceedings of the VLDB Endowment*, 5(5), 406-417.

[21] Bast, H., Funke, S., Sanders, P., & Schultes, D. (2007). Fast routing in road networks with transit nodes. *Science*, 316(5824), 566.

[22] Hilger, M., Kohler, E., Mhring, R. H., & Schilling, H. (2006). Fast point-to-point shortest path computations with Arc-Flags. *9th DIMACS Implementation Challenge*.

[23] Lauther, U. (2004). An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. In IfGIprints 22, Institut fuer Geoinformatik, Universitaet Muenster, (pp. 219-230).

[24] Sanders, P., & Schultes, D. (2005). Highway hierarchies hasten exact shortest path queries. ESA 2005, LNCS 3669, 568-579.

[25] Hart, P. E., Nilsson, N. J., & Raphael, B. (1972). Correction to a Formal Basis for the Heuristic Determination of Minimum Cost Paths. SIGART Newsletter. 37, 28-29.

[26] Nilsson, N. J. (2014). Principles of Artificial Intelligence, Springer, reprint..

[27] Akiba, T., Iwata, Y., Kawarabayashi, K., & Kawata, Y. (2014). Fast shortest-path distance queries on road networks by pruned highway labeling. *Proceedings of the 16th Meeting on Algorithm Engineering and Experiments (ALENEX)*.

[28] Akiba, T., Hayashi, T., Nori, N., Iwata, Y., & Yoshid, Y. (2015). Efficient Top-k Shortest-Path Distance Queries on Large Networks. Pruned Landmark Labeling. Association for the Advancement of Artificial Intelligence.

[29] Bast, H., Funke, S., Matijevic, D., Sanders, P., & Schultes, D. (2007). In transit to constant time shortest-path queries in road networks. In ALENEX, 2007.

[30] Bauer, R., Delling, D., Sanders, P., Schieferdecker, D., Schultes, D., & Wagner, D. (2008). Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. *Journal of Experimental Algorithmics,* 15, 303-318, .

[31] Goldberg, A. V., & Harrelson, C. (2005). Computing the shortest path: A* search meets graph theory. *In SODA: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, (pp.156-165).

[32] Johnson, E. L. (1972). On shortest paths and sorting. *Proceedings 25th ACM Annual Conference*, (pp. 510-517).

[33] Hung, M. S., & Divoky, J. J. (1988). A computational study of efficient shortest path algorithms. *Computers and Operations Research*, 15(6), 567-576.

[34] Bertsekas, D. P. (1993). A simple and fast label correcting algorithm for shortest paths. *Networks*, December, 23(8), 703-709.

[35] Cormen, T. H., Leiserson, C. E., Stein, C., & Rivest, R. (2009). Introduction to Algorithms (3rded.) MIT Press.

[36] Zhan, F. B., & Noon, C. E. (1998). Shortest Path Algorithms: An Evaluation using Real Road Networks. *Transportation Science*, February, 32(1), 65-73.

[37] Wagner, D., & Willhalm, T. (2005). Drawing graphs to speed up shortest-path computations. *In Proceedings of the 7th Workshop on Algorithm Engineering and Experiments*.

[38] Sankaranarayanan, J., Alborzi, H., & Samet, H. (2005). Efficient Query Processing on Spatial Networks. In Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems, (pp. 200-209), Bremen, Germany, November.

[39] Hjaltason, G. R., & Samet, H. (1991). Ranking in spatial databases. *In Proceedings of the 4th International Symposium on Large Spatial Databases*, (pp. 83-95).

[40] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. (1994). An optimal algorithm for approximate nearest neighbor searching. *In Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, (pp. 573-582).

[41] Broder, A. J. (1990). Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23(1-2), 171-178.

[42] Hjaltason, G. R., & Samet, H. (1999). Distance Browsing in Spatial Databases. *In TODS,* 24(2), 265-318.

[43] Roussopoulos, N., Kelley, S., & Vincent, F. (1995). Nearest Neighbor Queries. *In Proceedings SIGMOD,* (pp. 71-79).

[44] Ailon, N., & Chazelle, B. (2006). Approximate nearest neighbors and the Fast Johnson-Lindenstrauss Transform. *Proceedings of the Symposium on Theory of Computing (STOC)*.

[45] Panigrahy, R. (2008). An Improved Algorithm Finding Nearest Neighbor using KD-Trees. *In 8th Latin American conference on Theoretical informatics*, Bzios, Brazil, (pp. 387-398).

[46] Kolahdouzan, M., & Shahabi, C. (2004). Voronoi-Based Nearest Neighbor Search for Spatial Network Databases. *In Proceedings of VLDB*, (pp. 840-851).

[47] Huang, X., Jensen, C. S., & Saltenis, S. (2005). The Islands Approach to Nearest Neighbor Querying in Spatial Networks, DB Tech Report TR-12. Department of Computer Science, Aalborg University.

[48] Jensen, C. S., Kolarvr, J., Pedersen, T. B., & Timko, I. (2003). Nearest Neighbor Queries in Road Networks. *In Proceedings 11th ACM International Symposium on Advances in Geographic Information Systems*, (pp. 1-8).

[49] Yianilos, P. N. (1993). Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. *In 4th Annual ACM-SIAM Symposium on Discrete Algorithms,* Austin, Texas, USA, (pp. 311-321).

[50] Bajramovic, F., Mattern, F., Butko, N., & Denzler, J. (2006). A comparison of nearest neighbor search algorithms for generic object recognition. In 8th International Conference on Advanced Concepts for Intelligent Vision Systems, Antwerp, Belgium, (pp. 1186-1197).

[51] Beyer, K. S., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1998). When is nearest neighbour meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*.

[52] Andoni, A. (2009). Nearest Neighbor Search - The Old, the New, and the Impossible, Ph.D. dissertation, Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[53] Yoo, J. S., & Shekhar, S. (2005). In-route nearest neighbor queries. *Geo Informatica*, 9(2), 117-137.

[54] Schiller, J., & Voisard, A. (2004). *Location based services,* Elsevier.

[55] Ferraro, R., & Aktihanoglu, M. (2011). Location aware applications, Manning Publishers.