An Extensive Literature Review on Towards Online Shortest Path Computation

Rashmi Shukla[1] **& Prof. Pankaj Singh**[2]
[1]M-Tech Research Scholars, [2]Research Guide, Deptt. of Computer Science Engineering
SAM College of Engineering and Technology, Bhopal

Abstract: In this review paper we have analyzed the shortest path computation methods as when someone drive to somewhere 'far away', he will leave your current location via one of only a few 'important' traffic junctions. Starting from this informal observation, we study the algorithmic approach transit node routing that allows us to reduce quickest-path queries in road networks to a small number of table lookups. The ability of expressing path problems, such as shortest paths or bill of materials, is considered to be a substantial extension of conventional database languages. To realize this extension efficiently, path algorithms from graph theory are used. There exist a lot of path algorithms. Most of them, however, were designed and investigated mainly in the context of main memory.

Keywords - Shortest Path, Online Suggestions, Traffic, Journey, Travelling.

I. INTRODUCTION

Computing shortest paths in graphs (networks) with nonnegative edge weights is a classical problem of computer science. From a worst case perspective, the problem has largely been solved that gave an algorithm that finds all shortest paths from a starting node s using at most m+n priority queue operations for a graph G = (V,E) with n nodes and m edges.

However, motivated by important applications (e.g., in transportation networks), there has recently been considerable interest in the problem of accelerating shortest path queries, i.e., the problem to find a shortest path between a source node s and a target node t. In this case, Dijkstra's algorithm can stop as soon as the shortest path to t is found.

A classical technique that gives a constant factor speedup is bidirectional search which simultaneously searches forward from s and backwards from t until the search frontiers meet. All further speedup techniques either need additional information (e.g., geometry information for goal directed search) or pre computation. There is a trade-off between the times needed for pre computation, the space needed for storing the pre computed information, and the resulting query time. In review existing pre computation approaches, which have made significant progress, but still fall short of allowing fast exact shortest path queries in very large graphs.

In particular, from now on we focus on shortest paths in large road networks where we use 'shortest' as a synonym for 'fastest'. The graphs used for North America or Western Europe already have around 20000 nodes so that significantly superliner pre processing time or even slightly superliner space is prohibitive. To our best knowledge, all commercial applications currently only compute paths heuristically that are not always shortest possible. The basic idea of these heuristics is the observation that shortest paths "usually" use small roads only locally, i.e., at the beginning and at the end of a path. Hence the heuristic algorithm only performs some kind of local search from s and t and then switches to search in a highway network that is much smaller than the complete graph. Typically, an edge is put into the highway network if the information supplied on its road type indicates that it represents an important road.

The computation of shortest paths is an important task in many network and transportation related analyses. The development, computational testing, and efficient implementation of shortest path algorithms have remained important research topics within related disciplines such as operations research, management science, geography, transportation, and computer science some research efforts have produced a number of shortest path algorithms as well as extensive empirical findings regarding the computational performance of the algorithms.

When faced with the task of computing shortest paths, one must decide which algorithm to choose. Depending on the application, algorithm runtime can be an important consideration in the decision making process. Although a number of computational evaluations have been reported there is no clear answer as to which algorithm or set of algorithms, runs fastest on real road networks, the most common type of network faced by practitioners. The primary goal of this study is to identify which algorithms run the fastest on real road networks. A secondary goal is to better understand the sensitivity of algorithm performance to input data.

Past computational evaluations were mainly based on randomly generated networks. The methods for random network generation varied considerably. The resulting random networks ranged from complete networks with uniformly distributed arc lengths to highly structured grid networks. In comparison to real road networks, random networks often differ with respect to the degree of connectivity as indicated by the arc-to-node ratios. The real networks studied in this study have arc-to-node ratios ranging from 2.66 to 3.28. This is different from many randomly generated networks described in the literature where arc-to-node ratios are reported as high as 10 (cf. GALLO and PALLOTTINO, 1988).

Another aspect in which random networks can differ from real networks stems from the fact that random network arc lengths are usually randomly drawn in an independent fashion. This can result in network irregularities whereby a node may be “close” to two adjacent nodes that are “far” apart. Such irregularities can strongly favour certain types of algorithms and drastically slow others. The random network generators reviewed in the literature had one characteristic which felt resulted in significant differences in real versus random networks, namely, they apply a process for establishing connectivity or arc length generation in a homogeneous fashion across a network. Real network topology often contains areas of dense urban network surrounded by highly sub networked suburban areas which are then further surrounded by a rural road structure. Certain methods for random network generation may replicate one particular area well, for example, grid network generators for downtown areas, but real networks contain a mixed pattern of different types of road network topologies which are virtually impossible to simulate.

The problem of building a traffic aware route planning service. The idea is that users register continuous routing queries, specifying a set of <startpoint, endpoint> tuples. In response, the system monitors delays and sends query results as updates to users (e.g., via email or SMS) if the fastest route between any of these designated start-end pairs changes, based on real-time updates to traffic delays (in evaluation, the use of real-time delays from a traffic monitoring deployment have done on a network of 30 taxis in the Boston area). Like existing route planning services (e.g., Google Maps), our system uses a graph of road segments, and applies shortest-path planning algorithms to that graph to recommend routes to users. Unlike existing systems, however, our system maintains a large number of routes for pre-designated <source, target> pairs and updates those routes as traffic delays on road segments change. Our system additionally allows users to specify day and time slots along with the start and end points if they do not desire continuous monitoring (registration tuples in this case are of the form <startpoint, endpoint, day, time>). In this paper, we focus on the harder problem where all user queries require continuous monitoring.

Because small variations in delay won’t significantly affect a user’s travel time, we are not concerned with always finding the exact optimal route for any <start, end pair but in detecting if a previously reported route has become substantially non-optimal in the face of updates and in providing a new route that is near-optimal and much better (but not necessarily the exact optimal). By suggesting alternate time-saving routes before the user begins to drive, the service could prove extremely useful to commuters who tend to get stuck in peak hour traffic congestions. Our system is practically motivated: an iPhone application, iCarTel (available on the app store) that are extending with the methods in this study.

While adding support for ad-hoc traffic aware routing queries is intuitively simple, it is not immediately clear how a service could practically maintain the large number of designated routes it would need to continuously keep updated. Road network graphs contain millions of vertices and edges; even a sub-graph corresponding to a city and its surrounding suburbs can contain tens of thousands of segments – for instance, the sub graph corresponding to Boston’s road network has nearly 40,000 links. A naive recalculation of the optimal route on arrival of every update (or a set of updates) using a single source shortest-path algorithm such as Dijkstra’s algorithm (or A* search) for all registered continuous routing queries could turn out to be a major computational overhead given that real time traffic updates usually affect only a small part of the network. Though such an approach might work if the number of registered continuous routing queries is relatively small, it is unlikely to scale as the number of queries go up. Algorithms that are able to update shortest paths in the presence of link changes do exist (e.g., [2]), but they typically have a higher space or computation overhead than is acceptable for the setting.

II. LETERATURE SURVEY

L. H. U, H. J. Zhao, M. L. Yiu, Y. Li and Z. Gong, [1] the online shortest path problem aims at computing the shortest path based on live traffic circumstances. This is very important in modern car navigation systems as it helps drivers to make sensible decisions. To the best knowledge, there is no efficient system/solution that can offer affordable costs at both client and server sides for online shortest path computation. Unfortunately, the conventional client-server architecture scales poorly with the number of clients. A promising approach is to let the server collect live traffic information and then broadcast them over radio or wireless network. This approach has excellent scalability with the number of clients. Thus, authors develop a new framework called live traffic index (LTI) which enables drivers to quickly and effectively collect the live traffic information on the broadcasting channel. An impressive result is that the driver can compute/update their shortest path result by receiving only a small fraction of the index.

Their experimental study shows that LTI is robust to various parameters and it offers relatively short tune-in cost (at client side), fast query response time (at client side), small broadcast size (at server side), and light maintenance time (at server side) for online shortest path problem.

Holger Bast,Stefan Funke, Domagoj Matijevic, Peter Sanders, Dominik Schultes, [2] When you drive to somewhere 'far away', you will leave your current location via one of only a few 'important' traffic junctions. Starting from this informal observation, authors develop an algorithmic approach transit node routing that allows us to reduce quickest-path queries in road networks to a small number of table lookups. Authors present two implementations of this idea, one based on a simple grid data structure and one based on highway hierarchies. For the road map of the United States, their best query times improve over the best previously published figures by two orders of magnitude. Authors results exhibit various trade-offs between average query time (5 µs to 63 µs), preprocessing time (59 min to 1200 min), and storage overhead (21 bytes/node to 244 bytes/node).

Bin Jiang, [3] to establish the behaviour of algorithms in a paging environment, the author analyzes the input/output (I/O) efficiency of several representative shortest path algorithms. These algorithms include single-course, multisource, and all pairs ones. The results are also applicable for other path problems such as longest paths, most reliable paths, and bill of materials. The author introduces the notation and a model of a paging environment. The I/O efficiencies of the selected single-source, all pairs, and multisource algorithms are analyzed and discussed

N. Malviya, S. Madden and A. Bhattacharya, [4] In this study, authors address the problem of answering continuous route planning queries over a road network, in the presence of updates to the delay (cost) estimates of links. A simple approach to this problem would be to recompute the best path for all queries on arrival of every delay update. A naive approach scales poorly when there are many users who have requested routes in the system. Instead, authors proposed two new classes of approximate techniques - K-paths and proximity measures to substantially speed up processing of the set of designated routes specified by continuous route planning queries in the face of incoming traffic delay updates. Authors techniques work through a combination of pre-computation of likely good paths and by avoiding complete recalculations on every delay update, instead only sending the user new routes when delays change significantly. Based on an experimental evaluation with 7,000 drives from real taxi cabs, authors found that the routes delivered by techniques are within 5% of the best shortest path and have run times an order of magnitude or less compared to a naive approach.

N. Jing, Y. W. Huang and E. A. Rundensteiner, [5] Efficient path computation is essential for applications such as intelligent transportation systems (ITS) and network routing. In ITS navigation systems, many path requests can be submitted over the same, typically huge, transportation network within a small time window. While path pre computation (path view) would provide an efficient path query response, it raises three problems which must be addressed: 1) pre computed paths exceed the current computer main memory capacity for large networks; 2) disk-based solutions are too inefficient to meet the stringent requirements of these target applications; and 3) path views become too costly to update for large graphs (resulting in out-of-date query results). Propose a hierarchical encoded path view (HEPV) model that addresses all three problems. By hierarchically encoding partial paths, HEPV reduces the view encoding time, updating time and storage requirements beyond previously known path precomputation techniques, while significantly minimizing path retrieval time. Authors prove that paths retrieved over HEPV are optimal. Authors present complete solutions for all phases of the HEPV approach, including graph partitioning, hierarchy generation, path view encoding and updating, and path retrieval. In this study, they present an in-depth experimental evaluation of HEPV based on both synthetic and real GIS networks. Their results confirm that HEPV offers advantages over alternative path finding approaches in terms of performance and space efficiency.

III. PROBLEM IDENTIFICATION

In this paper we studied online shortest path computation; the shortest path result has been computed based on the live traffic circumstances. In the previous research work authors carefully analyzed and discuss their inapplicability to the problem (due to their prohibitive maintenance time and large transmission overhead). To address the problem, they suggest a architecture that broadcasts the index on the air. They first identified a feature of the hierarchical index structure which enables us to compute shortest path on a small portion of index. This important feature has been used in the solution. The LTI is a Pareto solution in terms of four performance factors for online shortest path computation but further it can be improved in the future in order to improve the system performance.

IV. CONCLUSION

The online shortest path problem aims at computing the shortest path based on live traffic circumstances. This is very important in modern car navigation systems as it helps drivers to make sensible decisions. There is no efficient system that can offer affordable costs at both client and server sides for online shortest path computation. The conventional client-server architecture scales poorly with the number of clients. A promising approach is to let the server collect live traffic information and then broadcast them over radio or wireless network. This approach has excellent scalability with the number of clients.

REFRENCES

[1] L. H. U, H. J. Zhao, M. L. Yiu, Y. Li and Z. Gong, "Towards Online Shortest Path Computation," in IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 4, pp. 1012-1025, April 2014.

[2] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes, “In Transit to Constant Time Shortest-Path Queries in Road Networks,” Proc. Workshop Algorithm Eng. and Experiments (ALENEX), 2007.

[3] Bin Jiang, "I/O-efficiency of shortest path algorithms: an analysis," Data Engineering, 1992. Proceedings. Eighth International Conference on, Tempe, AZ, 1992, pp. 12-19.

[4] N. Malviya, S. Madden and A. Bhattacharya, "A continuous query system for dynamic route planning," 2011 IEEE 27th International Conference on Data Engineering, Hannover, 2011, pp. 792-803.

[5] N. Jing, Y. W. Huang and E. A. Rundensteiner, "Hierarchical encoded path views for path query processing: an optimal model and its performance evaluation," in IEEE Transactions on Knowledge and Data Engineering, vol. 10, no. 3, pp. 409-432, May/Jun 1998.

[6] E.P.F. Chan and Y. Yang, “Shortest Path Tree Computation in Dynamic Graphs,” IEEE Trans. Computers, vol. 58, no. 4, pp. 541- 557, Apr. 2009.

[7] T. Imielinski, S. Viswanathan, and B.R. Badrinath, “Data on Air: Organization and Access,” IEEE Trans. Knowledge and Data Eng., vol. 9, no. 3, pp. 353-372, May/June 1997.

[8] J.X. Yu and K.-L. Tan, “An Analysis of Selective Tuning Schemes for Nonuniform Broadcast,” Data and Knowledge Eng., vol. 22, no. 3, pp. 319-344, 1997.

[9] A.V. Goldberg and R.F.F. Werneck, “Computing Point-to-Point Shortest Paths from External Memory,” Proc. SIAM Workshop Algorithms Eng. and Experimentation and the Workshop Analytic Algorithmic and Combinatory (ALENEX/ANALCO), pp. 26-40, 2005.

[10] M. Hilger, E. K€ohler, R. M€ohring, and H. Schilling, “Fast Point-to- Point Shortest Path Computations with Arc-Flags,” The Shortest Path Problem: Ninth DIMACS Implementation Challenge, vol. 74, pp. 41-72, American Math. Soc., 2009.

[11] A.V. Goldberg and C. Harrelson, “Computing the Shortest Path: Search Meets Graph Theory,” Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA), pp. 156-165, 2005.

[12] D. Delling and D. Wagner, “Landmark-Based Routing in Dynamic Graphs,” Proc. Sixth Int’l Workshop Experimental Algorithms (WEA), pp. 52-65, 2007.

[13] G. D’Angelo, D. Frigioni, and C. Vitale, “Dynamic Arc-Flags in Road Networks,” Proc. 10th Int’l Symp. Experimental Algorithms (SEA), pp. 88-99, 2011.

[14] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, “Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks,” Proc. Seventh Int’l Workshop Experimental Algorithms (WEA), pp. 319-333, 2008.

[15] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner, “Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra’s Algorithm,” ACM J. Experimental Algorithmics, vol. 15, article 2.3, 2010.