IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

Route-Saver: Leveraging Route APIs for Accurate and Efficient Query Processing At Location-Based Services

Ms. V.Bhagya Lakshmi , Mr.P.VIJAYA RAGHAVULU

Abstract: Location-based services (LBS) enable mobile users to query points-of-interest (e.g., restaurants, cafes) on various features (e.g., price, quality, variety). In addition, users require accurate query results with up-to-date travel times. Lacking the monitoring infrastructure for road traffic, the LBS may obtain live travel times of routes from online route APIs in order to offer accurate results. Our goal is to reduce the number of requests issued by the LBS significantly while preserving accurate query results. First, we propose to exploit recent routes requested from route APIs to answer queries accurately. Then, we design effective lower/upper bounding techniques and ordering techniques to process queries efficiently. Also, we study parallel route requests to further reduce the query response time. Our experimental evaluation shows that our solution is three times more efficient than a competitor, and yet achieves high result accuracy (above 98 percent).

1INTRODUCTION

The availability of GPS-equipped smartphones leads to a huge demand of location-based services (LBSs), like city guides, restaurant rating, and shop recommendation websites, e.g., Open Table, Hotels, and UrbanSpoon.1 They manage points-of-interest (POIs) specific to their applications, and enable mobile users to query for POIs that match with their preferences and time constraints. As an example, consider a restaurant rating website that manages a data set of restaurants P with various attributes like: location, food type, quality, price, etc. Via the LBS (website), a mobile user q could query restaurants based on these attributes as well as travel times on road network to reach them. Here are examples for a range query and a KNN query, based on travel times on road network.

Revenues. Similarly, high response time may drive users away from the LBS. Observe that the live travel times from user q to POIs vary dynamically due to road traffic and factors like rush hours,

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

congestions, road accidents. As a case study, query results (for range and KNN) would have low accuracy. Typical LBS lacks the infrastructure and resources (e.g., road-side sensors, cameras) for monitoring road traffic and computing live travel times. To meet the accuracy requirement (R1), the framework is proposed for the LBS to answer KNN queries accurately by retrieving live travel times (and routes) from online route APIs (e.g., Google Directions API which have live traffic .Given a query q, the LBS first filters POIs by local attributes in P. Next, the LBS calls a route API to obtain the routes (and live travel times) from q to each remaining POI, and then determines accurate query results for the user. As a remark, online maps on the other hand, cannot process queries for, because those queries may involve specific attributes (e.g., quality, price, facility) that are only maintained by the LBS.

Using online route APIs raises challenges for the LBS in meeting the response time requirement (R2). It is important for LBS to reduce the number of route requests for answering queries because a route request incurs considerable time (0.1-0.3 s) which is high compared to CPU time at LBS. obtains the latest travel times for queries from online route API. Though it guarantees accurate query results, it may still incur a considerable number of route requests. In this paper, we exploit an observation from, namely that travel times change smoothly within a short duration. Routes recently obtained from online route APIs may still provide accurate travel times to answer current queries. This property enables us to design a more efficient solution for processing range and KNN queries. Our experiments show that our solution is three times more efficient and yet achieves high result accuracy (above 98 percent).

Route-Saver keeps at the LBS the routes which were obtained in the past d minutes (from an online route API), where d is the expiry time parameter. For instance, we may set d to 10 minutes. These recent routes are then utilized to derive lower/upper bounding travel times to reduce the number of route requests for answering range and KNN queries. Another related work studies how to cache shortest paths for reducing the response times on answering shortest path. They mainly exploit the optimal sub path

property of shortest paths, i.e., all sub paths of a shortest path must also be shortest paths. Given a shortest path query, if both nodes s; t fall on the same (cached) shortest path, then the Shortest path from s to t can be extracted from that cached path. Unfortunately, this optimal sub path property is not powerful enough in reducing the number of route requests significantly in our problem. This is because each path contains a few data points and thus the probability for points lying on the same path with the query point is small. We show in an experiment that the optimal sub path property ('tL' in black) saves very few route requests, whereas our techniques provide the major savings in route requests. Furthermore not considered the expiry time requirement as in our work. To reduce the number of route requests while providing accurate results, we combine information across multiple routes in the log to derive tight lower/upper bounding travel times. **We also propose effective techniques** to compute such bounds efficiently. Moreover, we examine the effect of different orderings for issuing route requests on saving route requests. And we study how to parallelize route requests in order to reduce the query response time further. In the following, we first review related. Then, we describe the system architecture and our objectives. Our contributions are: _ Combine information across multiple routes in the log to derive lower/upper bounding travel times

2 RELATED WORKS

2.1 Query Processing on Road Networks

Indexing on road networks have been extensively studied in the. Various shortest path indices have been developed to support shortest path search efficiently how to process range queries and KNN queries over points-of-interest, with respect to shortest path distances on a road network. The evaluation of range queries and KNN queries can be further accelerated by specialized indices. In our problem scenario, query users require accurate results that are computed with respect to live traffic information. All the above works require the LBS to know the weights (travel times) of all road segments. Since the LBS lacks the infrastructure for monitoring road traffic, the above works are inapplicable to our problem. Some works attempt to model

the travel times of road segments as time-varying functions, which can be extracted from historical traffic patterns. These functions may capture the effects of periodic events (e.g., rush hours, weekdays). Nevertheless, they still cannot reflect live traffic information, which can be affected by sudden events, e.g., congestions, accidents and road maintenance. Landmark and distance oracle can be applied to estimate shortest path distance bounds between two nodes in a road network, which can be used to prune irrelevant objects and early detect results. The above works are inapplicable to our problem because they consider constant travel times on road segments (as opposed to live traffic). Furthermore, in this paper, we propose novel lower/upper travel time bounds derived from both the road network and the information of previously obtained routes.

2.2 Querying on Online Route APIs

Online route APIs. An online route API has access to current traffic information. It takes a route request as input and then returns a route along with travel times on the request is an HTTP query string, whose parameters contain the origin and destination locations in latitude-longitude, as well as the travel mode. In this example, the origin is at (44:94033;_93:22294), the destination is at (44:94198;_93:23722), and the user is at 'driving' mode. The response is an XML document that stores a sequence of route segments from the origin to the destination. Each segment, enclosed by <step> tags, contains its endpoints and its travel time by driving (see the <duration> tags). The segment in this example takes 8 seconds to travel. We omit the remaining segments here for brevity. Besides, the XML response contains the total travel time on this route (the sum of travel times on all segments).

Query processing algorithms. Thomsen et al. study the caching of shortest paths obtained from online route APIs. They exploit the optimal son cached paths to answer shortest path queries. As we discussed in the introduction and verified in experiments, this property cannot significantly reduce the number of route requests in our problem. Also, they have not studied the processing of range/KNN queries; the lower/upper bound techniques. It enables the LBS to process KNN

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

queries by using online route APIs. To reduce the number of route requests (for processing queries), exploits the maximum driving speed VMAX and the static road network GS (with only distance information) stored at the LBS. Upon receiving a KNN query from user q, the LBS first retrieves K objects with the smallest network distance from q and issues route requests for them. Let g be the Kth smallest current travel time (obtained so far). The LBS inserts into a candidate set C the objects whose network distance to q is within g _ VMAX. Next, groups the points in C to road junctions, utilizes historical statistics to order the road junctions, and then issues route requests for junctions in above order. However, they do not exploit the rich information of routes

that are specific in our problem. In our problem, the exact route from q to p reveals not only the current travel time to p, it may also provide the current travel times to other objects p0 on the route, and may even offer tightened lower/upper bounds of travel times to other objects.

3 PROBLEM STATEMENTS

In this section, we first describe the system architecture and then formulate the objectives of our problem. System architecture and notations. In this paper, we adopt the system architecture as depicted. It consists of the following entities: Online Route API. Examples are: Google/Bing route APIs. Such API computes the shortest route between two points on a road network, based on liv. It has the latest road network G with live travel time information. _ Mobile User. Using a mobile device (smartphone), the user can acquire his current geo-location q and then issue queries to a location-based server. In this paper, we consider range and KNN queries based on live traffic. _ Location-Based Service/Server. It provides mobile users with query services on a data set P, whose POIs (e.g., restaurants, cafes) are specific to the LBS's application. The LBS may store a road network G with edge weights as spatial distances, however G cannot provide live travel times. In case P and G do not fit in main memory, the LBS may store P as an R-tree and store the G as a disk-based adjacency list .We then define route, travel time, and queries formally.

4 QUERY PROCESSING

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

This section presents our approach Route-Saver for answering queries efficiently. First, we discuss the maintenance of the time-tagged road network G and the route log. Finally, we discuss the applicability of our techniques when no local maps are available. In subsequent discussion, we drop the subscript t in as we only use valid routes.

4.1 Maintenance of Structures at LBS

Conservative travel time bounds. Given an edge Observe that the lower-bound is limited by the euclidean distance of e and the maximum driving speed.

VMAX: (1) On the other hand, the upper-bound ¼ 1 because the travel time one can be arbitrarily long in case of traffic congestion. Structures. We employ a route log L and a time-tagged network G in the LBS. The route log L stores all routes obtained from an online route API within the last d time units, as described. Recall from that the timestamp of a route indicated by its subscript t. Assume that we use d ¼ 2 in Fig. 4a. At time tow ¼ 4, L keeps the routes obtained during time 2-4. To support query operations efficiently,

4.2 Exact Travel Times and Their Bounds In this section, we exploit the time-tagged road network G and the route log L to derive lowers and upper bounds of travel times for data points. As we will elaborate soon, these bounds enable us to save route requests during query processing. L contains only valid routes (not yet expired). For the time tagged network G, solid edges are valid while dotted edges are not.

4.3 Range Query Algorithm

In this section, we present our Route-Saver algorithm for processing a range query. It applies the travel time bounds discussed above to reduce the number of route requests. To guarantee the accuracy of returned results, it removes all expired routes CT in L. The algorithm first conducts a distance range search to obtain a set C of candidate points. Two phases to process the candida

4.4 KNN Query Algorithm

we extend our Route-Saver algorithm for processing KNN queries. We will also examine suitable orderings for processing candidates. Unlike range queries, KNN

queries do not have a (fixed) travel time limit T for obtaining a small candidate set. Instead, we first compute a (temporary) result set R so that it contains K candidates with the smallest. Recall that we can obtain these bounds/values for all candidates efficiently by traversal on G. Let g be the largest R. Having this value g, we can prune each candidate p that satisfies, as it cannot become the result. The pseudo-code of our KNN algorithm. First, we initialize the candidate set C with the data set P, insert K dummy pairs (with 1 travel time) into the result set R, and set g to the largest travel time in R. The algorithm consists of three phases. In the first phase, it obtains g by using the idea discussed above. In the second phase, it prunes candidates whose lower bounds or exact times are larger than g. In the third phase, it examines the candidates according to a certain order and issues route requests for them. The algorithm terminates when the candidate set contains exactly K objects, and then reports them as query results.

4.5 Applicability of Techniques without Map

In this section, we discuss how to adapt the Route-Saver in case the LBS cannot obtain the same map G used in the route service. We observe that, if the LBS uses the map G0 which are not the same with that used in route services, bounding travel times G can be over-estimated. For example, if the real shortest path from q top is missing local map G0, then it is possible that Route-Saver calculates a higher G for p and mistakenly prunes it from results. Therefore, the LBS are not allowed to use inaccurate maps. In case that the LBS cannot access to the map G used.

5 PARALLELIZED ROUTE REQUESTS

Our objective is to minimize the response time of queries. Optimizes the response time through reducing the number of route requests. In this section, we examine how to parallelize route requests in order to optimize user response time further. We propose two parallelization techniques that achieve different tradeoffs on the number of route requests and user response time. The execution of algorithms follows a sequential schedule. The user response time (i) the time

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

spent on route requests and (ii) local computation at the LBS (in white). Consider the sequential schedule in. An experiment reveals that the user response time is dominated by the time spent on route requests. Let a slot be the waiting period to obtain a route from the route API.2 the sequential schedule takes five slots for five route requests. Intuitively, the LBS may reduce the number of slots by issuing multiple route requests to a route API in parallel. a parallel schedule with two slots; each slot contains three route requests issued in parallel. Although parallelization helps reduce the response time, it may prevent sharing among routes and because extra route requests as we will explain later. Existing parallel scheduling techniques have not. Exploited this unique feature in our problem. We also want to avoid extra route requests because a route API may impose a daily route request or charge the LBS based on route requests. We proceed to present two parallelization techniques. They achieve different tradeoffs on the number of route requests and the number of slots. Our discussion focuses on range queries only. Our techniques can be extended to KNN queries as well. Greedy parallelization. Let me be the number of threads for parallel execution (per query). Our greedy parallelization approach dispatches route request to a thread as soon as it becomes available. Specifically, we modify as follows. Instead of picking one object p from the candidate set C we pick m candidate objects and assign their route requests to m threads in parallel.

6 EXPERIMENTAL EVALUATION

In this section, we compare the accuracy and the performance of our Route-Saver (abbreviated as RS) with an existing method SMashQ. lthough SMQ handles only KNN queries, we also adapt it to process range queries. Note that SMQ does not utilize any route log to save route requests. We also consider an extension of SMQ, called SMQ_, which keeps the routes within expiry time into a route log. SMQ_ applies only the optimal sub path property and retrieves exact travel times from the log; however, it does not apply the upper/lower bounding techniques in this paper. By default, RS uses the DESC and DIFF

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

orderings for range and KNN queries r pectively.

6.1 Experimental Setting Road networks.

For accuracy experiments on real traffic data, we will discuss the road network and traffic data For the performance and scalability study, we obtain three road maps in USA its number of route requests and user response time (summarizes the default values and ranges of parameters used in our experiments. The values for data set size. The default expiry time d is 10 minutes, according. To simulate the arrival of queries, we set the default query rate _ to 60 queries/ min and uniformly generate query points on the road network. This query rate (60 queries/min) is justified by visit statistics3 from restaurant and travel guide websites . All methods were implemented in C++ and ran on an Ubuntu 11.10 machine with a 3.4 GHz Intel Core i7-3770 processor and 16 GB RAM. In experiments, the route log contains at most 30,000 routes and occupies at most 30 MB. The largest road network (Florida) and data set occupies 87 and 1 MB respectively. Thus, the largest map, route log, anddata set can fit in the main memory.

6.2 Accuracy on Real Traffic Data

In this section, we test the result accuracy of the methods on real traffic data, for various expiry time d (2, 5, 10, 20 and 30 minutes). Other parameters are set to default values. Real traffic data. We downloaded historical real traffic on freeways in Los Angeles from PeMS.4 The corresponding road network contains 17,563 nodes and 17,694 edges. the travel times on edges are updated every 30 seconds. We also conduct this experiment with traffic data on other dates, and obtain similar results.Accuracy measure. Besides the methods discussed before, we also consider a baseline method which uses only local distance information to answer queries, without issuing route requests.

6.3 Performance and Scalability Study

For the sake of obtaining the user response time in our simulations, we measure the time of route requests on Google Directions API. On each roadmap, we randomly sample 400 pairs of points and issue route requests for

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

them to Google Directions API. plots the time of each route request versus its length (exact travel time), on the Erie roadmap. summarizes the average and standard deviation of route request time on all roadmaps.

6.3.1 Temporal Stability

In this section, we simulate the arrival of queries along a 60- minute (1-hour) timeline, while fixing all parameters to default. Thus, each test uses 60 _ _ ¼ 3;600 queries. The route log L is initially empty. To report temporal behavior, we measure (i) the route log size and (ii) the number of route requests of each query. We first conduct experiments with uniformly distributed queries and data sets. the number of routes in L of RS and SMQ_ versus the timeline, for range queries. SMQ is not plotted here as it does not utilize the log L.

6.3.2 Effect of Optimization Techniques

First, we investigate the effectiveness of our proposed lower/upper bound techniques. Recall that RS exploits the travel time information obtained from recent routes for three techniques: (i) retrieve the exact travel time of a point p, (ii) prune p by its lower bound; p:t_I and (iii) detect p as a true hit by its upper bound.

6.3.3 Scalability Experiments

As discussed before, in this section, we simulate the arrival of queries along 2d-minute time interval. And we measure the performance in terms of: (i) average number of route requests per query in the stable period, and (ii) average user response time per query in the stable.

6.4 Experiments on Google Directions API

We have implemented SMQ, SMQ_ and RS with Google Directions request/response format has been described. Due to the daily request limit (2,500) for evaluation users .we conduct this experiment on the Manhattan region .We randomly select 100 POIs5 in this region, and generate 100 queries (along a 100-second time period). depicts the number of route requests of each query versus the timeline, for range queries and KNN queries. RS outperforms SMQ and SMQ_ on both range queries and KNN queries.

7 CONCLUSION

In this paper, we propose a solution for the LBS to process range/KNN queries such that the query results have accurate travel times and the LBS incurs few number of route requests. Our solution Route-Saver collects recent routes obtained from an online route API (within d minutes). During query processing, it exploits those routes to derive effective lower-upper bounds for saving route requests, and examines the candidates for queries in an effective order. We have also studied the parallelization of route requests to further reduce query response time. Our experimental evaluation shows that Route-Saver is 3 times more efficient than a competitor, and yet achieves high result accuracy (above 98 percent). In future, we plan to investigate automatic tuning the expiry time d based on a given accuracy requirement. This would help the LBS guarantee its accuracy and improve their users' satisfaction.

8 REFERENCES

[1] 2011 Census TIGER/Line Shapefiles. (2011).[Online].Available:http://www.census.gov/cgi-bin/geo/shapefiles2011/main

[2] 9th DIMACS Implementation Challenge on Shortest Paths.(2013).[Online]. Available:http://www.dis.uniroma1.it/challenge9/ data/tiger/

[3] Bing Data Suppliers. (2013). [Online]. Available:http://windows.microsoft.com/en-HK/windows-live/about-bing-data-suppliers/

[4] Bing Maps API. (2013). [Online]. Available: http://www.microsoft.com/maps/developers/web.aspx

[5] Bing Maps Licensing and Pricing Information. (2013). [Online]. Available: http://www.microsoft.com/maps/product/licensing. aspx

[6] Google Directions & Bing Maps: Live Traffic Information.(2013).[Online]. Available: http://support.google.com/maps/bin/answer.py?hl=en&answer=2549020&topic=1687356&ctx=topic http://msdn.microsoft.com/en-us/library/aa907680.aspx

[7] Google Directions API. (2013). [Online]. Available:https://developers.google.com/maps/documentation/directions/

[8] Google Directions API Usage Limits. (2013).[Online].Available:https://developers .google.com/maps/faq#usagelimits

[9] Google Map Maker Data Download. (2013).[Online].Available:https://services.go ogle.com/fb/forms/mapmakerdatadownload/

[10] OpenStreetMap. (2013). [Online]. Available: http://www. openstreetmap.org/

[11] Statistics of Usage. (2013). [Online]. Available: http://www. quantcast.com

[12] US Maps from Government. (2013). [Online].Available:http://www.usgs.gov/pub prod/

[13] N. Bruno, S. Chaudhuri, and L. Gravano, "STHoles: A multidimensional workload-aware histogram," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2001, pp. 211–222.

[14] E. P. F. Chan and Y. Yang, "Shortest path tree computation in dynamic graphs," IEEE Trans. Comput., vol. 58, no. 4, pp. 541–557, Apr. 2009.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms. Cambridge, MA, USA: MIT Press, 2009.

[16] U. Demiryurek, F. B. Kashani, C. Shahabi, and A. Ranganathan, "Online computation of fastest path in time-dependent spatial networks," in Proc. 12th Int. Symp. Adv. Spatial Temporal Databases, 2011, pp. 92–111.

[17] A. Dingle and T. Partl, "Web cache coherence," Comput. Netw., vol. 28, pp. 907–920, 1996.

[18] M. Drozdowski, Scheduling for Parallel Processing, 1st ed. New York, NY, USA: Springer, 2009.

[19] H. Hu, D. L. Lee, and V. C. S. Lee, "Distance indexing on road networks," in Proc. 32nd Int. Conf. Very Large Data Bases, 2006, pp. 894–905.

[20] S. Jung and S. Pramanik, "An efficient path computation model for hierarchically structured topographical road maps," IEEE Trans. Knowl. Data Eng., vol. 14, no, 5, pp. 1029–1046, Sep./Oct. 2002.

[21] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding fastest paths on a road

network with speed patterns," in Proc. Int. Conf. Data Eng., 2006, p. 10.

[22] M. Kolahdouzan and C. Shahabi, "Voronoi-based K nearest neighbor search for spatial network databases," in Proc. 30th Int. Conf. Very Large Data Bases, 2004, pp. 840–851.

[23] H.-P. Kriegel, P. Kr€oger, P. Kunath, and M. Renz, "Generalizing the optimality of multi-step k -nearest neighbor query processing," in Proc. 10th Int. Symp. Adv. Spatial Temporal Databases, 2007, pp. 75–92.

[24] H.-P. Kriegel, P. Kr€oger, M. Renz, and T. Schmidt,, "Hierarchical graph embedding for efficient query processing in very large traffic networks," in Pro

[25] H.-P. Kriegel, P. Kr€oger, M. Renz, and T. Schmidt, "Proximity queries in large traffic networks," in Proc. 15th Annu. ACM Int. Symp. Adv. Geographic Inform. Syst., 2007, p. 21.

[26] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in Proc. 29th Int. Conf. Very Large Data Bases, 2003, pp. 802–813.

[27] M. Qiao, H. Cheng, L. Chang, and J. X. Yu, "Approximate shortest distance computing: A query-dependent local landmark scheme," in Proc. IEEE 28th Int. Conf. Data Eng., 2012, pp. 462–473.

[28] H. Samet, J. Sankaranarayanan, and H. Alborzi, "Scalable network distance browsing in spatial databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2008, pp. 43–54.

[29] J. Sankaranarayanan and H. Samet, "Distance oracles for spatial networks," in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 652–663.

[30] T. Seidl and H.-P. Kriegel, "Optimal multi-step k-nearest neighbor search," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1998, pp. 154–165.

[31] J. R. Thomsen, M. L. Yiu, and C. S. Jensen, "Effective caching of shortest paths for location-based services," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2012, pp. 313–324.

[32] D. Zhang, C.-Y. Chow, Q. Li, X. Zhang, and Y. Xu, "Efficient evaluation of k-NN queries using spatial mashups," in

Proc. 12th Int. Conf. Adv. Spatial Temporal Databases, 2011, pp. 348–366.

Author's Details

.P.VIJAY received M.Tech(CSE) Degree from School of Information Technology, Autonomous, and Affiliated to JNTUH, Hyderabad. He is currently working as Assistant Professor in the Department of Computer Science and Engineering in Modugula Kalavatamma Institute of Technology for Women, Anathapur,Andhra Pradesh India. His interests includes Object Oriented Programming, Operating System, Database Management System, Computer Networking, Cloud Computing and Software Quality Assurance.

Ms. V.Bhagya Lakshmi B.Tech Degree from Modugula Kalavatamma Institute of Technology for Women,anathapur. SHe is currently pursuing M.tech Degree in Computer Science and Engineering specialization in Modugula Kalavatamma Institute of Technology for Women,Anathapur,AndhraPradesh.