Predictive Query Processing On Moving Objects

Abdeltawab M. Hendawi
Expected Graduation: Summer 2014/15
Supervised by: Mohamed F. Mokbel
Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA
`{hendawi, mokbel}@cs.umn.edu`

***Abstract*—A fundamental category of location based services relies on predictive queries which consider the anticipated future locations of users. Predictive queries attracted the researchers' attention as they are widely used in several applications including traffic management, routing, location-based advertising, and ride sharing. This paper aims to present a generic and scalable system for predictive query processing on moving objects, e.g, vehicles. Inside the proposed system, two frameworks are provided to work on two different environments, (1) *Panda* framework for Euclidean space, and (2) *iRoad* framework for road network. Unlike previous work in supporting predictive queries, the target of the proposed system is to: (a) support long-term query prediction as well as short term prediction, (b) scale up to large number of moving objects, and (c) efficiently support different types of predictive queries, e.g., predictive range, KNN, and aggregate queries.**

I. INTRODUCTION

The fact that there are more than one billion smart phones [4] triggered the massive explosion of location based services [5], [9], [14], [18]. An important category of these services offers facilities based on the future location of a user rather than his/her location in the present time. Spatial queries in this categories come under the umbrella of predictive queries [8], [10], [11], where a service is supplied according to the predicted location of a user after some time in the future. Common types of predictive spatial queries include *predictive range* query, e.g., "find all hotels that will be located within two miles of a user's anticipated location after 30 minutes", *predictive KNN* query, e.g., "find the three taxis that most likely to pass by my location in the next 10 minutes", and *predictive aggregate* query, e.g., "how many cars expected to be around the stadium during the next 20 minutes".

In fact, *Predictive* queries can be employed in various types of real applications such as (1) traffic management, to predict areas with high traffic in the next half hour, so appropriate decisions can be taken before congestion appears, (2) location-aware advertising, to distribute coupons and sales promotions to customers more likely to show up around a certain store during the sale time in the next hour, (3) routing services, that take into consideration the predicted traffic on each road to find the shortest path for a user trip starting after 15 minutes from the present time, (4) ride sharing systems, to get the drivers that mostly will pass by a rider's location within few minutes, (5) store finders, to predict the closest restaurants to a user's route after half hour, (6) emergency response, to alert the three police cars expected to be the nearest to a stolen car in a couple of minutes.

The goal of this PhD thesis is to enable the practical realization of location-based services such that they can support common types of predictive queries on spatio-temporal data, i.e., moving objects. Therefore, this thesis proposes a generic and scalable predictive spatial query processing system. Inside this system, two different frameworks are introduced, namely, *Panda* framework [7] and *iRoad* framework [6]. Each one is customized according to the underlying work environment, euclidean space and road network graph, respectively.

Specifically, this thesis handles the following core challenges in processing of spatial predictive queries:

- **Prediction**. Unlike most of the existing related work [11], [19] that supports short-term prediction only, the proposed frameworks have the ability to evaluate long-term as well as short-term predictive queries. In addition, the employed prediction models do not rely on the historical data, as in many cases it is hard to obtain the historical data of the moving objects. For example, in new systems that there is no historical data or in confidential systems where the data are top secrets or at least private so it can not be released to the prediction model.
- **Salability**. The proposed frameworks can scale up to support heavy query workloads on a space with a large number of moving objects. The scalability of *Panda* is resulted from adjusting the underlying prediction function to be employed to filter out the objects having no possibility to show up in the query region at the specified time. This filtering saves a lot of the processing time for each single query. While the scalability of *iRoad* comes from introducing a novel data structure named *reachability tree* to prune the space around each object. Yet, it holds only those nodes, road intersections, reachable within a specified time period from the object current location.
- **Efficiency**. The goal here is to introduce an efficient query processing engine that utilizes the prediction of each object in the underlying space to answer the predictive queries in very fast response time. Thus, users do not have to wait to get the answer to their queries.
- **Generality**. The introduced solution can support the processing of many kinds of predictive queries including *predictive range* query, *predictive KNN* query, *predictive aggregate* query, and *predictive point* query. This is done

The work of this paper is supported in part by the National Science Foundation under Grants IIS-1218168 and IIS-0952977.

inside the running framework and using the same data structures and algorithms.

The main idea of the introduced system is to monitor those space areas that are highly accessed using predictive queries. For such areas, the system precomputes the prediction of objects being in these areas beforehand. Whenever a predictive query is received, the system checks if parts of this predictive query are included in those precomputed space areas. If this is the case, the system retrieves parts of its answer from the precomputed areas with a very low response time. For other parts of the incoming predictive query that are not included in the precomputed areas, the system has to dispatch the full prediction module to find out the answer, which will take more time to compute. It is important to note here that the aim is predict the answer for certain areas of the space rather than the whole space. Then, the overlap between the incoming query and the precomputed areas controls how efficient the query would be. This isolation between the precomputed area and the query area presents the main reason behind the generic nature of the proposed system as any type of predictive queries (e.g., range and kNN) can use the same precomputed areas to serve its own purpose.

The rest of this paper is organized as follows. Section II studies the related work. Section III gives an overview of the system architecture. The *Panda* and the *iRoad* frameworks are discussed in Section IV and Section V respectively. The system prototype and the experimental evaluation is provided in Section VI and Section VII. Finally, Section VIII concludes the paper.

II. RELATED WORK

In this section, we review the existing work for predictive query processing on moving objects. Existing techniques for predictive query processing can be classified according to the supported query type into the following categories:

(1) *Predictive range queries*, i.e., [11], [17], [20]. A predictive range query has a query region R and a future time t, and asks about the objects expected to be inside the R after time t. For example, a network mobility model [11] is used to predict the coming path of each of the underlying objects and employ the prediction results to evaluate predictive range queries.

(2) *Predictive k-nearest-neighbor queries*, i.e., [2], [15], [20]. A predictive K-nearest-neighbor query has point location P, a future time t, and asks about the K objects expected to be closest to P after time t. For example, two algorithms, RangeSearch, KNNSearchBF, [20] are introduced to traverse spatio-temporal index tree (TPR/TPR*-tree) to find the nodes that intersect with the query circular region for Range and KNN queries respectively.

(3) *Predictive aggregate queries*, i.e., [1], [7], [16]. A predictive aggregate query has a query region R and a future time t, and asks about the number of objects $\mathcal{N}$ predicted to be inside R after time t. For example, a comprehensive technique [16] provides an approximate answer for aggregate spatio-temporal queries for the future addition to the past, and the present. An integrated system [1] is used to perform predictive analysis on aviation data for air traffic management system. For more comprehensive study on the existing work in predictive spatio-temporal queries in general, we refer the reader to [8].

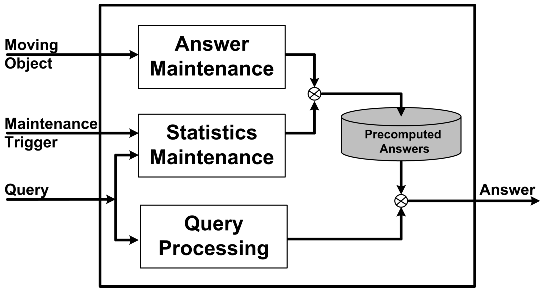


Fig. 1. The System Architecture

To summarize, the proposed system differentiates itself from existing related work is that it is the only work that can support predictive spatio-temporal query processing for both euclidean space and road networks. In addition, within its infrastructures including algorithms and data structures, the common types of predictive query, e.g., predictive *range*, *KNN*, and *aggregate*, can be efficiently evaluated. Further, the embedded prediction models do not depend on historical data to perform the prediction of objects future locations. Moreover, it provides a smooth scalable behavior to harmonize the realistic needs of large number of moving objects and massive query workloads.

III. SYSTEM ARCHITECTURE

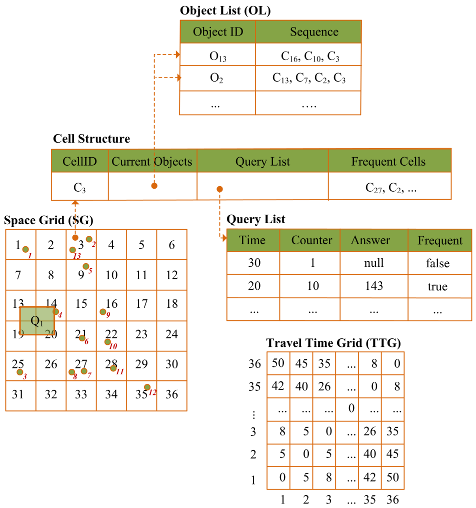
This section give the overall system overview for our proposed predictive query processing system. We briefly describe its main idea, and outline the system architecture and the key modules.

Figure 1 gives the system architecture which includes three main modules, namely, answer maintenance, statistics maintenance, and query processing. Each module is dispatched by an event, namely, an object movement, a trigger for statistic maintenance, and a query arrival, respectively. The system maintains a storage for precomputed answers, which is updated according to the objects movements and used to construct the final query answer for arriving queries. Below is a brief overview of the actions taken by the system for each event.

Object movement. Whenever the system receives an object movement, it dispatches the answer maintenance module to check if this movement affects any of the precomputed answers. If this is the case, the affected precomputed answers are updated accordingly.

Maintenance trigger. Based on a tunable threshold, a trigger may be fired to alert the system that the current set of statistics that judge on which answers to precompute need to be reset. The updated statistics affect which parts of query answers will be precomputed.

Query arrival. Once a query is received, the query processor divides the query area into two parts based on the answer precomputation. The first part is already precomputed where



Object List (OL)

Object ID	Sequence
O_{13}	C_{16}, C_{10}, C_3
O_2	C_{13}, C_7, C_2, C_3
...	....

Cell Structure

CellID	Current Objects	Query List	Frequent Cells
C_3			C_{27}, C_2, ...

Space Grid (SG)

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

Query List

Time	Counter	Answer	Frequent
30	1	null	false
20	10	143	true
...	...	...	...

Travel Time Grid (TTG)

36	50	45	35	...	8	0
35	42	40	26	...	0	8
⋮	...	...	...	0	...	...
3	8	5	0	...	26	35
2	5	0	5	...	40	45
1	0	5	8	...	42	50
	1	2	3	...	35	36

Fig. 2. Data Structures in *Panda*

its answer is just retrieved from the precomputed storage. The second part is not precomputed and needs to be evaluated from scratch through the computation of the prediction function against a candidate set of moving objects.

IV. Panda: Predictive Query Processing In Euclidian Space

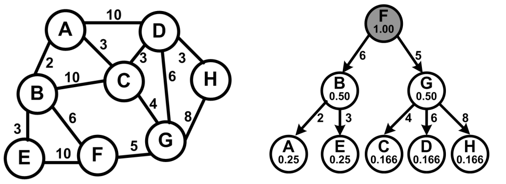
This section introduces of the *Panda* framework for predictive query processing for moving objects in euclidean space. We briefly overview the system and describe its basic data structures and explain how to handle each event. objects movements to precompute the predicted answer.

A. Data Structure

Figure 2 depicts the underlying data structure used by *Panda*. A brief overview of each data structure is outlined as follows. **Space Gird** SG. *Panda* partitions the whole space into $N \times N$ grid cells. For each cell $C_i \in SG$, we maintain: (1) *CellID* as an identifier, (2) *Current Objects* as the list of moving objects located inside C_i, (3) *Query List* as the list of predictive queries issued on C_i. (4) *Frequent Cells* as the list of cells that one of their precomputed answers should be updated with the movement of an object in C_i. **Object List** OL. This is a list of all moving objects in the system. **Travel Time Grid** TTG. This is a two-dimensional array of $N^2 \times N^2$ cells where each cell $TTG[i, j]$ has the average travel time between space cells C_i and C_j, where C_i and $C_j \in SG$.

B. Object movement

Basically, when an object moves, *Panda* checks if this movement has any effect on any of the precomputed answers. If this is the case, then *Panda* computes this effect by applying the prediction function to this object, Equation 1, then propagates it to precomputed answers in the all affected cells. By doing this, we either add this object along with its probability to the list of predicted objects at possible destination cells or remove it from the cells no longer possible destination to that object.



(a) Road Network (b) Tree of Node F

Fig. 3. Reachability Tree and Probability Assignment in iRoad

The long-term prediction function deployed in *Panda* is mainly an adaptation of the one introduced by Krumm [3], [12] to predict the final destination of a single object. Yet, the prediction model inside *Panda* is able to compute the probability that object O will be passing by the given cell C_i after time t, where t is specified in the predictive query. The numerator is the output of the original prediction function, and the denominator is the summations of the probabilities of all grid cells D_t that could be a possible destination of an object O after time t. D_t is the set of possible destinations of object O after time t.

$$P(C_i|S_o, t) = \frac{P(C_i|S_o)}{\sum_{d \in D_t} P(C_d|S_o)} \tag{1}$$

C. Maintenance trigger

This module runs periodically each t units to sweep over current statistics that decide which parts to precompute beforehand and update it. For a query Q to be considered for precomputation, it has to appear at least a number of times above a certain threshold in the last time period t. Otherwise, it will be computed at the time it is received.

D. Query arrival

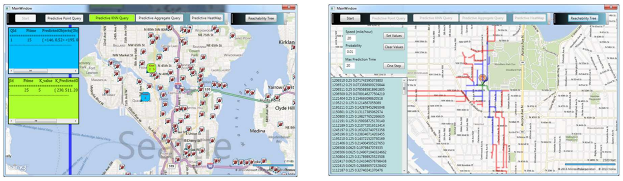
Upon the arrival of a new predictive spatio-temporal query Q, with an area of interest R, requesting a prediction about future time t, *Panda* first divides Q into a sets of grid cells C_f that overlap with the query region of interest R. For each cell $c \in C_f$, *Panda* does the following. Initially, it gets the query result from cell c, if it is already precomputed, otherwise it computes the result from scratch. Then, it maintains a set of statistics that help in deciding whether the answer of cell c, for a future time t, should be precomputed or not.

V. iRoad: Predictive Query Processing In Road Networks

This section shortly presents the *iRoad* framework for processing predictive queries for objects traveling on road networks.

A. Data Structures

In *iRoad*, there are three basic data structures to maintain: **Road Network Graph**. The given road network graph contains a set of nodes *N* and edges *E*, and the weights *W* of the edges represent the travel time. For each node *n* in the road network, we store additional information including: (1) list of *current objects*, to hold the moving objects that are currently around *n*, (2) list of *predicted objects* to carry the



(a) User View (Queries) (b) Internal View (Trees)

Fig. 4. The System User Interface

precomputed results. For each object at a node *n* in the road network graph we maintain the *current trip* list which holds the visited nodes by the object in its ongoing trip. **Reachability Tree**. A reachability tree uses a node n in the road network as a root, and stores all the reachable destinations, i.e., other nodes, based on the shortest paths from n within a determined time limit $\mathcal{T}$, i.e., 15 minutes, Figure 3(b). Yet, reachability tree is employed to handle objects movements such that we precompute and update the list of predicted objects at each node in the road network beforehand.

B. Object movement

The *iRoad* framework employs a novel data structure, named *reachability tree*, to hold the nodes reachable within a certain time frame $\mathcal{T}$ from an object current location. According to the movements of the underlying objects o, we leverage the *reachability trees* to precompute and store the predicted answer at each node in the underlying road network graph.

The prediction model employed by *iRoad* framework is based on the assumption that objects follow the shortest paths in their trips. The intuitions behind this assumption is based on the fact that in the most cases, the objects moving on road networks, e.g., vehicles, travel through shortest routes to their destinations [12], [13]. A probability value is assigned to each node in the object reachability tree, Equation 2, such that, nodes closer to object current node have higher probabilities than the far away ones. The probability of a node n_i being a destination to the object *o*, where n_i is a node in the *reachability tree* of *o* based on its current location n_o, is equal to the probability of n_j , parent node of n_i, being a destination to *o* divided by the number of children of n_j, Figure 3.

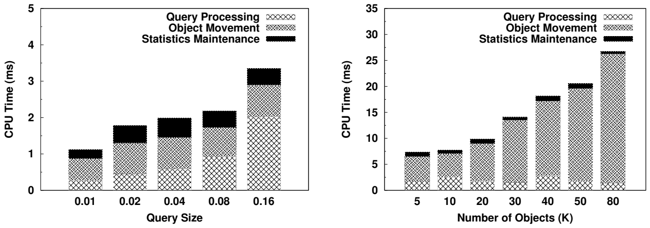
$$P(n_i|n_o) = \prod \frac{P(n_j|n_o)}{fanout(n_j)} \tag{2}$$

C. Maintenance trigger

This event does not apply in the *iRoad* framework, as there is no partial precomputation in it.

D. Query arrival

The idea of processing predictive queries in *iRoad* is to have the list predicted objects at each node precomputed and maintained in advance as a res by the movement handler module, so for coming queries, the query processor module fetches those results, adapts them according to the type of received query and returns the answer in a very fast response time.



(a) Large Sized Queries (b) Large Number Of Objects

Fig. 5. System Efficiency and Scalability

VI. SYSTEM PROTOTYPE

A nice graphical user interface is developed to allow end users to issue queries and visually inspect the results. As shown in Figure 4(a), users can issue different types of predictive queries including predictive point, range, KNN, and aggregate queries. Then the system responds by the list of objects predicted to show up at the query location after the specified future time along with objects probabilities. Also users will have an eye on the system inside by seeing how reachability trees are constructed and dynamically change according the objects movements, Figure 4(b).

VII. EXPERIMENTAL PERFORMANCE EVALUATION

This section illustrates the performance evaluation for the *Panda* framework, while the experimental evaluation for the *iRoad* is still under completion. The experiments given here focuses on the study of the scalability of the proposed system with large query sizes. In this set of experiments, Figure 5.(a), we notice that when the size increases, our system still behaves efficiently without significant increase in the total processing cost, for example when the query size increased by 16 times, from 0.01 to 0.16 of the total space, the average processing time per query increases only by three times, from 0.11 to 0.34 milliseconds/query. In our second set of scalability experiments, Figure 5.(b) depicts the behavior of the main components of the system when the number of moving objects increases from 5K to 80K. As noticed from the average CPU cost per query when the number of objects increases by 16 times, from 5K to 80K, the average cost per query increases only by less than four times, i.e., from 0.7 to 2.7 milliseconds/query.

VIII. CONCLUSION

This paper outlines a PhD thesis that proposes frameworks for predictive query processing for moving objects in two different environments, *Panda* for objects in euclidean space, and *iRoad* for objects on road networks. For each framework, we presented its main idea, architecture, and the embedded data structures. Three core challenges, necessary to realizing the introduced frameworks, along with the proposed approaches to solving these challenges were presented. These challenges include (1) Supporting long-term query prediction, many steps in the future, as well as short term prediction, next destination or step, (2) Scaling up to large number of moving objects, and large number of outsized predictive queries, (3) Supporting common types of predictive spatio-temporal queries including range, aggregate, and k-nearest-neighbor queries. Experimental evidence was given to prove the scalability and efficiency of the presented frameworks. As a future work, we plan to study how to deal with uncertainty in the underlying moving objects locations and directions, and to support different prediction models within the proposed framework.

REFERENCES

[1] S. Ayhan, J. Pesce, P. Comitz, G. Gerberick, and S. Bliesner. Predictive Analytics with Aviation Big Data. In *BigSpatial*, California, USA, Nov. 2012.

[2] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest and Reverse Nearest Neighbor Queries for Moving Objects. *VLDB Journal*, 15(3):229–249, 2006.

[3] J. Froehlich and J. Krumm. Route Prediction from Trip Observations. In *SAE*, Michigan, USA, Apr. 2008.

[4] GO-Gulf. Smartphone users around the world statistics and facts, Jan. 2012.

[5] Y. Gu, G. Yu, N. Guo, and Y. Chen. Probabilistic Moving Range Query over RFID Spatio-temporal Data Streams. In *CIKM*, pages 1413–1416, Hong Kong, China, Nov. 2009.

[6] A. M. Hendawi, J. Bao, and M. F. Mokbel. iRoad: A Framework For Scalable Predictive Query Processing On Road Networks. In *VLDB*, Riva Del Garda, Italy, Aug. 2013.

[7] A. M. Hendawi and M. F. Mokbel. Panda: A Predictive Spatio-Temporal Query Processor. In *ACM SIGSPATIAL GIS*, California, USA, Nov. 2012.

[8] A. M. Hendawi and M. F. Mokbel. Predictive Spatio-Temporal Queries: A Comprehensive Survey and Future Directions. In *MobiGIS*, California, USA, Nov. 2012.

[9] H. Hu, J. Xu, and D. L. Lee. A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects. In *SIGMOD*, pages 479–490, Maryland, USA, June 2005.

[10] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A Hybrid Prediction Model for Moving Objects. In *ICDE*, pages 70–79, Cancn, Mxico, Apr. 2008.

[11] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Path Prediction and Predictive Range Querying in Road Network Databases. *VLDB Journal*, 19(4):585–602, Aug. 2010.

[12] J. Krumm. Real Time Destination Prediction Based on Efficient Routes. In *SAE*, Michigan, USA, Apr. 2006.

[13] J. Krumm, R. Gruen, and D. Delling. From Destiantion Prediction To Route Prediction. *Technical Report. Microsoft Research*, 2011.

[14] M. F. Mokbel, X. Xiong, M. A. Hammad, and W. G. Aref. Continuous Query Processing of Spatio-temporal Data Streams in PLACE. *GeoInformatica*, 9(4):343–365, Dec. 2005.

[15] K. Raptopoulou, A. Papadopoulos, and Y. Manolopoulos. Fast Nearest-Neighbor Query Processing in Moving-Object Databases. *GeoInformatica*, 7(2):113–137, June 2003.

[16] J. Sun, D. Papadias, Y. Tao, and B. Liu. Querying about the Past, the Present, and the Future in Spatio-Temporal. In *ICDE*, pages 202–213, MASSACHUSETTS, USA, Mar. 2004.

[17] Y. Tao, C. Faloutsos, D. Papadias, and B. L. 0002. Prediction and Indexing of Moving Objects with Unknown Motion Patterns. In *SIGMOD*, pages 611–622, Paris, France, June 2004.

[18] H. Wang, R. Zimmermann, and W.-S. Ku. Distributed Continuous Range Query Processing on Moving Objects. In *DEXA*, pages 655–665, Krakow, Poland, Sept. 2006.

[19] R. Zhang, H. V. Jagadish, B. T. Dai, and K. Ramamohanarao. Optimized Algorithms for Predictive Range and KNN Queries on Moving Objects. *Information Systems*, 35(8):911–932, Dec. 2010.

[20] R. Zhang, H. V. Jagadish, B. T. Dai, and K. Ramamohanarao. Optimized Algorithms for Predictive Range and KNN Queries on Moving Objects. *Information Systems*, 35(8):911–932, Dec. 2010.