Query Optimization for Spatio-temporal Data Stream Management Systems*

Hicham G. Elmongui†
Department of Computer Science
Purdue University
305 N. University St
West Lafayette, IN 47907, USA
elmongui@cs.purdue.edu

ABSTRACT

Location-detection devices are used ubiquitously in moving objects due to the everyday decreasing cost and simplified technology. Usually, these devices will send the moving objects' location information to a spatio-temporal data stream management system that will be then responsible for answering spatio-temporal queries related to these moving objects. Most of the existing work focused on the continuous spatio-temporal query execution. However, several outstanding challenges have been either addressed partially or not at all in the existing literature. In this paper, we focus on the optimization of multi-predicate spatio-temporal queries on moving objects. In particular, we provide a costing mechanism for continuous spatio-temporal queries. We provide for the optimization of the parameters of the spatio-temporal operators. Finally, we propose the adaptive execution of the continuous queries for spatio-temporal data stream management systems.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query processing*

General Terms

Algorithms, Management

Keywords

Spatio-Temporal, Query Optimization, Multi-Predicate

*This work is supported in part by the National Science Foundation under Grant Number IIS-0811954.

†The author is also affiliated with Alexandria University, Alexandria, Egypt.

The SIGSPATIAL Special, Volume 1, Number 1, March 2009

1. INTRODUCTION

In recent years, the technology of location-detection devices became substantially cheaper to the point that this technology is being ubiquitously incorporated in moving objects. Not only we may find Global Positioning Systems (GPSs) in cars, ships, and airplanes, but also some mobile telephones and PDAs are already equipped with location detection devices. This has stimulated a large spectrum of research devoted for the spatio-temporal query processing and optimization and for providing location-aware services.

Usually, location-detection devices will send the moving objects' location information to a spatio-temporal data stream management system that will be responsible for answering spatio-temporal queries related to these moving objects. Mobile phones already send similar information to the cellular networks. Anyone who is carrying a cell phone is reporting its location to the network to receive phone calls wherever she goes.

Spatio-temporal data stream management systems (ST-DSMSs for short) have been prototyped (e.g., PLACE [8] and CAPE [14]) to make use of the massive number of moving objects being aware of their location. Location-aware services can be provided by these systems. Consequently, people can avoid traffic jams, parents can ensure their kids are safe, an enhanced 911 service [1] can be provided, and travelers can know easily about nearby facilities when they are driving.

ST-DSMSs receive their input as a stream of location updates from the moving objects. Streams are characterized by their high input rate. They cannot be stored and they have to be processed on the fly.

There are two types of queries that can be issued to ST-DSMSs. The first type is *snapshot queries*, which are evaluated only once. The query answer for this type depends on the spatio-temporal data reported to the ST-DSMS until the evaluation begins. The second type is *continuous queries*, which are registered with the ST-DSMS. Continuous queries are answered with the available location updates, and are updated whenever their answer changes with the arrival of new location updates from the moving objects.

Previous work on spatio-temporal queries on moving ob-

jects has focused on single predicate queries, for example, range queries and k-nearest-neighbor (kNN) queries. However, there are many interesting queries that consist of multiple predicates. None of the previous approaches consider multi-predicate queries (e.g.,[2, 3, 5, 6, 7, 8, 10, 9, 11, 12, 13, 15, 16, 18, 17]. Below, are a few examples of such queries.

Example 1: Joe is driving on a highway and needs to locate the closest motels in order to spend the night. However, he is only interested in those ahead of him. Joe would issue a continuous query to locate the nearest motels (a kNN predicate) that are ahead of him (a range predicate).

Example 2: It is required to identify trucks passing within a certain area. According to the selectivity of the range predicate and that of the selection (type="truck"), two different query evaluation plans may be optimal.

Example 3: The police department needs to investigate whether, at any time, there exists a region in which the number of suspects exceeds the number of police officers. In this case, the query would retrieve the suspects and police officers in each region (two range predicates), group by region to get the count of both groups, then perform a theta join to retrieve the required regions. Notice that both police officers and suspects are moving objects.

As we have shown in the previous examples, multi-predicate spatio-temporal queries may arise in several situations. This leads to a number of questions. How can we come up with a query evaluation plan for such queries? How can we determine whether a plan is optimal? And whether there are alternative plans that incur less system overhead?

We summarize our contribution as follows:

- Identifying the challenges related to the lack of spatio-temporal pipelined operators, and the impact of time, space, and their combination on the query plan optimality under different circumstances of query and object distributions.
- Proposing a cost estimation model for continuous spatio-temporal queries. We define the cost of different spatio-temporal query operators, and then we leverage the cost estimate from snapshot to continuous queries.
- Estimating selectivities of spatio-temporal queries. We consider not only free motion spatio-temporal histograms, but also road-network spatio-temporal histograms. We also consider using representative snapshots of the histograms in order to benefit from the periodicity property intrinsic in spatio-temporal data.
- optimizing the parameters of the spatio-temporal query operators. This is not limited to the existing spatio-temporal operators. We go further to define new operators and to optimize their performance as well.
- Proposing the adaptive execution of continuous spatio-temporal queries. We find it natural to benefit from adapting the query execution plan with the endless change in the selectivity of the query operators.

2. MOTIVATION

In this section, we present the challenges that confront the efficient execution of continuous spatio-temporal queries. First, there is no notion of pipelined spatio-temporal operators in the literature. Second, since the data is the location updates of moving objects, these updates form a stream of data that changes in properties (e.g., selectivities) over the time. Third, spatio-temporal servers usually serve a spatial area (e.g., a city). The properties of parts of this region governs how the objects are moving (e.g., their speed) and how their density is spread all over the cities. For instance, density of moving objects at downtown is usually higher than in a suburb. Forth, the combination of the spatial challenge and the temporal challenge forms a complex challenge as the selectivity of the operators change over both space and time and therefore no single query evaluation plan can withstand such change for a long period of time.

2.1 Spatio-temporal pipelined operators

We need to compose a pipeline of query operators to answer multi-predicate spatio-temporal queries. In the literature, there are many algorithms to answer spatio-temporal queries on moving objects. They all answer single-predicate queries — and none of them is pipelined.

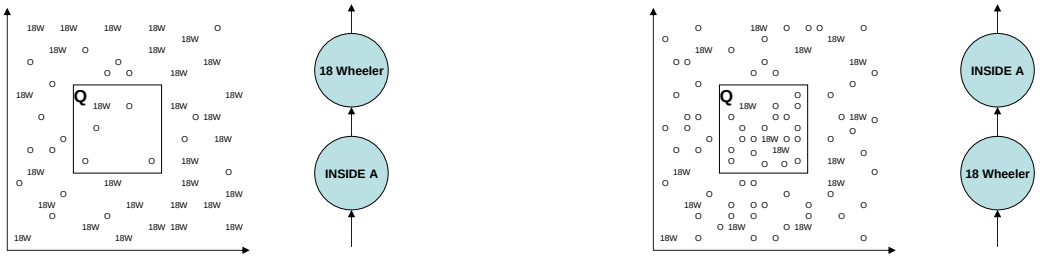
The input to the existing operators is a stream of the location updates of the moving objects. The output of these operators are the moving objects satisfying the query. The query answer can be reported periodically (or upon change) as the identifiers of all the objects forming the answer set. Alternatively, the answer set is reported progressively as a sequence of positive and negative updates [9]. A positive update indicates a new object being added to the answer set, and a negative update indicates an object that is being removed from the answer set.

Since the input of the current operators is different in format and semantic from their output, we cannot just plug the existing operators on top of each other to form a pipeline in order to answer a multi predicate spatio-temporal query.

2.2 Optimal Plan Varies with Time

Basically, the distribution of the moving objects changes continuously. For instance, downtown is full of vehicles during business hours leaving the suburbs with fewer cars. At night, most of the cars park, and hence deregister from the ST-DSMS. Consequently, the number of the cars in the ST-DSMS is less.

Consider the query Q in Table 1 that returns the ID and location of any 18 wheeler whose location is inside an area A. The INSIDE query operator is proposed in [9] to check whether or not a moving object is in a certain range. Initially, at Time t_1, the query optimizer finds that the selectivity of the INSIDE operator is less than the selectivity of the WHERE clause (Figure 1(a)). Thus, the query optimizer picks up the query execution plan in Figure 1(b) to be used to answer the query (Plan A). At Time t_2, many vehicles enter the area A and this increases the selectivity of the INSIDE operator, and meanwhile the number of trucks decreases (Figure 1(c)). Consequently, the WHERE clause is more selective than the INSIDE predicate. In this case,



(a) At time t_1 (b) Plan A (c) at time t_2 (d) Plan B

Figure 1: Moving vehicles (18W = 18 Wheeler, O = other) on the spatial space and the corresponding query execution plan.

```
SELECT M.ID, M.Location
FROM MovingObjects M
WHERE M.Type = "18 Wheeler"
INSIDE Area A
```

Table 1: Query Q

Plan A is suboptimal and the optimal query evaluation plan becomes Plan B (see Figure 1(d)).

2.3 Optimal Plan Varies with Location

Even at the same time instant, the mapping of a continuous spatio-temporal query into an efficient query evaluation plan is not one-to-one. Such challenge is illustrated with the following example. Consider the same range query in Table 1, but with two different areas A as shown in Figures 2(a) and 2(c).

Choosing an optimal execution plan is affected greatly by the distribution of the moving objects. In Figure 2(a), the INSIDE predicate is more selective than the predicate in the WHERE clause and hence the INSIDE operator comes first in the plan; Plan A (Figure 2(b)). On the other hand, for the same data distribution, and at the same time instant, but for a different INSIDE predicate (Figure 2(c)), Plan B (Figure 2(d)) is the optimal query evaluation plan.

2.4 Both Time and Location Change Continuously

The query optimization problem becomes more complex when a ST-DSMS receives moving queries on the moving objects. The selectivity of the query evaluation plan not only depends on the distribution of the moving objects on the space, but also on the location and speed of the moving-query focal object. When a query has more than a focal object, where all of them are moving, the query evaluation plan of the continuous query becomes suboptimal faster.

3. COST OF CONTINUOUS SPATIO-TEMPORAL QUERY

In this section, we propose a cost estimation model for continuous spatio-temporal queries. First, we define what the cost of a continuous spatio-temporal is. Next, we provide for the selectivity estimation that is essential to this cost model.

3.1 Cost Model for Continuous Spatio-temporal Queries

The location updates of the moving objects are streamed to the server. Location updates are batched each time period T, and are sent to the query executor. Therefore, the executor receives batches of location updates at times $t_0, t_1, \ldots$ such that for any $k > 0$, $t_k - t_{k-1} = T$. The amortized cost of a continuous spatio-temporal query is defined as the cost of updating the query answer for the location updates during any of these periods divided by the number of these location updates.

This amortized cost is a function of the operators in the query pipeline and the location updates of the moving objects. We are adopting the positive and negative update paradigm. Therefore, for each operator, we estimate the number of positive and negative updates coming out of each operator as a result of the location updates in any time period.

3.2 Spatio-temporal Selectivity Estimation

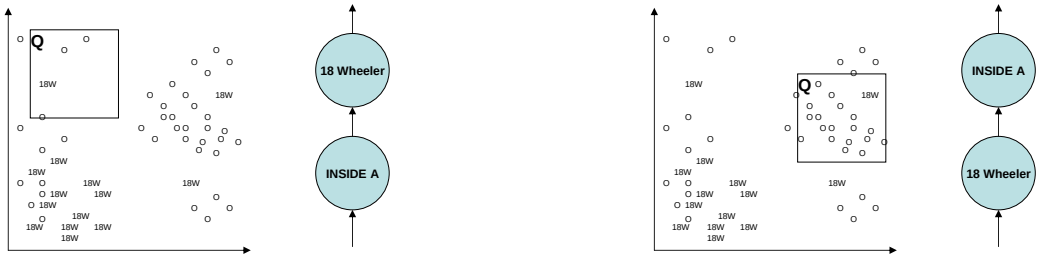
ST-Histograms are devised to estimate the selectivity of spatio-temporal operators [4]. They are continuously maintained to provide good estimates for the selectivity estimation of continuous pipelined query operators. Unlike traditional histograms that sample and/or examine all incoming data tuples, ST-Histograms are built and maintained using feedback from the query executor.

Periodically, the actual selectivity of the existing continuous queries is being sent from the query executor to the ST-Histogram manager. The query executor already knows the selectivity of each operator as part of its execution. Hence, ST-Histograms are maintained with almost no overhead to the system.

Rather than wasting system resources in maintaining accurate histograms for the whole spatial space, we only maintain accurate histograms for that part of the space that is relevant to the current existing queries. The rest of the space has less accurate histograms.

3.2.1 ST-Histograms for Free-motion Objects

ST-Histogram is a grid-based histogram, in which the spatial space is divided into a grid. Each grid cell is associated with its selectivity. The selectivity of a query is a weighted



(a) (b) Plan A (c) (d) Plan B

Figure 2: Moving vehicles (18W = 18 Wheeler, O = other) on the spatial space and the corresponding query execution plan.

sum of the selectivities of the parts of the grid cells that intersect the query range. The actual selectivity reported by the query executor is disseminated using similar weights. These weights are computed by taking into account that moving objects may move freely in the space. For instance, if a query covers a quarter of a grid cell, using a uniformity assumption, the selectivity of the query will be the quarter of the selectivity of this grid cell.

3.2.2 Road Network ST-Histograms

Many applications assume that the moving objects are constrained by a road network. Therefore, the uniformity assumption that the moving object may exist in any place in the grid cell cannot hold. In this case, we use a similar uniformity assumption, in which the fraction of lengthes of the network segments covered by the query is being used.

3.2.3 Representative ST-Histogram Snapshots

We analyze spatio-temporal data and realize certain patterns in the histogram. Mainly, periodicity patterns are detected in spatio-temporal data. This periodicity is a result of many periodic habits of the moving objects. For instance, people go to work and return home every day at the same time. This is materialized as rush hours, which occur daily at the same time. Moreover, people tend to take the same route to work each time they make this trip. Similarly, other patterns occur periodically on different periods, e.g., people traveling on weekends.

Instead of maintaining ST-histograms online, we can switch to the periodicity mode. In such mode, an ST-Histogram is run online for a certain time to learn the time instants when we can take representative snapshots from it. Later on, the ST-Histogram is exchanged by these representative snapshots, which will be used in their corresponding time interval. For instance, we may have a snapshot for the rush hour in the morning and another one for the rush hour in the evening. A snapshot may be taken for late night and so on.

3.3 From Snapshot to Continuous Query Cost

When a continuous spatio-temporal query is being submitted to the server, we need to estimate its cost to see whether or not we can afford executing it and hence admit it. This cost needs future estimates from ST-Histogram. Since ST-Histogram provides current estimates only, we provide for a mechanism to estimate future selectivities from current ones.

In this model, we compute the probabilities that an object, located inside the query range, will exit the range. This will materialize as a negative update out of the query operator. Similar probabilities are computed for positive updates. These probabilities are computed numerically and are looked up in a table whenever we need to compute the cost of the continuous spatio-temporal query.

4. SPATIO-TEMPORAL OPERATOR OPTIMIZATION

We find the optimization of the parameters of each query operator an integral part of the query optimization of continuous spatio-temporal queries. In the next subsection, we start with optimizing existing spatio-temporal operators such as kNN operators. Next, we introduce a new query operator called CANN to answer the continuous aggregate nearest neighbor query and provide mechanisms for optimizing its parameters in Section 4.2.

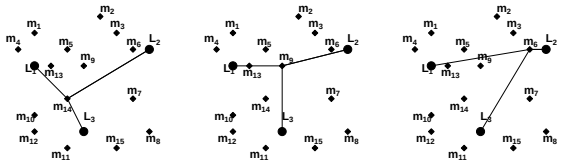
4.1 Optimizing kNN Query Operators

A kNN query operator continuously retrieves the k objects that are nearest to a query pivot. We provide optimizations for the parameters of this operator, whose execution is described in [7].

The configuration of this operator involves defining the radius of the search space that needs to be input to the operator. Also, to avoid the uncertainty in the query answer when objects outside the range move outside the search space, a safety region is proposed to be included in the search space [7].

The radius of the search space is dynamically computed with the arrival of location updates. If an update arrives late, this could affect the performance of the operator. Instead, we propose computing this radius based on the probability of having a moving object around the query pivot. This can be deduced from the selectivity estimates provided by ST-Histogram. Similar reasoning applies to compute the safety region.

Also, in the case where a kNN operator appears on top of an INSIDE operator in a query pipeline, the radius of the



(a) Sum (b) Max (c) Min

Figure 3: Examples of Aggregate Nearest Neighbor Queries

kNN operator search space needs to be adjusted since the global nearest neighbor may not be part of this constrained nearest neighbor query. We provide mechanisms to compute the radius of the operator in this case.

4.2 Optimizing CANN Query Operators

In this section, we summarize a new operator we devised to answer a new type of spatio-temporal queries, namely continuous aggregate nearest neighbor (CANN) queries. A CANN query specifies a set of landmarks, an integer k, and an aggregate distance function f (e.g., min, max, or sum), where f computes the aggregate distance between a moving object and each of the landmarks. The answer to this continuous query is the set of k moving objects that have the smallest aggregate distance f.

As an example, consider Figure 3. This figure gives a snapshot of a scenario in which $\{m_1, m_2, \ldots, m_{15}\}$ are a set of moving objects and $\{L_1, L_2, L_3\}$ are three landmarks. When the aggregate function is *sum*, the sum-ANN query retrieves the moving object whose sum of distances to the three landmarks is minimal. Object m_{14} represents such object in Figure 3(a). The max-ANN query (associated with the *max* aggregate function) retrieves the moving object whose maximum distance to any of the landmarks is minimal. In this case (see Figure 3(b)) the result is object m_9. Object m_6 is the result of min-ANN in Figure 3(c) and its minimum distance to any landmark is minimal.

We analyze the properties of the three aforementioned aggregate functions and prove that the locus of the points of equal aggregate distance to a set of foci is convex in the case of *sum* and *max* and is not in the case of *min*. Aggregate functions with a convex loci are called *class A* aggregate functions, whereas *class B* aggregate functions do not have a convex loci. We provide an incremental algorithm that would work for both aggregate functions classes. We also provide a progressive algorithm that provides a significant improvement in performance in the case of class A aggregate functions.

The Progressive Aggregate Nearest Neighbor algorithm (PANN, for short) retrieves the continuous query answer incrementally. PANN does not recompute the answer each period. Instead, it produces the differences from the current answer and the previous one. PANN is progressive in a way that the search space (i.e., the moving objects) gets pruned after checking each individual focus (landmark) in a certain order.

The foci order determines how many moving objects get pruned after being tested against each focus, and therefore, directly affects the performance of the algorithm. Computing the optimal foci order requires an exhaustive search among all possible foci permutation. We propose three policies for computing foci orderings. We also perform experiments to show which of these policies should be used with which class A aggregate function, and how far this selection will perform from the optimal order.

5. ADAPTIVE EXECUTION OF A CONTINUOUS ST QUERY

As pointed out in Section 2, no query execution plan can remain optimal for an extended period of time. Therefore, we envision the adaptive execution of a continuous spatio-temporal query as an essential component of any spatio-temporal stream management system.

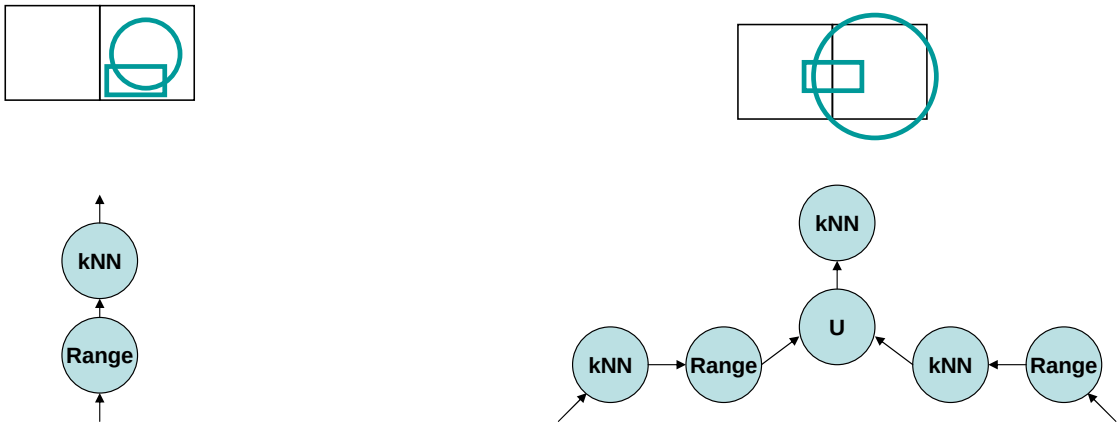
By monitoring the execution of the continuous query, we can monitor when the current statistics become far away from those statistics that are used to compute the query evaluation plan. In this case, the continuous query needs to be re-optimized.

To prevent a query plan from being executed on all location updates arriving to the ST-DSMS, we use a spatio-temporal data-to-plan grid. A location update is forwarded only to queries whose answer may be affected by this location update. This will materialize into having a sub-query for each grid cell whose input may be part of the query answer. Location updates in a certain grid cell are pushed into the sub-queries that belong to this grid cell. Answers to the sub-queries are merged or aggregated to form the answer to the original query. Therefore, an arriving location update may get dropped if it is located in a region in which no query is interested.

When the cost of the plan changes due to the motion of the query and/or the objects, the operators in the query evaluation plan might get reshuffled, the plan may get decomposed into sub-plans, or the sub-plans may reconsolidate again to form one plan. This is illustrated in Fig. 4(a)–4(b), where we have a continuous query Q consisting of a range predicate and a kNN operator. Initially, Q is covered by one data-to-plan grid cell, and according to the selectivity of both operators we depict the query evaluation plan shown in Fig. 4(a). Next, in Fig. 4(b), the range predicate moves and covers two grid cells. Consequently, a sub-plan is created for each grid cell, and the query answer will consist of the results of both sub-plans merged together. Later on, the query may move again and falls completely into another grid cell, the two sub-plans may reconsolidate and form a new query evaluation plan that takes into account the selectivity of all operators.

6. CONCLUSION

In this paper, we presented query optimization of continuous spatio-temporal queries. Different from most existing work, we focus on the optimization of multi-predicate spatio-temporal queries on moving objects. Specifically, we provide a costing mechanism for continuous spatio-temporal queries. We provide for the optimization of the parameters of the spatio-temporal operators. Finally, we propose the adaptive execution of the continuous queries for spatio-temporal data



(a) At Time t_1 (b) At Time t_2

Figure 4: Dynamic Plan Formation (An Example)

stream management systems.

7. REFERENCES

[1] FCC: Enhanced 911 - Wireless Services. http://www.fcc.gov/911/enhanced/.

[2] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects. In *IDEAS*, 2002.

[3] C. Böhm. A cost model for query processing in high dimensional data spaces. *TODS*, 25(2), 2000.

[4] H. G. Elmongui, M. F. Mokbel, and W. G. Aref. Spatio-temporal Histograms. In *SSTD*, 2005.

[5] G. Kollios, D. Gunopulos, and V. J. Tsotras. Nearest Neighbor Queries in a Mobile Environment. In *STDBM*, 1999.

[6] H. Li, H. Lu, B. Huang, and Z. Huang. Two ellipse-based pruning methods for group nearest neighbor queries. In *ACM-GIS*, 2005.

[7] M. F. Mokbel and W. G. Aref. GPAC: generic and progressive processing of mobile queries over mobile data. In *MDM*, 2005.

[8] M. F. Mokbel and W. G. Aref. PLACE: A Scalable Location-aware Database Server for Spatio-temporal Data Streams. *Data Engineering Bulletin*, 28(3), 2005.

[9] M. F. Mokbel, X. Xiong, and W. G. Aref. SINA: scalable incremental processing of continuous queries in spatio-temporal databases. In *SIGMOD*, 2004.

[10] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou. Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In *SIGMOD*, 2005.

[11] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group Nearest Neighbor Queries. In *ICDE*, 2004.

[12] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui. Aggregate nearest neighbor queries in spatial databases. *TODS*, 30(2), 2005.

[13] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query Processing in Spatial Network Databases. In *VLDB*, 2003.

[14] E. A. Rundensteiner, L. Ding, T. Sutherland, Y. Zhu, B. Pielech, and N. Mehta. CAPE: Continuous query engine with heterogeneous-grained adaptivity. In *VLDB*, 2004.

[15] Z. Song and N. Roussopoulos. K-Nearest Neighbor Search for Moving Query Point. In *SSTD*, 2001.

[16] X. Xiong, M. F. Mokbel, and W. G. Aref. SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-temporal Databases. In *ICDE*, 2005.

[17] M. L. Yiu, N. Mamoulis, and D. Papadias. Aggregate Nearest Neighbor Queries in Road Networks. *TKDE*, 17(6), 2005.

[18] X. Yu, K. Q. Pu, and N. Koudas. Monitoring K-Nearest Neighbor Queries Over Moving Objects. In *ICDE*, 2005.