Analysis of Predictive Spatio-Temporal Queries

YUFEI TAO
City University of Hong Kong, Hong Kong, China
JIMENG SUN
Carnegie Mellon University, Pittsburgh, Pennsylvania
and
DIMITRIS PAPADIAS
Hong Kong University of Science and Technology, Hong Kong, China

Given a set of objects S, a spatio-temporal *window query q* retrieves the objects of S that will intersect the window during the (future) interval q_T. A *nearest neighbor query q* retrieves the objects of S closest to q during q_T. Given a threshold d, a spatio-temporal *join* retrieves the pairs of objects from two datasets that will come within distance d from each other during q_T. In this article, we present probabilistic cost models that estimate the *selectivity* of spatio-temporal window queries and joins, and the *expected distance* between a query and its nearest neighbor(s). Our models capture any query/object mobility combination (moving queries, moving objects or both) and any data type (points and rectangles) in arbitrary dimensionality. In addition, we develop specialized spatio-temporal histograms, which take into account both location and velocity information, and can be incrementally maintained. Extensive performance evaluation verifies that the proposed techniques produce highly accurate estimation on both uniform and non-uniform data.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*selection process*

General Terms: Theory

Additional Key Words and Phrases: Database, spatio-temporal, selectivity, nearest distance, histogram

This work was supported by grants HKUST 6180/03E, 6197/02E, and 6081/01E from Hong Kong RGC.
Authors' addresses: Y. Tao, Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Hong Kong, China; email: taoyf@cs.cityu.edu.hk; J. Sun, Department of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA; email: jimeng@cs.cmu.edu; D. Papadias, Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China; email: dimitris@cs.ust.hk.

1. INTRODUCTION

Spatio-temporal databases have received considerable attention [Kollios et al. 1999; Agarwal et al. 2000; Pfoser et al. 2000; Saltenis et al. 2000; Hadjieleftheriou et al. 2002; Saltenis and Jensen 2002; Tao and Papadias 2003] in recent years due to the emergence of numerous applications (e.g., traffic supervision, flight control, weather forecast, etc.) that require management of continuously moving objects. An important operation in these systems is to predict objects' future location based on information at the current time. For this purpose, object movement is usually represented as a linear function of time. For example, given the location $o(0)$ of object o at the current time 0 and its velocity o_V, its position at some future time t can be computed as $o(t) = o(0) + o_V \cdot t$. Instead of location information, the system stores the function parameters so that an update to the database is necessary only when some parameter (i.e., o_V) changes.

A *spatio-temporal window query* (STWQ) specifies a (static or moving) region q_S, a future time interval q_T, and retrieves all data objects that will intersect (or will be covered by) q_S during q_T (e.g., "Based on its current motion, find all residential areas that will be covered by the typhoon in an hour"). A *spatio-temporal k-nearest neighbor* (STkNN) query returns the k objects that will be closest to q_S during q_T, where the distance between two objects during q_T is defined as the minimum of their distances at all timestamps $t \in q_T$ (e.g., "According to the ship's present movement, which will be its 2 nearest ports tomorrow 9–11 am?"). Given two datasets S_1, S_2, a *spatio-temporal* ("within-distance") *join* (STJ) obtains all pairs of objects (o_1, o_2) in the cartesian product $S_1 \times S_2$ such that the distance between o_1, o_2 during q_T is smaller than a constant d (e.g., "Find all pairs of flights that will come closer than 1 km from each other within the next 10 minutes"). The special case of $d = 0$ corresponds to *intersection* joins.

The above queries have been extensively studied in spatial databases, where objects and queries are static, and q_T corresponds to the current time. Existing analysis on window queries (spatial joins) focuses on estimating the *selectivity* [Kamel and Faloutsos 1993; Huang et al. 1997; Belussi and Faloutsos 1995; Acharya et al. 1999; Theodoridis et al. 2000], which is defined as the number of retrieved objects (object pairs) divided by the dataset cardinality (the size of the cartesian product). For kNN retrieval, where the concept of selectivity is not relevant (i.e., the output size is simply k), the goal of analysis is to compute the *nearest distance* (ND_k) from the query point to the farthest (kth) retrieved neighbor [Berchtold et al. 1997; Bohm 2000; Berchtold et al. 2001]. In addition to their significance as stand-alone measures in several applications (e.g., in air-traffic control systems it is important to know the expected distance of the nearest airport at any time, or the pairs of airplanes within collision course), selectivity and nearest distance are crucial for query optimization, due to their close connection with the query costs. The results derived for static data, however, are not applicable in dynamic environments, where the problems are more complex (intuitively, spatial databases constitute a special case where objects and queries have zero velocities). Related research in spatio-temporal databases is scarce and limited to STWQ selectivity; currently, there does not exist any published work on STkNN and STJ.

This article addresses these problems by presenting a comprehensive probabilistic study for spatio-temporal queries that covers (i) all common queries, (ii) all query/object mobility combinations (i.e., moving objects, moving query, or both), and (iii) arbitrary types of data (points or rectangles) in any dimensionality. Specifically, starting with uniform data, we first propose accurate cost models that estimate the query selectivity (for STWQ, STJ) and nearest distance (for STkNN), during a (future) query time interval using the current location and velocity information. Our analysis is based on a novel reduction technique, which transforms complex problems (e.g., involving moving rectangle data) to simpler ones (i.e., involving only static points). As a second step, we devise specialized spatio-temporal histograms to deal with nonuniform datasets. The proposed equations provide significant insight into the behavior of alternative query types, and are directly applicable to query optimization.

The underlying assumption of our techniques is that the current locations and velocities are known, and there are no updates between the current and the future query time. Applications that satisfy these conditions involve objects (ships, airplanes, weather patterns) moving in unobstructed spaces. Obviously, the prediction horizon depends on the (velocity) update rate. For instance, given that ships move with slow, linear movements for long periods, it is meaningful to estimate queries that refer to several hours in the future. For air traffic control, the prediction horizon should be in the order of minutes. On the other hand, velocity-based prediction is meaningless in applications, such as road network databases, where objects update their speed or direction of movement within very short time intervals (i.e., a car must turn or stop when it reaches the end of the current road segment). This point will be elaborated further in the sequel.

The rest of the article is organized as follows. Section 2 surveys the existing methods for estimating the selectivity and nearest distance. Section 3 provides the basic definitions used in our analysis. Then, Section 4 presents cost models for STWQ selectivity estimation, while Sections 5, 6 discuss STkNN and STJ, respectively. Section 7 develops a new spatio-temporal histogram for nonuniform data that supports incremental updates. Section 8 contains an extensive experimental evaluation to verify the accuracy of the proposed technique. Finally, Section 9 concludes the paper with directions for future work.

2. RELATED WORK

In this section, we review the previous work directly related to ours. Section 2.1 focuses on spatial databases, introducing methods for predicting the selectivity and nearest distance. Then, Section 2.2 discusses the selectivity of spatio-temporal window queries.

2.1 Selectivity and Nearest Distance in Spatial Databases

- *Window Query Selectivity*

The first models of window query selectivity on uniform datasets appear in Kamel and Faloutsos [1993] and Pagel et al. [1993]. Specifically, given two

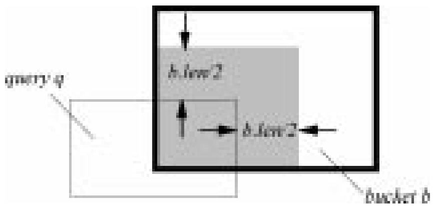


Fig. 1. Estimating the selectivity inside a histogram bucket.

m-dimensional rectangles r, q such that (i) they uniformly distribute in the data space $[U_{\min\_i}, U_{\max\_i}]^m$ (i.e., the ith axis has range $[U_{\min\_i}, U_{\max\_i}]$), and (ii) the extent of $r(q)$ on the ith dimension ($1 \leq i \leq m$) has length $r_i(q_i)$, then the probability that r intersects q equals $(1/U_{vol})\prod_{i=1}^{m}(r_i + q_i)$, where $U_{vol} = \prod_{i=1}^{m}(U_{\max\_i} - U_{\min\_i})$ is the volume of the data space. Hence, for a uniform dataset with cardinality N, the number of objects intersecting query q is $(N/U_{vol})\prod_{i=1}^{m}(r_i + q_i)$.

Query selectivity for nonuniform (rectangular) data can be estimated by maintaining a histogram that partitions the data space into a set of *buckets*, and assuming that object distribution in each bucket is (almost) uniform. Specifically, each bucket b contains (i) the number $b.num$ of objects whose centroids fall in b, and (ii) the average extent $b.len$ of such objects. Figure 1 illustrates an example in the 2D space, where the gray area corresponds to the intersection between b and the extended query region, obtained by enlarging each edge of q with distance $b.len/2$. Following the analysis on uniform data (i.e., results of [Kamel and Faloutsos 1993; Pagel et al. 1993] as described earlier), the expected number of qualifying objects in b is approximately $b.num \cdot I.area/b.area$, where $I.area$ and b.area are the areas of the intersection region and b, respectively [Acharya et al. 1999]. The total number of objects intersecting q is predicted by summing the results of all buckets. Evidently, satisfactory estimation accuracy depends on the degree of uniformity of objects' distributions in the buckets. This can be maximized using various algorithms [Muralikrishna and DeWitt 1988; Poosala and Ioannidis 1997; Acharya et al. 1999; Gunopulos et al. 2000; Jin et al. 2000; Bruno et al. 2001], which differ in the way that buckets are structured. For example, in Muralikrishna and DeWitt [1988] buckets have similar sizes (i.e., "equi-width") or cover approximately the same number of objects (i.e., "equi-depth"), while in Poosala and Ioannidis [1997] and Acharya et al. [1999] bucket extents minimize the so-called "spatial skew." Jin et al. [2000] adopt the density file, which is similar to the histograms in Muralikrishna and DeWitt [1988] but is augmented with additional statistics. In the above methods, bucket extents are disjoint, while Gunopulos et al. [2000] and Bruno et al. [2001] relax this constraint with specialized algorithms.

In addition to the previous techniques, window query selectivity on nonuniform data can be estimated using *fractals and power laws* [Belussi and Faloutsos 1995; Proietti and Faloutsos 1998, 2001], *sampling* [Olken and Rotem 1990; Palmer and Faloutsos 2000; Chaudhuri et al. 2001; Wu et al. 2001], *kernel estimation* [Blohsfeld et al. 1999], *single value decomposition* [Poosala and Ioannidis 1997], *compressed histograms* [Matias et al. 1998, 2000; Lee et al.

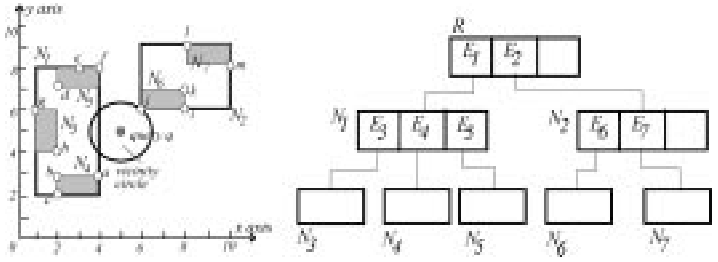


Fig. 2. Relation between the nearest distance and the cost of a NN query.

1999; Thaper et al. 2002], *maximal independence* [Deshpande et al. 2001], *Euler formula* [Sun et al. 2002b], etc. Furthermore, Aboulnaga and Naughton [2000] discusses the problem on general polygon objects.

• *Nearest Neighbor Distance*

The cost of a kNN query is closely related to the nearest distance ND_k between the query and its k-th NN. Figure 2 shows a set of static points $(a, b, \ldots, m)$ indexed by an R-tree [Guttman 1984; Beckmann et al. 1990] with three levels, the minimum bounding rectangles (MBR) of the tree nodes, and a single nearest neighbor query q (whose NN is point j). As shown in [Papadopoulos and Manolopoulos 1997; Berchtold et al. 1997], an optimal kNN algorithm (e.g., the one in Hjaltason and Samet [1999]) must visit those nodes whose MBR intersect the *vicinity circle* that centers at q with radius ND_k. To answer the single NN query in Figure 2, for example, the algorithm must access nodes (i.e., the root, N_1, N_2, and N_6), overlapping the circle centered at q with radius equal to the distance of q and j. Consequently, deriving the expected value of ND_k is a vital step in the kNN analysis [Papadopoulos and Manolopoulos 1997; Berchtold et al. 1997; Ciaccia et al. 1998; Weber et al. 1998; Beyer et al. 1999; Bohm 2000].

Focusing on single NN queries, Berchtold et al. [1997] derives the nearest neighbor distance ND_1 by first computing the probability $P_{single}(l)$ that an object is within distance l from q. Since ND_1 is less than l, if and only if, at least one object is within distance l from q, the probability $P_{ND1}(l)$ that $ND_1 \leq l$ equals $1-(1-P_{single}(l))^N$ (where N is the dataset cardinality). Let $p(ND_1 = l)$ be the probability density function of $P_{ND1}(l)$ (i.e., taking its derivative with respect to l); ND_1 can then be solved as $ND_1 = \int_0^\infty l \cdot p(ND_1 = l)dl$. Bohm [2000] generalizes the derivation to kNN queries.

• *Spatial Join Selectivity*

The analysis of (intersection) spatial joins is usually based on that of window queries, because a join (involving datasets S_1 and S_2) can be regarded as a set of window queries (each corresponding to an object in S_1) performed on S_2. Specifically, as suggested in Aref and Samet [1994] and Huang et al. [1997], for uniform distribution the number of qualifying object pairs can be estimated as $(N_1 \cdot N_2/U_{vol}) \prod_{i=1}^{m}(s_{1i} + s_{2i})$, where (i) N_1, N_2 are the cardinalities, (ii) s_{1i}, s_{2i} the object extents of the participating datasets (on the ith dimension), and (iii) U_{vol} the volume of the data space. Theodoridis et al. [1998, 2000] utilize this

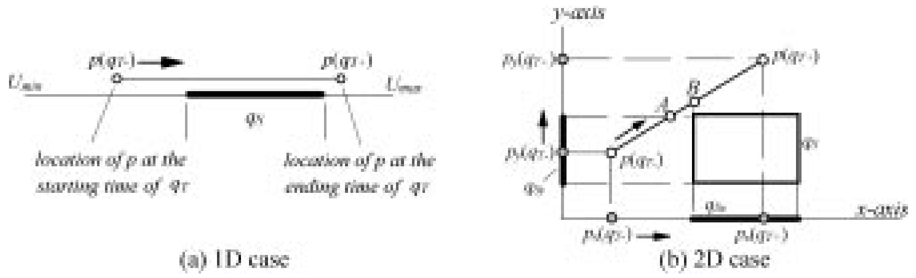


Fig. 3. Window queries in one- and two-dimensional spaces.

equation for nonuniform data, by considering bucket pairs from the histograms of the join datasets (in a way similar to Figure 1). An et al. [2001] maintains statistics about objects' edges and corners, while Belussi and Faloutsos [1998] and Faloutsos et al. [2000] apply power laws. Sun et al. [2002a] studies join selectivity restricted in a part of the data-space, and Mamoulis and Papadias [2001] addresses multi-way spatial join selectivity.

2.2 STWQ Selectivity Estimation

None of the previous methods is applicable in spatio-temporal databases, where the volatile nature of objects and/or queries invalidates their basic assumptions. Choi and Chung [2002] discusses the selectivity of STWQ for moving point data and static queries (i.e., the query region remains fixed). Starting from the one-dimensional case, where the spatial universe is a line segment [$U_{\min}$, $U_{\max}$], the model predicts the number of points that will intersect the query extent q_S during the query interval $q_T = [q_{T-}, q_{T+}]$ ($0 \leq q_{T-} \leq q_{T+}$, the current time is 0). Figure 3(a) shows q_S and the positions $p(q_{T-})$ and $p(q_{T+})$ of a data point p at the starting q_{T-} and ending q_{T+} timestamps of q_T, respectively (velocity directions are indicated with arrows). The distance between $p(q_{T-})$ and $p(q_{T+})$ depends on the velocity p_V of p, which distributes uniformly in the range [$V_{\min}$, $V_{\max}$]. Clearly, point p satisfies the query if and only if the segment connecting $p(q_{T-})$ and $p(q_{T+})$ intersects q_S. Assuming that the location of p at the current time 0 follows uniform distribution in [$U_{\min}$, $U_{\max}$], the probability (i.e., also the selectivity of q) that a data point qualifies q is a function of $U_{\min}$, $U_{\max}$, $V_{\min}$, $V_{\max}$, and the query parameters [Choi and Chung 2002].

The multidimensional version of the problem is converted to the 1D case by projecting objects and queries onto individual dimensions [Choi and Chung 2002]. In particular, the probability that p satisfies q is computed as $\prod_{i=1}^{m} Sel_i$, where m is the dimensionality and Sel_i is the 1D selectivity (i.e., the probability that the projection p_i of point p on the ith dimension intersects the projection q_i of the query during interval q_T). This, however, is inaccurate due to the fact that, a data point may still violate a query q, even if its projection intersects that of q on every dimension. For instance, in Figure 3(b), p is not a qualifying point because it never appears in the region q_S. However, the projections of its trajectory (during q_T) on both dimensions intersect the corresponding projections of q_S (i.e., segments q_{Sx} and q_{Sy}). Therefore, $\prod_{i=1}^{m} Sel_i$ overestimates the actual probability.

In general, an object o satisfies a spatio-temporal window query q if (i) the trajectory projection of o intersects that of q on each dimension (i.e., the *spatial condition*), and (ii) the intersection time intervals on all dimensions must overlap (i.e., the *temporal condition*). Let T_A and T_B be the timestamps when p reaches location A and B in Figure 3(b); then, the y-intersection interval (i.e., the period when y-projections of p and q intersect) is $[q_{T-}, T_A]$, while that on the x-dimension is $[T_B, q_{T+}]$. Point p does not satisfy the query because the two intersection intervals are disjoint, violating condition (ii). The estimation in Choi and Chung [2002] ignores the temporal condition (hence, in the sequel we refer to the method as the *time-oblivious approach*), which may lead to significant estimation error.

Hadjieleftheriou et al. [2003] proposes two alternative solutions for STWQ selectivity estimation on point objects. Using the *duality transformation* [Kollios et al. 1999], the first method converts the (linear) trajectory of each object to a point in the 4D *dual space*, which permits the direct employment of conventional multidimensional histograms (e.g., *minskew* [Acharya et al. 1999]). Accordingly, a query is transformed to a simplex search region in the dual space; the objective is to predict, using the histogram, the number of data points in this simplex query region. The second approach, instead of building a separate histogram, utilizes the extents of the leaf nodes of an underlying index (e.g., R-tree) as the histogram buckets. Although this method supports dynamic maintenance (by resorting to the index), it incurs high space consumption and large error, as shown in the experiments of Hadjieleftheriou et al. [2003].

3. DEFINITIONS

For all query types, we assume that either the data objects and/or the query are moving. Let r be a moving rectangle in the m-dimensional space. The extent of r at the current time 0 is a $2m$-dimensional vector $r_S = \{r_{S1-}, r_{S1+}, r_{S2-}, r_{S2+}, \ldots, r_{Sm-}, r_{Sm+}\}$, where $[r_{Si-}, r_{Si+}]$ is the extent along the ith dimension ($1 \leq i \leq m$). The *spatial length* of r on each axis is $r.L_i = r_{Si+} - r_{Si-}$. Vector $r_V = \{r_{V1-}, r_{V1+}, r_{V2-}, r_{V2+}, \ldots, r_{Vm-}, r_{Vm+}\}$ represents the velocities of r, such that $r_{Vi-}(r_{Vi+})$ is the velocity of the lower (upper) boundary of r on the ith dimension ($1 \leq i \leq m$). Similar to $r.L_i$, we define $r.LV_i = r_{Vi+} - r_{Vi-}$ as the *velocity length*. In the sequel, we assume that the spatial and velocity lengths are always nonnegative (which implies that a rectangle does not disappear in some future time). The extent $r_S(t)$ (also a $2m$-dimensional vector) of r at some future time t can be computed from r_S and r_V as: $r_S(t) = r_S + t \cdot r_V$ ($r_S(0) = r_S$). A point p is represented in a similar way: $p_S = \{p_{S1}, p_{S2}, \ldots, p_{Sm}\}$ and $p_V = \{p_{V1}, p_{V2}, \ldots, p_{Vm}\}$ are the coordinates and velocities on the m dimensions, respectively; the spatial and velocity lengths of p are zero on each axis. In all cases, we follow the common assumption in the literature that an object's velocity remains constant until explicitly updated.

For the ith dimension ($1 \leq i \leq m$), the data space has extent $[U_{\min\_i}, U_{\max\_i}]$, and possible velocity values fall in the range $[V_{\min\_i}, V_{\max\_i}]$ (i.e., the *velocity space*). In case of uniform datasets: (i) the spatial (velocity) length of each object has the same value L_i (LV_i) on the ith axis, (ii) the coordinate r_{Si-} (or p_{Si} of a point p) of the lower boundary uniformly distributes in $[U_{\min\_i}, U_{\max\_i} - L_i]$, (iii) the velocity r_{Vi-} (or p_{Vi} of a point p) uniformly distributes in $[V_{\min\_i}, V_{\max\_i} - LV_i]$, and (iv) all dimensions are independent. These assumptions will be later removed for non-uniform datasets. We discuss spatio-temporal versions of window queries, nearest neighbors and joins:

—Given a set of objects, a STWQ q (i) specifies a rectangle (with current extent q_S and velocity vector q_V) and a future time interval $q_T = [q_{T-}, q_{T+}](0 \le q_{T-} \le q_{T+})$, and (ii) retrieves all objects o that intersect q during q_T, or more formally, there exists some timestamp $t \in [q_{T-}, q_{T+}]$ such that $o_S(t)$ intersects $q_S(t)$.

—A STkNN q returns the k objects that are closest to q_S during q_T. Specifically, the distance $dist(o, q, q_T)$ between objects o, q is the minimum of their distances during q_T, or more formally: $dist(o, q, q_T) = \min\{\|o(t), q(t)\| \text{ for all } t \in q_T\}$, where $\|o(t), q(t)\|$ represents the Euclidean distance between $o(t)$, $q(t)$ at timestamp t.[1] Let $o_{NN1}, o_{NN2}, \ldots, o_{NNk}$ be the k NN of q (in ascending order of their distances to q); then, the nearest distance ND_k of q corresponds to $dist(o_{NNk}, q, q_T)$.

—Given two sets S_1, S_2 of objects, STJ outputs all pairs of objects (r_1, r_2) from the Cartesian product $S_1 \times S_2$ such that their distance $dist(r_1, r_2, q_T)$ is not larger than a constant d, where q_T and d are the join parameters. If $d = 0$, the join condition degenerates to intersection.

A *query* (STWQ, STkNN, STJ) is called *current*, if q_{T-} (the starting timestamp of q_T) equals 0 (i.e., the current time). When $q_{T-} = q_{T+}$, the query constitutes a *timestamp query*; otherwise, it is an *interval query*. The goal of our analysis is to represent the selectivity (for STWQ and STJ) and expected nearest distance (for STkNN) as a function of universe constants $U_{\min\_i}$, $U_{\max\_i}$, $V_{\min\_i}$, $V_{\max\_i}$, data properties L_i, LV_i, and the query parameters. Focusing on uniform datasets, the next section discusses STWQ, while Sections 5 and 6 solve the problems for STkNN and STJ, respectively. Section 7 extends the techniques to arbitrary distributions with the aid of spatio-temporal histograms. Table I summarizes the main symbols that will appear in our derivation.

4. SPATIO-TEMPORAL WINDOW QUERY SELECTIVITY

We derive the selectivity of STWQ based on the observation that any instance of the problem can be reduced to predicting the selectivity of a moving rectangle query on a set of static points. Section 4.1 studies this basic problem, and then, Sections 4.2 and 4.3 illustrate the reduction of other cases to the basic problem. Section 4.4 quantifies the error of the time-oblivious approach.

4.1 Static Point Data

A static point p satisfies a moving query window q, if p lies inside $q_S(t)$ at some timestamp $t \in q_T$. For the sake of simplicity, we first focus on current

[1]The distance between a point and a rectangle can be computed as shown in Roussopoulos et al. [1995]; distance computation between two rectangles is discussed in Corral et al. [2000].

Table I. Frequent Symbols in the Analysis

Symbol	Description
m	dimensionality of the data space
U_{vol}	volume of the data space
$[U_{\min\_i}, U_{\max\_i}]$	extent of the space on the ith dimension
$[V_{\min\_i}, V_{\max\_i}]$	velocity range on the ith dimension
$r_S = \{r_{S1-}, r_{S1+}, r_{S2-}, r_{S2+}, \ldots, r_{Sm-}, r_{Sm+}\}$	extent of moving rectangle r at the current time
$r_V = \{r_{V1-}, r_{V1+}, r_{V2-}, r_{V2+}, \ldots, r_{Vm-}, r_{Vm+}\}$	velocity vector of moving rectangle r
$p_S = \{p_{S1}, p_{S2}, \ldots, p_{Sm}\}$	coordinates of moving point p at the current time
$p_V = \{p_{V1}, p_{V2}, \ldots, p_{Vm}\}$	velocities of moving point p
L_i	spatial length of an object on the ith dimension
LV_i	velocity length of an object on the ith dimension
q_S	query extent vector at the current time (for STWQ, STkNN)
q_V	query velocity vector (for STWQ, STkNN)
q_T=$[q_{T-}, q_{T+}]$	query time interval
$dist(o, q, q_T)$	minimum distance between o and q during interval q_T
d	distance parameter for STJ
$CX(q)$	convex hull of corner points of $q_S(q_{T-})$ and $q_S(q_{T+})$
$ECX(q, l)$	extended convex hull of corner points of $q_S(q_{T-})$ and $q_S(q_{T+})$
Sel	selectivity of a query (for STWQ, STJ)
ND_k	nearest distance of STkNN
N	cardinality of the dataset (only used in STkNN)

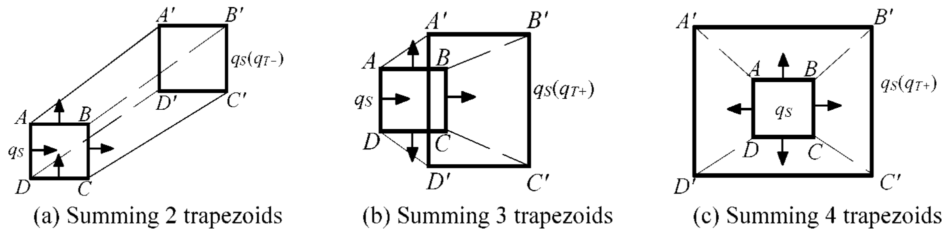


Fig. 4. All cases in calculating the area of $CX(q)$.

queries (i.e., $q_{T-} = 0$). Figure 4(a) shows a 2D moving query q, where q_S and $q_S(q_{T+})$ (i.e., rectangles $ABCD$ and $A'B'C'D'$) indicate the positions of q at the starting (0) and ending time (q_{T+}) of q_T, respectively. Let $CX(q)$ be the convex hull of all the corner points of q_S and $q_S(q_{T+})$ (i.e., polygon $ADCC'B'A'$ in Figure 4(a)). $CX(q)$ corresponds to the region that is "swept" by q during q_T, and consequently, a data point p will be retrieved if and only if it lies in $CX(q)$.

Since the data distribution is uniform, the probability for a point to fall inside $CX(q)$ is the ratio between the area (volume in higher dimensions) of $CX(q)$ and that of the spatial universe, which is also the selectivity Sel_{static_pt} of q:[2]

$$Sel_{static\_pt}(q_{Si-}, q_{Si+}, q_{Vi-}, q_{Vi+}, q_T) = \frac{volume(CX(q))}{U_{vol}} \tag{4.1}$$

The area (volume) of $CX(q)$ depends on the velocity directions of q_V. In Figure 4(a), for example, q_{Vi-} and q_{Vi+} have the same direction along all dimensions, in which case the area of $CX(q)$ is the sum of rectangle $ABCD$ (i.e., query's extent at the current time), and two trapezoids $ABB'A'$ and $BCC'B'$.

[2]In Eq. (4.1), the subscript "i" denotes that the corresponding parameter of Sel_{static_pt} ranges over all dimensions; similar notations are adopted in subsequent formulas.

```
Algorithm compute_CX_vol (q)
1. $sum=\prod_{i=1}^{m}[q_{Si+}(q_{T-})-q_{Si-}(q_{T-})]$ /* $[q_{Si-}(q_{T-}), q_{Si-}(q_{T-})]$ is the extent of $q_S$ at time $q_{T-}$ */
2. for each dimension $1 \le i \le m$
3.     if $q_{Vi-}<0$ and $q_{Vi+}<0$ then, $sum=sum+volume(Trapezoid_{lower\_i})$ (as in equation 4-2)
4.     if $q_{Vi-}>0$ and $q_{Vi+}>0$ then, $sum=sum+ volume\ (Trapezoid_{upper\_i})$ (as in equation 4-3)
5.     if $q_{Vi-}<0$ and $q_{Vi+}>0$ then, $sum=sum+ volume\ (Trapezoid_{lower\_i})+ volume\ (Trapezoid_{upper\_i})$
6. return $sum$
End compute_CX_vol
```

Fig. 5. Algorithm for computing the volume of $CX(q)$.

Trapezoid $ABB'A'$ ($BCC'B'$) is the region swept by segment AB (BC) during q_T. Figure 4b shows another case where q_{Vx-} and q_{Vx+} still have the same direction, while q_{Vy-} and q_{Vy+} are opposite. Then, the area of $CX(q)$ is the sum of rectangle $ABCD$, and three trapezoids $ABB'A'$, $BCC'B'$, and $DD'C'C$ (swept by segments AB, BC, CD, respectively). Figure 4c illustrates a third case, where velocities on all dimensions have opposite directions, and the area of $CX(q)$ is the sum of rectangle $ABCD$ and four trapezoids $ABB'A'$, $BCC'B'$, $DD'C'C$, $AA'D'D$ (swept by segments AB, BC, CD, DA).

Computing the area of a single trapezoid is straightforward. Consider, for example, trapezoid $ABB'A'$, where the lengths of AB and $A'B'$ are $(q_{Sx+} - q_{Sx-})$ and $(q_{Sx+} - q_{Sx-}) + (q_{Vx+} - q_{Vx-}) \cdot (q_{T+} - q_{T-})$, respectively. Furthermore, note that the vertical distance between AB and $A'B'$ is $q_{Vy+} \cdot (q_{T+} - q_{T-})$; thus, the area of trapezoid $ABB'A'$ is given by[3]:

$$area(ABB'A') = \frac{1}{2}[2(q_{Sx+} - q_{Sx-}) + (q_{Vx+} - q_{Vx-})(q_{T+} - q_{T-})] \cdot q_{Vy+}(q_{T+} - q_{T-})$$

In general m-dimensional spaces, each trapezoid is the region swept by a boundary of q_S, which is a $(m-1)$-dimensional hyper-rectangle. Specifically, the trapezoid volumes decided by the lower and upper boundaries on the ith dimension ($1 \le i \le m$) can be calculated using Eqs. (4.2) and (4.3), respectively:

$$\begin{aligned} volume(Trapezoid_{lower\_i}) = \frac{1}{2}\Bigg\{ & \prod_{j \neq i}(q_{Sj+} - q_{Sj-}) + \prod_{j \neq i}[(q_{Sj+} - q_{Sj-}) \\ & + (q_{Vj+} - q_{Vj-})(q_{T+} - q_{T-})]\Bigg\} |q_{Vi-}|(q_{T+} - q_{T-}), \quad (4.2) \end{aligned}$$

$$\begin{aligned} volume(Trapezoid_{upper\_i}) = \frac{1}{2}\Bigg\{ & \prod_{j \neq i}(q_{Sj+} - q_{Sj-}) + \prod_{j \neq i}[(q_{Sj+} - q_{Sj-}) \\ & + (q_{Vj+} - q_{Vj-})(q_{T+} - q_{T-})]\Bigg\} |q_{Vi+}|(q_{T+} - q_{T-}). \quad (4.3) \end{aligned}$$

Figure 5 shows the algorithm for computing the volume of $CX(q)$ in m-dimensional spaces, after which the selectivity of the query can be obtained

[3]The area of a trapezoid equals ½ times the product of the sum of parallel edges, and the distance between them.

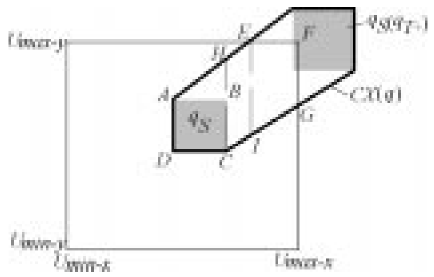


Fig. 6. Intersection area computation when $CX(q)$ is not completely in DS.

using Eq. (4.1). The handling of noncurrent queries (i.e., $q_{T-} > 0$) is straightforward. The only difference is that $CX(q)$ should be the convex hull of the corner points of rectangles $q_S(q_{T-})$ and $q_S(q_{T+})$. The volume of $CX(q)$ can still be calculated using the algorithm of Figure 5.

So far we have assumed that $CX(q)$ lies entirely in the spatial universe DS, while it is possible that the query moves out of DS during q_T as shown in Figure 6. In such cases, the probability that a data point satisfies q corresponds to the area of the intersection between $CX(q)$ and DS. In Figure 6, for example, the intersection region is hexagon $AEFGCD$, which can be obtained using a standard polygon intersection algorithm [Berg et al. 1997]. In particular, for 2-dimensional spaces the computation time is $O(1)$, due to the fact that $CX(q)$ and DS have constant complexities (i.e., they contain at most 6 and exactly 4 edges, respectively). After obtaining the intersection polygon, its area can be computed by decomposing the polygon into a set of trapezoids, and then summing their areas. The hexagon $AEFGCD$ in Figure 6, for instance, is divided into three trapezoids $ADCH$, $HCIE$, $EIGF$. Note that this algorithm also has constant computation time, because the intersection polygon contains at most 6 edges (i.e., the complexity of $CX(q)$).

Generalizing the above approach to compute the intersection volume between $CX(q)$ and DS in higher dimensional spaces, however, results in excessively complex equations that require expensive evaluation. Instead, we adopt the Monte-Carlo method. First, a set of α points (in our experiments, $\alpha = 2000$) are generated uniformly in the spatial universe. Then, we count the number β of points that fall in $CX(q)$, and the intersection volume is approximated as $\beta/\alpha \cdot U_{vol}$, where U_{vol} is the volume of the data space. Deciding if a static point p lies in $CX(q)$ can be achieved using an algorithm proposed in Saltenis et al. [2000] for determining whether two moving rectangles intersect each other during a time interval (i.e., p is in $CX(q)$ if and only if there exists a timestamp $t \in q_T$ such that p is in $q_S(t)$).

4.2 Moving Point Data

In this section, we discuss selectivity estimation for moving data points, where the location p_{Si} and velocity p_{Vi} of each point p along the ith ($1 \le i \le m$) dimension distributes uniformly in $[U_{\min\_i}, U_{\max\_i}]$ and $[V_{\min\_i}, V_{\max\_i}]$, respectively. Given a moving query q, we aim at deriving the probability $P(u_1, u_2, \ldots, u_m)$ that a point p satisfies q when its velocity p_{Vi} takes a specific value u_i

$(1 \le i \le m)$. Once $P(u_1, u_2, \ldots, u_m)$ has been derived, the query selectivity Sel_{moving_pt} can be obtained by integrating all possible values of p_{Vi}:

$$Sel_{moving\_pt}(q_{Si-}, q_{Si+}, q_{Vi-}, q_{Vi+}, q_T)$$
$$= \int_{V_{\min\_1}}^{V_{\max\_1}} \int_{V_{\min\_2}}^{V_{\max\_2}} \cdots \int_{V_{\min\_m}}^{V_{\max\_m}} P(u_1, u_2, \ldots, u_m) f(u_1, u_2, \ldots, u_m)\, du_m \cdots du_2 du_1. \tag{4.4}$$

where $f(u_1, u_2, \ldots, u_m)$ is the joint probability density function[4] of u_1, $u_2, \ldots, u_m$. Since all dimensions are independent and u_i uniformly distributes in $[V_{\min\_i}, V_{\max\_}i]$, we have:

$$f(u_1, u_2, \ldots, u_m) = f(u_1) \cdot f(u_2) \cdot \ldots \cdot f(u_m) = \prod_{i=1}^{m} \left(\frac{1}{V_{\max\_i} - V_{\min\_i}} \right)$$

Hence, Eq. (4.4) can be written as:

$$Sel_{moving\_pt}(q_{Si-}, q_{Si+}, q_{vi-}, q_{vi+}, q_T)$$
$$= \prod_{i=1}^{m} \left(\frac{1}{V_{\max\_i} - V_{\min\_i}} \right) \int_{V_{\min\_1}}^{V_{\max\_i}} \int_{V_{\min\_2}}^{V_{\max\_2}} \cdots \int_{V_{\min\_m}}^{V_{\max\_m}} P(u_1, u_2, \ldots, u_m)\, du_m \cdots du_2 du_1. \tag{4.5}$$

The derivation of $P(u_1, u_2, \ldots, u_m)$ can be reduced to the case of static points based on the following lemma:

LEMMA 4.1. *Let p be a m-dimensional point whose current location is p_S and velocity vector is $p_V = \{p_{V1}, p_{V2}, \ldots, p_{Vm}\}$. Given a moving query q, we formulate another query q' such that (i) its current extent q'_S and time interval q'_T are the same as q_S and q_T, and (ii) $q'_{Vi-} = q_{Vi-} - p_{Vi}$, $q'_{Vi+} = q_{Vi+} - p_{Vi}$. Then, p satisfies q, if and only if, query q' covers the static point p_S at some (future) timestamp $t \in q_T$.*

PROOF. Here we prove an even stronger statement: for *any* future $t \ge 0$, $q(t)$ covers $p(t)$ on *any* dimension $i (1 \le i \le m)$ if and only if $q'(t)$ covers static point p_S on the *same* dimension. Notice that, $q(t)$ covering $p(t)$ on dimension i means $q_{Si}(t) \le p_{Si}(t) \le q_{Si+}(t)$, or equivalently: $q_{Si-} + q_{Vi-} \cdot t \le p_{Si} + p_{Vi} \cdot t \le q_{Si+} + q_{Vi+} \cdot t$. This inequality can be re-written as: $q_{Si-} + (q_{Vi-} - p_{Vi}) \cdot t \le p_{Si} \le q_{Si+} + (q_{Vi+} - p_{Vi}) \cdot t$. Note that, $q_{Vi-} - p_{Vi}$ and $q_{Vi+} - p_{Vi}$ are exactly the velocities q'_{Vi-} and q'_{Vi+} of the transformed query q'. In other words, we have: $q'_{Si-}(t) \le p_{Si} \le q'_{Si+}(t)$, which completes the proof. □

Lemma 4.1 indicates that deciding whether a moving point p intersects a moving rectangle q can be achieved by examining the intersection between a static point p_S and a moving rectangle q', where p_S is the current location of p, and q' is formulated as described above. Intuitively, q' captures the “relative” movement between p and q, or equivalently, q' can be regarded as the representation of q in a coordinate system that remains static to p (i.e., this system moves

[4]Namely, $\int_{V_{\min\_1}}^{V_{\max\_1}} \int_{V_{\min\_2}}^{V_{\max\_2}} \cdots \int_{V_{\min\_m}}^{V_{\max\_m}} f(u_1, u_2, \ldots, u_m)\, du_m \cdots du_2 du_1 = 1$.

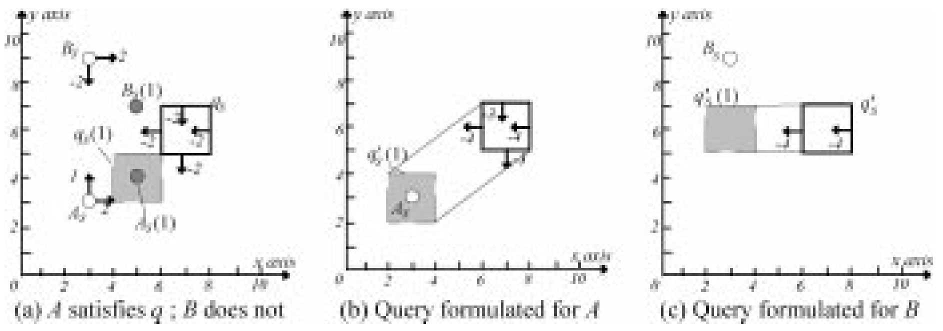

(a) A satisfies q ; B does not
(b) Query formulated for A
(c) Query formulated for B

Fig. 7. Illustration of Lemma 4.1.

at the same speed and direction as p). To illustrate this, consider Figure 7(a) which shows two moving points A, B and a moving query q with time interval $q_T = [0, 1]$. $A_S(1)$, $B_S(1)$, $q_S(1)$ correspond to the locations of points A, B, and query q at time 1, respectively. It is clear that A satisfies q, while B does not.

Figure 7(b) shows the formulated query q' in order to decide the intersection of A (observe how the velocities of q' change from those of q). In accordance with Lemma 4.1, the fact that A is a qualifying object guarantees that q' must cover static point A_S during q_T, which is indeed the case as shown in Figure 7(b). In general, given a data point p and a query q, the relative positions between $p_S(t)$ and $q_S(t)$ are always the same as those between static point p_S and the extent $q'_S(t)$ of the transformed query q' at any future time t. Figure 7(c) demonstrates the formulated query q' with respect to point B (notice that the y-velocities of q' are 0). Since B does not intersect q, by Lemma 4.1 we can infer that q' does not cover B_S.

Therefore, the probability $P(u_1, u_2, \dots, u_m)$ for a moving point p with velocities $u_1, u_2, \dots, u_m$ to intersect a query q equals the probability that the corresponding formulated query q' covers the static point p_S. Specifically, $P(u_1, u_2, \dots, u_m)$ can be represented as:

$$\begin{aligned} P(u_1, u_2, \dots, u_m) &= Sel_{static\_pt}(q'_{Si-}, q'_{Si+}, q'_{Vi-}, q'_{Vi+} q'_T) \\ &= Sel_{static\_pt}(q_{Si-}, q_{Si+}, q_{Vi-} - u_i, q_{Vi+} - u_i, q_T), \qquad (4.6) \end{aligned}$$

where Sel_{static_pt} is given by Eq (4.1). As discussed earlier, after solving $P(u_1, u_2, \dots, u_m)$, Eq (4.5) estimates the selectivity of spatio-temporal window queries on moving points. Since Eq. (4.5) involves an integral of several layers, we evaluate it numerically using the "trapezoidal rule" described in Press et al. [2002]. Specifically, to evaluate a general one-layer integral $\int_a^b f(x)\,dx$, the trapezoidal rule calculates the function values $f(x_i)$ at regular positions $x_i = a + i(b - a)/c (0 \le i \le c)$, where c is a constant (equal to 100 in our experiments) of the integral range $[a, b]$. Then, the integral value can be approximated as $\frac{b-a}{2c} \sum_{i=0}^{c-1} [f(x_i) + f(x_{i+1})]$. Extending the trapezoidal rule to multilayer integrals is straightforward, by integrating individual layers recursively. It is worth mentioning that the case of static queries over moving points discussed in Choi and Chung [2002] is merely a special instance of the problem solved above.

4.3 Moving Rectangle Data

This section analyzes moving data rectangles whose spatial length is L_i, velocity length equals LV_i, and the location and velocity of the lower boundary on each dimension i uniformly distributes in $[U_{\min\_i}, U_{\max\_i} - L_i]$ and $[V_{\min\_i}, V_{\max\_i} - LV_i]$, respectively. Similar to the analysis for moving points, we aim at deriving the probability $P(u_1, u_2, \ldots, u_m)$ that a rectangle r, whose r_{Vi-} takes specific a value u_i ($1 \le i \le m$), satisfies the query. Once $P(u_1, u_2, \ldots, u_m)$ is available, Sel_{rec} can be obtained by Eq. (4.7) (notice the changes in the upper limits of the integrals compared with Eq. (4.4)):

$$Sel_{rec}(q_{Si-}, q_{Si+}, q_{Vi-}, q_{Vi+}, q_T) = \int_{V_{\min\_1}}^{V_{\max\_1}-LV_1} \int_{V_{\min\_2}}^{V_{\max\_2}-LV_2} \cdots \int_{V_{\min\_m}}^{V_{\max\_m}-LV_m} P(u_1, u_2, \ldots, u_m) f(u_1, u_2, \ldots, u_m)\, du_m \cdots du_2 du_1. \tag{4.7}$$

Since u_i distributes uniformly in $[V_{\min\_i}, V_{\max\_i} - LV_i]$, we have:

$$f(u_1, u_2, \ldots, u_m) = f(u_1) \cdot f(u_2) \cdot \ldots \cdot f(u_m) = \prod_{i=1}^{m} \left(\frac{1}{V_{\max\_i} - LV_i - V_{\min\_i}} \right)$$

Thus, Eq. (4.7) becomes:

$$Sel_{rec}(q_{Si-}, q_{Si+}, q_{Vi-}, q_{Vi+}, q_T) = \prod_{i=1}^{m} \left(\frac{1}{V_{\max\_i} - LV_i - V_{\min\_i}} \right) \int_{V_{\min\_1}}^{V_{\max\_1}-LV_1} \int_{V_{\min\_2}}^{V_{\max\_2}-LV_2} \cdots \int_{V_{\min\_m}}^{V_{\max\_m}-LV_m} P(u_1, u_2, \ldots, u_m) du_m \cdots du_2 du_1. \tag{4.8}$$

The following lemma reduces the intersection examination between two moving rectangles r and q to that between a static point and a transformed moving rectangle q' (in a way similar to Lemma 4.1).

LEMMA 4.2. *Let r be a m-dimensional rectangle whose current extent is $r_S = \{r_{S1-}, r_{S1+}, r_{S2-}, r_{S2+}, \ldots, r_{Sm-}, r_{Sm+}\}$ and velocity vector is $r_V = \{r_{V1-}, r_{V1+}, r_{V2-}, r_{V2+}, \ldots, r_{Vm-}, r_{Vm+}\}$. Given a moving query q with $q_S = \{q_{S1-}, q_{S1+}, q_{S2-}, q_{S2+}, \ldots, q_{Sm-}, q_{Sm+}\}$, and $q_V = \{q_{V1-}, q_{V1+}, q_{V2-}, q_{V2+}, \ldots, q_{Vm-}, q_{Vm+}\}$, we formulate another query q' such that (i) $q'_T = q_T$, (ii) $q'_{Si-} = q_{Si-} - (r_{Si+} - r_{Si-})$, $q'_{Si+} = q_{Si+}$, and (iii) $q'_{Vi-} = q_{Vi-} - r_{Vi+}$, $q'_{Vi+} = q_{Vi+} - r_{Vi-}$. Then, r satisfies q, if and only if q' covers the static point $p = (r_{S1-}, r_{S2-}, \ldots, r_{Sm-})$ (i.e., a corner point of r_S) during time interval q'_T.*

PROOF. Similar to the proof of Lemma 4.1, we prove a stronger statement: for *any* future $t \ge 0$, $q(t)$ intersects $r(t)$ on *any* dimension i ($1 \le i \le m$) if and only if $q'(t)$ covers static point $(r_{S1-}, r_{S2-}, \ldots, r_{Sm-})$ on the *same* dimension. Notice that, $q(t)$ intersects $p(t)$ on dimension i means $\max\{q_{Si-}(t), r_{Si-}(t)\} \le \min\{q_{Si+}(t), r_{Si+}(t)\}$, or equivalently: $q_{Si-}(t) - (r_{Si+}(t) - r_{Si-}(t)) \le r_{Si-}(t) \le q_{Si+}(t)$. This can be rewritten as:

$$q_{Si-} + t \cdot q_{Vi-} - [(r_{Si+} - r_{Si-}) + (r_{Vi+} - r_{Vi-}) \cdot t] \le r_{Si-} + t \cdot r_{Vi-} \le q_{Si+} + t \cdot q_{Vi+}$$
$$\Leftrightarrow [q_{Si-} - (r_{Si+} - r_{Si-})] + (q_{Vi-} - r_{Vi+}) \cdot t \le r_{Si-} \le q_{Si+} + (q_{Vi+} - r_{Vi-}) \cdot t$$

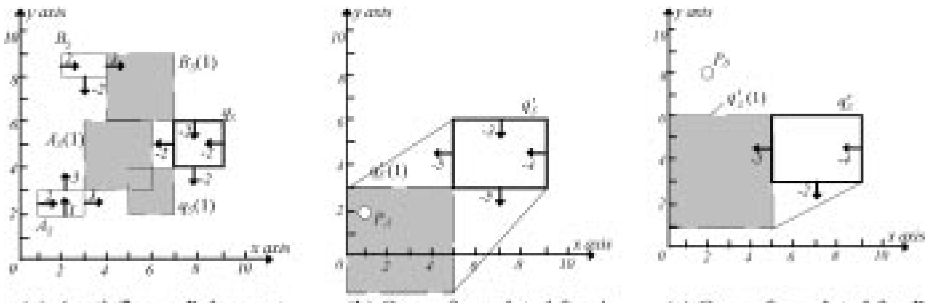

(a) A satisfies q; B does not (b) Query formulated for A (c) Query formulated for B

Fig. 8. Illustration for Lemma 4.2.

Since, $q'_{Si-} = q_{Si-} - (r_{Si+} - r_{Si-})$, $q'_{Vi-} = q_{Vi-} - r_{Vi+}$ and $q'_{Vi+} = q_{Vi+} - r_{Vi-}$, we have: $q'_{Si-}(t) \leq r_{Si-} \leq q'_{Si+}(t)$, which completes the proof. □

Consider Figure 8(a), which shows data rectangles A, B, query q (with interval $q_T = [0, 1]$), and their extents at time 1. Notice that A intersects q during q_T, while B does not. Figure 8(b) shows the transformed query q' with respect to A, as well as the lower-left corner point P_A of A_S. The current extent q'_S of q' is obtained by enlarging q_S with the size of A_S on each dimension. The value (-5) of q'_{Vx-} is computed by subtracting A_{Vx+} (3) from q_{Vx-} (-2). Since q' covers static point P_A during q_T, by Lemma 4.2 we can assert that the original rectangle A satisfies q. Similarly Figure 8(c) demonstrates the formulated query q' for B, which does not cover point P_B (lower-left corner of B_S) during q_T, indicating that B does not qualify q.

Hence, the probability $P(u_1, u_2, \ldots, u_m)$ that a moving rectangle r with $r_{Vi-} = u_i$ ($1 \leq i \leq m$) satisfies q can be represented as:

$$\begin{aligned} P(u_1, u_2, \ldots, u_m) &= Sel_{static\_pt}(q'_{Si-}, q'_{Si+}, q'_{Vi-}, q'_{Vi+}, q'_T) \\ &= Sel_{static\_pt}(q_{Si-} - L_i, q_{Si+}, q_{Vi-} - u_i - LV_i, q_{Vi+} - u_i, q_T) \end{aligned} \tag{4.9}$$

where Sel_{static_pt} is shown in Eq. (4.1), except that the volume of the universe should be modified to $\prod_{i=1}^{m}(U_{\max\_i} - L_i - U_{\min\_i})$ (i.e., the lower boundary of a data rectangle ranges in $[U_{\min\_i}, U_{\max\_i} - L_i]$). Since we have solved $P(u_1, u_2, \ldots, u_m)$, Eq. (4.8) can be used to estimate the selectivity for moving rectangles. Notice that, Lemmas 4.1 and 4.2 offer a general methodology of reducing complex STWQ selectivity estimation problems to simple ones; for example, their application to the time-oblivious approach [Choi and Chung 2002] automatically yields another method able to capture moving queries and rectangle objects. In the next section, however, we point out that this approach is erroneous in practice, by quantifying its error.

4.4 Error of the Time-Oblivious Approach

As discussed in Section 2.2, the time-oblivious approach estimates the selectivity Sel by simply taking the product of the qualifying probability Sel_i on each dimension ($1 \leq i \leq m$). Note that Sel_i can also be obtained from our derivation (i.e., the dimensionality equals 1); hence, by comparing the difference between Sel and $\prod_{i=1}^{m} Sel_i$ we can quantify the error of the time-oblivious approach. To illustrate the factors that affect the error, in the sequel we consider the case (moving points and static queries) targeted in Choi and Chong [2002], for which the resulting equations are simplest and can be solved into closed form. Specifically, given (i) a set S of 2D points such that, for each point $p \in S$, p_{Si} and p_{Vi} (i.e., its location/velocity on each dimension) uniformly distribute in $[0, U]$ and $[0, V]$, respectively, and (ii) a static query q whose extent is q_S and interval is $[0, q_{T+}]$ (i.e., a current query), the actual selectivity Sel follows Eq. (4.5), except that in this case the integral can be solved into the following closed form:

$$Sel = \frac{Vq_{T+}}{2U^2}[(q_{Sx+} - q_{Sx-}) + (q_{Sy+} - q_{Sy-})] + \frac{(q_{Sx+} - q_{Sx-})(q_{Sy+} - q_{Sy-})}{U^2} \quad (4.10)$$

The qualifying probability Sel_i on each dimension ($1 \le i \le m$) can be obtained with similar analysis:

$$Sel_i = \frac{q_{Si+} - q_{Si-}}{U} + \frac{Vq_{T+}}{2U}. \quad (4.11)$$

Thus, the estimation Sel' obtained by the time-oblivious approach is:

$$\begin{aligned} Sel' \; = \; Sel_x \cdot Sel_y = \frac{Vq_{T+}}{2U^2}[(q_{Sx+} - q_{Sx-}) + (q_{Sy+} - q_{Sy-})] \\ + \frac{(q_{Sx+} - q_{Sx-})(q_{Sy+} - q_{Sy-})}{U^2} + \frac{V^2 {q_{T+}}^2}{4U^2}. \end{aligned} \quad (4.12)$$

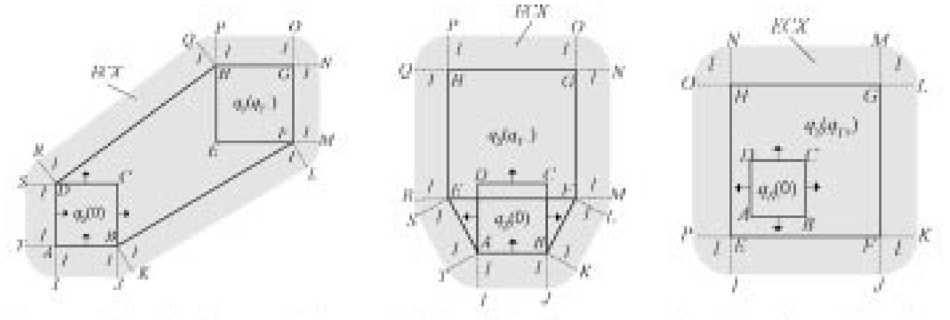
Comparing Eqs. (4.12) and (4.10), the relative error Err of Sel' is:

$$\begin{aligned} Err \; &= \; \frac{Sel' - Sel}{Sel} \\ &= \frac{V^2 {q_{T+}}^2}{2Vq_{T+}[(q_{Sx+} - q_{Sx-}) + (q_{Sy+} - q_{Sy-})] + 4(q_{Sx+} - q_{Sx-})(q_{Sy+} - q_{Sy-})}. \end{aligned} \quad (4.13)$$

Note that $(q_{Sx+} - q_{Sx-}) + (q_{Sy+} - q_{Sy-})$ and $(q_{Sx+} - q_{Sx-}) \cdot (q_{Sy+} - q_{Sy-})$ correspond to the perimeter and area of q_S, respectively. It is clear that the time-oblivious approach is accurate only when the query length $q_{T+} = 0$, because for timestamp queries ignoring the temporal condition does not cause any error: if an object satisfies a query q, then the intersection intervals on all dimensions are the same and consist of a single timestamp. The error grows, however, quadratically with q_{T+}, and the length of the velocity space V. On the other hand, the error is smaller for queries with larger extents q_S. Also, notice that the error is not affected by the length of the spatial universe.

5. NEAREST DISTANCE FOR SPATIO-TEMPORAL *K*NN SEARCH

In this section, we discuss the expected nearest distance for ST*k*NN queries assuming again uniform distribution (nonuniform data are collectively handled in Section 7.2). Adopting a methodology similar to the last section, Section 5.1 first solves the problem on static point data, and then Section 5.2 settles the



(a) Same velocity directions on both axes (b) Same direction on one axis (c) Different directions

Fig. 9. All cases of extended convex hulls $ECX(q, l)$.

general problem involving moving rectangles using reductions. Unlike previous studies (on spatial kNN) that discuss only point queries, our analysis also covers rectangle queries.

5.1 Static Point Data

Before solving general kNN queries, we consider a current rectangle query q (with interval $q_T = [0, q_{T+}]$) that retrieves a single nearest neighbor. To derive the expected nearest distance ND_1, we adopt the common paradigm [Berchtold et al. 1997; Bohm 2000] of spatial kNN analysis (reviewed in Section 2.1). We first obtain the probability $P_{static\_pt}\{dist \leq l\}$ that the minimum distance between a random data point p and a given query rectangle q during q_T (i.e., $dist(p, q, q_T)$) is smaller than a constant l. For this purpose, let us define, in the same way as Figure 4, $CX(q)$ as the convex hull of the vertices of $q_S(0)$ and $q_S(q_{T+})$, which are the extents of q at the current time 0 and q_{T+}, respectively. Further, we introduce the concept of the *extended convex hull* $ECX(q, l)$ which enlarges $CX(q)$ with length l on all directions. The extended convex hull is motivated by the *Minkowski sum*, a well-studied concept in computational geometry [Berg et al. 1997], which is popular in query analysis [Kamel and Faloutsos 1993; Pagel et al. 1993; Berchtold et al. 1997; Bohm 2000]. A useful property of the *Minkowski sum* is that it facilitates the adaptation of the proposed (i.e., Euclidean) solutions to other metrics.

Figure 9 shows the three types of ECX that correspond to the possible shapes of CX illustrated in Figure 4. In Figure 9(a), $ECX(q, l)$ consists of (i) $CX(q)$, (ii) rectangles $ADST$, $AIJB$, $GFMN$, $HGOP$ (we call them *side extensions* because they are extended from edges of $q_S(0)$ or $q_S(q_{T+})$), (iii) rectangles $BKLF$, $RDHQ$ (we call them *trajectory extensions* because they are extended from the trajectories of the corner points B, D), and (iv) a circle with radius l (which is the composition of six separate arcs with the same radius but centering at A, B, F, G, H, D, respectively). Similarly, $ECX(q, l)$ in Figure 9(b) consists of (i) $CX(q)$, (ii) side extensions $AIJB$, $FMNG$, $HGOP$, $REHQ$, (iii) trajectory extensions $STAE$, $BKLF$, and (iv) a circle with radius l. On the other hand, $ECX(q, l)$ in Figure 9(c) does not contain any trajectory extension (i.e., it involves $CX(q)$, side extensions $EIJF$, $FKLG$, $GMNH$, $OPEH$, and a circle).

```
Algorithm compute_ECX_area($q$, $l$)
1. $sum$=compute_CX_vol($q$)
/* invoke the function in Figure 5 */
2. $sum$=$sum$+compute_side_ext($q$,$l$)
3. $sum$=$sum$+compute_traj_ext($q$,$l$)
4. $sum$=$sum$+$\pi \cdot l^2$ /* the area of the circle */
5. return sum
End compute_ECX_area

Algorithm compute_side_ext($q$, $l$)
1. $sum$=0
2. for each dimension $1 \le i \le 2$
3.     $j$=2−$i$
4.     if ($q_{Vi-} \le 0$ & $q_{Vi\cdot} \le 0$) or ($q_{Vi-} \ge 0$ & $q_{Vi\cdot} \ge 0$)
5.         $sum$=$sum$+$l \cdot [2 \cdot q_{Sj\cdot} - 2 \cdot q_{Sj-} +$
           $(q_{Vj\cdot} - q_{Vj-}) \cdot (q_{T\cdot} - q_{T-})$
6.     if $q_{Vi-} < 0$ and $q_{Vi\cdot} > 0$
           $sum$=$sum$+$2 \cdot l \cdot [q_{Sj\cdot} - q_{Sj-} +$
           $(q_{Vj\cdot} - q_{Vj-}) \cdot (q_{T\cdot} - q_{T-})]$
7. return $sum$
End compute_side_ext

Algorithm compute_traj_ext($q$, $l$)
1. $sum$=0
2. if ($q_{V1-} \ge 0$, $q_{V1\cdot} \ge 0$, and $q_{V2-} < 0$) or
   ($q_{V1-} < 0$, $q_{V2-} \ge 0$, and $q_{V2\cdot} \ge 0$)
3.     $sum$=$sum$+$l \cdot (q_{T\cdot} - q_{T-}) \cdot \sqrt{q_{V1-}^{\ 2} + q_{V2-}^{\ 2}}$
4. if ($q_{V1\cdot} \ge 0$, $q_{V2-} \ge 0$, and $q_{V2\cdot} \ge 0$) or
   ($q_{V1-} < 0$, $q_{V1\cdot} < 0$, and $q_{V2-} < 0$)
5.     $sum$=$sum$+$l \cdot (q_{T\cdot} - q_{T-}) \cdot \sqrt{q_{V1+}^{\ 2} + q_{V2-}^{\ 2}}$
6. if ($q_{V1\cdot} \ge 0$, $q_{V2-} < 0$, and $q_{V2\cdot} < 0$) or
   ($q_{V1-} < 0$, $q_{V1\cdot} < 0$, and $q_{V2\cdot} \ge 0$)
7.     $sum$=$sum$+$l \cdot (q_{T\cdot} - q_{T-}) \cdot \sqrt{q_{V1+}^{\ 2} + q_{V2+}^{\ 2}}$
8. if ($q_{V1-} \ge 0$, $q_{V1\cdot} \ge 0$, and $q_{V2\cdot} \ge 0$) or
   ($q_{V1-} < 0$, $q_{V2-} < 0$, and $q_{V2\cdot} < 0$)
9.     $sum$=$sum$+$l \cdot (q_{T\cdot} - q_{T-}) \cdot \sqrt{q_{V1-}^{\ 2} + q_{V2+}^{\ 2}}$
10. return $sum$
End compute_traj_ext
```

Fig. 10. Computing the area of $ECX(q, l)$ (for two-dimensional space).

A crucial observation is that, $dist(p, q, q_T) \le l$, if and only if, p falls in $ECX(q, l)$ and hence, for uniform data, the probability $P_{static\_pt}\{dist \le l\}$ equals the area of $ECX(q, l)$ divided by U_{vol} (i.e., that of the data space). Figure 10 illustrates the algorithms for computing this area in 2D space which, although seemingly complex, can be written as a closed quadratic function of l. Specifically, function *compute_side_ext* (*compute_traj_ext*) returns the total area of all the side (trajectory) extensions, which is a linear function of l, while the total area of $ECX(q, l)$ also includes those of $CX(q)$ (independent of l) and a circle (quadratic with l).

$ECX(q, l)$ in higher dimensionality, however, is more complicated, and the derivation of its volume leads to complex analysis beyond the scope of this article. Instead, we once again resort to the Monte-Carlo method, by first generating α uniform points in the universe (in our experiments, $\alpha = 2000$), and then counting the number β of points that fall in ECX. In particular, a point is in ECX, if and only if, its distance to $q_S(t)$ is smaller than l for some time $t \in q_T$, which can be decided using the algorithm in Benetis et al. [2002]. Then, the volume of $ECX(q, l)$ is computed as $\beta/\alpha \cdot U_{vol}$. Similar to Figure 6, if part of $ECX(q, l)$ falls outside the spatial universe, only the intersection region (between $ECX(q, l)$ and DS) should be considered. In this case, the Monte-Carlo method is always invoked.

Having computed $P_{static\_pt}\{dist \le l\}$, we proceed to derive $P_{static\_pt}\{ND_1 \le l\}$, the probability that the nearest distance ND_1 of q is smaller than l, or equivalently, there exists at least one data point p such that $dist(p, q, q_T) \le l$. Assuming that the dataset contains N points, $P_{static\_pt}\{ND_1 \le l\}$ can be derived as:

$$P_{static\_pt}\{ND_1 \le l\} = 1 - (1 - P_{static\_pt}\{dist \le l\})^N. \tag{5.1}$$

Taking the derivative of $P_{static\_pt}\{ND_1 \le l\}$ (with respect to l), we can obtain its probability density function $p_{static\_pt}(ND_1 = l)$. As a result, the expected nearest distance ND_1 can be represented as:

$$ND_1 = \int_0^{\infty} l \cdot p_{static\_pt}(ND_1 = l)\, dl. \tag{5.2}$$

This equation can also be evaluated numerically using the "trapezoidal rule" described in Section 4.2. The above analysis can be extended to kNN retrieval ($k > 1$). The difference is that the nearest distance $ND_k \le l$, if and only if, there exist at least k objects such that their distances to q during q_T are smaller than l. To derive $P_{static\_pt}\{ND_k \le l\}$ (again, from $P_{static\_pt}\{dist \le l\}$), we consider the complementary probability $P_{static\_pt}\{ND_k > l\}$, which equals the probability that at most $k-1$ objects are within distance l to q. Towards this, we further distinguish k cases where there are exactly $0, 1, \ldots, k-1$ objects within distance l, respectively; $P_{static\_pt}\{ND_k > l\}$ corresponds to sum of the probabilities of all cases. Specifically, the probability for exactly i objects ($0 \le i \le k-1$) is $\binom{N}{i}(P_{static\_pt}\{dist \le l\})^i(1 - P_{static\_pt}\{dist \le l\})^{N-i}$; thus, $P_{static\_pt}\{ND_k > l\}$ and $P_{static\_pt}\{ND_k \le l\}$ are given by:

$$P_{static\_pt}\{ND_k > l\} = \sum_{i=0}^{k-1} \left\{ \binom{N}{i} (P_{static\_pt}\{dist \le l\})^i (1 - P_{static\_pt}\{dist \le l\})^{N-i} \right\} \tag{5.3}$$

$$P_{static\_pt}\{ND_k \le l\} = 1 - P_{static\_pt}\{ND_k > l\}. \tag{5.4}$$

Then, ND_k is derived as in Eq. (5.2), except that $P\{ND_1 \le l\}$ and $p(ND_1 = l)$ should be replaced with $P\{ND_k \le l\}$ and $p(ND_k = l)$, respectively. Finally, the above discussion also applies to noncurrent queries (i.e., $q_{T-} \ne 0$), where ECX should be computed based on $q_S(q_{T-})$ and $q_S(q_{T+})$.

5.2 Moving Rectangles

In this section, we discuss the expected nearest distance ND_k of a kNN query for general moving rectangle objects, by reducing the problem to the basic case (on static points) solved in the previous section. Specifically, we start from single NN retrieval, and focus on deriving the probability $P_{rec}\{dist \le l\}$ that $dist(r, q, q_T)$ is smaller than constant l, based on the assumption that the lower boundary velocity r_{Vi-} of a data rectangle r is uniform in the range $[V_{\min\_i}, V_{\max\_i} - LV_i]$ on the ith dimension (where LV_i is the velocity length). Towards this, we first calculate the probability $P(u_1, u_2, \ldots, u_m)$ that $dist(r, q, q_T)$ is smaller than l, given that r_{Vi-} takes a specific value u_i. Once $P(u_1, u_2, \ldots, u_m)$ is available, the overall probability $P_{rec}\{dist \le l\}$, which considers all velocity values for r_{Vi-}, can be represented as (similar to Eq. (4.8)):

$$P_{rec}\{dist \le l\} = \prod_{i=1}^{m} \left(\frac{1}{V_{\max\_i} - LV_i - V_{\min\_i}} \right) \int_{V_{\min\_1}}^{V_{\max\_1} - LV_1} \int_{V_{\min\_2}}^{V_{\max\_2} - LV_2} \cdots \int_{V_{\min\_m}}^{V_{\max\_m} - LV_m} P(u_1, u_2, \ldots, u_m)\, du_m \cdots du_2 du_1. \tag{5.5}$$

The analysis of $P(u_1, u_2, \ldots, u_m)$ can be reduced to the static problem as indicated in the following lemma (analogous to Lemmas 4.1 and 4.2):

LEMMA 5.1. *Let r be a m-dimensional rectangle whose current extent is* $r_S = \{r_{S1-}, r_{S1+}, r_{S2-}, r_{S2+}, \ldots, r_{Sm-}, r_{Sm+}\}$ *and velocity vector is* $r_V = \{r_{V1-}, r_{V1+}, r_{V2-}, r_{V2+}, \ldots, r_{Vm-}, r_{Vm+}\}$. *Given a moving query q with* $q_S = \{q_{S1-}, q_{S1+}, q_{S2-}, q_{S2+}, \ldots, q_{Sm-}, q_{Sm+}\}$, *and* $q_V = \{q_{V1-}, q_{V1+}, q_{V2-}, q_{V2+}, \ldots, q_{Vm-}, q_{Vm+}\}$, *we formulate another query* q' *such that (i)* $q'_T = q_T$, *(ii)* $q'_{Si-} = q_{Si-} - (r_{Si+} - r_{Si-})$, $q'_{Si+} = q_{Si+}$, *and (iii)* $q'_{Vi-} = q_{Vi-} - r_{Vi+}$, $q'_{Vi+} = q_{Vi+} - r_{Vi-}$. *Then, the minimum distance between r and q is smaller than l at some time* $t \in q_T$, *if and only if, the static point* $p = (r_{S1-}, r_{S2-}, \ldots, r_{Sm-})$ *(i.e., a corner point of* r_S*) is within distance l from* $q'(t)$ *at time t.*

PROOF. We briefly review the computation of the distance between two m-dimensional rectangles r_S and q_S, since it is fundamental to the proof for the lemma. The distance $dist_i(r_S, q_S)$ between r_S and q_S on the ith dimension $(1 \le i \le m)$ is defined as [Roussopoulos et al. 1995]:

$$dist_i(r_{Si}, q_{Si}) = \begin{cases} (q_{Si-} - r_{Si+}) & \text{if} \quad q_{Si-} > r_{Si+} \\ (r_{Si-} - q_{Si+}) & \text{if} \quad q_{Si+} < r_{Si-} \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

where $[r_{Si-}, r_{Si+}]$ denotes the extent of r_S on the ith dimension (similarly, $[q_{Si-}, q_{Si+}]$ represents the same information for q_S). Thus, the distance between r_S and q_S is obtained as $[\sum_i (dist_i(r_{Si}, q_{Si}))^2]^{1/2}$.

Given a transformed query q', we prove a stronger statement: for *any* future $t \ge 0$, the distance $dist_i(r_{Si}(t), q_{Si}(t))$ between $r_{Si}(t)$ and $q_{Si}(t)$ on *any* dimension i $(1 \le i \le m)$ equals that (denoted as $dist_i(r_{Si-}, q'_{Si}(t))$) between $q'_{Si}(t)$ and r_{Si-}(i.e., the ith coordinate of static point $(r_{S1-}, r_{S2-}, \ldots, r_{Sm-})$). For this purpose, we write $dist_i(r_{Si}(t), q_{Si}(t))$ as follows (in accordance with Eq. (5.6)):

$$dist_i(r_{Si}(t), q_{Si}(t)) = \begin{cases} (q_{Si-}(t) - r_{Si+}(t)) & \text{if} \quad q_{Si-}(t) > r_{Si+}(t) \\ (r_{Si-}(t) - q_{Si+}(t)) & \text{if} \quad q_{Si+}(t) < r_{Si-}(t) \\ 0 & \text{otherwise.} \end{cases} \Leftrightarrow$$

$$dist_i(r_{Si}(t), q_{Si}(t)) \\ = \begin{cases} (q_{Si-} + q_{Vi-} \cdot t - r_{Si+} - r_{Vi+} \cdot t) & \text{if} \quad q_{Si-} + q_{Vi-} \cdot t > r_{Si+} + r_{Vi+} \cdot t \\ (r_{Si-} + r_{Vi-} \cdot t - q_{Si+} - q_{Vi+} \cdot t) & \text{if} \quad r_{Si-} + r_{Vi-} \cdot t > q_{Si+} + q_{Vi+} \cdot t \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

In the same way, $dist_i(r_{Si-}, q_{Si}{}'(t))$ can be written as follows:

$$dist_i(r_{Si-}, q'_{Si}(t)) = \begin{cases} (q'_{Si-} + q'_{Vi-} \cdot t - r_{Si-}) & \text{if} \quad q'_{Si-} + q'_{Vi-} \cdot t > r_{Si-} \\ (r_{Si-} - q'_{Si+} - q'_{Vi+} \cdot t) & \text{if} \quad r_{Si-} > q'_{Si+} + q'_{Vi+} \cdot t \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

It is easy to verify that, by substituting q' with q as stated in the lemma, Eqs. (5–7) and (5–8) turn out to be exactly the same, which completes the proof. □

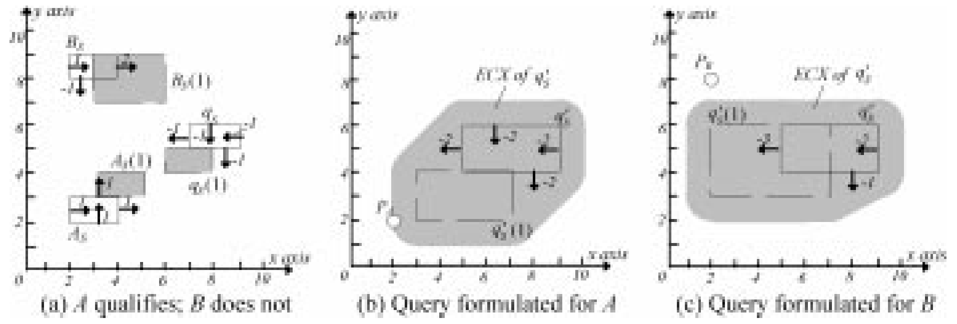


Fig. 11. Illustration of Lemma 5.1 (distance constant $l = 1$).

Figure 11 illustrates the lemma with moving rectangles A, B, and query q, assuming that $q_T = [0, 1]$ and $l = 1$. As shown in Figure 11(a), at time 1 the minimum distance between $A_S(1)$ and $q_S(1)$ equals 1, while the distance between $B_S(t)$ and $q_S(t)$ is larger than 1 at any $t \in q_T$. Figure 11(b) shows the transformed query q' for A together with the corresponding $ECX(q')$ (obtained from the vertices of $q'_S(0)$ and $q'_S(1)$ in the same way as Figure 9). Since $dist(A, q, q_T) \leq 1$, according to the lemma we can assert that $ECX(q')$ covers the corner P_A of A_S (which is indeed the case as shown in the figure). Similarly, Figure 11(c) demonstrates the formulated query q' for B. Notice that in this case $ECX(q')$ does not cover the corner P_B of B_S, confirming the fact that $dist(B, q, q_T) > 1$.

Lemma 5.1 converts the minimum distance computation between two moving rectangles, to that between a static point and a transformed moving rectangle. Consequently, given a data rectangle r whose r_{Vi-} takes specific value u_i, the probability $P(u_1, u_2, \ldots, u_m)$ (that $dist(r, q, q_T)$ is smaller than l) can be computed as:

$$\begin{aligned} P(u_1, u_2, \ldots, u_m) &= P_{static\_pt}\{dist \leq l, q'_{Si-}, q'_{Si+}, q'_{Vi-}, q'_{Vi+}, q'_T\} \\ &= P_{static\_pt}\{dist \leq l, q_{Si-} - L_i, q_{Si+}, q_{Vi-} - u_i - LV_i, \\ &\quad q_{Vi+} - u_i, q_T\}, \end{aligned} \tag{5.9}$$

where q' is obtained as described in the lemma, and $P_{static\_pt}\{dist \leq l\}$ is computed as in Figure 10. Having derived $P(u_1, u_2, \ldots, u_m)$, we can solve $P_{rec}\{dist \leq l\}$ using Eq. (5.5), and then obtain the expected nearest distance ND_1 with Eqs. (5.1) and (5.2). The extension of the above analysis to general STkNN ($k > 1$) is trivial: we only need to substitute $P_{static\_pt}\{dist \leq l\}$ in Eq. (5.3) with $P_{rec}\{dist \leq l\}$ (computed by Eq. (5.5)).

6. SELECTIVITY ESTIMATION FOR SPATIO-TEMPORAL JOIN (STJ)

Given two m-dimensional datasets S_1, S_2, a constant d and an interval q_T, STJ reports all object pairs (o_1, o_2) from $S_1 \times S_2$, such that $dist(o_1, o_2, q_T) \leq d$. The data and velocity spaces for S_1 (S_2) are $[U_{1,\min\_i}, U_{1,\max\_i}]$ ($[U_{2,\min\_i}, U_{2,\max\_i}]$) and $[V_{1,\min\_i}, V_{\max\_i}]$ ($[V_{2,\min\_i}, V_{2,\max\_i}]$) (i.e., we allow datasets with different universes), and each object of S_1 (S_2) has spatial length $L_{1,i}$ ($L_{2,i}$) and velocity length $LV_{1,i}$ ($LV_{2,i}$) along the ith dimension. The goal of our analysis is to derive

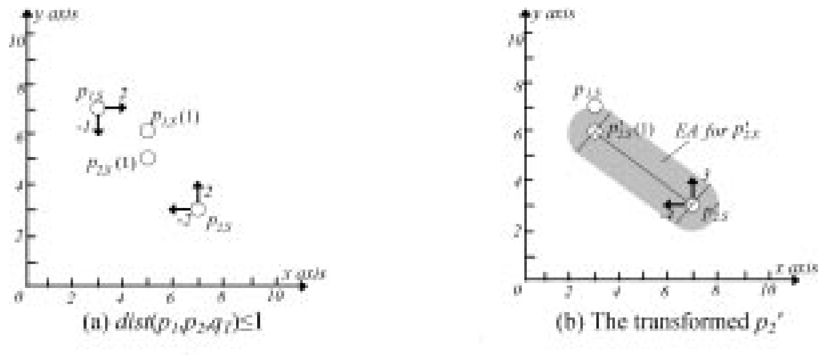


Fig. 12. Calculating $P(u_i, v_i, y_i)$ for point data ($d = 1, q_T = 1$).

the selectivity of the join, which also corresponds to the probability that a pair of objects (o_1, o_2) satisfies the join predicate. In the sequel, we first discuss the case where (i) the query interval q_T starts from the current time 0 (i.e., a current join) and (ii) S_1 and S_2 consist of only point data (i.e., $L_{1,i} = L_{2,i} = 0$, and $LV_{1,i} = LV_{2,i} = 0$); then we solve the general problem involving rectangles and arbitrary q_T.

Consider a pair of points $(p_1, p_2) \in S_1 \times S_2$ such that the velocities of p_1 and p_2 take specific values $(u_1, u_2, \ldots, u_m)$, $(v_1, v_2, \ldots, v_m)$, respectively, and the initial position of p_2 is fixed to $(y_1, y_2, \ldots, y_m)$, while that of p_1 uniformly distributes in the data space. To compute the probability $P(u_i, v_i, y_i)$ that (p_1, p_2) satisfies the join predicate, we convert p_2 to another point p_2' such that (i) the current location of p_2' is the same as p_2, and (ii) $p'_{2,Vi} = p_{2,Vi} - p_{1,Vi}$ (i.e., we subtract the velocity of p_1) on the ith dimension. Then, based on Lemma 5.1, we assert that $dist(p_1, p_2, q_T) \leq d$ if and only if the distance between $p'_{2,S}(t)$ and a static point $p_{1,S}(0)$ (i.e., the current location of p_1) is smaller than d at some timestamp t during q_T. Figure 12(a) illustrates an example where $d = 1, q_T = [0, 1]$, and p_1, p_2 satisfy the join predicate because at q_{T+} their distance is 1. Figure 12(b) shows the transformed point p_2', whose distance from the static point $p_{1,S}$ is also 1 at q_{T+}.

The implication is that, (p_1, p_2) is a result pair if and only if the current location of p_1 lies in the extended area $EA(p_2')$ of p_2', which is obtained by enlarging the trajectory of p_2' (during q_T) with length d (the shaded area in Figure 12(b)). Since the location of p_1 uniformly distributes in the data space, the probability $P(u_i, v_i, y_i)$ (that (p_1, p_2) qualifies the join condition) equals the area of $EA(p_2')$ divided by U_{vol} (the volume of the data space). In 2D, the area of $EA(p_2')$ is the sum of a rectangle and a circle. In 3D, the volume of $EA(p_2')$ is the sum of a cylinder and a sphere. In arbitrary dimensionality m, the volume of $EA(p_2')$ can be computed as:

$$Volume[EA(p_2')] = \frac{\sqrt{\pi^m}}{\Gamma(m/2+1)} d^m + \frac{\sqrt{\pi^{m-1}}}{\Gamma\left(\frac{m-1}{2}+1\right)} d^{m-1} \cdot (q_{T+} - q_{T-}) \sqrt{\sum_{i=1}^{m} (p'_{2,Vi})^2} \tag{6.1}$$

where[5]

$$\Gamma(x+1) = x \cdot \Gamma(x), \Gamma(1) = 1, \Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

For 2D, the above equation can be simplified into:

$$Area[EA(p_2')] = \pi d^2 + 2d \cdot (q_{T+} - q_{T-})\sqrt{(p'_{2,V1})^2 + (p'_{2,V2})^2}. \tag{6.2}$$

So far we have assumed that $EA(p_2')$ lies completely in the data space of S_1, in which case $EA(p_2')$ does not depend on the location of p_2. If part of $EA(p_2')$ falls outside the universe, we should compute the intersection between $EA(p_2')$ and the universe.

Remember that $P(u_i, v_i, y_i)$ only captures the probability that (p_1, p_2) satisfies the join when the velocities of p_1, p_2 and the location of p_2 take specific values (while the location distribution of p_1 is uniform). The overall probability that an arbitrary point pair satisfies the join predicate equals the average of $P(u_i, v_i, y_i)$ over all possible values of u_i, v_i, and y_i. This probability also corresponds to the selectivity Sel_{pt} of STJ, and can be formally represented by Eq. 6.3 ($V_{2,\max\_i}$ denotes the maximum velocity on dimension i for dataset S_2, $V_{2,\min\_i}$ the minimum velocity, etc.).

$$\begin{aligned} Sel_{pt} = {} & \left[\prod_{i=1}^{m}\left(\frac{1}{U_{2,\max\_i} - U_{2,\min\_i}}\right)\right]\left[\prod_{i=1}^{m}\left(\frac{1}{V_{1,\max\_i} - V_{1,\min\_i}}\right)\right] \\ & \left[\prod_{i=1}^{m}\left(\frac{1}{V_{2,\max\_i} - V_{2,\min\_i}}\right)\right]\cdot \\ & \int_{U_{2,\min\_1}}^{U_{2,\max\_1}} \cdots \int_{U_{2,\min\_m}}^{U_{2,\max\_m}} \int_{V_{1,\min\_m}}^{V_{1,\max\_m}} \cdots \int_{V_{1,\min\_m}}^{V_{1,\max\_m}} \int_{V_{2,\min\_1}}^{V_{2,\max\_1}} \cdots \int_{V_{2,\min\_m}}^{V_{2,\max\_m}} \\ & P(u_i, v_i, y_i)\, dv_m \cdots dv_1 du_m \cdots du_1 dy_m \cdots dy_1. \end{aligned} \tag{6.3}$$

Next we discuss STJ for moving rectangles. Similar to the point case, we first consider an object pair $(r_1, r_2) \in S_1 \times S_2$ such that (i) r_1 uniformly distributes in the data space, while $r_{2,i-}$ (i.e., the lower boundary coordinate of r_2 on the ith axis) is fixed to y_i, and (ii) the lower boundary velocities of r_1 and r_2 take specific values $(u_1, u_2, \ldots, u_m)$ and $(v_1, v_2, \ldots, v_m)$, respectively. Let $P(u_i, v_i, y_i)$ be the probability that (r_1, r_2) satisfies the join predicate under these conditions. To derive $P(u_i, v_i, y_i)$, we formulate another object r_2' such that (i) r_2' has the same current extent as r_2, and (ii) $r'_{2,Vi+} = r_{2,Vi+} - r_{1,Vi-}$, $r'_{2,Vi-} = r_{2,Vi-} - r_{1,Vi+}$. By Lemma 5.1, $dist(r_1, r_2, q_T) \le d$ if and only if the distance between $r'_{2,S}(t)$ and the static point $(r_{1,S1-}, r_{1,S2-}, \ldots, r_{1,Sm-})$ (i.e., a corner point of $r_{1,S}(0)$) is less than d for some time $t \in q_T$. Thus, $P(u_i, v_i, y_i)$ equals the volume of $ECX(r_2')$ divided by U_{vol}. After obtaining $P(u_i, v_i, y_i)$, the join selectivity Sel_{rec} can be

[5]Γ is a common function used to describe the volume of a sphere in arbitary dimensionality [Bohm 2000].

computed by integrating all possible values for u_i, v_i, y_i, or specifically:

$$Sel_{rec} = \left[\prod_{i=1}^{m}\left(\frac{1}{U_{2,\max\_i} - U_{2,\min\_i} - L_{2,i}}\right)\right]\left[\prod_{i=1}^{m}\left(\frac{1}{V_{1,\max\_i} - V_{1,\min\_i} - LV_{1,i}}\right)\right]$$
$$\left[\prod_{i=1}^{m}\left(\frac{1}{V_{2,\max\_i} - V_{2,\min\_i} - LV_{1,i}}\right)\right]. \tag{6.4}$$
$$\int_{U_{2,\min\_1}}^{U_{2,\max\_1}-L_{2,1}} \cdots \int_{U_{2,\min\_m}}^{U_{2,\max\_m}-L_{2,m}} \int_{V_{1,\min\_1}}^{V_{1,\max\_m}-LV_{1,1}} \cdots \int_{V_{1,\min\_m}}^{V_{1,\max\_m}-LV_{1,m}} \int_{V_{2,\min\_1}}^{V_{2,\max\_1}-LV_{2,1}} \cdots$$
$$\int_{V_{2,\min\_m}}^{V_{2,\max\_m}-LV_{2,m}} P(u_i, v_i, y_i)dv_m \cdots dv_1 du_m \cdots du_1 dy_m \cdots dy_1.$$

Finally, the above discussion can be easily extended to queries whose time intervals do not start from the current time, in which case the areas of the extended regions (for point data) and extended convex hulls (for rectangle data) should be computed based on $q_S(q_{T-})$.

7. SPATIO-TEMPORAL HISTOGRAMS AND NONUNIFORM ESTIMATION

Next we extend the results to nonuniform data using spatio-temporal histograms (STHs). Specifically, Section 7.1 introduces the STH and discusses its incremental maintenance. Section 7.2 explains nonuniform estimation for each query type.

7.1 Incremental Spatio-Temporal Histograms

The objective of a histogram is to partition the data space into a set of buckets $b_1, b_2, \ldots, b_h$ (h is the number of buckets) such that the data distribution inside each bucket is uniform. A bucket b_j ($1 \le j \le h$) has spatial extents b_j.MBR, and velocity ranges b_j.VBR (where VBR stands for velocity bounding rectangle). In general, a m-dimensional dataset requires a $2m$-dimensional STH. Figure 13(a) illustrates a STH with 4 buckets, assuming that the data space contains only one dimension (i.e., $m = 1$). The MBR of b_1, for example, is [0, 40], while its VBR covers velocities [−20, 20] (i.e., the minimum and maximum velocity among all points in the bucket). Point p belongs to b_2, because its coordinate $p_S = 30$ and velocity $p_V = 25$ fall in b_2.MBR and b_2.VBR, respectively. Moving intervals (hyper-rectangles in higher dimensions), on the other hand, are assigned according to the coordinates and velocities of their centroids. For instance, interval r (with spatial extent [30, 60] and velocity extent [10, 20]) is allocated to bucket b_4, which contains the coordinate 45 and velocity 15 of its centroid.

In addition to MBR and VBR, each bucket b_j also stores (i) the number $b_j.num$ of assigned objects, and (ii) for hyper-rectangles, the sum of spatial $b_j.L_i$ and velocity $b_j.LV_i$ lengths of these objects along each dimension ($1 \le i \le m$). Similar to moving objects, the MBR of b_j also grows according to its VBR, and in the sequel we denote its MBR at future timestamp t as b_j.MBR(t). Such a STH can be constructed using any existing algorithm for conventional

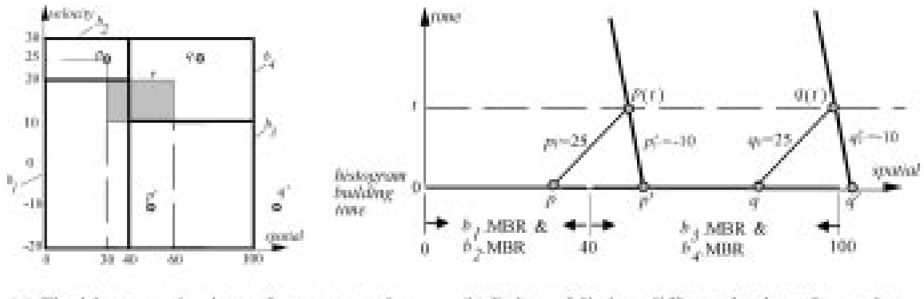

(a) The histogram buckets after construction
(b) Point p falls in a different bucket after update

Fig. 13. The spatio-temporal histogram and its update.

multidimensional histograms, by treating a m-dimensional moving object as a $2m$-dimensional box. Finally, note that STH differs from the histogram presented in Choi and Chung [2002] (see Section 2.2), which constructs the buckets by considering only objects' spatial MBRs. As shown in Tao et al. [2003b], ignoring the velocities during partitioning, leads to significant error. Further, STH generalizes the solution of Hadjieleftheriou et al. [2003] because it (i) supports also rectangle objects (Hadjieleftheriou et al. [2003] only considers point data), and (ii) can be incrementally maintained (Hadjieleftheriou et al. [2003] does not address dynamic maintenance) discussed as follows.

Assume that the histogram of Figure 13(a) is constructed at time 0, and point p updates its velocity (from 25 to -10) at some future time t (when its position is $p(t)$). After the change p does not belong to bucket b_2 any more, because its new velocity falls out of b_2.VBR [20, 30]. Furthermore, p cannot be inserted to the bucket that contains its current position $p(t)$ and velocity (-10), since the histogram is based on information at time 0 (meaning that future object positions are calculated based on the time elapsed with respect to time 0). To decide the new bucket for p, we must find its *projection point* p' at the histogram construction time (0), such that p' will reach the same position $p(t)$ with the updated velocity. To illustrate this, consider Figure 13(b), where the velocity of a point is represented as the slope of its trajectory. The projection point p' is the intersection of the spatial axis and the line with slope 10 that crosses $p(t)$, which spatially belongs to buckets b_3 and b_4, but only b_3.VBR covers the new velocity value.[6] To reflect the change, we should update $b_2.num$ ($= b_2.num - 1$) and $b_2.LV$ ($= b_2.LV - 25$), and modify b_3 accordingly ($b_3.num{+}= 1$, $b_3.LV{-}= 10$).

In some cases a projection point may fall outside the spatial universe leading to the enlargement of a boundary bucket. As an example, consider that in Figure 13 another point q also changes its velocity (from 25 to -10) at time t. Then, the MBR of b_3 must be expanded to cover the projection point q' (of q) as in Figure 13(b). Such bucket expansion will occur more frequently as the

[6]Here we assume the bucket extents are disjoint, which holds for many histograms (e.g., *minskew* [Acharya et al. 1999] used in our experiments), so that the bucket containing the projection point is unique. For histograms without this property, there may be multiple candidate buckets, in which case the final bucket can be selected randomly.

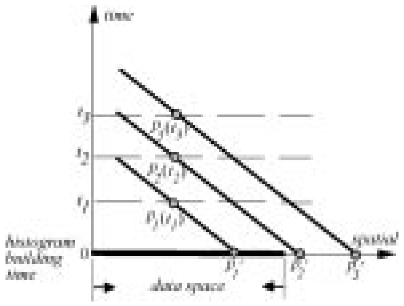


Fig. 14. Updated location falls out of the data space.

time progresses further. This is illustrated in Figure 14, where points p_1, p_2, p_3 change their velocities (to the same value) at the same spatial location, but at different timestamps $t_1 < t_2 < t_3$, respectively. Notice that, although the projection p'_1 (of p_1) lies inside the data space, p'_2 and p'_3 fall outside. This indicates that, after a sufficiently long period, most projection points will be outside the space, in which case the buckets covering inner areas become useless and the histogram efficiency drops. To avoid this, the histogram must be rebuilt periodically to reduce the difference between the current time and the histogram construction time.

Instead of performing a *physical rebuild* that needs detailed information about all objects, we propose a *logical rebuild* algorithm that creates a new histogram STH_{new} by only reading the original histogram STH_{old} (i.e., without scanning the data file). Specifically, the MBRs and VBRs of the buckets in STH_{new} are the same as those in STH_{old} except that, if a MBR (in STH_{old}) has been expanded, the part outside the data space is discarded (i.e., the bucket MBRs in STH_{new} are the same as in the initial histogram at time 0). Then, the algorithm estimates the new statistics for each bucket in STH_{new} at the current time T_C. To derive the number $b_1.num$ of objects in bucket $b_1 \in STH_{new}$, for example, we examine each bucket b_2 in the original histogram STH_{old}, and compute the number $N_{mig}(b_1, b_2, T_C)$ of objects that are originally in b_2, but covered by b_1 at time T_C (we say these objects *migrate* from b_2 to b_1). Then, $b_1.num$ is the sum of $N_{mig}(b_1, b_2, T_C)$ for all buckets $b_2 \in STH_{old}$. Similarly, $b_1.L_i$ and $b_1.LV_i$ (i.e., the sum of spatial and velocity lengths of objects in b_1) are estimated as the weighted sum in lines 10–11 of the logical rebuild algorithm shown in Figure 15.

Next we derive the expected number $N_{mig}(b_1, b_2, T_C)$ of objects migrated to b_1 from b_2. In particular, we focus on the probability $P_{mig}(b_1, b_2, T_C)$ that an object in b_2 can migrate to b_1. Let $[b_{1,Si-}, b_{1,Si+}]$ ($[b_{1,Vi-}, b_{1,Vi+}]$) be the extent of b_1.MBR (b_1.VBR) on the ith dimension (similar notation is used for b_2). We say that a object o in b_2 *partially migrates* on the ith axis at time T_C if $o_{Si}(T_C) \in [b_{1,Si-}, b_{1,Si+}]$ and $o_{Vi} \in [b_{1,Vi-}, b_{1,Vi+}]$. Notice that an object migrates if and only if it partially migrates along all dimensions. Hence, let $P_{mig\_i}(b_1, b_2, T_C)$ be the probability of partial migration on the ith axis; $P_{mig}(b_1, b_2, T_C)$ can be represented as:

$$P_{mig}(b_1, b_2, T_C) = \prod_{i=1}^{m} P_{mig\_i}(b_1, b_2, T_C). \tag{7.1}$$

```
**Algorithm logical_rebuild** ($STH_{old}$, $T_C$)
/* this function creates a new histogram $STH_{new}$ at the current time $T_C$ from the previous histogram
$STH_{old}$ constructed at time 0 */
1.   for each bucket $STH_{new}.b_j$
2.       $STH_{new}.b_j$.MBR= $STH_{old}.b_j$.MBR $\cap DS$;
3.       $STH_{new}.b_j$.VBR = $STH_{old}$. $b_j$.VBR
4.   for each bucket $b_1$ in $STH_{new}$
5.       $b_1.num$=0; $b_1.L_i$=0; $b_1.LV_i$=0
6.       for each bucket $b_2$ in $STH_{old}$
7.           compute $b_2$.MBR($T_C$) based on $b_2$.VBR
8.           if $b_2$.MBR($T_C$) intersects $b_1$.MBR
9.               $b_1.num$= $b_1.num$+**N_mig**($b_1$, $b_2$, $T_C$) /* see the derivation of $N_{mig}(b_1,b_2,T_C)$ */
10.              $b_1.L_i$=$b_1.L_i$+[$b_2.L_i$+($b_2.LV_i$)·$T_C$]·**N_mig** ($b_1$, $b_2$, $T_C$)/$b_2.num$;
11.              $b_1.LV_i$=$b_1.LV_i$+($b_2.LV_i$)·**N_mig** ($b_1$, $b_2$, $T_C$) /$b_2.num$;
12.  return $STH_{new}$
**End logical_rebuild**
```

Fig. 15. The logical re-build algorithm for STHs.

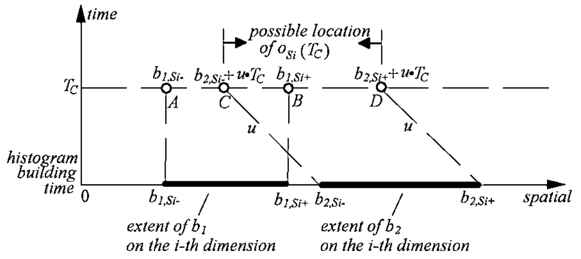


Fig. 16. Derivation of $P_{mig\_fix\_i}(u, b_1, b_2, T_C)$.

To derive $P_{mig\_i}(b_1, b_2, T_C)$, we consider the probability $P_{mig\_fix\_i}(u, b_1, b_2, T_C)$ that o partially migrates if its velocity o_{Vi} takes specific value u. Obviously, $P_{mig\_fix\_i}(u, b_1, b_2, T_C) = 0$, if $u \notin [b_{1,Vi-}, b_{1,Vi+}]$ (i.e., the velocity of o does not fall into b_1.VBR). Figure 16 illustrates the case where $u \in [b_{1,Vi-}, b_{1,Vi+}]$. Since the location o_{Si} of o uniformly distributes in $[b_{2,Si-}, b_{2,Si+}]$ (and o_{Vi} is fixed), it has equal probability to arrive at any position in segment $[C, D]$ at time T_C, where C (D) is the location reached by o if it is currently at $b_{2,Si-}(b_{2,Si+})$.

Since o partially migrates if it appears in segment $[A, B]$ (i.e., the spatial MBR of b_1), $P_{mig\_fix\_i}(u, b_1, b_2, T_C)$ equals the ratio between lengths of segments $[B, C]$ and $[C, D]$. In general, $P_{mig\_fix\_i}(u, b_1, b_2, T_C)$ is given by:

$$P_{mig\_fix\_i}(u, b_1, b_2, T_C)$$
$$= \begin{cases} 0 & if\,(u \notin [b_{1,Vi-}, b_{1,Vi+}]) \vee (b_{2,Si+} + u \cdot T_C \leq b_{1,Si-}) \vee (b_{2,Si-} + u \cdot T_C \geq b_{1,Si+}) \\ \dfrac{\min(b_{2,Si+} + u \cdot T_C, b_{1,Si+}) - \max(b_{2,Si-} + u \cdot T_C, b_{1,Si-})}{b_{2,Si+} - b_{2,Si-}} & \text{otherwise.} \end{cases} \tag{7.2}$$

Hence, $P_{mig\_i}(b_1, b_2, T_C)$ (the probability that an object o with any velocity in b_2 partially migrates) can be computed by integrating over all values of u:

$$P_{mig\_i}(b_1, b_2, T_C) = \frac{1}{b_{2,Vi+} - b_{2,Vi-}} \int_{b_{2,Vi-}}^{b_{2,Vi+}} P_{mig\_fix\_i}(u, b_1, b_2, T_C)\, du. \tag{7.3}$$

Having derived $P_{mig\_i}(b_1, b_2, T_C)$, the probability $P_{mig}(b_1, b_2, T_C)$ that an object migrates from b_2 to b_1 can be computed as Eq. (7.1). Given that bucket b_2 contains $b_2.num$ objects, the number $N_{mig}(b_1, b_2, T_C)$ of migrated objects equals $(b_2.num) \cdot P_{mig}(b_1, b_2, T_C)$. Note that, since bucket allocation for rectangles is based on their centroids, the above analysis also applies to rectangle objects.

Whenever the system receives an object update, the new information is intercepted to modify the histogram accordingly. Further, the histogram is, periodically, logically re-rebuilt using the algorithm in Figure 15. Although the incremental updating reduces the maintenance cost significantly, since the logical re-building only updates the statistics without changing the original buckets extents (created at time 0), the uniformity inside the buckets (which is the precondition for efficient estimation) may gradually deteriorate as the data (location and velocity) distributions vary. When such changes have accumulated considerably (which, as evaluated in the experiments, happens only after a very long period), a physical rebuild is still necessary to maintain satisfactory performance.

7.2 Nonuniform Estimation with Spatio-Temporal Histograms

STHs allow the application of uniform models inside each bucket, by regarding its MBR and VBR as the data and velocity space, respectively. Given a STWQ q, for each bucket b_j in STH, we estimate the selectivity $b_j.Sel$ for objects in b_j, by replacing $[U_{\min\_i}, U_{\max\_i}]$, $[V_{\min\_i}, V_{\max\_i}]$, L_i, LV_i in Eq. (4.8) with b_j.MBR, b_j.VBR, $b_j.L_i$, $b_j.LV_i$, respectively. Then, the number of objects in b_j satisfying q can be estimated as $b_j.Sel \cdot b_j.num$, and the overall selectivity of q can be computed from the results of all buckets:

$$Sel_{STWQ}(q) = \frac{\sum_{j=1}^{h}(b_j.Sel \cdot b_j.num)}{N} \tag{7.4}$$

where h is the number of buckets in STH, and N the cardinality of the dataset. Notice that we do not compute the selectivity for those buckets that cannot contain any qualifying objects. Figure 17 shows the extents b_1.MBR, b_2.MBR of buckets b_1, b_2 for point objects and a query ($q_{T-} = 0$) with current extent q_S. The shaded rectangles represent the extents b_1.MBR(q_{T+}), b_2.MBR(q_{T+}), $q_S(q_{T+})$ of b_1, b_2, and q at time q_{T+}, respectively. Estimation can be avoided for b_1, because its MBR does not intersect that of q during any time in q_T, indicating that none of the objects inside it can possibly satisfy the query. Bucket b_2, on the other hand, must be considered (i.e., a qualifying bucket). Thus, for estimation of STWQ, we scan the STH to identify the qualifying buckets and then evaluate the cost model for each one of them.

For STkNN queries, we need to evaluate Eq. (5.2) by substituting variables $[U_{\min\_i}, U_{\max\_i}]$, $[V_{\min\_i}, V_{\max\_i}]$, L_i, LV_i, N with the data properties around the query's trajectory. Specifically, we first identify the set of buckets $\{b_1, b_2, \ldots, b_s\}$ (s is the number of such buckets) whose spatial MBRs intersect query region q_S during query time q_T. Then, $U_{\min\_i}$, $U_{\max\_i}$($V_{\min\_i}$, $V_{\max\_i}$) are set to the minimum and maximum spatial (velocity) boundaries of these buckets $b_j (1 \le j \le s)$, respectively, L_i and LV_i equal the average spatial and velocity lengths of the

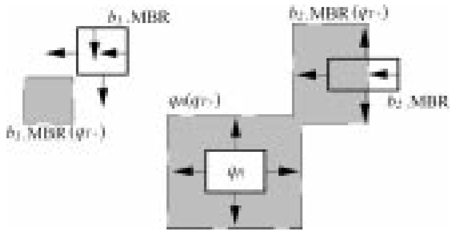


Fig. 17. Filtering buckets for selectivity estimation.

objects inside the buckets, and N is computed as the weighted sum of $b_j.num$ (weights are decided based on buckets' volumes):

$$\begin{aligned} U_{\min\_i} &= \min\{b_{j\cdot Si-}(1 \le j \le s)\}, U_{\max\_i} = \max\{b_{j\cdot Si+}(1 \le j \le s)\}, \\ V_{\min\_i} &= \min\{b_{j\cdot Vi-}(1 \le j \le s)\}, V_{\max\_i} = \max\{b_{j\cdot Vi+}(1 \le j \le s)\} \\ L_i &= \frac{\sum_{j=1}^{s} b_j.L_i}{\sum_{j=1}^{s} b_j.num}, LV_i = \frac{\sum_{j=1}^{s} b_j.LV_i}{\sum_{j=1}^{s} b_j.num}, \\ N &= \frac{\left[\sum_{j=1}^{s} b_j.num\right] \cdot \prod_{i=1}^{m}(U_{\max\_i} - U_{\min\_i})}{\sum_{j=1}^{s} vol(b_j.MBR)} \end{aligned} \tag{7.5}$$

where $[b_{j,Si-}, b_{j,Si+}]([b_{j,Vi-}, b_{j,Vi+}])$ is the spatial (velocity) extent of b_j.MBR (b_j.VBR) on the ith dimension, and $vol(b_j$.MBR) corresponds to the volume of b_j.MBR. To ensure satisfactory estimation accuracy, we must guarantee that the buckets invoked in Eq. (7.5), contain enough objects. For example, if the path of q crosses only those buckets with $b_j.num = 0$ (i.e., these buckets cover areas with zero data density), N is set to 0 and Eq. (5.2) yields nearest distance 0 (which is clearly unreasonable). Intuitively, in such cases, the nearest neighbor of q does not lie in the buckets intersecting its path; hence, we need to compute the statistics by including more distant buckets, or specifically identifying buckets whose MBRs are within some positive distance l from the query path. If the set of buckets thus obtained contains fewer than N_ε objects (i.e., $\sum_{j=1}^{s} b_j.num < N_\varepsilon$), we increase l (by certain constant) and repeat the process. The threshold N_ε should be large enough to ensure that the vicinity of the query contains a sufficient number of data points. In the experimental evaluation we set l to 1% of the axis length of the data space, and N_ε to 0.1% of the dataset cardinality. These values offer satisfactory estimation accuracy without incurring significant overhead.

For computing the selectivity of STJ assume that STH$_1$, STH$_2$ are the histograms for the participating datasets. Given a pair of buckets $(b_1, b_2)(b_1 \in$ STH$_1, b_2 \in$ STH$_2$), a partial selectivity Sel_{12} can be obtained from Eq. (6.4), treating b_1.MBR (b_1.VBR) and b_2.MBR (b_2.VBR) as the data (velocity) spaces. Then, the number of objects in (b_1, b_2) satisfying the join predicate can be estimated as $Sel_{12} \cdot b_1.num \cdot b_2.num$, and the overall selectivity is:

$$Sel_{STJ}(q) = \frac{\sum_{i=1}^{h_1} \sum_{j=1}^{h_2}(Sel_{ij} \cdot b_j.num.b_j.num)}{N_1 \cdot N_2} \tag{7.6}$$

Similar to the case of STWQ, we only apply the model for qualifying bucket pairs, that is, pairs that may contain objects satisfying the join condition.

8. EXPERIMENTS

This section experimentally evaluates the proposed cost models on a Pentium III 1 Ghz CPU with 256 Mbytes RAM, using both uniform and nonuniform data in the two-dimensional space where each axis has range [0, 10000]. A uniform rectangle dataset UNI_{rec} contains 1 million moving rectangles, such that on the ith dimension ($1 \leq i \leq 2$), (i) the spatial length L_i and velocity length LV_i of each rectangle r are fixed to 0.5 and 0, respectively (i.e., the extent size of an object remains fixed as it moves), and (ii) the coordinate r_{Si-} (velocity r_{Vi-}) of its lower boundary, is uniform in the range $[0, 10000 - L_i]$ ($[-10, 10 - LV_i]$). From UNI_{rec}, we create a uniform point dataset UNI_{pt} by taking the coordinates and velocities of each rectangle's centroid.

Due to the lack of real-world spatio-temporal data, we generate two nonuniform datasets CA and LA using a common methodology in the literature [Pfoser et al. 2000; Theodoridis et al. 1999; Tao and Papadias 2003; Choi and Chung 2002]. Specifically, for CA (LA), objects' locations (MBRs) are taken from a real spatial dataset (the *Tiger* collection [Tiger] of US Census Bureau) containing 2.2 (1.3) million points (rectangles) corresponding to places in California (Los Angeles). Then, each point (rectangle) o is associated with a velocity vector o_V such that on the ith dimension ($1 \leq i \leq 2$), (i) the absolute value of $o_{Vi}(o_{Vi-})$ follows a Zipf distribution in [0, 10] ($[0, 10 - o.LV_i]$) (skewed towards 0 with biased coefficient 0.8), and (ii) $o_{Vi}(o_{Vi-})$ has equal probability to be positive or negative. For rectangle objects the velocity length $o.LV_i$ follows Zipf (skewed towards 0, coefficient 0.8) in [0, 5] (i.e., objects' extent sizes can change at different rates). Note that object velocities on the two dimensions are independent (i.e., the moving direction is arbitrary) in the above datasets. For each nonuniform dataset, we create a (4-dimensional) spatio-temporal histogram using *minskew* [Acharya et al. 1999], fixing the number of buckets to 2000 such that each histogram consumes around 15 kbytes memory.

We evaluate our techniques on (i) prediction accuracy, (ii) computational overhead, and (iii) performance deterioration along with time (due to object updates). For comparison, we also demonstrate the results of Choi and Chung [2002] (referred to CC in the sequel) on STWQ selectivity estimation (no previous work exists for kNN and STJ estimation). The prediction accuracy is measured as the average error in answering a query workload of 200 queries with the same parameters (elaborated shortly for concrete query types). Specifically, let act_i and est_i be the actual and estimated values (i.e., selectivity for STWQ, STJ, nearest distance for STkNN) of the ith query ($1 \leq i \leq 200$) in the workload; we adopt the following workload error definition [Acharya et al. 1999]:

$$Err_{workload} = \frac{\left(\sum_{i=1}^{200} |est_i - act_i|\right)}{\left(\sum_{i=1}^{200} act_i\right)}. \tag{8.1}$$

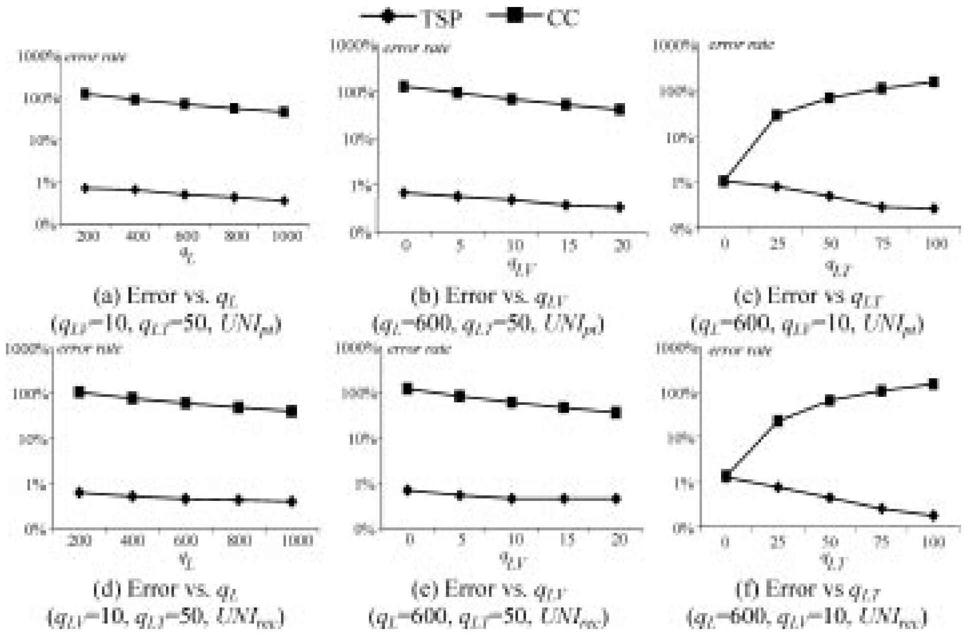
We use the above definition, instead of another common metric $\frac{1}{200}\sum_i[(est_i - act_i)/act_i]$, because the latter is often dominated by the large error of "small" queries (i.e., those with low act_i). As with previous work [Acharya et al. 1999; Gunopulos et al. 2000; Bruno et al. 2001], we aim at evaluating performance for relatively "large" queries, since in practice query optimization for small queries is trivial (i.e., index search, rather than sequential scan, should always be used). It is worth mentioning, however, that our solutions consistently outperform that of Choi and Chung [2002] under both error definitions.

8.1 STWQ Selectivity Estimation

The spatial extent q_S of each STWQ query is a square with side length q_L (e.g., if $q_L = 1000$, then the query window covers 1% of the space) whose distribution follows that of the corresponding dataset. The lower boundary velocity q_{Vi-} on the ith ($1 \leq i \leq 2$) axis is generated uniformly in $[-10, 10 - q_{LV}]$, where the velocity length q_{LV} is fixed for all dimensions. The starting timestamp of the query interval q_T is uniform in $[0, 100 - q_{LT}]$, where q_{LT} is the length of q_T. Thus, a query workload involves parameters q_L, q_{LV}, and q_{LT}.

The first set of experiments verifies the correctness of the probabilistic derivation for STWQ selectivity and compares our model (denoted as TSP) with CC. Since the original CC only captures static queries over moving objects, we apply our reduction techniques (i.e., Lemmas 4.1 and 4.2) to obtain the corresponding formulas for moving queries and rectangle objects. Figure 18a measures, for UNI_{pt} (uniform point dataset), the error rates by fixing q_{LV}, q_{LT} to 10, 50, respectively and varying the query (spatial) length q_L from 200 to 1000 (note that the y-axis is in logarithmic scale). TSP yields extremely accurate prediction (with maximum error less than 1%), while CC leads to substantial errors (greater than 100%), indicating that the temporal intersection condition cannot be ignored. The error rates of both methods are lower for larger query windows, a finding that is consistent with the previous studies on spatial window selectivity [Acharya et al. 1999; Jin et al. 2000; Theodoridis et al. 2000] (in general, probabilistic analysis achieves better accuracy as the output size increases). Figure 18(b) shows the error rate with respect to various velocity lengths q_{LV} (ranging from 0 to 20), fixing $q_L = 600$ and $q_{LT} = 50$. Again our model is precise whereas CC produces around 100% error. In Figure 18(c), we fix q_L and q_{LV}, and increase the interval length q_{LT} from 0 (i.e., timestamp queries) to 100. It is clear that CC is accurate only for $q_{LT} = 0$ (i.e., timestamp queries) and its error rate increases very fast with q_{LT}, as predicted by Eq. (4.13). Figures 18(d), 18(e), 18(f) repeat the experiments for rectangle dataset UNI_{rec}, confirming the above observations. Since CC is erroneous in almost all cases, we omit this technique in the following experiments.

Figure 19(a) plots, for dataset CA, the estimated and actual selectivity (averaged over all queries in a workload) as a function of the query length q_L, fixing q_{LV}, q_{LT} to their median values. The numbers below the estimated curve indicate the corresponding workload error. The output size increases with q_L, because higher q_L leads to greater area of $CX(q)$ (the convex hull of the vertices of $q_S(q_{T-})$ and $q_S(q_{T+})$), as discussed in Section 4. Figures 19(b) and 19(c) show the selectivity with respect to the velocity (q_{LV}) and interval length (q_{LT}), respectively. The output size also increases with q_{LV} (q_{LT}), which, similar to



(a) Error vs. q_L (q_{LV}=10, q_{LT}=50, UNI_{pt})
(b) Error vs. q_{LV} (q_L=600, q_{LT}=50, UNI_{pt})
(c) Error vs q_{LT} (q_L=600, q_{LV}=10, UNI_{pt})
(d) Error vs. q_L (q_{LV}=10, q_{LT}=50, UNI_{rec})
(e) Error vs. q_{LV} (q_L=600, q_{LT}=50, UNI_{rec})
(f) Error vs q_{LT} (q_L=600, q_{LV}=10, UNI_{rec})

Fig. 18. Accuracy of STWQ selectivity estimation (uniform data).

Figure 18, improves precision. The maximum estimation error is about 12%. Figures 19(d), 19(e), and 19(f) evaluate the models for nonuniform rectangles using dataset *LA* confirming similar observations. *LA* incurs larger error than *CA* because objects in *LA* have different spatial and velocity lengths, while each bucket stores only the average values, thus, incurring additional inaccuracy.

The above results are consistent with the recent evaluation in Hadjieleftheriou et al. [2003]. If we do not consider object updates, for point data the histogram in Hadjieleftheriou et al. [2003] achieves the same performance as our solution, since they are based on similar rationale and both adopt *minskew*. However, our approach is incrementally maintainable, which, as shown in Section 8.5, reduces the overhead significantly by avoiding frequent (physical) rebuilding. Furthermore, we cover all query types, while they focus explicitly on window queries.

It is worth mentioning that Hadjieleftheriou et al. [2003] also perform experiments using objects moving on road networks. Since the underlying assumption of their (and our) technique is that there is no update between now and the ending time of the query interval, they measure the "actual" query results by assuming that the objects maintain their velocities, even after they reach the end of the road segments (that they were on). In other words, for estimation purposes objects do not really move on a road network, but on a set of infinite lines defined by the original road segments. We do not include such experiments[7]

[7]We actually performed experiments, using the spatio-temporal generator of Brinkhoff [2002], simulating 500,000 objects (vehicles, pedestrians) moving on the road network of Oldenburg city (6105 nodes, 7035 edges). We found that, under typical settings of the generator (downloadable at the

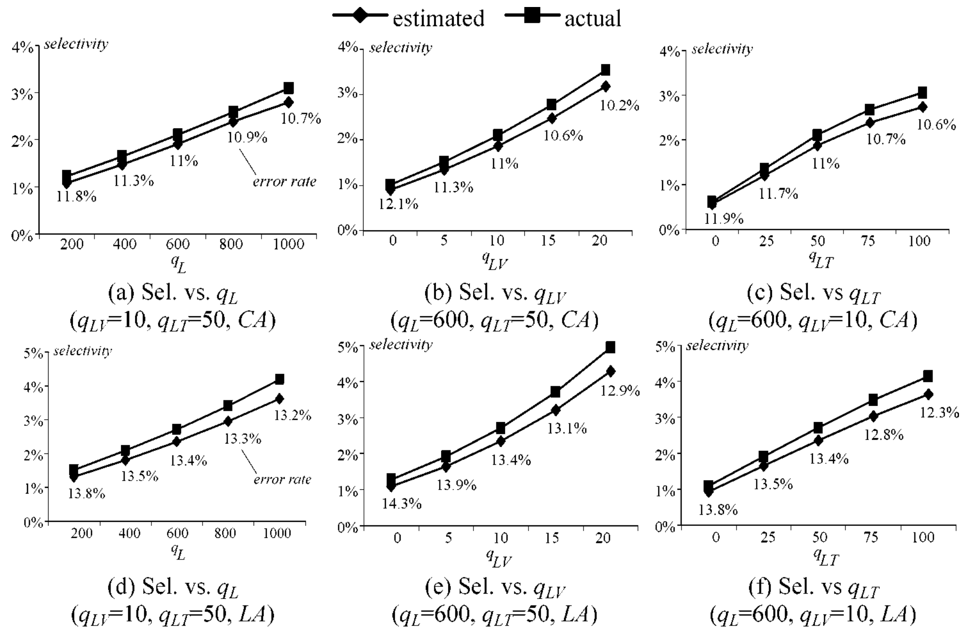


Fig. 19. STWQ selectivity estimation and its accuracy (*CA*, *LA*).

in this article because our models, and velocity-based prediction methods in general (including Choi and Chung [2002] and Hadjieleftheriou et al. [2003]), are not suitable for road networks, or any application where updates must occur before the query time. For such cases, conventional forecasting methods, like *exponential smoothing* [Gardner 1985], can estimate future results using only location (but not velocity) information about the present and the recent past (provided that historical information is available). The main idea is that although individual object velocities may change abruptly, the overall data distribution varies smoothly with time due to the continuity of movement; thus, the recent history can, presumably, predict the immediate future.

8.2 ST*k*NN Nearest Distance Prediction

Next we evaluate the formulas estimating the nearest distances (*ND*) for ST*k*NN. We distinguish between point and rectangle queries (i.e., with extents) because, as shown shortly, they have different characteristics. As with STWQ workloads, the location of a point query q is decided according to the spatial distribution of the underlying dataset. Its velocities q_{Vx}, q_{Vy} on the x- and y-dimensions are set to $|q_V| \cdot \cos\theta$ and $|q_V| \cdot \sin\theta$, respectively, where θ (randomly generated in [0, 2π)) is the angle between the query's movement and the positive direction of the x-axis, and $|q_V|(= (q_{Vx}^2 + q_{Vy}^2)^{1/2})$ corresponds to the *movement speed*. The starting timestamp of the query interval q_T follows uniform

following URL: www.fh-oow.de/institute/iapg/personen/brinkhoff/generator.shtml), the percentage of objects that issue updates per timestamp is about 95% that is, prediction is not useful even for the next timestamp.

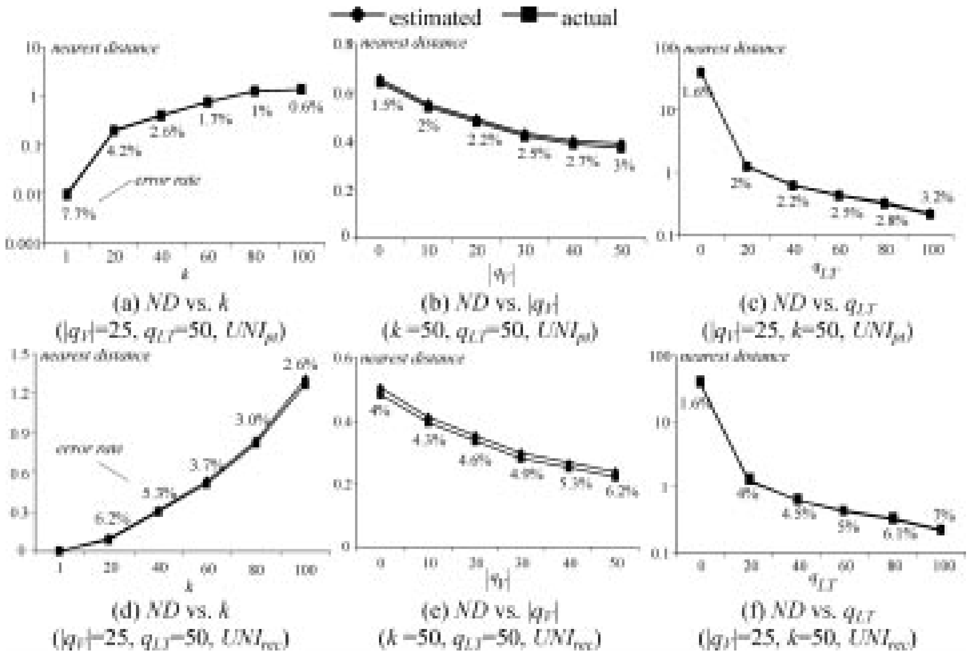


Fig. 20. STkNN nearest distance estimation and its accuracy (point queries, uniform data).

distribution in [0, $100 - q_{LT}$], where q_{LT} is the length of q_T. A rectangle query is created similarly, except that (i) the query extent is a square with side length q_L, and (ii) the velocity length is zero on all dimensions (the size of a query remains constant as it moves). Thus, a STkNN workload involves parameters k, $|q_V|$, q_{LT} (for all queries), and q_L (for rectangle queries).

Figure 20(a) shows the actual and estimated ND (again, averaged over all the queries in the workload), together with the error rate (Eq. (8.1)), as a function of the number k of neighbors to be retrieved for point dataset UNI_{pt} ($|q_V| = 25$ and $q_{LT} = 50$). As expected, the value of ND is very small for single NN, and increases with k (up to around 1 for 100 NN). Figure 20(b) measures ND for various movement speeds $|q_V|$. Obviously, the nearest distance is lower for queries that move with high speed, because a faster query travels longer distance and has a higher chance to encounter objects closer to its path. Figure 20(c) plots ND (for 50 NN) as a function of the query interval length q_{LT} ($|q_V| = 25$). When q_{LT} equals 0, the corresponding ND indicates the distance from the query to its kth NN at the query timestamp. As in the case of travel speed, increasing the query interval length causes the decrease of ND. The estimated values are very accurate in all cases, yielding maximum error 7.7%. Furthermore, observe that the errors decrease when the nearest distance becomes larger (similar to Figures 18 and 19, where the error decreases with the output size). Figures 20(d), 20(e), 20(f) demonstrate the results for rectangle dataset UNI_{rec}, validating the above findings. As expected, the nearest distances for rectangle data are smaller than those for points. Particularly, in Figure 20(d), the nearest distance for $k = 1$ is

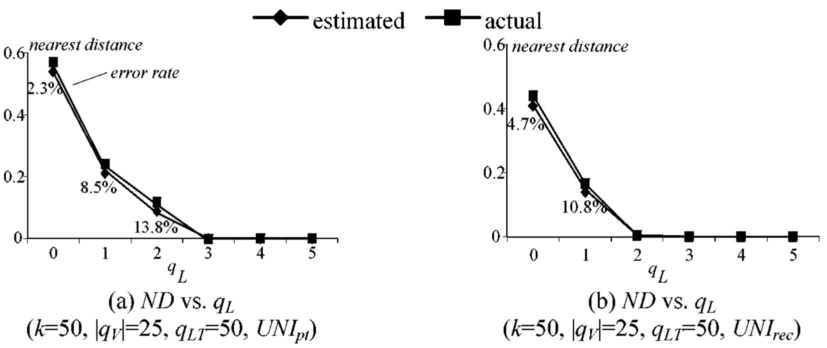


Fig. 21. STkNN nearest distance estimation and its accuracy (rectangle queries, uniform data).

zero (i.e., at least one object intersects the query extent q_S during q_T), in which case the error rate (Eq. (8.1)) is not defined.

In order to examine the accuracy of the model on rectangle queries, we fix $k, |q_V|, q_{LT}$ to 50, 25, 50, respectively, and vary q_L (i.e., the side length of the query window) from 0 to 5. Figure 21(a) shows the estimated and actual nearest distances for point data UNI_{pt}. Observe that, the nearest distance already drops to 0 when q_L equals 3, implying that there are more than k(=50) objects intersecting the query window q_S at some point during the interval q_T. Figure 21(b) illustrates the nearest distances for rectangle objects UNI_{rec}, where the expected ND is even lower. Notice that although the distances are very small, our model still provides fairly accurate estimation (maximum error below 15%).

Next we illustrate the results of point queries for datasets CA (Figures 22(a), 22(b), and 22(c)) and LA (Figures 22(d), 22(e), and 22(f)). Comparing with uniform data (Figure 20), the values of ND are lower, because in these datasets the data distributions are skewed, and hence, the nearest neighbors of a query lie in closer vicinity (recall that the query distribution follows that of the dataset). Rectangle queries are omitted because even with small query extent ($q_L < 1$), the expected ND becomes zero.

8.3 STJ Selectivity Estimation

The next set of experiments evaluates the cost models for spatio-temporal join selectivity. The two query parameters that we consider are: (i) the maximum distance d between the retrieved objects, and (ii) the query time interval [0, q_{LT}] (the starting timestamp is always at the current time). Figure 23(a) shows the actual and estimated (from Eq. (6.3)) selectivity of the self-join $UNI_{pt} \bowtie UNI_{pt}$, as a function of d ($q_{LT} = 50$). In Figure 23(b), the distance threshold d is fixed to 250, and the selectivity is measured with respect to q_{LT} (which ranges from 0 to 100).

The chance for two objects to move within distance d from each other increases with both d and q_{LT}. The self-join for UNI_{rec} produces almost the same selectivity and prediction error as UNI_{pt} (hence, we omit the diagrams), because the object extents (0.5) are significantly smaller than d ($\geq$100) and do not affect

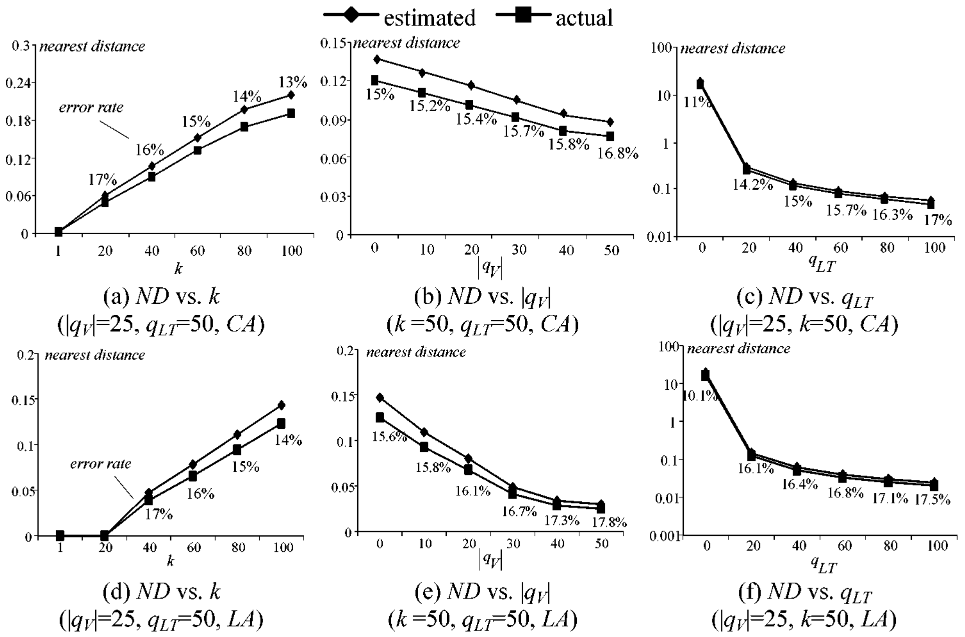


Fig. 22. ST*k*NN nearest distance estimation and its accuracy (point queries, *CA*, *LA*).

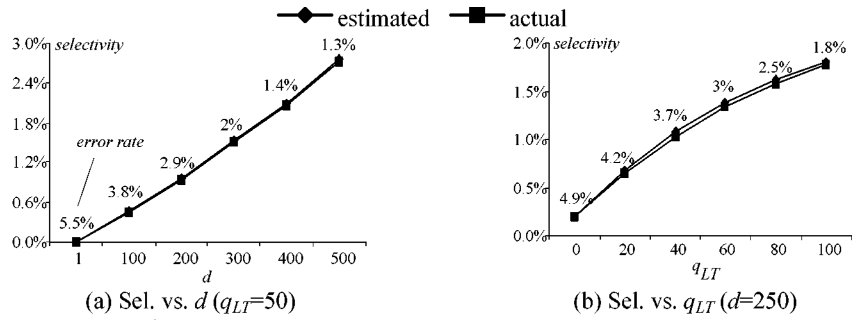


Fig. 23. STJ selectivity estimation and its accuracy ($UNI_{pt} \bowtie UNI_{pt}$).

the performance. Figure 24 repeats the experiments for $CA \bowtie LA$ (where the estimated values are obtained from Eq. (6.4)), illustrating similar phenomena.

8.4 Estimation Costs

Having demonstrated the accuracy of the models, we now evaluate their computation costs, starting with STWQ. Figures 25(a), 25(b), and 25(c) demonstrate the CPU time for obtaining an estimated value as a function of the query length (q_L), velocity length (q_{LV}), and interval length (q_{LT}) for dataset *CA*. The cost is very small (up to 300 ms) in all cases, and increases with each query parameter because we apply the model only for those histogram buckets that intersect the query window during the query interval. Higher q_L(q_{LV}, or q_{LT}) increases the

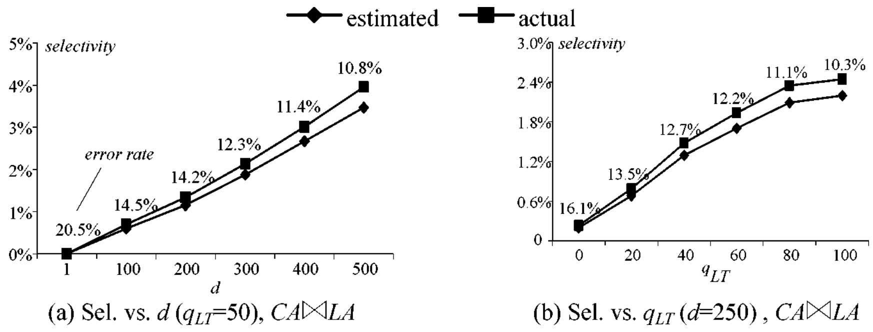


Fig. 24. STJ selectivity estimation and its accuracy.

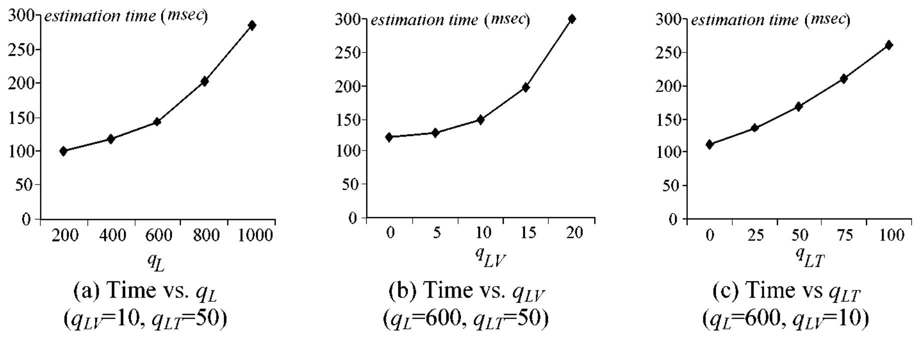


Fig. 25. STWQ estimation time (*CA*).

number of qualifying buckets, and consequently, the number of applications of the model.

For ST*k*NN, the time to estimate *ND* is negligible (around 10 ms) in all cases because, as discussed in Section 7.2, we first identify the set of buckets that intersect the query path, and then execute the uniform model only once using the data properties obtained from these buckets. On the other hand selectivity estimation for joins is expensive. Figures 26(a) and 26(b) demonstrate the cost as a function of d and q_{LT}, respectively ($CA \bowtie LA$). Since, we examine only those buckets (from the histograms of the joined datasets) that can contain qualifying object pairs, the cost increases with the number of qualifying pairs, which grows with d and q_{LT}. Despite the fact that the estimation time can reach up to 40 seconds, the model is still applicable for optimization since the CPU time of the corresponding actual join (using TPR-trees [Saltenis et al. 2000]) is more than 20 minutes.

8.5 Performance Deterioration with Time

The last set of experiments examines the effectiveness of the proposed logical rebuild algorithm (shown in Figure 15) for maintaining spatio-temporal histograms. The initial histogram is constructed at timestamp 0 from a nonuniform dataset. Then, for *LA* and *CA*, at each of the subsequent 2000 timestamps, approximately 1% of the objects (e.g., for *CA* the total number of changes is

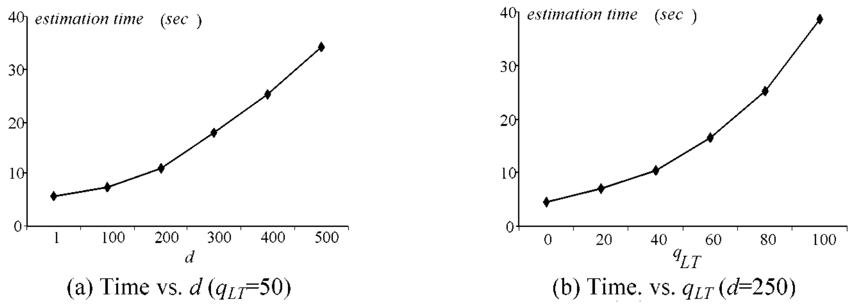


Fig. 26. STJ Selectivity estimation time ($CA \bowtie LA$).

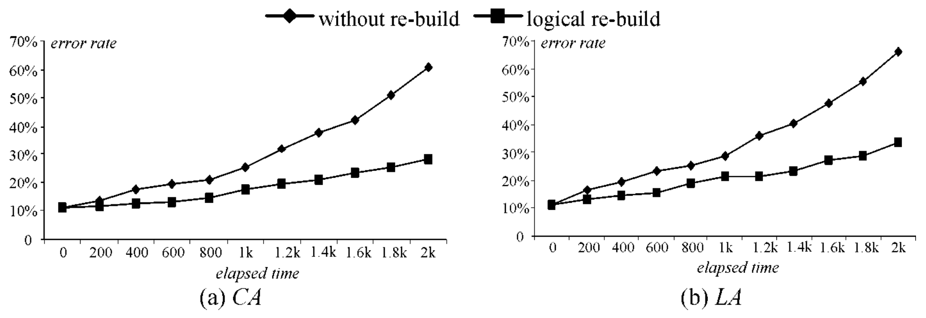


Fig. 27. Deterioration of STWQ selectivity estimation with time ($q_L = 600, q_{LV} = 10, q_{LT} = 50$).

around 40 million) are randomly selected to update their velocities, such that each velocity offset is uniformly distributed in [−1, 1]. Datasets generated this way incur gradual distribution changes. For each update, the histogram is modified as described in Section 7.1, and is logically rebuilt every 50 timestamps (a rebuild takes less than one second). We perform estimation on each dataset every 200 timestamps using the most updated histogram (at the query time), and compare the error with that obtained by using a histogram that is never re-built.

Figure 27 demonstrates the error rate of selectivity estimation as a function of elapsed time for STWQ ($q_L = 600$, $q_{LV} = 10$, $q_{LT} = 50$). Figures 28(a) (CA) and 28(b) (LA) illustrate the error rate of estimating the expected distance for STkNN, where $k = 50, |q_V| = 25, q_{LT} = 50$. Finally, Figure 29 shows the error of selectivity estimation for STJ ($CA \bowtie LA$) ($d = 50, q_{LT} = 50$). In all cases, the performance of the rebuilt histograms degrades (due to the change of distributions) very slowly. For example, if the tolerable maximum error rate is 35%, then physical rebuilding is unnecessary for all types of queries until 2000 timestamps after the initial construction. The precision of histograms that do not apply logical rebuilding drops much faster and eventually their error rates double those of the rebuilt histograms.

As a summary of this section, the experimental results suggest that the proposed formulas are very accurate and compare favorably with those of the corresponding analysis in spatial (static) databases although our problem is substantially harder. In addition, our models and spatio-temporal histograms are highly

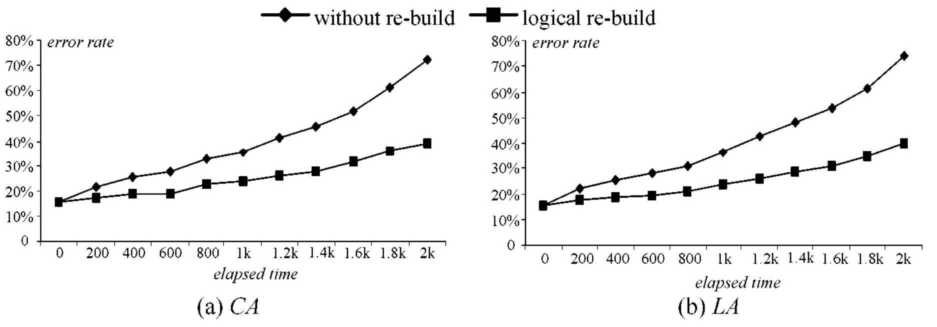


Fig. 28. Deterioration of nearest distance estimation with time ($k = 50$, $|q_V| = 25$, $q_{LT} = 50$).

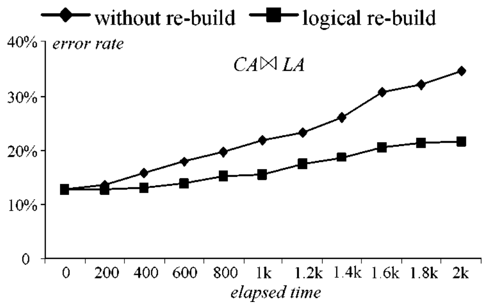


Fig. 29. Deterioration of STJ selectivity estimation with time ($d = 50$, $q_{LT} = 50$).

applicable in practice, since they incur small overhead (compared to the processing time required for the actual queries), and permit effective maintenance.

9. CONCLUSION

This article provides a comprehensive study of predictive spatio-temporal queries that covers the most common query types (i.e., window queries, k-nearest neighbors, and spatio-temporal joins), and any object/query mobility combination. Particularly, we present formulas for selectivity estimation of window queries and joins, as well as, the nearest distance for nearest neighbor queries. Our methodology is based on the reduction of complex problems into simple ones which involve only static objects and thus, can be efficiently solved. We also propose incremental spatio-temporal histograms that involve minimal maintenance cost and enable estimation for arbitrary location and velocity distributions. The efficiency of our techniques is confirmed through extensive experiments.

Although we focus on individual query types, our results can also support complex queries such as combinations of joins and window queries (e.g., retrieve all join pairs that will appear in a query window). The proposed models also constitute the theoretical foundation for analyzing the performance (in terms of the number of page accesses during query processing) of spatio-temporal access methods. Furthermore, they significantly enhance the understanding of spatio-temporal queries, which may motivate the development of novel index structures and processing algorithms. For instance, the model for window queries led us to the development of the TPR*-tree [Tao et al. 2003a], an optimized version of the TPR-tree [Saltenis et al. 2000]. In the future, we plan to investigate alternative forecasting methods (e.g., exponential smoothing) for applications, where velocity-based prediction is unsuitable due to intensive updates.

ACKNOWLEDGMENTS

We would like to thank George Kollios and Marios Hadjieleftheriou for the long discussions that helped us clarify several of the concepts presented in the paper.

REFERENCES

ABOULNAGA, A. AND NAUGHTON, J. 2000. Accurate estimation of the cost of spatial selections. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Feb.). pp. 123–134.

ACHARYA, S., POOSALA, V., AND RAMASWAMY, S. 1999. Selectivity estimation in spatial databases. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 13–24.

AGARWAL, P., ARGE, L., AND ERICKSON, J. 2000. Indexing moving points. In *Proceedings of the ACM Symposium on Principles of Database Systems* (*PODS*) (May). ACM, New York, pp. 175–168.

AN, N., YANG, Z., AND SIVASUBRAMANIAM, A. 2001. Selectivity estimation for spatial joins. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Apr.). pp. 368–375.

AREF, W. AND SAMET, H. 1994. A cost model for query optimization using R-trees. In *Proceedings of the Second ACM Workshop on Advances in Geographic Information Systems* (*GIS*) (Dec.). ACM, New York, pp. 1–7.

BECKMANN, N., KRIEGEL, H., SCHNEIDER, R., AND SEEGER, B. 1990. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD conference* (May). ACM, New York, pp. 322–331.

BELUSSI, A. AND FALOUTSOS, C. 1995. Estimating the selectivity of spatial queries using the correlation's fractal dimension. In *Proceedings of Very Large Database Conference* (*VLDB*) (Sep.). pp. 299–310.

BELUSSI, A. AND FALOUTSOS, C. 1998. Self-spatial join selectivity estimating using fractal concepts. *ACM Tran. Information Systems 16*, 2, 161–201.

BENETIS, R., JENSEN, C., KARCIAUSKAS, G., AND SALTENIS, S. 2002. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *Proceedings of International Database Engineering and Application Symposium* (July). pp. 44–53.

BERCHTOLD, S., BOHM, C., KEIM, D., AND KRIEGEL, H. 1997. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the ACM Symposium on Principles of Database Systems* (*PODS*) (May). ACM, New York, pp. 78–86.

BERCHTOLD, S., BOHM, C., KEIM, D., KREBS, F., AND KRIEGEL, H. 2001. On optimizing nearest neighbor queries in high-dimensional data spaces. In *Proceedings of International Conference on Database Theory* (*ICDT*) (Jan.). pp. 435–449.

BERG, M., KREVELD, M., OVERMAS, M., AND SCHWARZKOPF, O. 1997. Computational geometry: algorithms and applications. Springer, New York, ISBN: 3-540-61270-X.

BEYER, K., GOLDSTEIN, J., AND RAMAKRISHNAN, R. 1999. When is "nearest neighbor" meaningful? In *Proceedings of International Conference on Database Theory* (*ICDT*) (Jan.). pp. 217–235.

BLOHSFELD, B., KORUS, D., AND SEEGER, B. 1999. A comparison of selectivity estimators for range queries on metric attributes. In *Proceedings of the ACM SIGMOD conference* (June). ACM, New York, pp. 239–250.

BOHM, C. 2000. A cost model for query processing in high dimensional data spaces. *ACM Tran. Datab. Syst. 25*, 2, 129–178.

BRINKHOFF, T. 2002. A framework for generating network-based moving objects. *GeoInformatica. 6*, 2, 153–180.

BRUNO, N., GRAVANO, L., AND CHAUDHURI, S. 2001. STHoles: A workload aware multidimensional histogram. In *Proceedings of the ACM SIGMOD Conference* (May). ACM, New York, 211–222.

CHAUDHURI, S., DAS, G., DATAR, M., MOTWANI, R., AND NARASAYYA, V. 2001. Overcoming limitations of sampling for aggregation queries. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Apr.). pp. 534–542.

CHOI, Y. AND CHUNG, C. 2002. Selectivity estimation for spatio-temporal queries to moving objects. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 440–451.

CIACCIA, P., PATELLA, M., AND ZEZULA, P. 1998. A cost model for similarity queries in metric spaces. In *Proceedings of the ACM Symposium on Principles of Database Systems* (*PODS*) (June). ACM, New York, pp. 59–68.

CORRAL, A., MANOLOPOULOS, Y., THEODORIDIS, Y., AND VASSILAKOPOULOS, M. 2000. Closest pair queries in spatial databases. In *Proceedings of the ACM SIGMOD Conference* (May). ACM, New York, pp. 189–220.

DESHPANDE, A., GAROFALAKIS, M., AND RASTOGI, R. 2001. Independence is good: dependency-based histogram synopses for high-dimensional data. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 199–210.

FALOUTSOS, C., SEEGER, B., TRAINA, A., AND TRAINA JR., C. 2000. Spatial join selectivity using power laws. In *Proceedings of the ACM SIGMOD Conference* (May). ACM, New York, pp. 177–188.

GARDNER, E. 1985. Exponential smoothing: the state of the art. *J. Forecast. 4*, 1–28.

GUNOPULOS, D., KOLLIOS, G., TSOTRAS, V., AND DOMENICONI, C. 2000. Approximating multi-dimensional aggregate range queries over real attributes. In *Proceedings of the ACM SIGMOD Conference* (May). ACM, New York, pp. 463–474.

GUTTMAN, A. 1984. R-Trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 47–57.

HADJIELEFTHERIOU, M., KOLLIOS, G., AND TSOTRAS, V. 2003. Performance evaluation of spatio-temporal selectivity estimation techniques. In *Proceedings of Statistical and Scientific Database Management* (*SSDBM*) (July). pp. 202–211.

HADJIELEFTHERIOU, M., KOLLIOS, G., TSOTRAS, V., AND GUNOPULOS, D. 2002. Efficient indexing of spatiotemporal objects, In *Proceedings of Extending Data Base Technology* (*EDBT*) (Mar.). pp. 251–268.

HJALTASON, G. AND SAMET, H. 1999. Distance browsing in spatial databases. *ACM Trans. Datab. Syst. 24*, 2, 265–318.

HUANG, Y., JING, N., AND RUNDENSTEINER, E. 1997. A cost model for estimating the performance of spatial joins using R-trees. In *Proceedings of Statistical and Scientific Database Management* (*SSDBM*) (Aug.). pp. 30–38.

JIN, J., AN, N., AND SIVASUBRAMANIAM, A. 2000. Analyzing range queries on spatial data. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Feb.). pp. 525–534.

KAMEL, I. AND FALOUTSOS, C. 1993. On packing R-trees. In *Proceedings of International Conference on Information and Knowledge Management* (*CIKM*) (Nov.). ACM, New York, pp. 490–499.

KOLLIOS, G., GUNOPULOS, D., AND TSOTRAS, V. 1999. On indexing mobile objects. In *Proceedings of the ACM Symposium on Principles of Database Systems* (*PODS*) (May). ACM, New York, pp. 261–272.

LEE, J., KIM, D., AND CHUNG, C. 1999. Multidimensional selectivity estimation using compressed histogram information. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 205–214.

MAMOULIS, N. AND PAPADIAS, D. 2001. Multiway spatial joins. *ACM Trans. Datab. Syst. 26*, 4, 424–475.

MATIAS, Y., VITTER, J., AND WANG, M. 1998. Wavelet-based histograms for selectivity estimation. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 448–459.

MATIAS, Y., VITTER, J., AND WANG, M. 2000. Dynamic maintenance of wavelet-based histograms. In *Proceedings of Very Large Database Conference* (*VLDB*) (Sept.). pp. 101–110.

MURALIKRISHNA, M. AND DEWITT, D. 1988. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 28–36.

OLKEN, F. AND ROTEM, D. 1990. Random sampling from database files: A survey. In *Proceedings of Statistical and Scientific Database Management* (*SSDBM*) (Apr.). pp. 92–111.

PAGEL, B., SIX, H., TOBEN, H., AND WIDMAYER, P. 1993. Towards an analysis of range query performance in spatial data structures. In *Proceedings of the ACM Symposium on Principles of Database Systems* (*PODS*) (May). ACM, New York, pp. 214–221.

PALMER, C. AND FALOUTSOS, C. 2000. Density biased sampling: an improved method for data mining and clustering. In *Proceedings of the ACM SIGMOD conference* (June). ACM, New York, pp. 82–92.

PAPADOPOULOS, A. AND MANOLOPOULOS, Y. 1997. Performance of nearest neighbor queries in R-trees. In *Proceedings of International Conference on Database Theory* (*ICDT*) (Jan.). pp. 394–408.

PFOSER, D., JENSEN, C, AND THEODORIDIS, Y. 2000. Novel approaches to the indexing of moving object trajectories. In *Proceedings of Very Large Database Conference* (*VLDB*) (Sept.). pp. 395–406.

POOSALA, Y. AND IOANNIDIS, Y. 1997. Selectivity estimation without the attribute value independence assumption. In *Proceedings of Very Large Database Conference* (*VLDB*) (Aug.). pp. 486–495.

PRESS, W., FLANNERY, B., TEUKOLSKY, S., AND VETTERLING, W. 2002. *Numerical Recipes in C++* (second edition). Cambridge University Press, Cambridge, Mass., ISBN 0-521-75034-2.

PROIETTI, G. AND FALOUTSOS, C. 1998. Selectivity estimation of window queries for line segment datasets. In *Proceedings of International Conference on Information and Knowledge Management* (*CIKM*) (Nov.). ACM, New York, pp. 340–347.

PROIETTI, G. AND FALOUTSOS, C. 2001. Accurate modeling of region data. *Trans. Knowl. Data Eng.* (*TKDE*) *13*, 6, 874–883.

ROUSSOPOULOS, N., KELLEY, S., AND VINCENT, F. 1995. Nearest neighbor queries. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 71–79.

SALTENIS, S. AND JENSEN, C. 2002. Indexing of moving objects for location-based services. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Feb.). pp. 463–472.

SALTENIS, S., JENSEN, C., LEUTENEGGER, S., AND LOPEZ, M. 2000. Indexing the positions of continuously moving objects. In *Proceedings of the ACM SIGMOD Conference* (June). ACM, New York, pp. 331–342.

SUN, C., AGRAWAL, D., AND EL ABBADI, A. 2002a. Exploring spatial datasets with histograms. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Feb.). pp. 93–102.

SUN, C., AGRAWAL, D., AND EL ABBADI, A. 2002b. Selectivity estimation for spatial joins with geometric selections. In *Proceedings of Extending Data Base Technology* (*EDBT*) (Mar.). pp. 609–626.

TAO, Y. AND PAPADIAS, D. 2003. Spatial queries in dynamic environments. *ACM Tran. Datab. Syst.* 28, 2, 101–139.

TAO, Y., PAPADIAS, D., AND SUN, J. 2003a. The TPR*-Tree: An optimized spatio-temporal access method for predictive queries. In *Proceedings of Very Large Database Conference* (*VLDB*) (Sept.), pp. 790–801.

TAO, Y., SUN, J., AND PAPADIAS, D. 2003b. Selectivity estimation for predictive spatio-temporal queries. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Mar.). pp. 417–428.

THAPER, N., GUHA, S., INDYK, P., AND KOUDAS, N. 2002. Dynamic multidimensional histograms. In *Proceedings of the ACM SIGMOD conference* (June). ACM, New York, pp. 428–439.

THEODORIDIS, Y., SILVA, J., AND NASCIMENTO, M. 1999. On the generation of spatiotemporal datasets. In *Proceedings of Symposium on Large Spatial Databases* (*SSD*) (July). pp. 147–164.

THEODORIDIS, Y., STEFANAKIS, E., AND SELLIS, T. 1998. Cost models for join queries in spatial databases. In *Proceedings of International Conference on Data Engineering* (*ICDE*) (Feb.). pp. 476–483.

THEODORIDIS, Y., STEFANAKIS, E., AND SELLIS, T. 2000. Efficient cost models for spatial queries using R-trees. *Tran. Knowl. Data Eng.* (*TKDE*). 12, 1, 19–32.

TIGER. http://www.census.gov/geo/www/tiger/.

WEBER, R., SCHEK, H., AND BLOTT, S. 1998. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of Very Large Database Conference* (*VLDB*) (Aug.). pp. 194–205.

WU, Y., AGRAWAL, D., AND EL ABBADI, A. 2001. Applying the golden rule of sampling for selectivity estimation. In *Proceedings of the ACM SIGMOD conference* (June). ACM, New York, pp. 449–460.

Received October 2002; revised May 2003; accepted June 2003