Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮
www.elsevier.com/locate/infosys

Spatio-temporal join selectivity ☆

Jimeng Sun[a], Yufei Tao[b], Dimitris Papadias[c,*], George Kollios[d]

[a]Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[b]Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Hong Kong
[c]Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
[d]Department of Computer Science, Boston University, Boston, MA, USA

Received 12 July 2002; received in revised form 11 February 2005; accepted 11 February 2005

Abstract

Given two sets S_1, S_2 of moving objects, a future timestamp t_q, and a distance threshold d, a *spatio-temporal join* retrieves all pairs of objects that are within distance d at t_q. The selectivity of a join equals the number of retrieved pairs divided by the cardinality of the Cartesian product $S_1 \times S_2$. This paper develops a model for spatio-temporal join selectivity estimation based on rigorous probabilistic analysis, and reveals the factors that affect the selectivity. Initially, we solve the problem for 1D (point and rectangle) objects whose location and velocities distribute uniformly, and then extend the results to multi-dimensional spaces. Finally, we deal with non-uniform distributions using a specialized spatio-temporal histogram. Extensive experiments confirm that the proposed formulae are highly accurate (average error below 10%).

1. Introduction

An important operation in spatio-temporal databases and mobile computing systems is to predict objects' future location based on information at the current time (e.g., for collision detection). For this purpose the database usually represents object movement as a function of time and stores the function parameters [1–4]. As an example, given the location $o(0)$ of object o at the current time 0 and its velocity o_V, its position at future time t can be computed as $o(t) = o(0) + o_V t$. According to this representation, an update is necessary only when the function parameters (i.e., o_V) change.

Given two sets S_1, S_2 of objects, a *spatio-temporal join* retrieves all pairs of objects $\langle o_1, o_2 \rangle$ such that $o_1 \in S_1$, $o_2 \in S_2$, and $|o_1(t_q), o_2(t_q)| \leqslant d$, i.e., the distance $|o_1(t_q), o_2(t_q)|$ between objects o_1 and o_2 at a (future) query timestamp t_q is below a threshold d. For instance, consider the query

☆ Recommended by Yannis Ioannidis.

*Corresponding author. Tel.: +852 23586971; fax: +852 23581477.
E-mail addresses: jimeng@cs.cmu.edu (J. Sun), taoyf@cs.cityu.edu.hk (Y. Tao), dimitris@cs.ust.hk (D. Papadias), gkollios@cs.bu.edu (G. Kollios).
URL: http://www.cs.ust.hk/~dimitris/.

0306-4379/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.is.2005.02.002

"retrieve all pairs of airplanes that will come closer than 10 miles 5 minutes from now". This (self-join) example outputs pairs of moving objects; in some cases one of the inputs can be static: "retrieve all pairs $\langle$ city c, typhoon $t\rangle$ such that t will cover c at 9 a.m. tomorrow according to the current spreading speed of t". An important variation is the *constrained join*, which involves an additional constraint window to limit the data space of interest. For instance, in the previous example, an analyst may be interested only in cities in the southeast US region.

In this paper we discuss the *selectivity* of spatio-temporal joins, which is defined as the number of result pairs divided by the size of the Cartesian product of the input data sets. Estimating the join selectivity is important for several reasons.

- As with conventional and spatial [5] databases, selectivity estimation is vital to query optimization for computing the best execution plan.
- In many applications users are interested in the number of joined pairs (i.e., a *join counting* query) rather than the concrete results. For example, prediction of potential congestions requires the traffic volume rather than the IDs of cars [6]. Furthermore, stream (spatio-temporal) databases [7,8] may maintain only aggregate information in order to deal with voluminous updates.
- Performing an exact join (which is time consuming) is meaningless in applications with very frequent motion function changes because the result may already be invalidated before the join processing terminates. In such cases, a fast estimation of the output size is the only meaningful computation that can be performed given the tight time limit.

Although spatial join selectivity can be computed using several methods [9–11], their application in spatio-temporal scenarios leads to significant error. Similarly, existing work [12] on estimation of spatio-temporal window queries on a single data set cannot be efficiently adapted for joins. Motivated by this, we address the problem by first proposing fundamental probabilistic formulae for spatio-temporal joins on uniform (point and rectangular) objects. Then, we integrate these equations with spatio-temporal histograms to support non-uniform data. Compared to the previous approaches, the proposed histogram achieves significantly lower estimation error and is incrementally updatable (whereas previous solutions require frequent re-construction). We evaluate the efficiency of our methods with extensive experimentation.

The rest of the paper is organized as follows. Section 2 reviews previous work on spatial joins, histograms, and spatio-temporal prediction. Section 3 analyzes spatio-temporal join selectivity on uniform data, while Section 4 extends the results to non-uniform data (using histograms). Section 5 experimentally evaluates the proposed methods, and Section 6 concludes the paper with directions for future work.

2. Related work

Section 2.1 overviews spatial join algorithms (assuming knowledge of R-trees [13,14]), and join selectivity estimation on static objects. Section 2.2 introduces MinSkew, a multi-dimensional histogram that constitutes the starting point of our spatio-temporal histogram. Section 2.3 discusses spatio-temporal range selectivity and elaborates why it cannot be applied for join selectivity estimation. Finally Section 2.4 reviews the time parameterized R-tree (TPR-tree), an index structure for moving objects which is employed in our histogram techniques.

2.1. Spatial join selectivity

Consider two objects o_1, o_2 belonging to spatial data sets S_1, S_2 and satisfying the join condition $|o_1, o_2| \leqslant d$. Join processing algorithms [15–17] are based on the observation that o_1 and o_2 should reside in R-tree nodes E_1 and E_2, respectively, whose minimum bounding rectangles (MBRs) satisfy the property $|E_1, E_2| \leqslant d$. Thus, qualifying pairs are retrieved by synchronously traversing the R-trees of S_1 and S_2 in a top-down fashion, recursively following node pairs that are within the distance constraint. The special case where $d = 0$

corresponds to the intersection join and has received considerable attention. Notice, however, that intersection join is not meaningful for point data since it reduces to equality of points.

Spatial join selectivity was first studied in [18], which presents a formula for uniform data using the results of previous work [19,20] on window query selectivity. Several histogram-based approaches have been proposed for non-uniform distributions. In particular, the histograms of [5,21] divide the universe regularly, while more sophisticated techniques [9,11] perform the partitioning according to the data distribution. On the other hand, [10] employs a different approach based on power laws. Further, [5] studies the selectivity of complex queries that involve multiple data sets. All the above methods require the knowledge of distributions of the join data sets. In spatio-temporal databases, however, the distribution continuously changes due to object movements. Hence, it is extremely expensive (both in terms of computation time and storage cost) to pre-compute the distributions for future timestamps. Furthermore, even if such distributions are obtained, they will soon be invalidated due to subsequent updates, rendering re-computation necessary. Therefore, traditional approaches for spatial join selectivity are insufficient for moving objects. Effective techniques should take the specialized problem characteristics into account.

2.2. MinSkew

MinSkew [22] is a spatial histogram originally proposed for selectivity estimation of window queries on non-uniform data sets. It partitions the space into a set of buckets such that the MBRs of all buckets are disjoint, and their union covers the entire universe. Each bucket b_i contains the number $b_i.num$ of objects whose centroids fall inside b_i.MBR, and the average extent $b_i.len$ of these objects. Fig. 1 illustrates a query q and an intersecting bucket b in the 2D space. The gray area corresponds to the intersection between b.MBR and the *extended query region*, obtained by enlarging each edge of q with distance $b.len/2$. The expected number of rectangles in b intersecting q is estimated as $b.num \times areaG/\text{area}(b.\text{MBR})$,

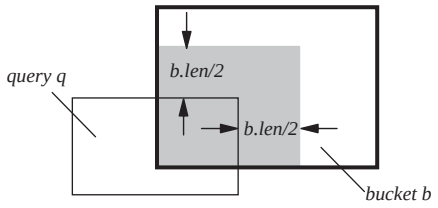

Fig. 1. Estimating the selectivity inside a bucket.

where $areaG$ and area(b.MBR) are the areas of the gray region and b.MBR, respectively. The estimated selectivity is obtained by summing the results for all such intersecting buckets.

To ensure satisfactory accuracy, the above estimation requires that objects in each bucket b have similar sizes and their centroids distribute uniformly in b.MBR. To quantify the degree of uniformity, [22] defines the *spatial-skew* (denoted as $b.skew$) for a bucket b as the variance[1] of the spatial densities[2] of all points inside it. Since a small spatial-skew indicates better uniformity, MinSkew aims at minimizing $\sum_{i=1\sim n}(b_i.num \cdot b_i.skew)$, i.e., the weighted sum of the spatial-skews of the buckets. Computing the optimal buckets, however, is NP-hard [23]. To reduce the computation cost, [22] partitions the original space into a grid with $H \times H$ regular cells (where H is a parameter called the *resolution*), and associates each cell c with (i) the number $c.num$ of objects whose centroids fall in c.MBR, (ii) the average extent length $c.len$ of objects satisfying (i), and (iii) the density $c.den$ of the cell (i.e., the number of objects intersecting c.MBR). Fig. 2a shows an example ($H = 3$) for a data set with 8 objects, and Fig. 2b illustrates the information associated with the cells (*len* is not shown because it is not needed for partitioning). A greedy algorithm builds the histogram that minimizes the total-skew, under the constraint that each bucket must cover an integer number of cells. The final buckets are shown in Fig. 2c, together with their associated information (notice that the

[1] Given n numbers $a_1, a_2, \ldots, a_n$, the variance equals $\frac{1}{n}\sum_{i=1}^{n}(a_i - \overline{a})^2$, where $\overline{a}$ is the average of $a_1, a_2, \ldots, a_n$.

[2] The density of a point is defined as the number of objects that cover the point.

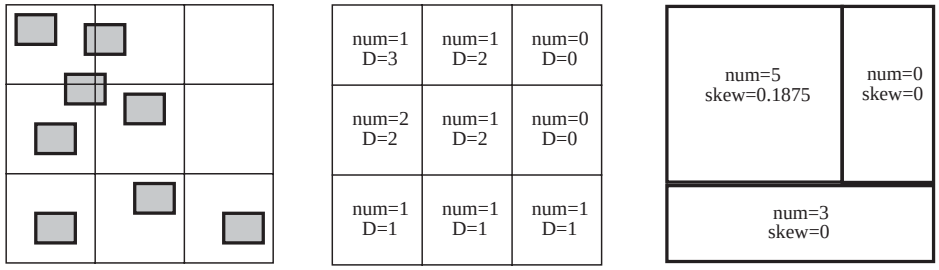

Fig. 2. Building the histogram. (a) 3×3 grid, (b) cell information, (c) the final buckets.

spatial-skews are very small) computed as follows:

$$b.num = \sum_{each\ cell\ c\ in\ b} c.num,$$

$$b.len = \frac{\sum_{each\ cell\ c\ in\ b} c.num \cdot c.len}{\sum_{each\ cell\ c\ in\ b} c.num},$$

$$b.skew = \frac{1}{|C|} \sum_{each\ cell\ c\ in\ b} \left(c.den - \overline{den}\right)^2,$$

where $|C|$ is the number of cells covered by b, and $\overline{den}$ denotes their average density. MinSkew can be applied in arbitrary dimensionality with straightforward modifications.

2.3. Spatio-temporal range selectivity

Given a set of moving objects, a *spatio-temporal window query* retrieves all the objects that will appear in a query region at the query time t_q. Choi and Chung [12] extend MinSkew with velocities to estimate the selectivity of window queries (defined as the ratio between the number of qualifying objects and the data set cardinality). Fig. 3a shows 8 moving points, where the arrows (numbers) indicate the directions (values) of their velocities (a minus value indicates movement in the negative direction of the axis). Fig. 3b shows the corresponding spatio-temporal histogram built in two steps. First, the spatial extents of the buckets are determined in the same way as the traditional MinSkew algorithm (by ignoring the velocities). Then, each bucket b is associated with a velocity bounding rectangle (VBR) (b_{Vx-}, b_{Vx+}, b_{Vy-}, b_{Vy+}), such that (i) b_{Vx-} (b_{Vx+}) equals the minimum (maximum) velocity along the x-dimension of the objects inside, and (ii) b_{Vy-} (b_{Vy+}) is defined similarly with respect to the y-dimension. In Fig. 3b, the VBRs of buckets b_1, b_2, and b_3 are $(-2, 1, -3, 1)$, $(0, 0, 0, 0)$, and $(-1, 2, -1, -1)$, respectively.

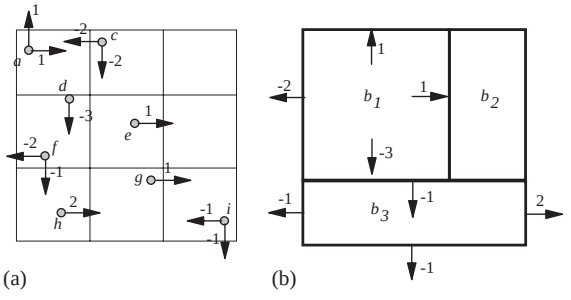

Fig. 3. A spatio-temporal histogram. (a) Object location and velocities, (b) the final buckets.

Accurate spatio-temporal selectivity estimation requires that the location (velocities) of the objects inside a bucket uniformly distribute(s) in the bucket's MBR (VBR). The data partition in Fig. 3, however, is decided according to spatial information; thus, the uniformity of velocity cannot be guaranteed, which may lead to significant estimation error. Furthermore, the histogram is not incrementally updatable and must be re-built very frequently to maintain satisfactory accuracy ([12] suggests re-building at every single timestamp). To see this, assume that in Fig. 3b the

y-velocity of object d (which determines b_{1Vy-}) changes to -1, after which b_{1Vy-} should be adjusted to the y-velocity of c (i.e., -2), because it is now the minimum y-velocity of all objects in b_1. This, however, is not possible because the histogram does not contain detailed information about the velocities of individual objects.

Based on the above histogram, Choi and Chung [12] discuss selectivity estimation for (predictive) range queries on point data. Specifically, given a point q, distance d, and timestamp t_q, a range query retrieves all the objects whose distances from q at time t_q are less than d. The selectivity is defined as the number of retrieved objects divided by the data set cardinality. This technique could estimate join selectivity (on point data sets) by regarding a join as the combination of multiple range queries. Specifically, given two data sets S_1, S_2, we build a histogram on S_1. Then, for each point $p_2 \in S_2$, the number of objects p_1 in S_1 that qualify the join condition with p_2 (i.e., $|p_1(t_q), p_2(t_q)| \leqslant d$) can be predicted using the method in [12]. The total number of qualifying pairs equals the sum of the estimates for all points in S_2. This approach, however, has the following problems. First, it incurs significant computation overhead because, the number of range selectivity predictions equals the cardinality of S_2 (which can be huge in practice). Second, it requires maintaining all the objects in one data set, leading to expensive space consumption. Further, for large data sets, some data might need to be stored on disk, in which case selectivity estimation would involve I/O accesses, further compromising the estimation time. In Section 3, we develop alternative solutions to overcome these problems.

2.4. The TPR-tree

The TPR-tree [3] is an extension of the R-tree that can answer predictive queries on moving objects. Each object is represented with an MBR that bounds its extents at the current time, and a velocity vector. Fig. 4a shows the representation of two objects u and v. As before, the arrows indicate the velocity directions, and the numbers correspond to their values. A non-leaf entry (e.g., E in Fig. 4a) stores an MBR and a VBR. Specifically, as in traditional R-trees the MBR tightly encloses the extents of all entries in the node at the current time, while the definition of VBR is similar to those of the spatio-temporal buckets in Fig. 3b (i.e., it tightly bounds the velocities of the objects in its subtree). The MBR of an entry E continuously grows with time according to its VBR, which ensures that the MBR always bounds the underlying objects (but is not necessarily tight).

Fig. 4b shows u, v and the enclosing node E at time 1. Observe how the extents and positions of u, v, E change, and that the rectangle of E is larger than the tightest MBR for u and v. Future MBRs (e.g., those in Fig. 4b) are not stored explicitly, but are computed based on the entries' current extents and velocities. A spatio-temporal window query is processed in exactly the same way as in the R-tree, except that the extents of the MBRs at some future

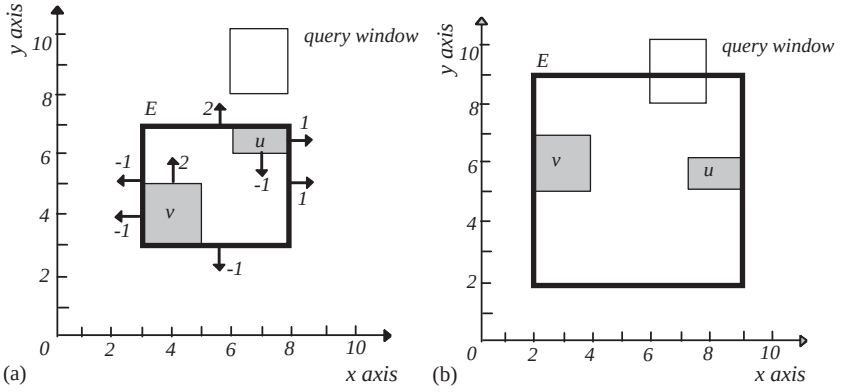

Fig. 4. Representation of entries in the TPR-tree. (a) The boundaries at current time 0, (b) the boundaries at future time 1.

time are first calculated dynamically before being compared to the query window. For the query (retrieving objects intersecting the query window at timestamp 1) shown in Fig. 4, for example, node E must be visited because its MBR intersects the query at time 1 (although its MBR at time 0 does not).

3. Spatio-temporal join selectivity

Formally, given two sets S_1, S_2 of m-dimensional moving objects (points or rectangles), a future timestamp t_q, and distance threshold d, a *spatio-temporal join* retrieves all pairs of objects $\langle o_1, o_2 \rangle$ such that $o_1 \in S_1$, $o_2 \in S_2$, and $|o_1(t_q), o_2(t_q)| \leqslant d$, where $|o_1(t_q), o_2(t_q)|$ is the distance between o_1 and o_2 at t_q. We consider that the distance $|o_1(t_q), o_2(t_q)|$ is computed according to the L_p norm. Specifically, denoting the coordinates of an m-dimensional point p as $p.x_0, p.x_1, \ldots, p.x_m$, the L_p-distance $|o_1(t_q), o_2(t_q)|$ equals $\left(\sum_{i=1-d} |p_1.x_i - p_2.x_i|^p\right)^{1/p}$. A *constrained spatio-temporal join* is similar to a normal spatio-temporal join, except that it involves a *constraint window* W_q which is an m-dimensional rectangle. A pair of qualifying objects $\langle o_1, o_2 \rangle$ must satisfy all the conditions of a normal join, and the *additional* predicate that o_1 and o_2 both intersect W_q at time t_q. The selectivity of the (normal or constrained) join is the ratio between the number of result pairs and the size of the Cartesian product $S_1 \times S_2$.

Interestingly, as discussed in [10], the join selectivity under various L_p norms (for different p) differs only by a constant factor. As a result, to deal with arbitrary L_p norm, it suffices to solve the problem under a particular value of p. In the sequel, we focus on $L_{\infty;}$ (i.e., $|o_1(t_q), o_2(t_q)| = \max_{i=1,2,\ldots,m} |p_1.x_i - p_2.x_i|$), since the resulting equations are the simplest in this case. The distance between two (hyper-) rectangles r_1, r_2 is the minimum of the distances between all pairs of points in r_1 and r_2, respectively, or more formally: $|r_1, r_2| = \min \{|p_1, p_2| \text{ for all } p_1 \in r_1, \text{ and } p_2 \in r_2\}$. Under $L_{\infty;}$, two (point or rectangle) objects o_1, o_2 are within distance d if and only if the extended region of o_1, obtained by enlarging its extents with length d along all dimensions, intersects o_2.

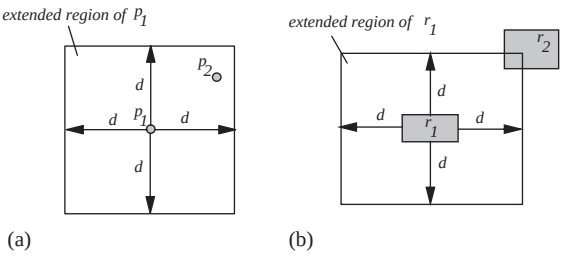

Fig. 5. Objects within distance d (the $L_{\infty;}$ norm) from each other. (a) Points, (b) rectangles.

Fig. 5a and b illustrate this for 2D points and rectangles, respectively. When $d = 0$, the condition $|o_1(t_q), o_2(t_q)| \leqslant d$ reduces to simple intersection.

We use R_1 (V_1) to represent the MBR (VBR) that tightly encloses the location (velocities) of all the objects in S_1, and similarly, $R_2(V_2)$ for S_2. Note that R_1 and R_2 may cover different sub-spaces of the universe (i.e., we allow objects of the two sets to distribute in different areas). The objective is to predict the join selectivity based solely on R_1, R_2, V_1, V_2, assuming that (i) the location of the objects inside S_1 (S_2) distributes uniformly in R_1 (R_2), (ii) the object velocities are uniform in V_1 (V_2), and (iii) all the dimensions are independent. Section 3.1 first solves the problem for 1D space, and Section 3.2 extends the results to higher dimensionality, both assuming L_∞. Section 3.3 extends the results to the constrained join and other L_p norms. In Section 4 we overcome the uniformity assumptions with the aid of histograms. Table 1 lists the symbols that will appear frequently in our analysis.

3.1. Solution for one-dimensional space

We first consider the case of 1D point data and then solve the problem for 1D intervals. Let $[R_1.x_-, R_1.x_+]$ be the extent of R_1 (the MBR of S_1), and $[V_1.v_-, V_1.v_+]$ the range of V_1 (the VBR of S_1). The Cartesian product $S_1 \times S_2$ consists of $N_1 N_2$ pairs of points. Let PQ_{1Dpt} be the probability for a pair to qualify the join condition; the expected number of qualifying pairs can be represented as $N_1 N_2 PQ_{1Dpt}$. Furthermore, since the selectivity equals the number of qualifying

pairs divided by N_1N_2, PQ_{1Dpt} directly corresponds to the join selectivity. In particular, the derivation of PQ_{1Dpt} is equivalent to the following problem: Given two points p_1, p_2 such that p_1 (p_2) uniformly distributes in the range $[R_1.x_-, R_1.x_+]$ ($[R_2.x_-, R_2.x_+]$), and the velocity of p_1 (p_2) uniformly distributes in $[V_1.v_-, V_1.v_+]$ ($[V_2.v_-, V_2.v_+]$), compute the probability PQ_{1Dpt} that $|p_1(t_q), p_2(t_q)| \leqslant d$.

Table 1
Frequent symbols in the analysis

Symbol	Description
M	Dimensionality of the data space
t_q	The query timestamp
S_1, S_2	The two data sets participating the join
$R_1(R_2)$	The MBR of $S_1(S_2)$ covering the location of objects in the set
$V_1(V_2)$	The VBRs of $S_1(S_2)$ covering the velocities of objects in the set
$N_1(N_2)$	The number of objects in $S_1(S_2)$
$p.x_i$	The ith coordinate of point p
$[R.x_{i-}, R.x_{i+}]$	The extent along the ith dimension of MBR R
$[V.v_{i-}, V.v_{i+}]$	The extent along the ith dimension of VBR V
$PQ_{\{1Dpt,\ 1Dintv\}}$	The qualifying probability for 1D {point, interval} objects
$Sel_{\{1Dpt,1Dintv,\ pt,\ rect\}}$	The join selectivity for {1D points, 1D intervals, mD points, mD rectangles}
W_q	The constrained window

For two particular points p_1, p_2, the probability for $\langle p_1, p_2 \rangle$ to qualify depends on their relative positions. To see this, consider Fig. 6a, where the x- and y-axes correspond to the spatial and temporal dimensions, respectively (the current time is 0). The thick line segments (on the spatial dimension) represent the MBRs of S_1 and S_2, while their VBRs are illustrated by the lines passing through the end points of the corresponding MBRs (the slopes of these lines indicate the velocity values). At query time t_q, the entire range that can be reached by p_1 constitutes line segment AB, where point A (B) is the extreme position if p_1 travels with the minimum (maximum) velocity $V_1.v_-$ ($V_1.v_+$). The probability that p_1 reaches *any* point on segment AB is the same, due to the fact that the velocity of p_1 uniformly distributes in the range $[V_1.v_-, V_1.v_+]$ (note that: every position on segment AB can be reached via a *unique* velocity of p_1). Similarly, segment CD consists of all the possible location of p_2 at time t_q. Assuming the distance threshold d is 0, $\langle p_1, p_2 \rangle$ may qualify because, as shown in Fig. 6a, segments AB and CD intersect (i.e., it is possible for p_1 and p_2 to reach the same position at t_q).

Fig. 6b shows similar situation except that p_1 and p_2 are farther apart from each other at the current time. Notice that, in this case AB and CD do not intersect, indicating that $\langle p_1, p_2 \rangle$ does not belong to the result. Motivated by this, we denote with $P_{pair}(l_1,l_2)$ the probability that pair $\langle p_1, p_2 \rangle$ qualifies if p_1 (p_2) lies at location l_1 (l_2). Consequently, the qualifying probability PQ_{1Dpt} corre-

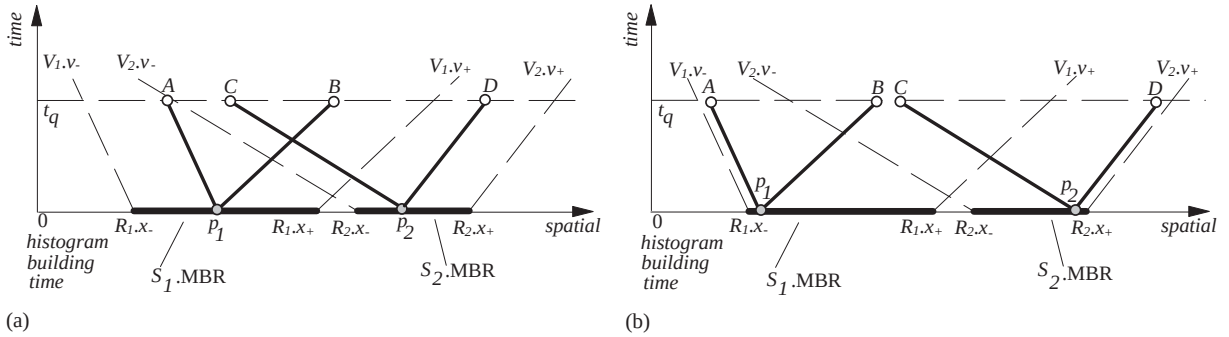

Fig. 6. Qualifying pairs (distance threshold $d = 0$). (a) $\langle p_1, p_2 \rangle$ may qualify, (b) $\langle p_1, p_2 \rangle$ cannot qualify.

sponds to the average of $P_{pair}(l_1,l_2)$ over all possible positions for l_1 and l_2, or formally:

$$PQ_{1Dpt} = \frac{1}{(R_1.x_+ - R_1.x_-)(R_2.x_+ - R_2.x_-)} \times \int_{R_1.x_-}^{R_1.x_+} \int_{R_2.x_-}^{R_2.x_+} P_{pair}(l_1,l_2)\,\mathrm{d}l_2\,\mathrm{d}l_1. \quad (1)$$

Given two points at l_1 and l_2, respectively, $P_{pair}(l_1,l_2)$ corresponds to $P\{|p_1(t_q),p_2(t_q)|\leqslant d\}$, i.e., the probability that the distance of p_1 and p_2 at time t_q is not greater than the threshold d. Assuming the velocities of p_1 and p_2 to be u_1 and u_2, respectively, $p_1(t_q)$ and $p_2(t_q)$ can be represented as

$$p_1(t_q) = l_1 + t_q \cdot u_1 \quad \text{and} \quad p_2(t_q) = l_2 + t_q \cdot u_2.$$

Thus,

$$P_{pair}(l_1,l_2) = P\{|p_1(t_q),p_2(t_q)|\leqslant d \,|\, p_1 = l_1 \ \textit{and}\ p_2 = l_2\} = P\{|(l_1-l_2)+t_q\cdot(u_1-u_2)|\leqslant d\}.$$

The above equation can be converted to

$$P_{pair}(l_1,l_2) = P\left\{\frac{l_1-l_2-d}{t_q} + u_1 \leqslant u_2 \leqslant \frac{l_1-l_2+d}{t_q} + u_1\right\}. \quad (2)$$

Since u_1 and u_2 uniformly distribute in $[V_1.v_-, V_1.v_+]$ and $[V_2.v_-, V_2.v_+]$, respectively, they satisfy the following probability density functions:

$$f(u_1) = \frac{1}{V_1.v_+ - V_1.v_-} \quad \text{and} \quad f(u_2) = \frac{1}{V_2.v_+ - V_2.v_-}.$$

Therefore, the right-hand side of Eq. (2) can be written as:[3]

$$P\left\{\frac{l_1-l_2-d}{t_q} + u_1 \leqslant u_2 \leqslant \frac{l_1-l_2+d}{t_q} + u_1\right\}$$

[3]In this paper we follow the convention that if $a>b$, then $\int_a^b f(x)\,\mathrm{d}x = 0$.

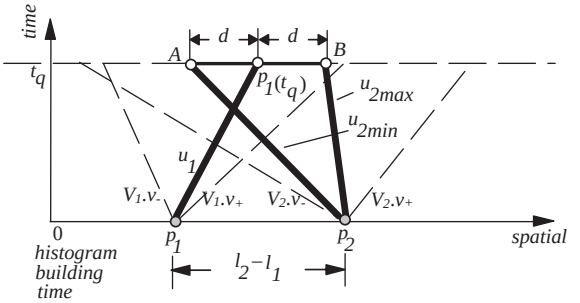

Fig. 7. Computing the probability $P_{pair}(l_1, l_2)$.

$$= \int_{V_1.v_-}^{V_1.v_+} \left\{ f(u_1) \times \left[\int_{\max(V_2.v_-,u_1+(l_1-l_2-d)/t_q)}^{\min(V_2.v_+,u_1+(l_1-l_2+d)/t_q)} f(u_2)\,\mathrm{d}u_2 \right] \right\} \mathrm{d}u_1$$
$$= \frac{1}{(V_1.v_+ - V_1.v_-)(V_2.v_+ - V_2.v_-)} \times \int_{V_1.v_-}^{V_1.v_+} \int_{\max(V_2.v_-,u_1+(l_1-l_2-d)/t_q)}^{\min(V_2.v_+,u_1+(l_1-l_2+d)/t_q)} 1\,\mathrm{d}u_2\,\mathrm{d}u_1. \quad (3)$$

The above integral can be solved into closed form as presented in the appendix. To understand the formula, consider Fig. 7, where point $p_1(t_q)$ shows the location of p_1 at t_q when it travels at speed u_1 ($\in [V_1.v_-, V_1.v_+]$). Then, line segment AB corresponds to the set of positions for a qualifying point p_2 at time t_q (i.e., $p_2(t_q)$ is within distance d from $p_1(t_q)$).

Let u_{2min} and u_{2max} be the velocities according to which p_2 reaches points A and B at time t_q, respectively. It follows that the probability that $\langle p_1,p_2\rangle$ is a join result, is $(u_{2max}-u_{2min})/(V_2.v_+ - V_2.v_-)$, i.e., the probability that the velocity u_2 of p_2 falls in the range $[u_{2min}, u_{2max}]$, given that u_2 uniformly distributes in $[V_2.v_-, V_2.v_+]$. So far we have considered a particular value of u_1, while, as shown in Eq. (3), in order to compute $P_{pair}(l_1, l_2)$ we must consider all possible values in $[V_1.v_-, V_1.v_+]$ (i.e., the outer integral in the formula). Combining Eqs. (1)–(3), we have represented PQ_{1Dpt} as a function of R_1, R_2, V_1, V_2; the following equation gives the complete formula for

join selectivity of 1D points:

$$Sel_{1D\text{-}pt}(d,t_q) = \frac{1}{(R_1.x_+ - R_1.x_-)(R_2.x_+ - R_2.x_-)(V_1.v_+ - V_1.v_-)(V_2.v_+ - V_2.v_-)} \times \int_{R_1.x_-}^{R_1.x_+} \int_{R_2.x_-}^{R_2.x_+} \int_{V_1.v_-}^{V_1.v_+} \int_{\min(V_2.v_+,\ u_1+(l_1-l_2-d)/t_q)}^{\min(V_2.v_+,\ u_1+(l_1-l_2+d)/t_q)} 1\,du_2\,du_1\,dl_2\,dl_1. \quad (4)$$

Extending the results to interval objects is straightforward. Assume that S_1 (S_2) contains intervals with lengths I_1 (I_2); then, similar to the discussion for point objects, the selectivity corresponds to the qualifying probability PQ_{1Dintv} that a pair of intervals $\langle i_1,i_2\rangle$ in the Cartesian product $S_1 \times S_2$ has distance no longer than d at future time t_q. Observe that i_1 and i_2 are closer than d, if and only if their centroids are within distance $d + (I_1 + I_1)/2$. This is illustrated in Fig. 8, where intervals i_1 and i_2 (with lengths I_1 and I_2) travel at velocities u_1 and u_2, respectively. Therefore, the selectivity PQ_{1Dintv} can be estimated using Eq. (4), except that, as shown in Eq. (5), (i) d should be replaced with $d + (I_1 + I_1)/2$, and (ii) the lower/upper limit of the integral should be modified to capture the fact that the centroid of i_1 distributes in $[R_1.x_- + I_1/2, R_1.x_+ - I_1/2]$ (the range is $[R_2.x_- + I_2/2, R_2.x_+ - I_2/2]$ for i_2).

$$Sel_{1D\text{-}intv}(d,I_1,I_2,t_q) = PQ_{1Dintv} = \frac{1}{(R_1.x_+ - R_1.x_-)(R_2.x_+ - R_2.x_-)(V_1.v_+ - V_1.v_-)(V_2.v_+ - V_2.v_-)} \times \int_{R_1.x_- + I_1/2}^{R_1.x_+ - I_1/2} \int_{R_2.x_- + I_2/2}^{R_2.x_+ - I_2/2} \int_{V_1.v_-}^{V_1.v_+} \int_{\min(V_2.v_+,\ u_1+(l_1-l_2-d-I_1/2-I_2/2)/t_q)}^{\min(V_2.v_+,\ u_1+(l_1-l_2+d+I_1/2+I_2/2)/t_q)} 1\,du_2\,du_1\,dl_2\,dl_1. \quad (5)$$

3.2. Arbitrary dimensionality

In this section we present the equations for spatio-temporal join selectivity in m-dimensional spaces, starting with point data sets before extending the results to rectangles. The MBR R_1 (VBR V_1) of set S_1 is now an m-dimensional rectangle, and its extent along the ith dimension ($1 \leqslant i \leqslant m$) is $[R_1.x_{i-}, R_1.x_{i+}]$ ($[V_1.v_{i-}, V_1.v_{i+}]$); similar notations are used for S_2. Let PQ_{pt} be the probability that a pair of points $\langle p_1,p_2\rangle$ in $S_1 \times S_2$ satisfies the join condition. The crucial observation is that (due to the definition of L_∞;) $|p_1(t_q), p_2(t_q)| \leqslant d$ if and only if $|p_1.x_i(t_q) - p_2.x_i(t_q)| \leqslant d$ for all dimensions $1 \leqslant i \leqslant m$, where $p_1.x_i(t_q)$ represents the ith coordinate of p_1 at time t_q. Let $PQ_{1Dpt\text{-}i}$ ($1 \leqslant i \leqslant m$) be the probability that the ith coordinates of p_1 and p_2 qualify. Since the dimensions are independent, we have:

$$PQ_{pt} = \prod_{i=1}^{m} PQ_{1Dpt\text{-}i}. \quad (6)$$

The computation of $PQ_{1Dpt\text{-}i}$ along each dimension is based on Eq. (4) (passing d, t_q), except that $R_1.x_-$, $R_1.x_+$, $V_1.v_-$, $V_1.v_+$ should be replaced with $R_1.x_{i-}$, $R_1.x_{i+}$, $V_1.v_{i-}$, and $V_1.v_{i+}$, respectively. As in the 1D case, PQ_{pt} corresponds to the selectivity $Sel_{point}(d, t_q, m)$.

In order to estimate the join selectivity for rectangular objects, we denote I_{1i} (I_{2i}) as the extents of the objects in S_1 (S_2) along the ith dimension. Given a pair of rectangles $\langle r_1,r_2\rangle$, let their extents (on the ith dimension) at time t_q be $r_1.i_i(t_q)$ and $r_2.i_i(t_q)$, respectively. In analogy with the point case, $\langle r_1,r_2\rangle$ is a join result if and only if

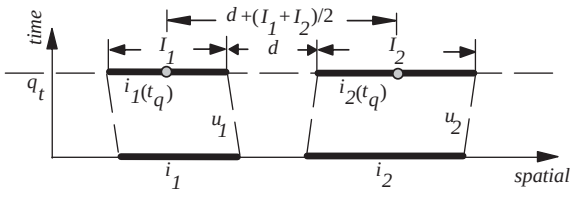

Fig. 8. A pair of qualifying intervals.

$\langle r_1.i_i(t_q),\ r_2.i_i(t_q)\rangle$ qualifies along the ith dimension $(1 \leqslant i \leqslant m)$, or equivalently, $|r_1.i_i(t_q), r_2.i_i(t_q)| \leqslant d$. Hence, the selectivity for m-dimensional rectangles is represented as:

$$Sel_{rect}(d, t_q, m) = \prod_{i=1}^{m} PQ_{1Dintv\text{-}i}, \tag{7}$$

where $PQ_{1Dintv\text{-}i}$ is the probability that $\langle r_1.i_i(t_q), r_2.i_i(t_q)\rangle$ qualifies, and is computed according to Eq. (5) (passing d, I_{1i}, I_{2i}, and t_q).

Consequently, the minimum u_{2min} (maximum u_{2max}) velocity of p_2 is decided by the slope of segment connecting p_2 and $C(B)$. Therefore, the probability $P_{C\text{-}pair}(l_1, l_2, u_1, W_q)$ that $\langle p_1,p_2\rangle$ qualifies (given l_1, l_2, u_1) equals $(\min(V_2.v_+, u_{2max}) - \max(V_2.v_-, u_{2min}))/(u_{2max} - u_{2min})$, taking into account the fact that u_2 is in $[u_{2min}, u_{2max}]$. Similar to the discussion in Section 3.1, the overall probability $P_{C\text{-}1Dpt}$ (i.e., the selectivity $Sel_{C\text{-}1Dpt}$) can be obtained by integrating $P_{C\text{-}pair}(l_1, l_2, u_1, W_q)$ over all the possible values for l_1, l_2, u_1, as shown in Eq. (8):

$$\begin{aligned} Sel_{C\text{-}1Dpt}(d, t_q, W_q) &= P_{C\text{-}1Dpt}(d, t_q, W_q) \\ &= \frac{1}{(R_1.x_+ - R_1.x_-)(R_2.x_+ - R_2.x_-)(V_1.v_+ - V_1.v_-)(V_2.v_+ - V_2.v_-)} \\ &\quad \times \int_{R_1.x_-}^{R_1.x_+} \int_{R_2.x-}^{R_2.x_+} \int_{u_{1\min}}^{u_{1\max}} (\min(V_2.v_+, u_{2\max}) - \max(V_2.v-, u_{2\min}))\, \mathrm{d}u_1\, \mathrm{d}l_2\, \mathrm{d}l_1. \end{aligned} \tag{8}$$

3.3. Extensions

In this section, we first study the selectivity estimation for constrained joins (again for L_∞ metric), and then explain the extension to arbitrary L_p norms. Our analysis of the constrained join follows the same framework as the discussion on normal joins. Specifically, we first solve the fundamental problem involving 1D points, and then tackle the general version (multi-dimensional, rectangle data) by reducing it to the fundamental case. Fig. 9 illustrates an example, where line segment AB denotes the constrained window W_q at time t_q (i.e., if $\langle p_1,p_2\rangle$ is a qualifying join pair, then both p_1 and p_2 must appear in AB at t_q). Similar to the derivation in Section 3.1, we fix the location of p_1 and p_2 at time 0 to specific positions l_1, l_2, respectively, and the velocity of p_1 to a particular value u_1. Note that the permissible values of u_1 (in order to appear in W_q at t_q) depend on the location l_1 of p_1. In Fig. 9, for instance, the minimum (maximum) value u_{1min} (u_{1max}) of u_1 is such that p_1 will reach point A (B) (i.e., an end point of W_q) at this speed.

Having fixed $p_1(t_q)$, the possible location of $p_2(t_q)$ (for $\langle p_1,p_2\rangle$ to qualify the join predicate) is confined to segment CB, which contains all the positions that are in W_q and have distances at most d from $p_1(t_q)$.

Next we study constrained joins on 1D intervals. Fig. 9 shows two intervals i_1, i_2 (with lengths I_1, I_2, respectively) that qualify the join (note that both $i_1(t_q)$ and $i_2(t_q)$ intersect the constraint region W_q). The crucial observation is that, the centroids of i_1, i_2 must satisfy the following conditions: (i) the distance between them is within $d + (I_1 + I_2)/2$ (similar to Fig. 8), and (ii) the centroid of $i_1(i_2)$ should have distance no more than $W_q/2 + I_1/2$ ($W_q/2 + I_2/2$) from the centroid of W_q in order for $i_1(t_q)$ ($i_2(t_q)$) to intersect W_q. Hence, the selectivity for intervals can be reduced to Eq. (8) by considering the interval centroids (as with Eq. (5), the integral ranges must be modified to ensure that both i_1 and i_2 lie in R_1 and R_2, respectively). The extension to multiple dimensions is trivial: we simply multiply the selectivity on each individual axis, as shown in Section 3.2.

We finish this section by clarifying the application of our techniques to general L_p norms other than L_∞. For this purpose, we review the concept of *sphere ratio* $sr(L_{p1},L_{p2},m)$ between two different norms L_{p1}, L_{p2} $(p_1 \neq p_2)$ for dimensionality m. Specifically, $sr(L_{p1},L_{p2},m)$ equals the ratio between the volumes of m-dimensional spheres with the same radius in L_{p1} and L_{p2} (it can be easily verified that the ratio is a constant independent of the radius). To illustrate this, Fig. 10 demonstrates the spheres with radius d under L_1, L_2, L_∞ in the 2D

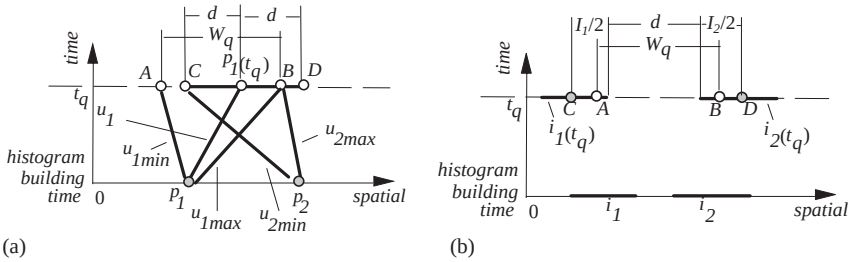

Fig. 9. Analysis of constrained joins. (a) Deriving $P_{C\text{-}pair}(l_1, l_2, u_1, W_q)$ for point data, (b) a pair of qualifying intervals.

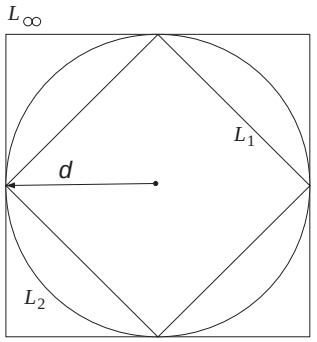

Fig. 10. Spheres in L_1, L_2, $L_{\infty;}$ norms.

space (particularly, the sphere of L_2 is simply a circle). The sphere ratio between L_1 (L_∞) and L_2, for example, equals the area of the inner (outer) square divided by that of the circle. Faloutsos et al. [10] show the following interesting result: if Sel_{p1} and Sel_{p2} are the join selectivities under the L_{p1} and L_{p2} metrics and m is the dimensionality, then Sel_{p1}/Sel_{p2} equals $sr(L_{p1},L_{p2},m)$. Based on this observation, they estimate the selectivity of arbitrary L_p norm by multiplying the selectivity of $L_{\infty;}$ with $sr(L_{p1},L_{p2},m)$. The same method can also be applied in our case.

4. Spatio-temporal histograms

In the previous section we presented the formulae for estimating the join selectivity for data sets where objects' location and velocities distribute uniformly in their corresponding ranges. The uniformity assumption, however, does not usually hold in real data sets, and thus direct application of the above models will lead to significant error. In this section we deal with this problem using histograms that partition the universe into separate buckets, such that the data distribution within a bucket is nearly uniform. Then, the previous equations are applied locally (in each bucket), and the overall prediction is obtained by summing up the individual estimations. The accuracy of this approach depends on the quality of the histogram, which must guarantee that in each bucket both the location and the velocity distributions are as uniform as possible. In Section 4.1, we first show that existing histograms cannot achieve this (thus, leading to erroneous estimation), and then provide an alternative solution to avoid their defects. Section 4.2 explains how to use the proposed histogram to perform estimation, as well as various approaches to reduce the computation cost.

4.1. Histogram construction and maintenance

The spatio-temporal of [12] first partitions the objects based on their spatial location using the conventional MinSkew algorithm, and then decides the VBRs of the buckets during a second step. Since the velocity information is not considered during data partition, the resulting histogram cannot ensure the uniformity of velocity distribution in the buckets. Assume, for example, that we want to build a histogram with two buckets for the data set in Fig. 11a. In Fig. 11b the buckets are decided based on the objects' location. In particular, the first two columns of cells are grouped into the same bucket because they all contain

exactly one point (i.e., no variance), while cells in the last column (with 2 points each) constitute the second bucket. Although the location distribution is fairly uniform, the velocity distribution is rather skewed. Consider the left bucket in Fig. 11b, whose (x_) velocity range is [−10,8] (i.e., decided by the velocities of points a and e). Notice that, there are 5 points with velocities in the range [−10, −2], while only one (i.e., e) has positive velocity (8). Similarly, the velocity range of the right bucket is [−8, 10], but ranges [−8, 0] and [2,10] contain 2 and 4 points.

An effective spatio-temporal histogram should partition data using both location and velocity information. Continuing the previous example, Fig. 11c shows such a histogram, where the left and right buckets contain the first and the last two columns of cells, respectively. Compared with Fig. 11b, the spatial uniformity is slightly worse (only in the right bucket), while the velocity uniformity is significantly better. Specifically, the velocities distribute uniformly in the ranges [−10, −6] and [−8, 10] for the two buckets, respectively. As a result, the new histogram is expected to produce better prediction.

The overall velocity distribution for the data set of Fig. 11 is uniform. If the distribution is skewed, ignoring the velocities during partitioning is even more problematic. Consider, for example, Fig. 12a where object velocities have only two values −10 and 10. Partitioning the spatial universe is useless because (i) the overall location distribution is already fairly uniform (i.e., 2 points in each cell), and (ii) for all possible partitions, the resulting buckets still have extremely skewed velocity distribution. In fact, in this case the best partition should be based entirely on the velocity dimension. Specifically, the first bucket (Fig. 12b) contains all the points with negative velocities while the second one (Fig. 12c) involves those with positive ones. Notice that the resulting buckets have uniform location (one point from each cell) and velocity (all points have the same velocity) distributions.

Motivated by this observation, given a 2D spatial universe, we employ a histogram in the 4D space consisting of two spatial (same as the

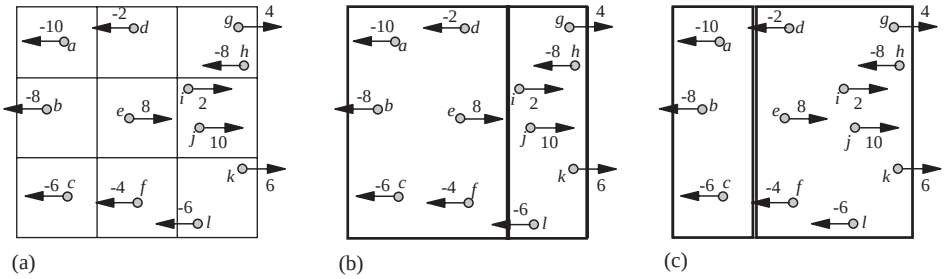

Fig. 11. Uniform velocity distribution. (a) Cell information, (b) considering only location, (c) location and velocity.

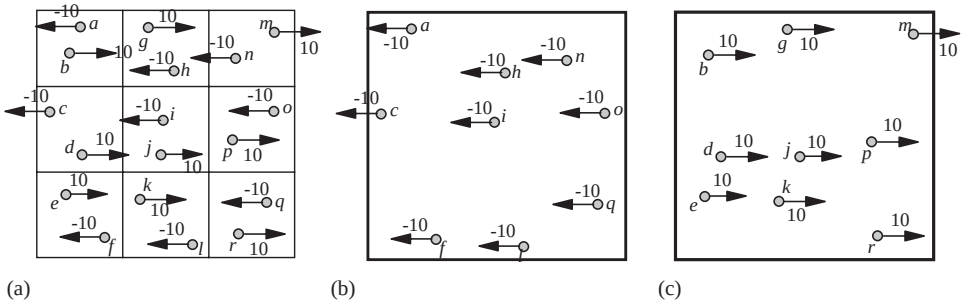

Fig. 12. Skewed velocity distribution. (a) Cell information, (b) bucket 1, (c) bucket 2.

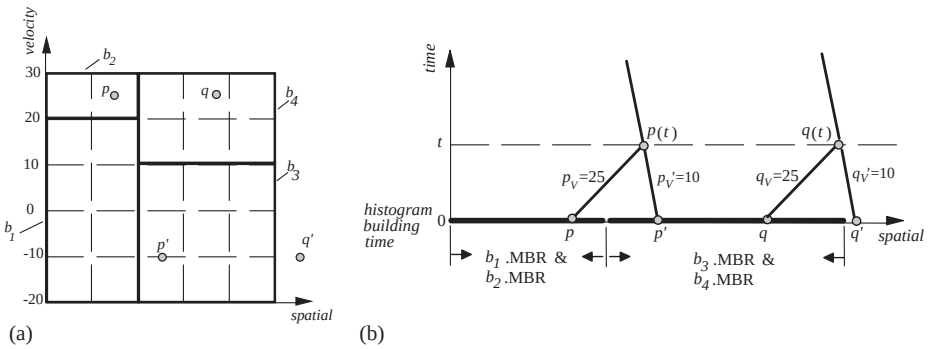

Fig. 13. Updating the histogram. (a) The cells and buckets, (b) point p falls in a different bucket after update.

original universe) and two velocity (decided by the maximum and minimum velocities along the corresponding spatial axis) dimensions. Specifically, a point $(p.x_1, p.x_2)$ with velocities $(p.v_1, p.v_2)$ is converted into a 4D point $(p.x_1, p.x_2, p.v_1, p.v_2)$, and similarly a rectangle r is transformed to a 4D one with the same extents on the spatial and velocity dimensions. The histogram is constructed using the MinSkew algorithm with an initial grid that partitions the space into H^4 regular cells. Each bucket b is associated with b.MBR that encloses the MBRs of all the cells in b, and with b.VBR that tightly bounds the cell VBRs. For point data sets, $b.num$ records the number of 4D points in b. For rectangle sets, $b.num$ is the number of 4D rectangles whose centroids lie in b, and $b.len$ ($b.vlen$) is the average spatial (velocity) extents of these rectangles. The discussion generalizes to arbitrary dimensionality in a straightforward manner.

An important property of the proposed histogram is that it can be incrementally maintained. Fig. 13 illustrates this using point data with one space/velocity dimension (extending to higher dimensionality and rectangles is straightforward). Fig. 13a and b illustrate the bucket extents in the space–velocity and space–time universe, respectively. Particularly, in the space–time universe, a velocity is represented as the slope of a line. Consider point p which falls in bucket b_2, given its current location (as shown in both figures) and velocity (25). Assume that p generates a velocity change (to -10) at future time t (when its location is $p(t)$). To decide the bucket affected by $p(t)$ (in the histogram constructed at time 0), we find a point p' (called the *surrogate point*) at time 0, such that p' will reach the same position $p(t)$ with the updated velocity (-10) of p. As shown in Fig. 13b, p' is the intersection of the spatial axis and the line with slope -10 that crosses $p(t)$. The bucket affected by $p(t)$ is the one (b_3) that contains p' in the space–velocity universe, as shown in Fig. 13a. Since the new bucket b_3 is different from the old one b_2, the histogram must be modified by decreasing $b_2.num$ and increasing $b_3.num$ (by 1). In some cases, the surrogate point may fall outside the universe, in which case the boundary bucket needs to be enlarged. As an example, the MBR of bucket b_3 must be expanded to cover the surrogate point q' (of q) in Fig. 13. It is worth mentioning that, the VBR of the selected bucket for expansion includes the updated velocity of q' (i.e., hence b_4 is not expanded).

An exhaustive scan over all buckets is necessary to identify the ones affected by an update. To avoid this, we build an in-memory[4] TPR-tree on the MBRs and VBRs of the buckets (stored at the leaf level of the tree), so that the affected ones can be identified very quickly by performing a window query using the information of p and p', respectively. Since, as with normal TPR-trees, the non-leaf levels account for a fraction of the total size, the space overhead is very small (less than 15% in our implementation). Similarly, building the tree

[4]As currently there is no version of the TPR-tree specifically designed for main memory, our implementation follows the disk-resident version, although with a different node size ($=10$). This choice of the node capacity is discussed towards the end of the section.

after the histogram has been constructed requires less than 1% of the total construction time. Furthermore, notice that for most updates (except those requiring expansion) the bucket extents are not changed, and thus the TPR-tree only needs to be maintained infrequently.

Updating the histogram incrementally avoids the frequent histogram re-building, and hence reduces the maintenance cost considerably. Whenever the system receives an object update, the new information is intercepted to modify the histogram accordingly. However, the uniformity (in buckets) may gradually deteriorate along with time as the data (location and velocity) distributions vary. When the distribution changes significantly, the histogram eventually needs to be re-built in order to ensure satisfactory estimation accuracy. In order to formulate such a *re-building condition*, notice that, if sufficient distribution changes have accumulated, the histogram must have been modified many times. Therefore, a simple heuristic to ensure satisfactory estimation accuracy is to re-construct the histogram when the number of modifications reaches a certain threshold. As evaluated in the experiments, high prediction precision can be achieved with a very large threshold which leads to only occasional re-building.

4.2. Performing estimation with histograms

Given the histograms H_1 and H_2 for data sets S_1 and S_2, respectively, the expected number of qualifying join pairs can be obtained by summing up the results of bucket pairs from $H_1 \times H_2$. Specifically, for two buckets b_i, b_j ($b_i \in H_1$ and $b_j \in H_2$), the number of result pairs produced from objects inside them is $b_i.num \cdot b_j.num \cdot Sel_{ij}(d, t_q, m)$, where $Sel_{ij}(d,t_q,m)$ is computed using Eqs. (6) (for points), and (7) (for rectangles), by replacing R_i, R_j, V_i, V_j with b_i.MBR, b_j.MBR, b_i.VBR, b_j.VBR, respectively. Thus, the join selectivity can be estimated as

$$Sel(d, t_q, m) = \frac{\sum_{\text{for all buckets } b_i \in H_1 \text{ and } b_j \in H_2} b_i.num \cdot b_j.num \cdot Sel_{ij}(d, t_q, m)}{N_1 \cdot N_2}, \quad (9)$$

where N_1 and N_2 are the cardinalities of S_1 and S_2, respectively.

As explained in the appendix, computing $Sel_{ij}(d,t_q,m)$ requires integral evaluation, and hence should be avoided as much as possible in order to minimize the estimation time. For point data, given two buckets b_i, b_j, such computation is necessary only if the distance between $b_i(t_q)$ (i.e., its extent at query time t_q) and $b_j(t_q)$ is closer than the distance threshold d (in the sequel we call $\langle b_i, b_j \rangle$ a *qualifying bucket pair*). Consider, for example, Fig. 14 which shows the MBRs (at the current time) of three buckets: $b_1 \in H_1$ and b_2, $b'_2 \in H_2$, together with the VBRs. The dashed rectangles represent the extents of the buckets at query time t_q. Note that, computing $Sel_{ij}(d, t_q, m)$ is necessary for $\langle b_1,b_2 \rangle$ ($|b_1(t_q), b_2(t_q)| = 0 < d$), but not for $\langle b_1,b_2' \rangle$ because $|b_1(t_q), b_2'(t_q)| > d$, indicating that no points from b_1 and b_2' can produce a joined pair. Similarly, for rectangular data, we do not consider two buckets if $|b_1(t_q), b_2(t_q)| > d + 1/2(b_1.len + b_2.len)$, where $b_1.len$ corresponds to the side length of the rectangles in b_1 (similar for b_2.len). This is because (i) b_1.MBR covers the centroids of the rectangles counted in b_1, and (ii) $|r_1, r_2| \leqslant d$ if and only if their centroids are closer than $d + 1/2(r_1.len + r_2.len)$ along all dimensions (as explained in Section 3), where $r_i.len$ denotes the side length of r_i.

In order to avoid checking all possible bucket pairs, we take advantage of the main-memory TPR-trees (used also for efficient updating of the histograms). Specifically, we adopt a synchronous traversal algorithm (i.e., *SpatialJoin* [15]) that traverses the TPR-trees of the two histograms

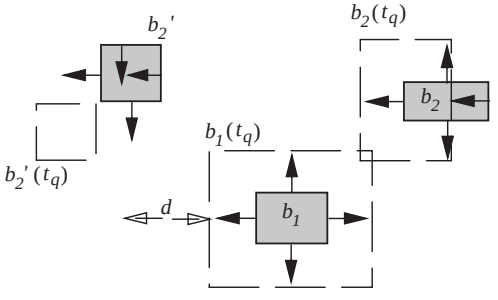

Fig. 14. Buckets that need to be examined.

synchronously, only following pairs of nodes that may lead to qualifying buckets. Compared with the brute-force approach, this algorithm avoids checking pairs whose parent entries do not qualify, thus leading to lower estimation time. Note that the node size influences the estimation time considerably. Specifically, a large size leads to a tree with few levels, in which case the intermediate entries have large MBRs and VBRs. Thus, the synchronous traversal needs to descend almost all pairs of (intermediate) entries, degenerating into the naïve algorithm (that checks all pairs of buckets). On the other hand, a small node size increases the number of tree levels, resulting in longer processing time on intermediate entries. In our experiments, we set the node size to 10 entries per node, which leads to the smallest traversal cost.

5. Experiments

In this section we present an extensive experimentation to prove the effectiveness of the proposed methods. All experiments were performed using a Pentium IV 1 GHz CPU with 256 Mbytes memory. Due to the lack of real spatio-temporal data, we created synthetic data sets by generating velocities for objects in real spatial data sets. Fig. 15 shows the distributions of the selected static data sets (the universe is normalized to [0, 10 000]), where *LB*, *CA* contain point and *LA*, *GER* rectangle objects. From each data set X (= *LB*, *CA*, *LA*, or *GER*), we created two spatio-temporal sets *X_u* and *X_s* where objects' velocities are generated according to uniform and Zipf distributions (with skew coefficient 0.8), respectively.

The accuracy of our model is examined using (i) the histogram in [12] where bucket extents are decided by considering only the spatial dimensions (referred to as 2D in the sequel), and (ii) the 4D histogram presented in Section 4. Both histograms are created using the MinSkew algorithm introduced in Section 2, while the resolution H is set to 50 and 10 for 2D and 4D, respectively (i.e., there are $50^2 = 2500$ (2D) and $10^4 = 10\,000$ (4D), cells in the initial grid before MinSkew starts). As shown in [22], the accuracy of Minskew initially improves as H increases, but actually deteriorates when H grows beyond a certain threshold. Values 50 and 10 (for 2D and 4D) are selected because they lead to the best performance for the corresponding histograms (in particular, 50 is also the value used in [12]). The number of buckets in a histogram is set to 200 (requiring around 8K bytes) in all cases. The 2D (4D) building time is around 0.5 (0.75) seconds after the initial grid is ready. A join query has two parameters: (i) the query timestamp t_q (assuming the histogram is constructed at current time 0), and (ii) a distance threshold d. Given the actual *act* and estimated *est* selectivity, the error rate is computed as $|est-act|/\min(est, act)$.

Fig. 16a shows the estimation error for joining data sets *LB_u* and *CA_u* (i.e., uniform velocity distributions) by fixing the distance threshold d to 250 and varying t_q from 0 (i.e., the current time) to 500 timestamps in the future. Both histograms demonstrate similar behavior and provide precise estimation (with less than 5% error). Note that, since the velocity distribution is uniform, the 4D histogram partitions mainly on the spatial dimensions, and thus behaves similarly to 2D (i.e., ignoring velocities is not important in this case). Fig. 16b illustrates the error rate as a function of d with t_q fixed to 250. Again both histograms have

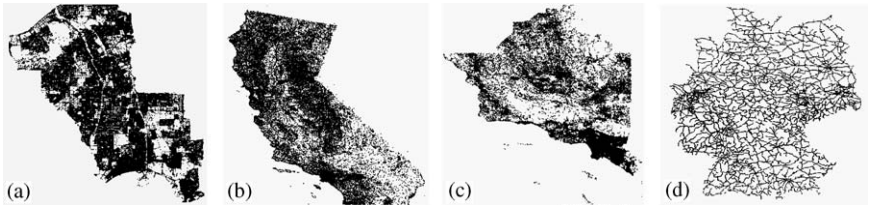

Fig. 15. Location distributions. (a) *LB* (53K points), (b) CA (62K points), (c) *LA* (130K rectangles), (d) *GER* (36K rectangles).

similar accuracy, indicating the correctness of the proposed equations. The error rates are relatively higher (around 10%) when d is small. This is not surprising because for low values of d the output size is small, rendering accurate estimation more difficult.

Fig. 17a and b demonstrate the results of similar experiments using data sets *LB_s* and *CA_s* (skewed velocity distribution). The performance of 2D is very poor in most cases (with up to 60% error) while the proposed 4D approach yields significantly lower error (up to 10%). As shown in Fig. 17a, 2D gives better estimation only when t_q equals 0 (i.e., at the current time), while its precision deteriorates very quickly when t_q increases. This is expected because the selectivity estimation at the current time depends on only objects' current location (i.e., not on velocities); thus, the 2D histogram is more accurate because it achieves better spatial uniformity in the buckets. 4D, on the other hand, outperforms it very quickly at small t_q, confirming the importance of considering velocities in building the histogram. Fig. 17b illustrates the error rate as a function of d (fixing t_q to 250). Similar to Fig. 16b, the error rates remain stable when d is sufficiently large.

Figs. 18 and 19 repeat the same set of experiments for rectangle data, where similar phenomena can be observed (again, 2D is erroneous for skewed velocities while 4D is accurate in all cases). Note that rectangles incur higher error than points because although rectangles have variable sizes, only the average size (in each bucket) is used for estimation.

The next experiment evaluates the efficiency of the cost models for constrained joins. Since the selectivity depends on the concrete position of the constraint window, we measure the average error of a workload consisting of 50 joins, where the constraint area of each join is a square uniformly distributed in the universe. Fig. 20a and b demonstrate the results (obtained from Eq. (8)) for point and rectangle data sets (with skewed velocities), respectively (d and t_q are set to 250). The error rate is plotted as a function of the constraint window size, denoted as the percentage

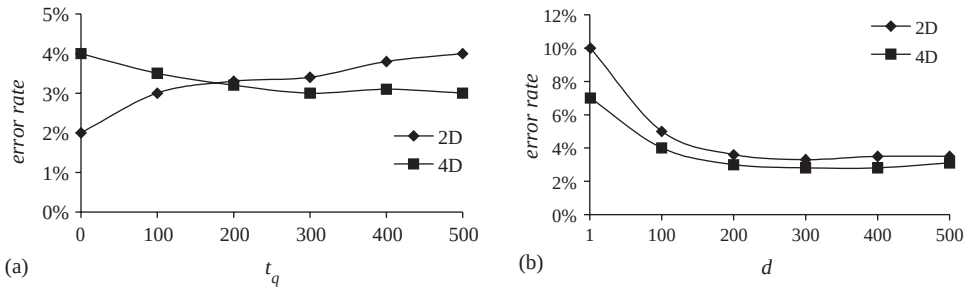

Fig. 16. *LB_u* joins *CA_u* (point data—uniform velocities). (a) Error vs. t_q ($d = 250$), (b) error vs. d ($t_q = 250$).

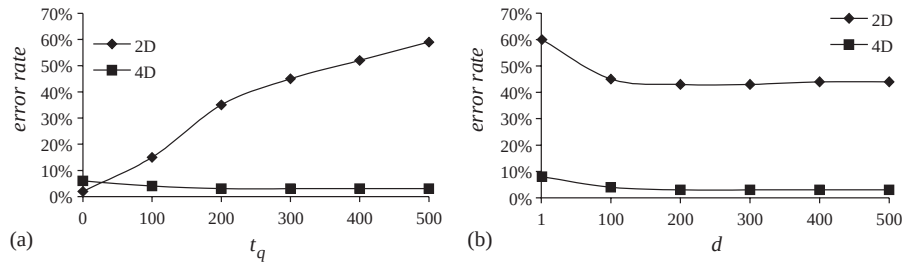

Fig. 17. *LB_s* joins *CA_s* (point data—skewed velocities). (a) Error vs. t_q ($d = 250$), (b) error vs d ($t_q = 250$).

of its side length over the axis length (e.g., a 1% window covers 0.01% of the universe area). It is clear that 4D gives much more accurate prediction than 2D. The error rates are relatively higher (up to 15% for 4D) for small windows since they lead to high selectivity. On the other hand, when the window is sufficiently large, the error rate converges to that of a normal join.

As mentioned in Section 4.1, our histogram can be maintained incrementally to avoid frequent rebuilding. To study the effects of updates we created dynamic data sets as follows. At timestamp 0, the location and velocities of objects are generated as described earlier. Then, at each of the subsequent 1000 timestamp, 10% of the objects are randomly chosen to produce updates. Specifically, the velocity change (along each dimension) is generated randomly in [−10, 10], while the new location of the object is decided based on its previous location and velocity. As discussed in

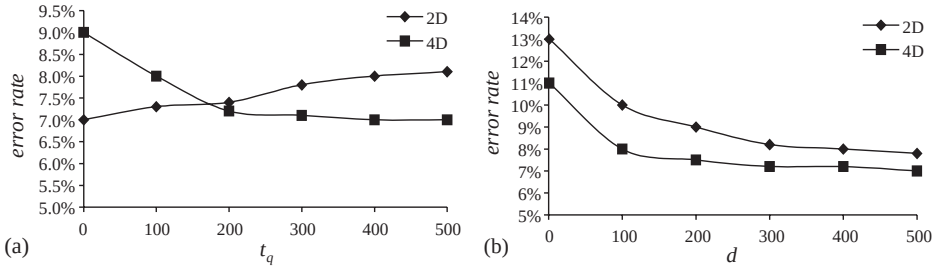

Fig. 18. *LA_u* joins *GER_u* (rectangle data—uniform velocities). (a) Error vs. t_q ($d = 250$), (b) error vs. d ($t_q = 250$).

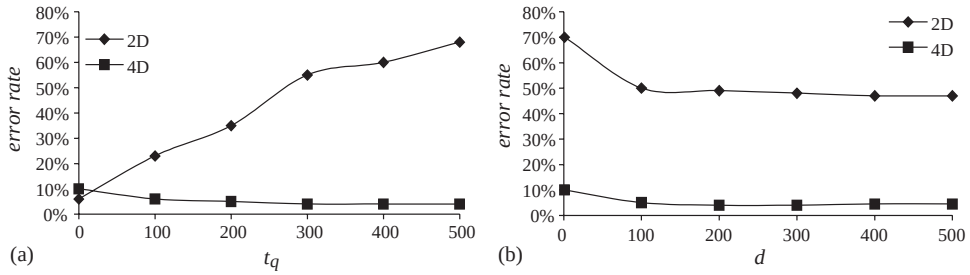

Fig. 19. *LA_s* joins *GER_s* (rectangle data—skewed velocities). (a) Error vs. t_q ($d = 250$), (b) error vs. d ($t_q = 250$).

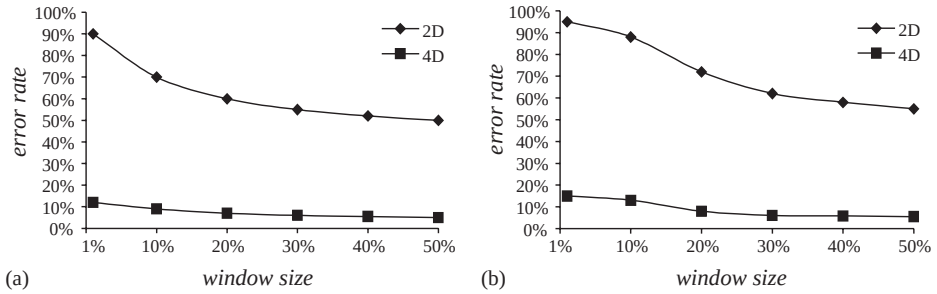

Fig. 20. Error rates for constrained join ($d = 250$, $t_q = 250$). (a) *LB_s-CA_s* (points—skewed velocities), (a) *LA_s-GER_s* (rectangles—skewed velocities).

Section 4.2, we build a main-memory TPR-tree on the histogram of each data set, with a node capacity of 10 entries. The qualifying buckets are identified by synchronously traversing the two trees. The construction time for each tree is around 1% of the histogram building time. Further, the TPR-tree handles object updates efficiently in less than 0.1 ms per update.

Since the generated data sets incur (slow) distribution changes, whereas we do not modify the bucket extents, the uniformity inside each bucket may gradually deteriorate, thus affecting the estimation accuracy. To investigate the degradation rate, we measure the error of joins with the same parameters ($t_q = 250$, $d = 250$) every 100 timestamps (i.e., based on objects' latest information) using the histogram incrementally maintained. Fig. 21 shows the results (of the 4D histogram) for joining *LB_s*, *CA_s* and *LA_s*, *GER_s*. The accuracy decreases slowly with time (error rate up to 15% after 1000 timestamps). This implies that, using incremental maintenance we only need to re-build the histogram very infrequently in order to ensure good estimation accuracy. Note that similar experiments are not applicable to 2D histograms, which require rebuilding at every timestamp. This fact severely limits the applicability of 2D histograms in practice, where frequent updates may invalidate the histogram even before it is constructed.

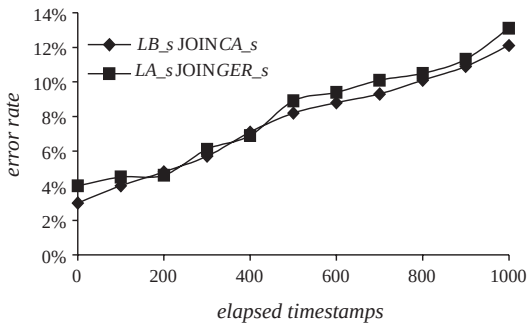

Fig. 21. The accuracy degradation of 4D histogram ($t_q = 250$, $d = 250$).

As discussed in Section 4.2, given two histograms, a brute-force estimation algorithm would examine all pairs of buckets, and hence its computation time would be quadratic to the number of buckets. Instead we utilize the TPR-tree for each histogram and apply a spatial join algorithm to avoid examining unnecessary buckets. Fig. 22 compares the evaluation time of the brute-force and TPR approaches in joining *LB_s* and *CA_s* (results for other data sets are similar). Specifically, Fig. 22a shows the time as a function of t_q, fixing d to 250. The cost of the brute-force algorithm is constant because it checks the same number of bucket pairs for all queries. The cost of TPR, on the other hand, increases with time because, for larger t_q, the entry MBRs (at t_q) of the TPR-trees are larger, leading to more intersecting pairs. Fig. 22b shows similar results by varying d and fixing t_q to 250 (i.e., higher distance thresholds also increase the number of intersecting pairs).

In summary, we have shown that the proposed models accurately capture the selectivity of spatio-temporal joins in a wide range of settings (including constrained joins), yielding average error below 10%. Furthermore, we also confirm the necessity of considering velocity distributions in building spatio-temporal histograms, by demonstrating that the proposed 4D histogram has significantly lower error than the 2D counterpart.

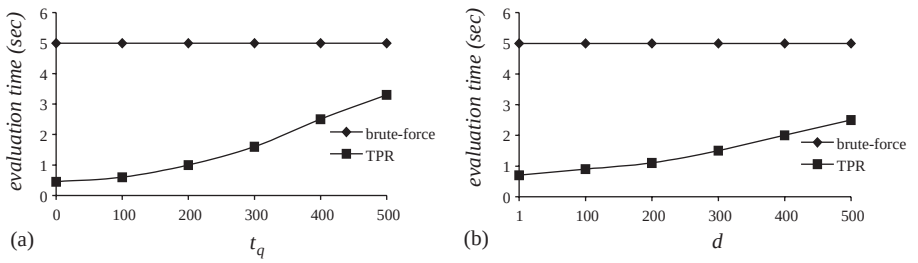

Fig. 22. Evaluation time (*LB_s* join CA_*s*). (a) Evaluation time vs. t_q ($d = 250$), (b) evaluation time vs d ($t_q = 250$).

6. Conclusion

The paper first discusses the mathematical preliminaries for spatio-temporal join selectivity and then solves the problem with a specialized histogram that can be efficiently maintained in main memory. Extensive experiments confirm that the proposed techniques produce accurate prediction with average error less than 10%. Our techniques are based on the assumption that objects move with linear and known velocities, which is common in the literature of spatio-temporal prediction, e.g., [1–4,12]. Applications that satisfy these conditions involve objects (ships, airplanes, weather patterns) moving in unobstructed spaces. The prediction horizon depends on the (velocity) update rate; e.g., 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 [24]. On the other hand, in some situations (e.g., cars on a road network) the motion parameters may be unknown or may change so fast that velocity-based prediction is meaningless. In such cases, selectivity estimation should be based on alternative techniques. For instance, [25] applies *exponential smoothing* using only location information. Although this method does not rely on velocity assumptions, it has to maintain historical data since future prediction is based on the recent past (whereas velocity-based prediction requires only information about the present).

Acknowledgements

This work was supported by grants HKUST 6180/03E, CityU 1163/04E from Hong Kong RGC, and NSF CAREER grant IIS-0133825.

Appendix

This appendix discusses the evaluation of the integrals in Eqs. (3) and (4). For the sake of simplicity, we use the notation $\alpha = (l_1 - l_2 - d)/t_q$ and $\beta = (l_1 - l_2 + d)/t_q$; thus, the integral in Eq. (3) can be written as

$$\int_{V_1.v_-}^{V_1.v_+} \int_{\max(V_2.v_-,u_1+\alpha)}^{\min(V_2.v_+,u_1+\beta)} 1 \, du_2 \, du_1.$$

The value of the integral equals the area of the intersection (the shaded region in Fig. A.1) of (i) the rectangle bounded by lines $\{u_1 = V_1.v_-, u_1 = V_1.v_+, u_2 = V_2.v_-, u_2 = V_1.v_+\}$, and (ii) the region between parallel lines LN_1: $u_2 = u_1 + \alpha$ and LN_2: $u_2 = u_1 + \beta$. As shown in Fig. A.1b and c, the shaded area is the difference of areas Ar_1 and Ar_2, i.e., the intersection of the rectangle in (i) and the lower half planes of lines LN_1 and LN_2, respectively. Obviously Ar_1 depends on the intercept (i.e., α) of LN_1, which decides the relative positions of LN_1 and the rectangle; Table A.1 illustrates its values for the all six possible cases. Due to the symmetry, values of Ar_2 (as well as the corresponding conditions) are the same as those in Table 1, except that α should be replaced with β.

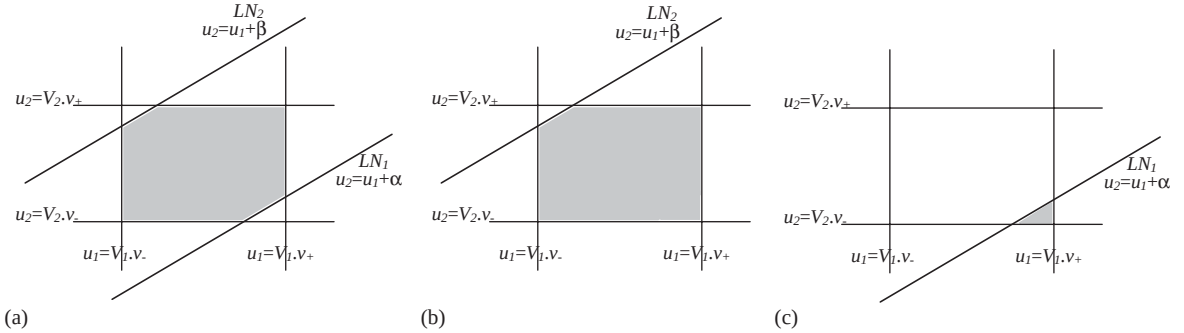

Fig. A.1. Integral area = $Ar_1 - Ar_2$. (a) Integral area, (b) area Ar_1, (c) area Ar_2.

Table A.1
Values of Ar_1 and corresponding conditions

Values of Ar_1	Conditions
$(V_1.v_+ - V_1.v_-)(V_2.v_+ - V_2.v_-)$	$\alpha \in (V_2.v_+ - V_1.v_-, \infty)$
$\frac{1}{2}[(V_1.v_- - V_2.v_- + \alpha) + (V_2.v_+ - V_2.v_-)](V_2.v_+ - V_1.v_- - \alpha) + (V_1.v_+ - V_2.v_+ + \alpha)(V_2.v_+ - V_2.v_-)$	$\alpha \in \left(\begin{array}{l} \max\{V_2.v_- - V_1.v_-, V_2.v_+ - V_1.v_+\}, \\ V_2.v_+ - V_1.v_- \end{array}\right]$
$\frac{1}{2}[(V_1.v_+ - V_2.v_+ + \alpha) + (V_2.v_+ - V_2.v_- + \alpha)](V_2.v_+ - V_2.v_-)$	$\alpha \in (V_2.v_+ - V_1.v_+, V_2.v_- - V_1.v_-]$
$\frac{1}{2}[(V_1.v_- - V_2.v_- + \alpha) + (V_1.v_+ - V_2.v_- + \alpha)](V_1.v_+ - V_1.v_-)$	$\alpha \in (V_2.v_- - V_1.v_-, V_2.v_+ - V_1.v_+]$
$\frac{1}{2}(V_1.v_+ - V_2.v_- + \alpha)(V_1.v_+ - V_2.v_- + \alpha)$	$\alpha \in \left(\begin{array}{l} V_2.v_- - V_1.v_+, \\ \min\{V_2.v_- - V_1.v_-, V_2.v_+ - V_1.v_+\} \end{array}\right]$
0	$\alpha \in (-\infty, V_2.v_- - V_1.v_+]$

Notice that since the integral in Eq. (3) equals $Ar_1 - Ar_2$, we have solved the equation into closed form.

Because the exact solution of Eq. (4) is more complex, in our implementation we evaluate it using the trapezoidal method [26]. Particularly, the numerical approach is applied to the two outer integral layers (recall that Eq. (4) has four layers), while the inner two (i.e., corresponding to Eq. (3)) can be solved accurately as described earlier.

References

[1] G. Kollios, D. Gunopulos, V. Tsotras, On indexing mobile objects, ACM PODS, 1999.
[2] P.K. Agarwal, L. Arge, J. Erickson, Indexing moving points, ACM PODS, 2000.
[3] S. Saltenis, C.S. Jensen, S.T. Leutenegger, M.A. Lopez, Indexing the positions of continuously moving objects, ACM SIGMOD, 2000.
[4] S. Saltenis, C.S. Jensen, Indexing of moving objects for location-based services, IEEE ICDE, 2002.
[5] N. Mamoulis, D. Papadias, Multiway spatial joins, ACM TODS 26 (4) (2001) 424–475.
[6] D. Papadias, Y. Tao, P. Kalnis, J. Zhang, Indexing spatio-temporal data warehouses, IEEE ICDE, 2002.
[7] J. Gehrke, F. Korn, D. Srivastava, On computing correlated aggregates over continual data streams, ACM SIGMOD, 2001.
[8] A. Dobra, M. Garofalakis, J. Gehrke, R. Rastogi, Processing complex aggregate queries over data streams, ACM SIGMOD, 2002.
[9] N. An, Z. Yang, Sivasubramaniam, A. Selectivity estimation for spatial joins, IEEE ICDE, 2001.
[10] C. Faloutsos, B. Seeger, A. Traina, C. Traina Jr., Spatial join selectivity using power laws, ACM SIGMOD, 2000.
[11] C. Sun, D. Agrawal, A. Abbadi, Selectivity estimation for spatial joins with geometric selections, EDBT, 2002.
[12] Y. Choi, C. Chung, Selectivity estimation for spatio-temporal queries to moving objects, ACM SIGMOD, 2002.
[13] A. Guttman, R-trees: a dynamic index structure for spatial searching, ACM SIGMOD, 1984.
[14] N. Beckmann, H. Kriegel, R. Schneider, B. Seeger, The R*-tree: an efficient and robust access method for points and rectangles, ACM SIGMOD, 1990.
[15] T. Brinkhoff, H. Kriegel, B. Seeger, Efficient processing of spatial joins using R-trees, ACM SIGMOD, 1993.
[16] Y.-W. Huang, N. Jing, E. Rundensteiner, Spatial joins using R-trees: breadth first traversal with global optimizations, VLDB, 1997.
[17] N. Koudas, K. Sevcik., High dimensional similarity joins: algorithms and performance evaluation, IEEE ICDE, 1998.

[18] W. Aref, H. Samet, A cost model for query optimization using R-trees, ACM GIS, 1994.
[19] I. Kamel, C. Faloutsos, On packing R-trees, CIKM, 1993.
[20] B.U. Pagel, H.W. Six, H. Toben, P. Widmayer, Towards an analysis of range query performance in spatial data structures, ACM PODS, 1993.
[21] Y. Theodoridis, E. Stefanakis, T. Sellis, Cost models for join queries in spatial databases, IEEE ICDE, 1998.
[22] S. Acharya, V. Poosala, S. Ramaswamy, Selectivity estimation in spatial databases, ACM SIGMOD, 1999.
[23] S. Muthukrishnan, V. Poosala, T. Suel, On rectangular partitionings in two dimensions: algorithms, complexity, and applications, ICDT, 1999.
[24] Y. Tao, J. Sun, D. Papadias, Analysis of predictive spatio-temporal queries, ACM TODS 28 (4) (2003) 295–336.
[25] J. Sun, D. Papadias, Y. Tao, B. Liu, Querying about the past, the present, and the future in spatio-temporal databases, IEEE ICDE, 2004.
[26] W. Press, B. Flannery, S. Teukolsky, W. Vetterling, Numerical Recipes in C++, second ed., Cambridge University Press, Cambridge, 2002.