Max-Count Aggregation Estimation for Moving Points[*]

Yi Chen Peter Revesz
Dept. of Computer Science and Engineering,
University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Abstract

Many interesting problems regarding moving objects can be reduced to the following question: Given a set S of moving points on a line and two other movings points A and B on the same line, what is the maximum number of points in S that will be simultaneously between A and B within a time interval (t_1, t_2)? We propose an algorithm that can estimate the answer for arbitrary A and B and any fixed S in a chosen constant time. We show that the error rate of the estimation is related to this chosen constant and some other parameters of the input data. Our experimental results show that high accuracy estimation can be achieved when S has a large number of points and A and B are not too close to each other.

1 Introduction

Spatio-temporal databases are increasingly important in various areas including e-commerce, meteorology, telecommunications, and transportation. In querying such spatio-temporal databases, the common aggregation operations of *Count* and *Max* occur frequently. In addition to these, Revesz and Chen [3] recently introduced a third aggregate operator called *Max-Count*, which arises only in the case of moving points and has no analogue in relational databases. The three aggregate operators mentioned can be defined as follows:

Let S be a set of N moving points in a database, and let Q be a query rectangle Q.

Count: For a given time, count the number of points of S that are in Q.

Max: For a given time and a value for each point of S, find the maximum of the values of the points of S that are in Q.

Max-Count: Find the maximum number of points of S that are simultaneously in Q within a time interval (t_1, t_2). (Optional: Return also the earliest time when the maximum is reached.)

In one-dimensional space, instead of a query rectangle, we talk about a *query interval*, whose endpoints are called *query points*.

Acharya et al. [1] gave an algorithm that can estimate the *Count* of the rectangles in the database that intersect a new query rectangle. Choi and Chung [2] and Tao et al. [4] proposed methods that can estimate the *Count* of the moving points in the plane that intersect a new query rectangle. Hence while estimation in the case of *Count* is an old idea, its consideration in the case of *Max-Count* is new.

In Section 2, we give an algorithm that can estimate the *Max-Count* aggregate operator on spatio-temporal databases that represent a set of one-dimensional and linearly moving points. In Section 3, we present experimental results that show that our estimation algorithm provides accurate estimation over various queries.

2 Max-Count Aggregation Estimation

We first discuss the special case when the set of moving points in one dimensional space has uniform distribution of initial position (at time $t = 0$) and the velocity. (This special case is corresponds to the case of one bucket of a histogram in Definition 2.2. We generalize later this case to an arbitrary number of buckets.)

Let S be a set of N moving points in one dimensional space. The position of a point $P_i \in S$ at time t can be represented by a linear function of time $P_i(t) = a_i t + b_i$. In the dual plane, this point can be represented as a static point with the coordinate (a_i, b_i). Suppose that the N points represented in the dual plane are distributed uniformly in a rectangular area R as shown in Figure 1.

Definition 2.1 The *dual space* of the one-dimensional moving point set is a two dimensional space in which

[*]This research was supported in part by USA NSF grant EIA-0091530. Authors' email:{ychen, revesz}@cse.unl.edu.

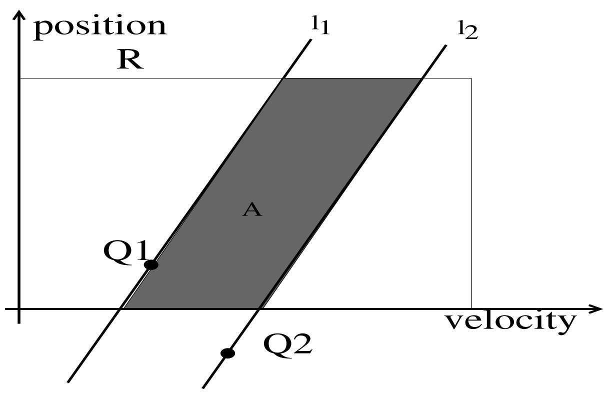

Figure 1. Estimation idea assuming uniformly distributed point sets.

the x and y-coordinates denote the speed and the initial position, respectively, of the moving points.

Definition 2.2 The spatio-temporal *histogram* consists of a finite partitioning into a set of rectangular areas, called *buckets*, the two dimensional dual space of the one-dimensional moving point set. Each bucket is described by its corner vertices and the total number of points in it.

Definition 2.3 Given two moving query points let Q_1 and Q_2 be their duals and let lines l_1 and l_2 cross them, respectively, with slopes $-t$ as shown in Figure 1. Then the *query band* is the area between the lines l_1 and l_2.

In the above definition, the slope of the lines change with the variable t.

Lemma 2.1 Let Q_1 and l_1 be as in Definition 2.3. Then at any time t, the moving points whose duals lie below (or above) l_1 in the dual plane are exactly those that are to the left (or right) of Q_1 is the original one-dimensional line.

Lemma 2.2 Let S be a set of N moving points which are all mapped with a uniform distribution within a rectangular area R in the dual plane as shown in Figure 1. Then the number of points in S that lie between $Q_2(t)$ and $Q_1(t)$ at time t can be estimated to be $N \cdot A/R$, where A is the intersection of the rectangular area R and the query band.

Hence if we can calculate the area of the intersection, we can efficiently calculate the estimated aggregation result. Let A_1 be the area in the rectangle R below l_1. Similarly, let A_2 be the area in the rectangle R below l_2. If l_1 is above l_2, then area of the intersection A can be represented as $A = A_1 - A_2$. If l_1 is below l_2, then we have $A = A_2 - A_1$.

It is also clear that A_1 and A_2 can be calculated in a constant time. For example, given a time instance t, we have (i) if l_1 is above the rectangular area, then $A_1 = R$; (ii) if l_1 is below the rectangular area, then $A_1 = 0$; (iii) if l_1 intersects the rectangular area, we have the following cases as shown in Figure 2:

1. Only the upper-right vertex is above l_1.
2. Only the upper-left vertex is above l_1.
3. Upper-left and upper-right vertexes are above l_1.
4. Upper-left and lower-left vertexes are above l_1.
5. Upper-right and lower-right vertexes are above l_1.
6. Only lower-left vertex is below l_1.
7. Only the lower-right vertex is below l_1.

Lemma 2.2 and the above imply that we need a constant number of calculations to find the *Count* aggregate, because we need to consider only one value of t, hence the slopes of l_1 and l_2 are fixed. When we pose *Max-Count* aggregates on moving points, the situation is more complex, because we have to consider all possible t values in the time interval (t_1, t_2), meaning that the slopes of l_1 and l_2 vary. This looks a daunting task. However, in the following, we show that we still need only a constant number of calculations to find *Max-Count* aggregates.

Lemma 2.3 Let S be a set of moving points in one dimensional space, such that in the dual plane they are uniformly distributed in a rectangular area R. Let Q_1 and Q_2 be two moving points. Given a query time interval (t_1, t_2), we can estimate the *Max-Count* aggregation by a constant number of calculations.

Lemma 2.4 Let R be a rectangle and l a line. Then, the area in R that is below (or above) l can always be represented by a function of the form $A = a \cdot t + \frac{b}{t} + c$, where a, b and c are constants.

Now we prove that the area that is the intersection of R and the query band can also be represented by a similar function of time.

Lemma 2.5 Let S be a set of moving points in one dimensional space, and let $Q_1(t)$ and $Q_2(t)$ be two moving query points. Then for any given histogram H of S, the estimated number of points that are located between the two query points at time t can be represented by a function of the form

$$count(t) = a \cdot t + \frac{b}{t} + c$$

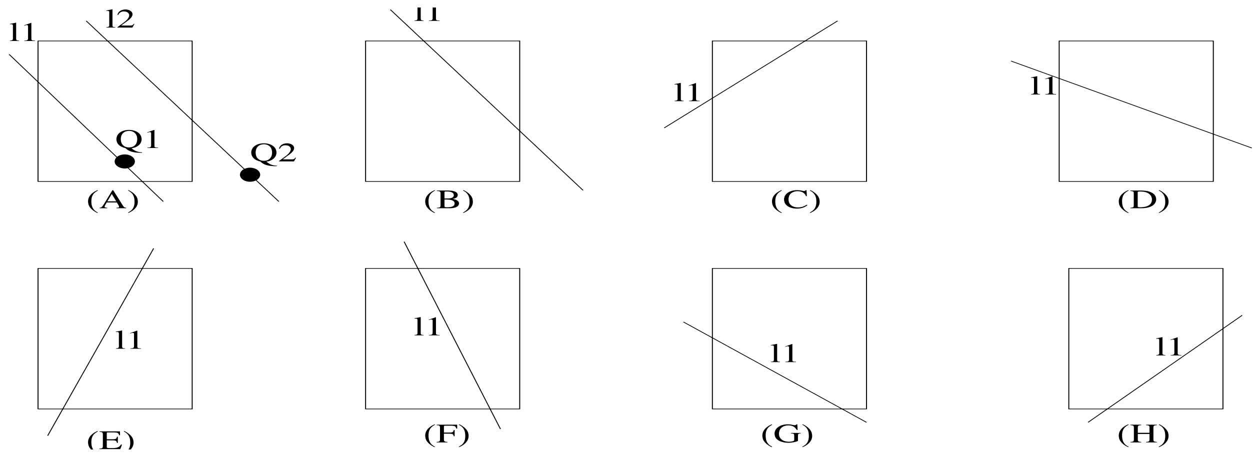

Figure 2. Cases with one bucket and one line.

where a, b and c are constants and $t \neq 0$. When $t = 0$, then $count(t) = d$ where d is a constant.

Lemma 2.6 Suppose the dual plane is partitioned into rectangular buckets. We can calculate the *Max-Count* of a query band during a query time interval when the query band covers the same set of corner points of the buckets.

Definition 2.4 Let H be a histogram. Let $Q_1(t)$ and $Q_2(t)$ be two query points. Let $(t^{[}, t^{]})$ be the query time interval. We define the *Time Partition Order* to be the set of time instances $TP = \{t_1, t_2, ..., t_i, ..., t_k\}$, such that k is a constant and $t_1 = t^{[}$ and $t_k = t^{]}$ and for each time interval $[t_i, t_{i+1})$ the set of bucket corner vertices that lie within the query band remains the same.

Note that a query band changes with t as the slope of the lines l_1 and l_2 changes. For the query band to remain in one of the states shown in Figure 2 during a time interval $[t_i, t_{i+1})$, it cannot change so much that it either leaves a corner vertex or adds a new corner vertex of a bucket.

Therefore, throughout the time interval $[t_i, t_{i+1})$ the number of points within the query band can be estimated by the same function of the form $a \cdot t + \frac{b}{t} + c$, where a,b and c are constants.

All the above lemmas and observations lead to the following algorithm to estimate the *Max-Count* value.

Algorithm 2.1 *Max-Count Algorithm*
Input: A histogram H, query points $Q_1(t)$ and $Q_2(t)$ and a query time interval $(t^{[}, t^{]})$.
Output: The estimated *Max-Count* value.

1. Find all bucket corner vertices in H. Find the lines between the corner vertices and the dual of the query points. Order the lines by their slopes. Find the Time Partition Order of the time interval $(t^{[}, t^{]})$.
2. For each time interval associated with the Time Partition Order calculate the function of time having the form $a \cdot t + \frac{b}{t} + c$, where a, b and c are constants.
3. For each such function of time, calculate the maximum value within the corresponding time interval. Store the result in a list.
4. The maximum value in the list is the final result.

Theorem 2.1 Let H be a histogram with B number of buckets. Let $Q_1(t)$ and $Q_2(t)$ be two moving query points, and let $(t^{[}, t^{]})$ be a time interval. It takes $O(B \log B)$ time to calculate the estimated *Max-Count* value.

Example 2.1 We show in Figure 3 a histogram which contains three buckets and in which P and Q are the duals of the two moving query points. There are a total of eight corner vertices for the buckets in the histogram, as shown in the figure. Figure 3(t_i) shows the query band at time t_i. The query band consists of two parallel lines which have the slope $-t_i$. The line crossing Q also crosses G. This means that at time t_i, F lies in the query band, and G enters the query band. We sweep the query band clockwise as time increases and slope decreases. Then we find that at a later time t_{i+1}, G still lies in the query band, but F is leaving the query band, as shown in Figure 3(t_{i+1}).

During the time interval $[t_i, t_{i+1})$, the query band intersects with all three buckets. Moreover, the intersection between each bucket and the query band remains in one of the states shown in Figure 2. For example, for the intersection of the query band and bucket 1 remains

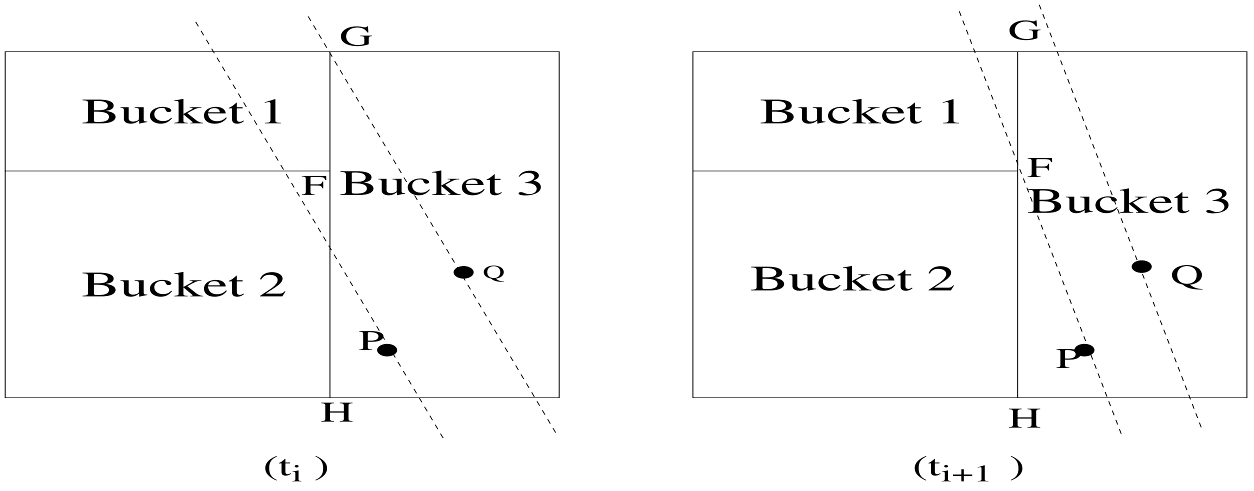

Figure 3. The query band at two different times.

in the case shown in Figure 2(5). Hence, the area of the intersection between the query band and bucket 1 can be represented by the same function of time. According to Lemmas 2.2 and 2.4, the number of points can be estimated by a function of time $f_1 = a_1 \cdot t + \frac{b_1}{t} + c_1$. Similarly, the number of points in the intersection of the query band and buckets 2 and 3 can be estimated by functions $f_2 = a_2 \cdot t + \frac{b_2}{t} + c_2$ and $f_3 = a_3 \cdot t + \frac{b_3}{t} + c_3$. Then, the total number of points during the time interval $[t_i, t_{i+1})$ can be estimated by the function of time $f = f_1+f_2+f_3 = (a_1+a_2+a_3)\cdot t+\frac{b_1+b_2+b_3}{t}+(c_1+c_2+c_3)$. Observe that the *Max-Count* value during this time interval can be calculated with constant time. Since the *Time Partition Order* forms $O(B)$ number of such time intervals, it takes $O(B)$ time to calculate the *Max-Count* of all such intervals and the final result.

3 Experiments

We study the impact of various parameters for the performance of the algorithm. We systematically generate several synthetic datasets that consist of a large number of one-dimensional moving points. Both the initial positions and the speeds of these points are distributed between 0 and $10,000$ according to the Zipf distribution. In the Zipf distribution we assumed that the higher speed and higher displacement points were denser. This is similar to the dataset used in [2, 4]. Therefore, in the dual plane the dataset was distributed within a rectangular area with height $10,000$ and width $10,000$ with a greater density of points in the upper and right regions of the histogram.

3.1 Experimental Parameters

We consider the estimation accuracy with respect to the following parameters:

Number of Buckets: We used the histogram algorithm of [1] that allowed us to specify the number of buckets as an input. We used either 10 or 20 buckets in our experiments.

Number of Points: This is the number of points in the histogram. Since we used the same Zipf distribution in all of our experiments, the higher number of points also mean a higher density of the points. We varied the number of points from 8000 to 40000.

Query Range: This is the distance between the *duals* of the two moving query points. We varied the query range from 400 to 2000, that is, from 2% to 20% of the width of the histogram.

Query Type: The position of the dual of the two moving query points can be either in a dense region or a sparse region of the histogram. We used one dense and one sparse query in the experiments.

Originally we did not consider the query type as a parameter. However, we added it when we realized that it is actually an important parameter. Presenting only an average running time of a set of different queries would actually hide an interesting and non-obvious relationship.

3.2 Dense Queries

In our first set of experiments we considered queries where the location of the duals of the moving query points was in a dense region of the histogram. We call these cases *dense queries*.

3.2.1 10 Buckets

We fixed the number of buckets to be 10 and varied the number of points and the query range.

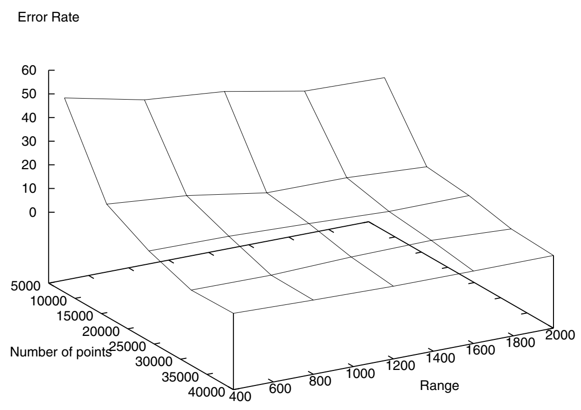

Figure 4. Performance measures for a dense query and 10 buckets.

Figure 4 shows that the *error rate* (the absolute value of the difference between the estimated and the actual values divided by the actual value) decreases exponentially as the number of points increase. The error rate also decreases slightly as the query range increases.

Discussion: These findings were as expected. Obviously, as the number of points increases, the points are more clearly Zipf distributed. With a clearer Zipf distribution in the entire plane, the bucketing algorithm can find buckets in which the points are more uniformly distributed than before, because it has to consider only the Zipf factor and less random factors. Hence the accuracy increases.

The query range data is harder to explain. Intuitively, in general the higher the intersection area between a bucket and the query band the less is the error rate. When the query range is wider the intersection areas between the buckets and the query band tends to be greater, in fact, the query band may completely cover many buckets. For those buckets that are completely covered the estimation would be accurate.

3.2.2 20 Buckets

Figure 5 shows that the error rate decreases exponentially as the number of points increase. The error rate also decreases slightly as the query range increases. This results are similar to the results in Section 3.2.1, with a slightly lower error rate here than in the previous section for most combinations of number of points and query ranges.

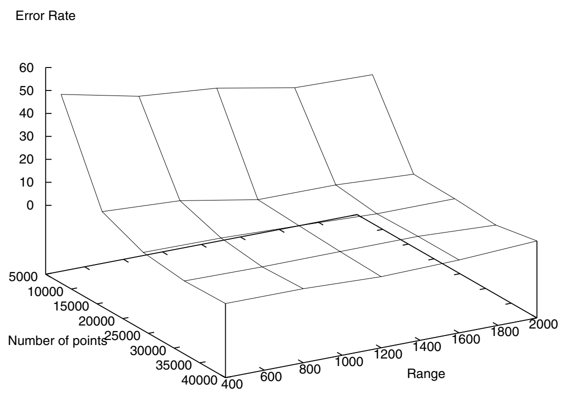

Figure 5. Performance measures for a dense query and 20 buckets.

Discussion: Intuitively, when we allow more buckets in the histogram, the distribution in each bucket is more uniform, hence the total error rate should be lower. However, there is no visible decrease of the error rate when the number of buckets increases from 10 to 20. Apparently most of the extra buckets do not intersect with the query band, hence increasing the number of buckets does not significantly lower the error rate.

For dense queries, with either 10 or 20 buckets, a slight change in time could result in a large change in the estimate of the number of points in the query band. This explains why the error rate can be high (up to 50%) in the case of a relatively few number of points but remains low in the case of a high number of points. Apparently the estimate and the actual values change more in tandem with the higher density.

3.3 Sparse Queries

By *sparse queries* we mean queries that are the opposite of dense queries. In sparse queries the duals of the moving query points are located in a sparse region of the histogram.

3.3.1 10 Buckets

Figure 6 shows the relationship of the error rate, number of points and query range when the number of buckets

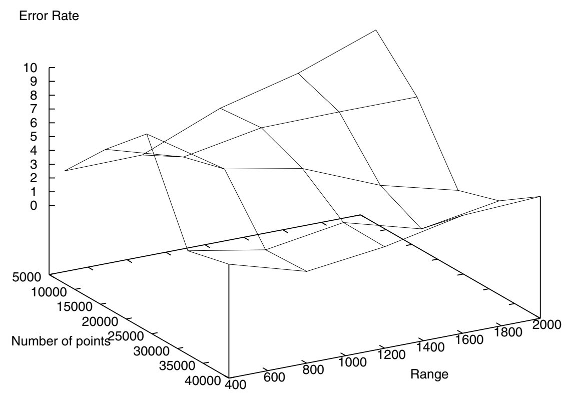

Figure 6. Performance measures for a sparse query and 10 buckets.

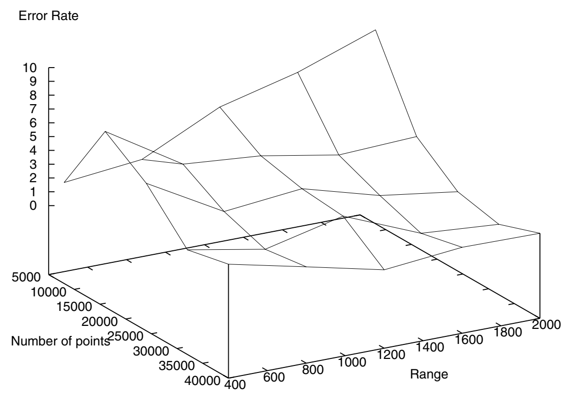

Figure 7. Performance measures for a sparse query and 20 buckets.

is 10 and we have sparse queries.

The error rate is always relatively small, that is, it is always below 10%. There is no clear relationship between the error rate and query range. In fact, the error rate decreases about linearly when the number of points is 24,000, but it increases linearly when the number of points is 8,000 and 40,000. Similarly, there is no clear relationship between the error rate and the number of points. For example, the error rate goes up and down for query range 400 and down and up for query range $2,000$.

Discussion: The lack of a clear relationship between the error rate and the query rate in this case may be due simply to the fact that the error rate remains lower than 7% in most cases. With such a relatively small error rate the ups and downs in Figure 6 cannot be statistically significant.

3.3.2 20 Buckets

Figure 7 shows that the error rate is again very small in most cases when we use sparse queries and fix the number of buckets to be 20. The highest error rate occurs in one corner of the picture when the number of points is $8,000$ and the query range is $2,000$. There seems to be a decrease in the error rate as we go away from that corner in any direction, either decreasing the query range or increasing the number of points.

Discussion: In many ways these results were similar to those in Section 3.3.1. The most surprising result again was that the error rate was small, always less than 10%. It was also noteworthy that in the case of sparse queries the average error rate seems to be slightly lower with 20 buckets than with 10 buckets.

For sparse queries, with either 10 or 20 buckets, a slight change in time results only in a small change in the estimate of the number of points in the query band. This explains why the error rate is always small even when we have relatively few number of points.

Our experiments show that the query type is an important, perhaps the most important, parameter in the performance of the *Max-Count* estimation algorithm. That is surprising, because it is a less obvious variable than the others. However, even in the case of dense queries a good performance can be guaranteed if the number of points is high, the query range is not too small, and the number of buckets is 10 or higher.

References

[1] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation in spatial databases. In *Proc. ACM SIGMOD*, pages 13–24, 1999.

[2] Y.-J. Choi and C.-W. Chung. Selectivity estimation for spatio-temporal queries to moving objects. In *Proc. ACM SIGMOD*, 2002.

[3] P. Revesz and Y. Chen. Efficient aggregation on moving objects. In *TIME-ICTL*, 2003.

[4] Y. Tao, J. Sun, and D. Papadias. Selectivity estimation for predictive spatio-temporal queries. In *ICDE*, 2003.