Efficient Traffic Crash and Snow Complaint GIS System

Anthony Ngo
Lincoln Public Works Department
901 W Bond Street, Suite 100
Lincoln, NE 68521, USA
ango@lincoln.ne.gov

Peter Revesz
Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588, USA
revesz@cse.unl.edu

ABSTRACT

We describe the design and implementation of a traffic crash and snow complaint GIS system developed for the Lincoln Public Works department. We also describe a novel geocoding algorithm that was used to move data from the older Criminal Justice Information System, which is a relational database, to the new GIS system. In addition, we describe the implementation of several indexing algorithms that enable the system to efficiently answer rectangular range queries and queries about the relative locations of moving objects.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

I.3.5 [**Computational Geometry and Object Modeling**]: Geometric algorithms, languages, and systems

General Terms

Design, Performance, Experimentation

Keywords

Dominance, Geocoding, Indexing, Traffic Crash, Snow Removal

1. INTRODUCTION

An increasing number of city governments provide GIS-based public services. These systems give city officials and the public access to real and accurate data and planning tools which were not previously available [7]. In this paper, we describe our design and development of the Lincoln Public Works (LPW) department's traffic crash data and snow complaint GIS system.

Motivation for traffic crash database: The Lincoln Public Works and the Lincoln Police Department generated and maintained for many years the traffic crash reports and associated statistics using the Criminal Justice Information System (CJIS), which is a relational database system. In 2008, the CJIS system recorded 7,890 crashes with a total monetary loss estimated at nearly 200 million dollars including wage loss, medical expenses, property damage, and insurance administrative costs [6].

On October 2009, LPW decided to convert the CJIS data to a new GIS-based system for mapping and analyzing traffic crashes in order to conduct a detailed crash study throughout the city of Lincoln. The primary goal of this project, called *CJIS to GIS*, was to provide a high performance GIS-based tool for identifying frequent traffic crash locations and provide realistic solutions which aid in reducing the total number of traffic crashes and their monetary impact [6]. Identifying frequent or otherwise problematic traffic crash locations will lead to a better prioritization and planning of transportation improvements and to developing more effective countermeasures that address the identified crash patterns at all locations having frequent traffic crashes.

Motivation for snow complaint database: According to a New York Times article [14] about New York City government and a large snow storm:

> "*The blizzard prompted a political crisis that became legendary in the annals of municipal politics, nearly brought down the administration of Mayor John V. Lindsay and offered an instructive lesson to elected officials in the politics of snow removal.*"

To avoid such a political crisis and other damages, cities need to be prepared for winter snow storms and have a plan to keep the streets cleared and safe. To manage snow removal efforts, cities need GIS-based systems that enable real-time visualization of snow-related complaints, such as "street is icy", "parking ban info", "snow push into street", "property damage", "plowing wrong side street," etc.

We combine crash reports data and snow complaints data in the same database because they are closely related. According to city statistics, January is a high-fatality month with many traffic crashes due to snow-related issues. A common GIS-based system for managing traffic crash and snow removal data can help visualize and analyze this relationship more deeply. That would help improve street improvement operations, which turn out which would reduce the number of snow-related traffic crashes.

Outline: This paper is organized as follows. Section 2 describes related work. Section 3 presents a novel and efficient geocoding algorithm that converts CJIS traffic crash data into GIS data. Section 4 describes an implementation of rectangular range queries. Section 5 gives an overview of the snow complaints database. Section 6 describes the implementation of an efficient moving points estimation algorithm. Finally, Section 7 suggests some future work.

2. RELATED WORK

Some recent research considered mapping only intersection crashes using GIS technology. [9] describes the application of GIS to map intersection crashes at only 20 intersections for 24 months of study. [10] discusses an efficient tool for transportation safety engineers and policymakers to analyze the traffic crash data typically obtained from police reports without using GIS technology. [2] reports their development of a GIS-based traffic network analysis system, which provides a graphical analysis platform to transportation planners and researchers for transportation network analysis. However, almost all of these known public researches discussed to date fall into one of three categories. Either they are just a non-GIS crash data analysis tool, or they are a GIS crash data analysis tool, or they are so simple that they are not useful in solving many traffic crash problems that involve geographic visualization and querying.

Other existing traffic crash databases such as [4], [15], and [13] use traditional geocoding, which requires postal addresses. However, we cannot apply for our application because LPD never has recorded any postal address. On the other hand, [19] introduces a new type of geocoding task as segment-type geocoding in contrast with classic geocoding that takes a street address or intersection. This approach is so specific that it is useful in solving only one or two problems. Moreover, this approach is dependent on different types of street intersections (one node or multiple nodes). In this paper we describe a *not-too-general* or *not-too-specific* paradigm that can be precisely specified and can be used to solve many traffic crash related problems.

Our traffic crash and snow removal system has unique indexing features. As a result, our system is able to answer efficiently many queries of the type "find the number of (traffic crash or snow complaint) events on the map on a particular street segment." Efficient answering of these types of queries is based on the implementation of the main indexing algorithm in [11], which in turn is a based on Bentley's ECDF-trees [3].

ESRI's ArcGIS system provides some visualization tools for spatial analysis. However, those tools are limited to small amount of data because of inefficient processing methods implemented within ArcGIS. For example, the ArcGIS query task default parameter *maxresultsize* is normally set to 500 (version 9.x) and 1000 (version 10.x). Our system avoids this limitation and could handle much larger data sets by using the indexing algorithm in [10, 11].

3. THE GEOCODING PROBLEM

Geocoding is the problem of converting non-geographical information, such as street address, into valid geographical information that GIS systems or constraint database systems [8] can use, typically (x, y) coordinates.

For example, the CJIS relational database records *traffic crash report data* in a form which needs to be converted to a GIS form. Since CJIS is a very large database, manual conversion of the data is prohibitively expensive. It may take between 15 and 30 minutes for a GIS specialist to manually add one *traffic crash report* into an ESRI ArcMap/ArcInfo shapefile. Since CJIS has over 100 thousand traffic crash report data, it may take between 25 and 50 thousand hours to manually convert each traffic crash report data to a GIS data. Hence we developed an efficient conversion algorithm that automatically converts the old data into the GIS representation.

Now let us analyze the problem of locating a traffic crash on a GIS map. There are two cases, either the crash happens at an intersection or at a non-intersection location, which is at the middle of a segment of a street. The first case is easy and can be converted by an ESRI ArcGIS geocoding tool. The second case is not solved by the ArcGIS tool. Hence we focus on the latter case. In contrast to the segment-type geocoding method in [19], our geocoding method is independent of the different types of street intersections because we use the street center line GIS map shown in Figure 1, which yields only a single intersection node for streets that cross each other.

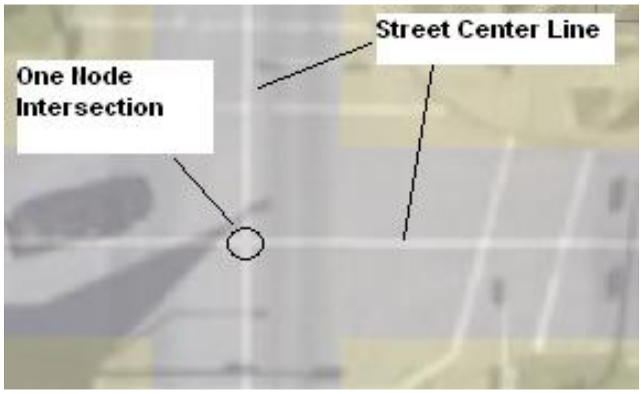


Figure 1: Street center line and one-node intersection.

Each CJIS traffic crash report is purely relational and does not include any information about location coordinates (latitude and longitude) or street address used in traditional geocoding [16]. Instead, as shown in Table 1, the CJIS database gives the names of *OnStreet*, *AtStreet*, and *BtStreet*.

The particular geocoding problem that we considered is to find the GIS street segment, called *midBlockStreet*, on *OnStreet*, where the traffic crash happened. Before presenting a solution, let us first recall the following important definitions regarding point dominance [12]:

Definition 1. Point $Q(x_Q,y_Q)$ dominates point $C(x_C,y_C)$, written as $Q > C$, if and only if $x_Q > x_C$ and $y_Q > y_C$

Definition 2. A block street dominates point $C(x_C,y_C)$, written as blockStreet > C, if and only if for every point $Q(x_Q,y_Q)$ on blockStreet $x_Q > x_C$ and $y_Q > y_C$

Table 1. Example CJIS traffic crash street information

ONST	ATST	BTST
P ST	N 11TH ST	N 12TH ST
N 27TH ST	Vine ST	T ST

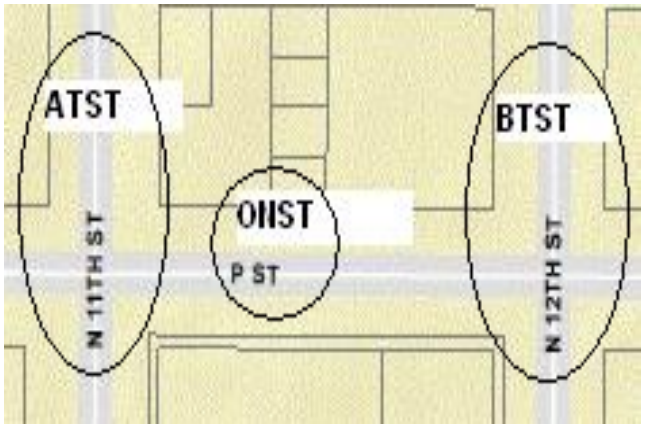


Figure 2: Searching street segment on ONST between ATST and BTST.

Now we can give the following solution:

Let I_1 be the intersection point at ONST and ATST.

Let I_2 be the intersection point at ONST and BTST.

Find *midBlockStreet* on ONST between ATST and BTST.

Without loss of generality, we assume that $I_1 < I_2$. Then the following must hold:

$$I_1 < midBlockStreet < I_2 \qquad (1)$$

Below are two steps to find midBlockStreet.

Step 1. Find *onStreetSegmentsSet*, which is the set of all street segments of ONST.

Step 2. Find *midBlockStreet*, which is the street segment in *onStreetSegmentsSet* that is dominated by I_2, and dominates I_1, by condition (1).

In the GIS database the middle point of *midBlockStreet* can represent the approximate traffic crash location.

4. RECTANGULAR RANGE QUERIES

For developing a practical crash analysis tool, we need to provide a function that finds all crashes that happened in a rectangular area. Such a function could be used to identify problematic rectangular areas with a high frequency of traffic crashes.

Consider the red rectangular area shown in Figure 3. The location of a traffic crash C(x,y) lies within the rectangle if C(x,y) is dominated by the upper right point Q(x,y) and C(x,y) dominates the bottom left corner point P(x,y). Hence we have:

$$X_C < X_Q \text{ and } Y_C < Y_Q \text{ and } X_P < X_C \text{ and } Y_P < Y_C$$

To implement dominance queries efficiently, we store all traffic crash records in an ECDF-tree, which runs in O(log N) time for each dominance query and requires only O(N logN) space where N is the number of traffic crash records. We investigated the computational complexity of finding traffic crashes dominated by a point Q. We experimentally compared the runtime performance of two algorithms for finding traffic crashes dominated by a point Q; the first algorithm used ECDF-trees [3] and the second is algorithm did not use any indexing.

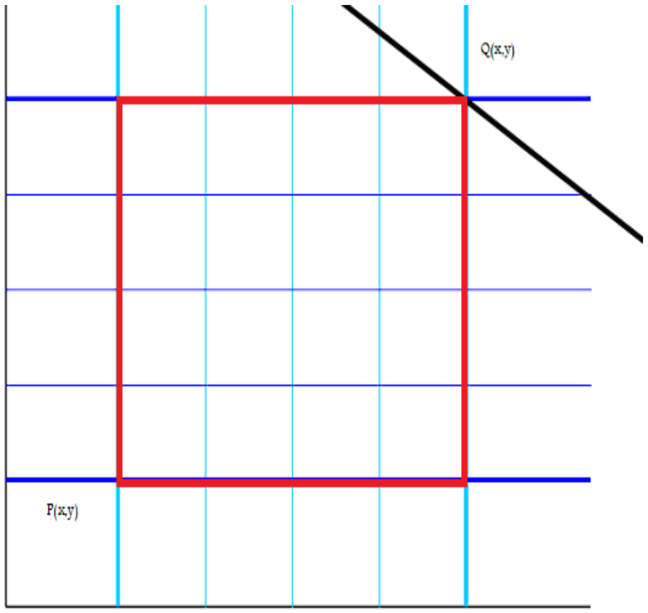


Figure 3: Traffic crashes in the red rectangular area are dominated by point Q(x,y) and dominate point P(x,y).

The results were visualized by the system. For example, Figure 4 shows traffic crashes dominated by point Q for January 2010.

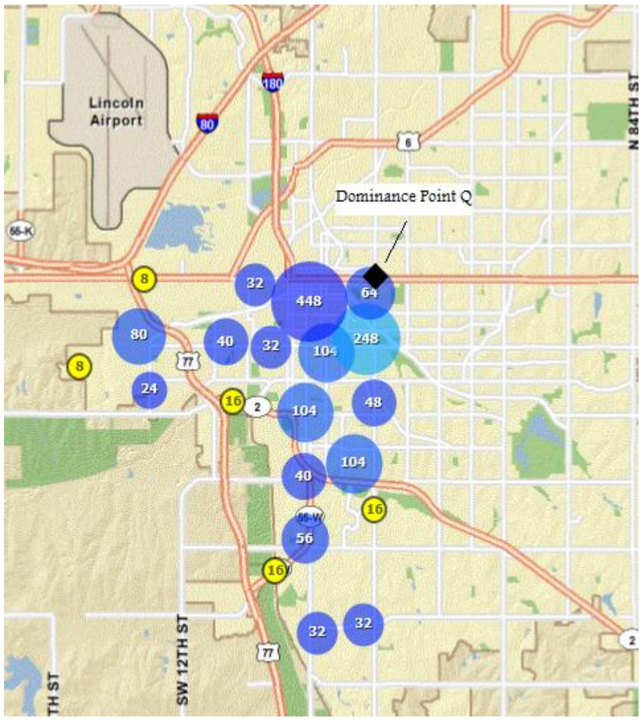


Figure 4: Traffic crashes dominated by point Q for January 2010.

Figure 5 and Table 2 show that as the total number of traffic crashes increase, the implementation with indexing is running faster than the one without indexing.

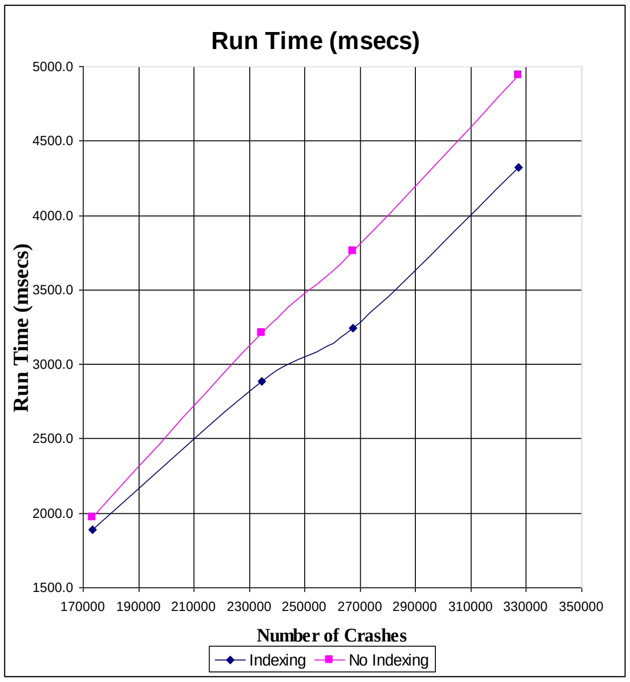


Figure 5: Run time using indexing versus no indexing.

Table 2: Run times using indexing versus no indexing.

Number of Crashes	Dominated Points	Indexing Run Time (msecs)	No Indexing Run Time (msecs)
7358	2029	46.8	46.8
14899	4134	73.0	78.0
173200	48028	1887.6	1971.0
234580	65448	2888.0	3213.6
267496	74856	3244.8	3760.0
327092	91404	4321.2	4945.2

5. THE SNOW COMPLAINT DATABASE

The city of Lincoln is prepared in case of any snowstorm to implement its snow removal plan. There are currently three snow removal districts, namely, the North Eastern, the South Eastern, and the Western districts.

In the past, street maintenance operations were maintained using Microsoft Access and Microsoft Excel applications for recording labor, snow removal equipments, material, and snow operations. The Access and Excel methods are inefficient for managing snow removal operations as the city grows.

In addition, in order to create a GIS map showing the current snow complaints, only a professional GIS specialist knows how to manually geocode a snow complaint. That poses a significant challenge to less-skilled end users who may need to use GIS technology. Hence there was a need to have a better tool for data management, especially applying GIS technology by less-skilled end users to handle day-to-day street operations. That was a major motivation for developing the *Street Maintenance Manager* application shown in Figure 7. The *Street Maintenance Manager* provides a Lincoln snow complaint map (Figure 9) that has been designed and developed using the current cutting-edge ESRI ArcGIS technology and state of the art non-traditional geocoding for non-GIS end-users.

The snow complaint application also has the same two main problems as the traffic crash application, that is, geocoding and rectangular range querying, which have already been addressed above. Therefore, we will not repeat them in this section. Instead we would like to present in Figure 6 an overview of the Street Operation Manager system.

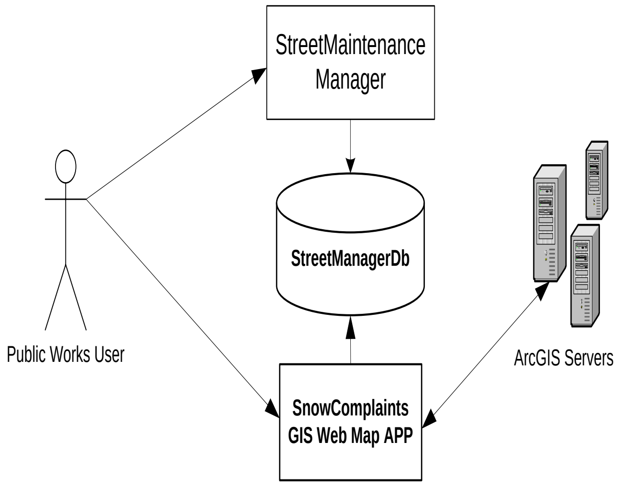
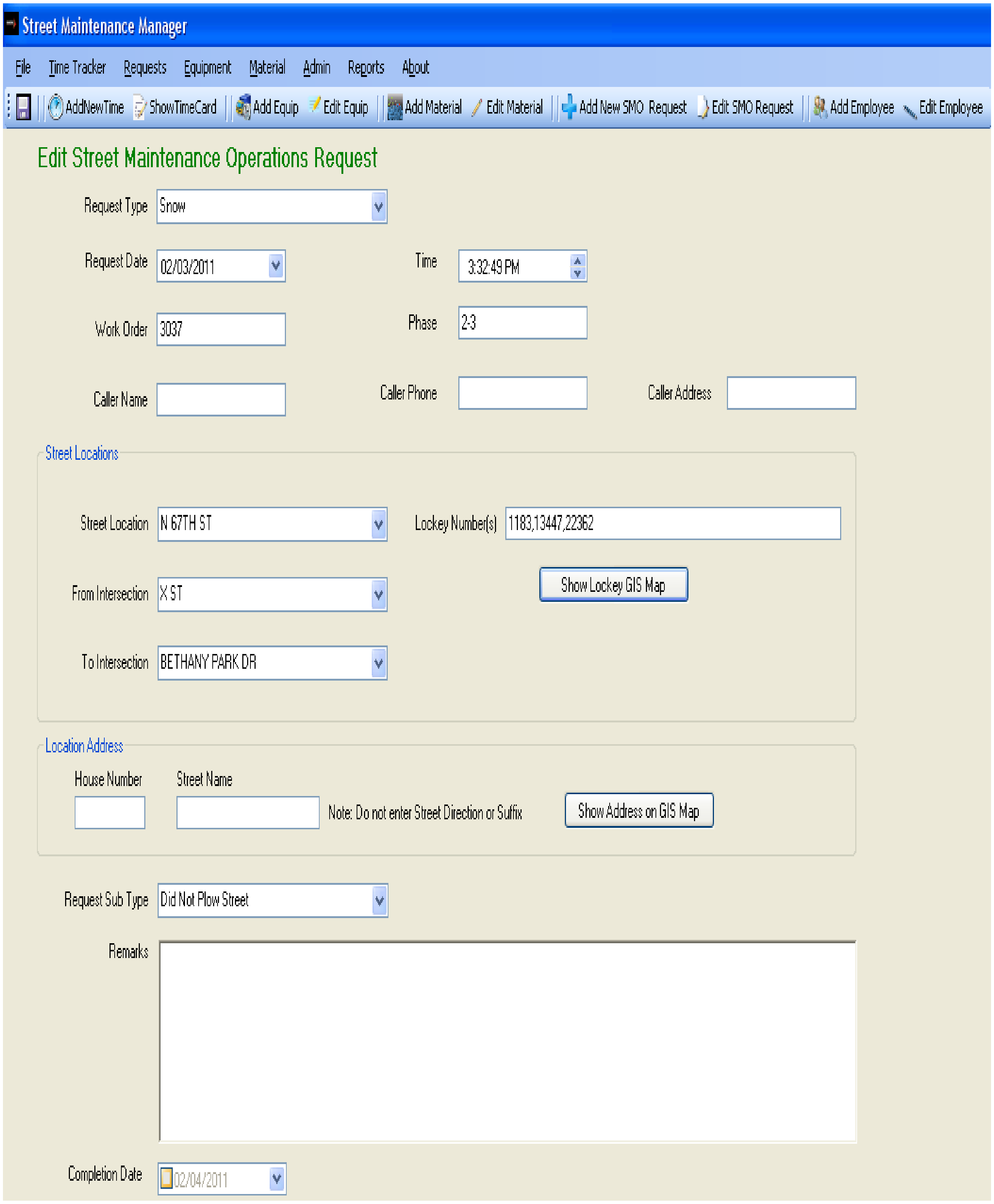


Figure 6: Street Maintenance Manager software architecture.

We developed the *Street Maintenance Manager* using the following steps. First, we developed a day-to-day input form including some street location fields for geocoding (such as On Street, At Street, and Between Street) for recording a snow complaint (see Figure 7). The input form also has a built-in street dictionary to quickly correct a street name, making sure that the complaint location does exist. This feature ensures that snow crews are sent to the correct locations.

Second, by geocoding the street location information, we can find the location of a snow complaint and display it in **real-time** on a map as soon as a public works user adds such a complaint to the *StreetManagerDb* database (see Figures 8 and 9). That is a useful new feature because in the past public works database users used a Microsoft Access program to record snow complaints and some days later a GIS specialist may have used street location information to manually geocode the Microsoft Access data using an ESRI ArcGIS geocoding tool. Therefore, the map could not be generated in real-time.


Street Maintenance Manager
File Time Tracker Requests Equipment Material Admin Reports About
AddNewTime ShowTimeCard Add Equip Edit Equip Add Material Edit Material Add New SMO Request Edit SMO Request Add Employee Edit Employee
Show Lockey GIS Map
Show Address on GIS Map

Edit Street Maintenance Operations Request

Request Type: Snow

Request Date: 02/03/2011
Time: 3:32:49 PM

Work Order: 3037
Phase: 2-3

Caller Name:
Caller Phone:
Caller Address:

Street Locations

Street Location: N 67TH ST
Lockey Number(s): 1183,13447,22362

From Intersection: X ST

To Intersection: BETHANY PARK DR

Location Address

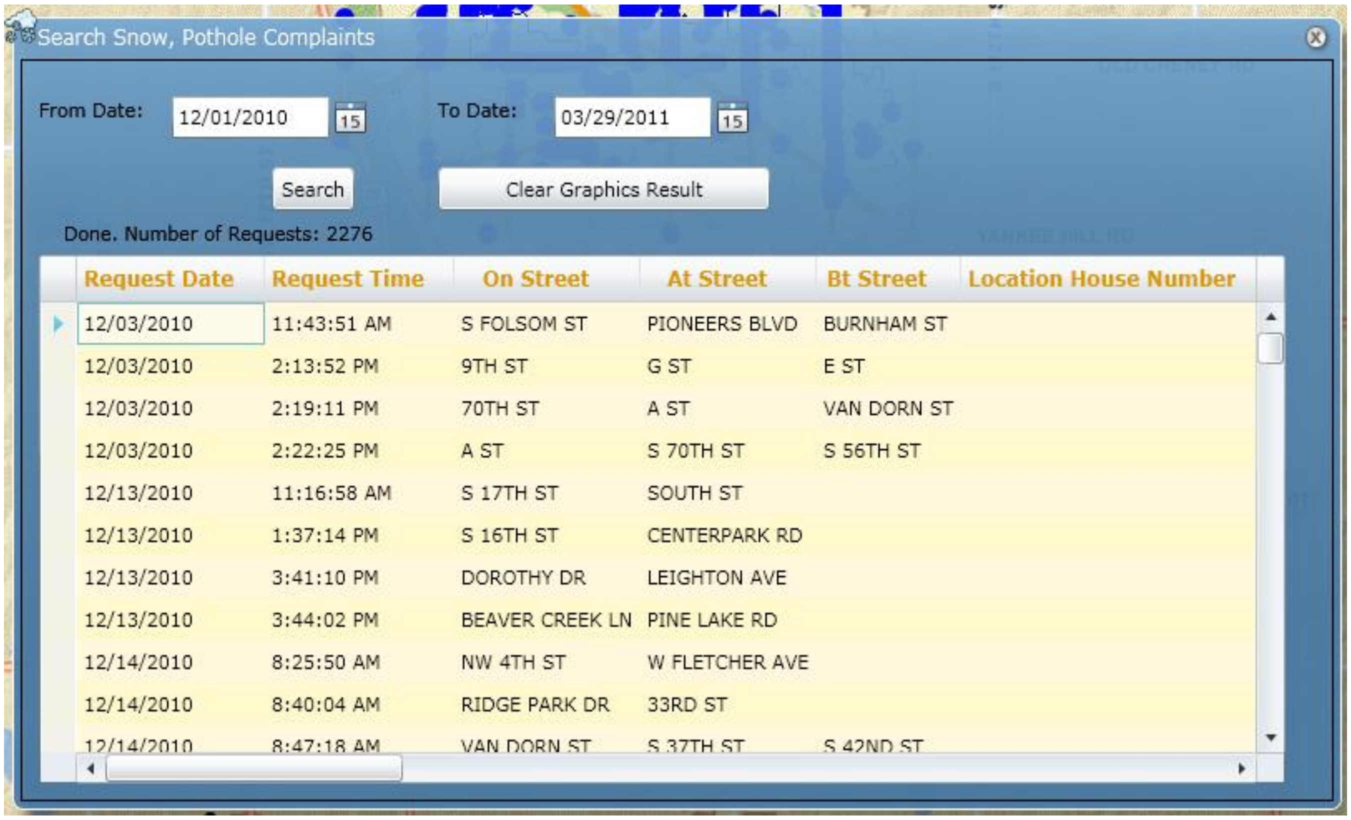
House Number:
Street Name:
Note: Do not enter Street Direction or Suffix

Request Sub Type: Did Not Plow Street

Remarks:

Completion Date: 02/04/2011

Figure 7: Snow complaint data entry form.


Search Snow, Pothole Complaints

From Date: 12/01/2010 To Date: 03/29/2011

Search | Clear Graphics Result

Done. Number of Requests: 2276

Request Date	Request Time	On Street	At Street	Bt Street	Location House Number
12/03/2010	11:43:51 AM	S FOLSOM ST	PIONEERS BLVD	BURNHAM ST	
12/03/2010	2:13:52 PM	9TH ST	G ST	E ST	
12/03/2010	2:19:11 PM	70TH ST	A ST	VAN DORN ST	
12/03/2010	2:22:25 PM	A ST	S 70TH ST	S 56TH ST	
12/13/2010	11:16:58 AM	S 17TH ST	SOUTH ST		
12/13/2010	1:37:14 PM	S 16TH ST	CENTERPARK RD		
12/13/2010	3:41:10 PM	DOROTHY DR	LEIGHTON AVE		
12/13/2010	3:44:02 PM	BEAVER CREEK LN	PINE LAKE RD		
12/14/2010	8:25:50 AM	NW 4TH ST	W FLETCHER AVE		
12/14/2010	8:40:04 AM	RIDGE PARK DR	33RD ST		
12/14/2010	8:47:18 AM	VAN DORN ST	S 37TH ST	S 42ND ST	

Figure 8: Search snow complaints from 12/01/2010 to 03/29/2011.

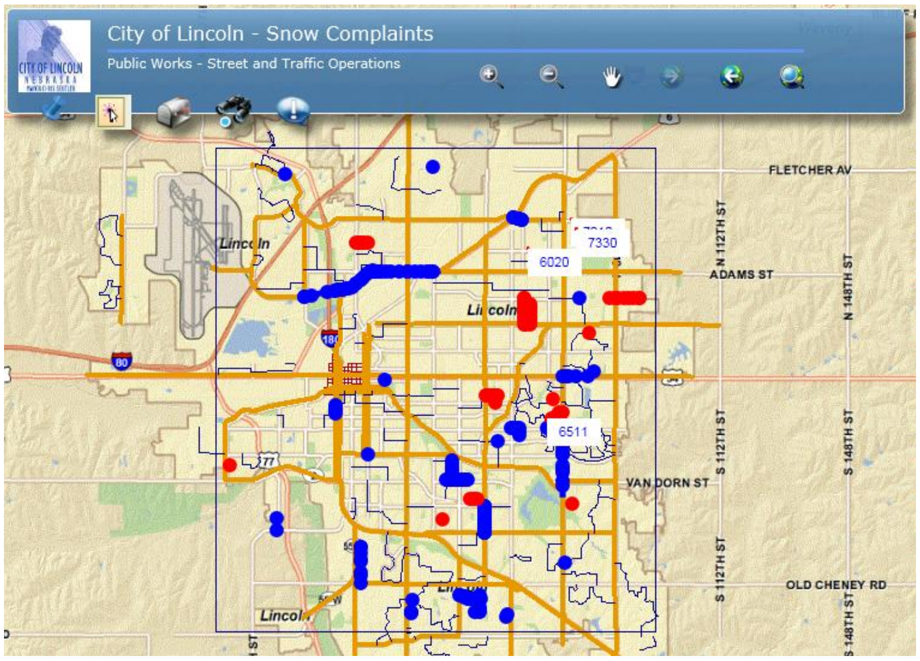


Figure 9: Range queries.

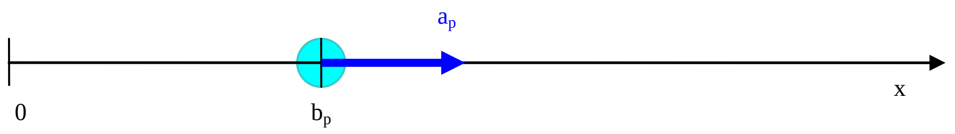


Figure 10: The moving point P starting at b_P and moving with speed a_P.

6. EFFICIENT MOVING POINTS ESTIMATION

As shown in Figure 10, the position of any point P moving linearly along the x-axis can be represented by a function $a_p.t + b_p$ where a_p is the speed and b_p is the starting position of point P. Several GIS applications need to efficiently solve the following:

Count Problem for Moving Points: *Given n moving points P_0, P_1… P_{n-1} in one dimensional space with parametric functions $P_i = a_P.t + b_P$ for $i = 0,…, n-1$ and a query point Q with $Q = a_Q.t + b_Q$, find the number of P_i to the left of Q at given time t.*

For example, we may want to know how many cars are to the left of a slow moving vehicle, which may be a snow removal truck.

[11] presents an indexing algorithm that finds an approximate count for this problem in only O(m log N) time, where m is a constant parameter that controls accuracy of the result and N is the number of moving vehicles. A key element of the algorithm is representing each moving point P = $a_p.t + b_p$ by a static point P’ = (a_p, b_p) in a dual plane as shown below in Figure 11.

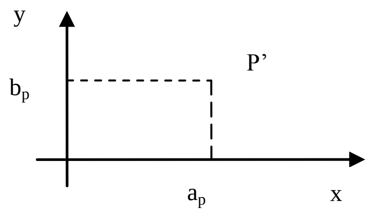


Figure 11: The dual representation of point P(a_P,b_P).

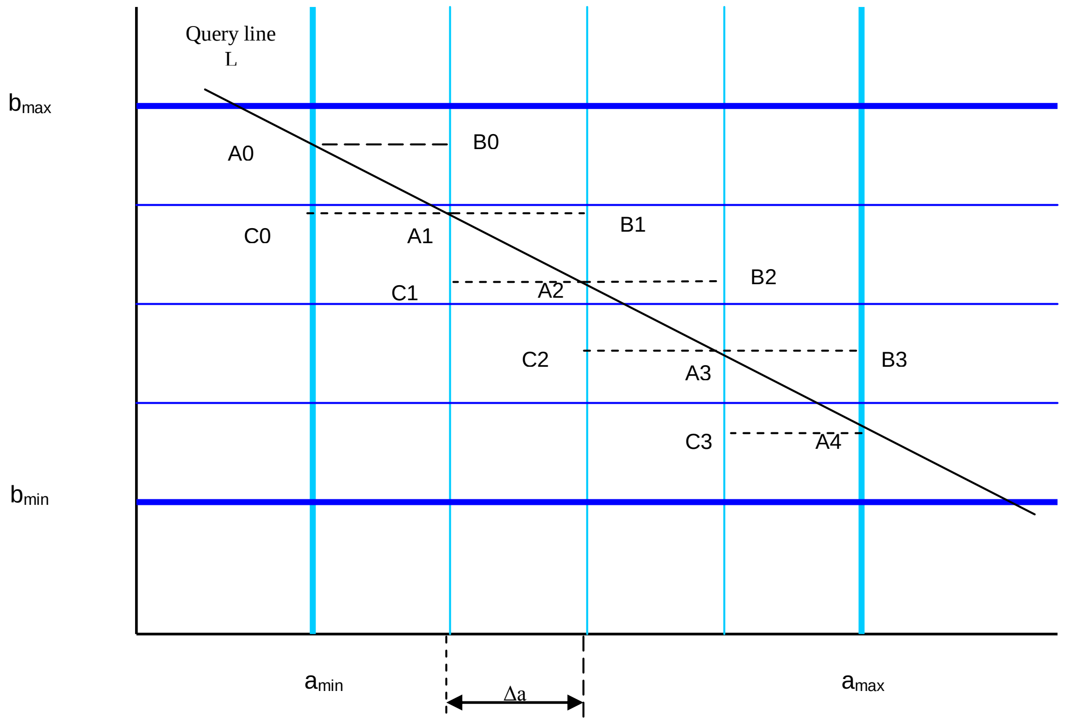


Figure 12: Approximating points below query line L.

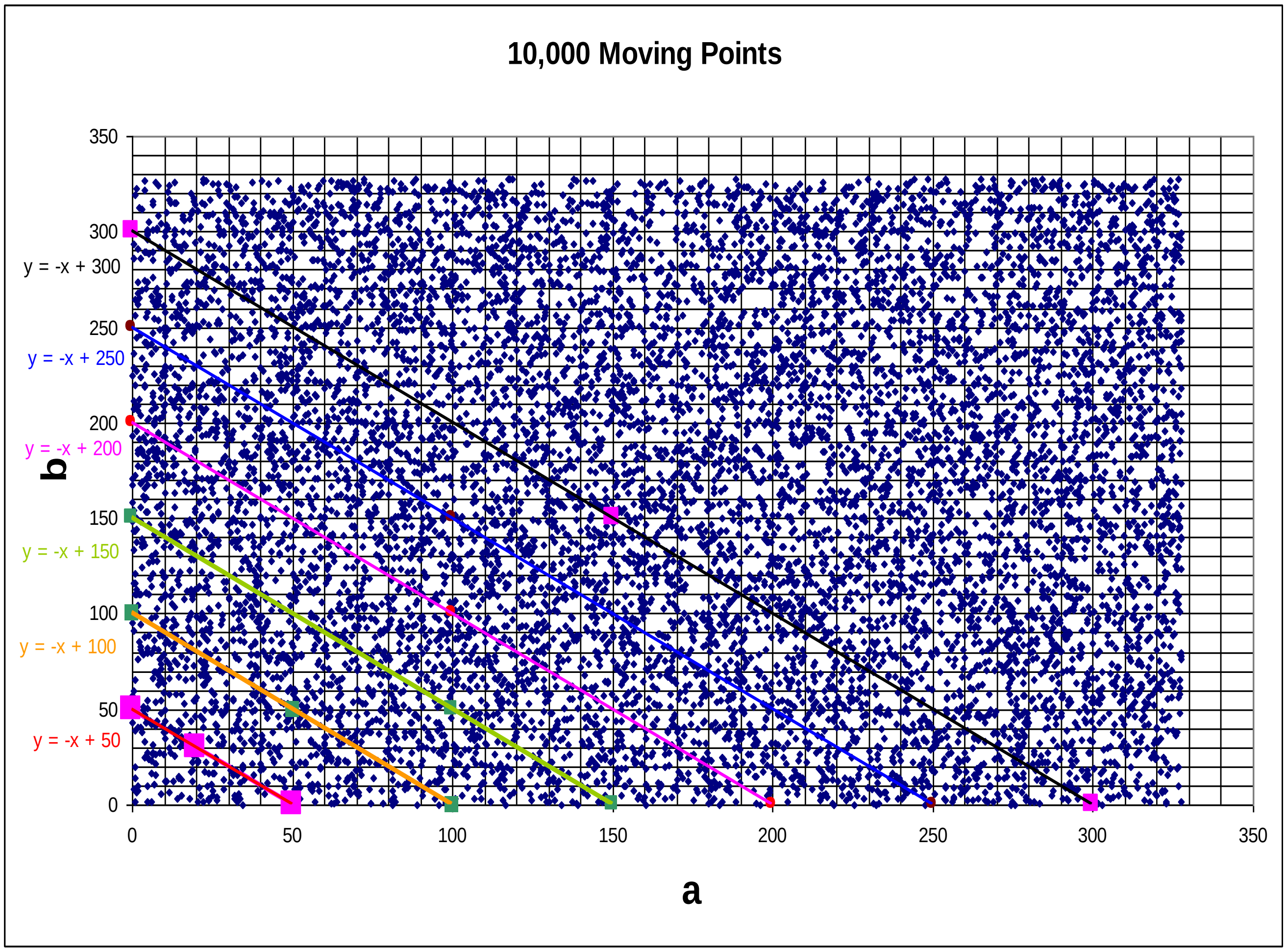


Figure 13: 10,000 moving points and query lines in dual plane.

This dual representation is attractive because of the following well-known lemma:

Lemma 1 Let P = $a_p.t + b_p$ and Q = $a_Q.t + b_Q$ be two moving points in one-dimensional space, and let P'= (a_p, b_p) and Q' = (a_Q, b_Q) be their corresponding static points in the dual plane. If P overtakes Q or vice versa at time instance t, then the following must hold:

$$-t = \frac{b_P - b_Q}{a_P - a_Q}$$

That is, -t is the slope of the line P'Q'. Hence, the **Count Problem for Moving Points** reduces to the problem of finding how many points are below a ***query line*** **L**, where **L** is a line crossing Q' with slope –t in the dual plane.

Further, [11] reduces the problem of finding the number of points below a line to an O(m) number of ECDF-tree queries as shown in Figure 12, which leads to the following algorithm.

MovingPoints Algorithm

Inputs -- n points $P_0(a_{P0}, b_{P0}),\ldots, P_{n-1}(aP_{n-1}, bP_{n-1})$
– query point Q (a_Q, b_Q)
– m: the number of line divisions
– t: the query time in seconds

Output: the number of points P_i dominated by Q at time t.

1. Find $a_{max} = \max(a_{p0},\ldots,a_{pn-1})$, $a_{min} = \min(a_{p0},\ldots,a_{pn-1})$, and $b_{max} = \max(b_{p0},\ldots,b_{pn-1})$, $b_{min} = \min(b_{p0},\ldots,b_{pn-1})$.
2. Compute a = $(a_{max} – a_{min})$/m and b = $(b_{max} – b_{min})$/m.
3. Compute arrays A[m], B[m-1], and C[m-1].
4. Compute and return the following approximate count of the number of points below line L through Q with slope –t

$$Approx\#below = \frac{\#(A_1, I) + \#(A_{m+1}, I) + \sum_{i=0}^{m} \#(B_i, I) - \#(C_i, I)}{2}$$

where, I is the ECDF-tree that stores the dual representations of the moving points and $\#(A_i, I)$ is the number of points dominated by point A_i, $\#(B_i, I)$ is the number of points dominated by point B_i, and $\#(C_i, I)$ is the number of points dominated by point C_i.

We implemented the indexing method of [11] on a data set that contained 10,000 random moving points that were created by allowing both a_P and b_P to vary uniformly between 0 and 33. Figure 13 shows a graph of 10,000 moving points and seven query lines in the dual plane.

Table 3 records the running time for counting number of points below each query line L with different m and Q at time t = 1 second. In Table 3 we use the following parameters:

(1) ***# of dominated points***: the number of points dominated by Q at time t.

(2) ***Counting time***: The total of counting time for counting the number points dominated by Q at time t (excluding the time for setting up and creating the ECDF-tree).

The last column of Table 3 is used to show the accurate answers. We can see that the answer with m = 10 had a maximum error of 26, while with m = 100 had a maximum error of only 6. In both cases the maximum error occurred with the query point Q(150, 150). For Q(150, 150), the error was 0.6 percent with m = 10 and only 0.14 percent with m = 100.

Table 3: Running time results for point dominance queries.

Line		Estimate with m = 2	Estimate with m = 5	Estimate with m = 10	Estimate with m = 100	Estimate with m =10,000
Q(20,30), t = 1 sec	# of dominated points	356	140	116	103	105
	Counting time	1 msec	1 msec	1 msec	9 msecs	764 msecs
Q(50,50), t = 1 sec	# of dominated points	750	490	445	446	446
	Counting time	< 1 msec	1 msec	1msec	8 msecs	763 msecs
Q(100,50), t = 1 sec	# of dominated points	1129	1046	1022	1018	1019
	Counting time	< 1 msec	< 1 msec	2 msec	8 msecs	765 msecs
Q(100,100), t = 1 sec	# of dominated points	2036	1815	1806	1823	1820
	Counting time	< 1 msecs	< 1 msecs	1 msecs	8 msecs	767 msecs
Q(100,150), t = 1 sec	# of dominated points	3159	2895	2874	2875	2875
	Counting time	1 msec	1 msec	1 msec	8 msecs	770 msecs
Q(150,150), t = 1 sec	# of dominated points	4324	4174	4176	4144	4150
	Counting time	< 1 msec	< 1msec	1 msec	8 msecs	776 msecs
Q(330,330), t = 1 sec	# of dominated points	10,000	10,000	10,000	10,000	10,000
	Counting time	< 1 msec	< 1 msec	< 1 msec	8 msecs	808 msecs

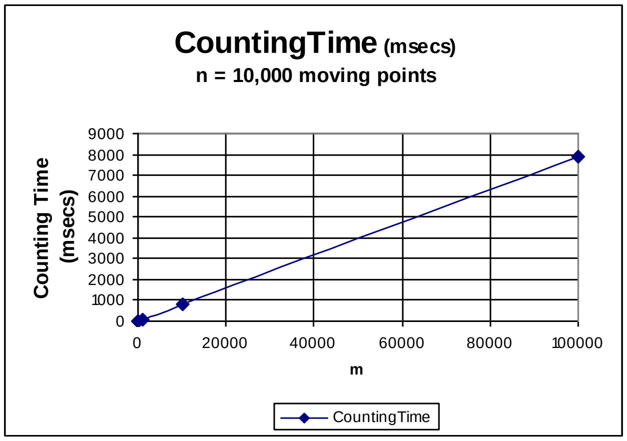


Figure 14: Time vs. m graph with n = 10,0000 moving points.

Figure 14 shows that the counting time varies linearly with the value of m, as expected. Therefore, there is a natural trade-off between the accuracy of the approximation and the counting time.

7. FUTURE WORK

In the future, we would like to extend our system to show all the moving snow removal trucks. This would be an interesting synthesis of several different types of indexing algorithms. For example, the efficient max-count algorithm of [1] could find the maximum number of snow removal trucks in a rectangular area between two time instances, and the 1-dimensional moving point algorithm could find the number of snow removal complaint locations on a straight road ahead of a particular snow removal truck.

Instead of the static ECDF trees [3], we also plan to implement the dynamic ECDF trees [19], which would allow efficient addition and deletion of moving objects. Finally, we also plan to consider the problem of estimating aggregate values of the number of moving vehicles in an area when the movement of the vehicles is not linear but polynomial, based on the recent model of [17].

8. ACKNOWLEDGMENTS

We would like to thanks Alicea McCluskey, Shane Dostal, Scott Opfer, Al McCracken, and Angela Chesnut of Lincoln Public Works for providing information and suggestion about street maintenance operations and traffic crash analysis; Mark Wieting of Lincoln Information System for providing traffic crash data from CJIS; Frank Larson, Tracy Schuppan, and Tan Pham of Lincoln CEIS for providing GIS technical advice; James Anderson of Lincoln IS for supporting the ArcGIS Server; Virendra Singh of Lincoln Public Works for expert advice regarding traffic crash analysis; Tim Pratt of Lincoln CEIS for managing and overseeing software development, and Roger Figard for allowing and encouraging us to publish these results.

The second author served as a Jefferson Science Fellow at the U.S. Department of State during part of this research work. The opinions expressed in this paper are only the authors' own.

9. REFERENCES

[1] Anderson, S. and Revesz, P. Efficient max-count and threshold operators of moving objects. *Geoinformatica*, 13(4): 355-96, 2009.

[2] Bao, J., Ying, J., and Hayamizu, S. Development of traffic analysis system using GIS. *Proceedings of the 2nd International Conference on Information Science and Engineering*, page 3648, 2010.

[3] Bentley, J.L.. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4), 1980.

[4] Bigham, J.M., Rice, T.M., Pande, S., Lee, J., Park, S.H., Gutierrez, N., and Ragland, D.R. Geocoding police collision report data from California: A comprehensive approach. *International Journal of Health Geographics*, 8, 2009.

[5] City of Lincoln, Criminal justice information system (CJIS): http://cjis.lincoln.ne.gov/~lpd/cfstoday.htm

[6] City of Lincoln, 2008 crash study: http://lincoln.ne.gov/city/pworks/engine/crash/index.htm

[7] Kamal, M.A. City planning and development using geographic information systems. *Proceedings of the 11th International Conference on Computer and Information Technology*, page 3, 2008.

[8] Kanellakis, P., Kuper, G., and Revesz, P., Constraint query languages, *Journal of Computer and System Sciences*, 51(1):26-52, 1995.

[9] Nwaneri, S.O. Mapping intersection accidents with GIS technology in Huntsville, Alabama, U.S.A. Proceedings of the *IEEE International Geoscience and Remote Sensing Symposium*, volume 6, page 3727, 2003.

[10] Parrish, L.S., Dixon, B., Cordes, D., Vrbsky, S.,and Brown,.D. CARE: An automobile crash data analysis tool. *IEEE Computer Society*, 36(6):22, June 2003.

[11] Revesz, P. Efficient rectangle indexing algorithms based on point dominance. *Proceedings of the 12th International Symposium on Temporal Representation and Reasoning*, IEEE Press, pages 210-12, Burlington, VT, June 2005.

[12] Revesz, P. *Introduction to Databases: From Biological to Spatio-Temporal*, Springer, 2010.

[13] Steiner, R., Bejleri, I., Yang, X., and Kim, D. Improving geocoding of traffic crashes using a custom ArcGIS address matching application. *Proceedings of the 22nd Environmental Systems Research Institute International User Conference*, San Diego, 2003.

[14] Sewell, C. Remembering a snowstorm that paralyzed the city, *New York Times*, February 10, 2009.

[15] Steenberghen, T., Dufays, T., Thomas, I., and Flahaut, B. Intra-urban location and clustering of road accidents using GIS: A Belgian example. *International Journal of Geographical Information Science*, 18(2):169–181, 2004.

[16] Wikipedia article on geocoding http://en.wikipedia.org/wiki/Geocoding

[17] Yue, H., Jones, E., Revesz, P., Local polynomial regression models for vehicle speed estimation and forecasting in linear constraint databases. *Proceedings of the 17th International Symposium on Temporal Representation and Reasoning*, IEEE Press, pages 154-161, Paris, France, September 2010.

[18] Zhang, D., Tsotras, V. J., Gunopulos, D., Efficient aggregation over objects with extent, *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 121-132, 2002.

[19] Zhang, J., Chen, L., You, S., and Chen, C. A hybrid approach to segment-type geocoding of New York City traffic data. *Proceedings of the 1st International Conference on Computing for Geospatial Research and Applications, ACM Digital Library, http://dx.doi.org/10.1145/1823854.1823871, 2010.*