An Enhanced Vector-Based Update Algorithm for Network-Constrained Moving Clients

Wen Wu
Computer Science Department of
Hangzhou Dianzi University
HangZhou, China
wenwujuanwu@gmail.com

Jian Xu, Ming Xu, Ning Zheng
Computer Science Department of
Hangzhou Dianzi University
HangZhou, China
{jian.xu, mxu, nzheng}@hdu.edu.cn

Abstract—**Location based services (LBS) is one of the fastest growing areas in recent years. Location update of mobile clients is fundamental in all types of LBS. But algorithms proposed in this field generally didn't concern the restricted context of road network and caused some unnecessary update. This paper proposed a revised vector-based update algorithm which taking characteristics of road network into account. The improved approach adopts two-step correction to revise the predicted position of the moving clients. First, the erect correction is implemented. Second, horizontal correction is carried out based on the erect correction. The proposed algorithm is evaluated via the simulator and compared with the vector-based algorithm. Experimental results show our algorithm can significantly decrease the number of updates compared to barely vector-based strategy while having lower error, and is scalable and stable.**

Keywords- location update; road network; erect correction; horizontal correction

I. Introduction

In recent years, the number of GPS-enabled devices has drastically increased. In 2009, there was an estimated 27 million GPS-equipped smart phones sold, resulting in the world-wide GPS user-based to at least 68 million in the consumer market alone [4, 5]. So LBS is becoming more and more popular. The main idea of LBS [7, 8] is to provide the user with a service that is dependent on the user's positional information, so the client's position is the most important factor in LBS.

Services based on the client's position may require different degrees on the tracking of geographical positions of moving clients. To accomplish tracking mobile clients with certain accuracy, every GPS device monitors its real position (i.e. its GPS position) and then it compares this with a local copy of the position that the central database assumes. When needed to achieve the required accuracy in the database side, the GPS device issues an update to the server. The database may predict the future position of a mobile client in different ways. In general case, the database explicitly informs the mobile client about how it predicts the mobile device's position.

There are two key challenges that may affect the system's performance and service quality in future mobile systems which support location-based services and applications: (1)the high cost of network bandwidth and energy consumed on the mobile clients for frequent location tracking and updates at the location servers[6]; (2) the challenge of scaling large amount of location updates at the location server as the number of mobile clients demanding to be tracked increases in a location determination system.

Predicting future positions of the clients aims at minimizing the number of updates between the mobile clients and the server. Reduction of updates reduces communication and server-side updates processing. So how to predict the future positions of the clients in the database is a key problem which this paper focuses on.

This paper has three main contributions. First, we introduce erect correction algorithms to revise the predictive position obtained from vector-based strategy. Second, we introduce horizontal correction algorithm to further correct the predictive location. Third, we provide experiments to prove the performance of our algorithm in number of updates, average error, scalability and stability.

The remainder of this paper is organized as follows: Section Ⅱ surveys the related work. In section Ⅲ, we discuss the system model. We expand our algorithm in section Ⅳ. Section Ⅴgives out analysis and experimental results. Finally, we conclude the paper in section Ⅵ.

II. Related Work

A number of positioning systems are publicly available for tracking the location update of mobile clients moving on the road network, such as Google's Latitude and Skyhook wireless WiFi positioning system [3]. Frequent location updates make the server to keep track of client's current location and ensure the accuracy of query results. The algorithm clients employed to determine when and where to update their locations is often referred to as the location update strategy, we below describe some typical algorithms for reducing the updates between the mobile clients and the server.

Periodic update algorithm.

This strategy is the simplest time-based location update strategy, in which the location server maintains the location update for each mobile client at a fixed time interval. This update algorithm implies that mobile clients are treated as stationary between updates. What's more, as it doesn't take threshold into account, this strategy can't ensure the accuracy of the recorded locations at the server side. With the incremental demand of the accuracy of positioning, this algorithm which doesn't consider the error threshold can't be widely used. What's more, it produces too many updates which is a fatal drawback.

Point-based update algorithm.

This approach uses the distance-based scheme and the server only records an update when the mobile client travels more than a delta threshold away in distance from the location of last update. The number of location updates per unit time will depend on the speed of the mobile user. This algorithm may also produce redundant updates when the mobile clients are moving on a straight road segment.

Segment-based update algorithm.

A segment-based updated strategy [9] utilizes the underlying road network to minimize the number of updates. Mobile clients are assumed to move at a constant speed on their current road segment. An update is sent when the distance between the current and the predicted location is larger than a system-defined threshold. It assumes that mobile clients change their velocities at the end of each segment. Thus an update will be sent when the mobile client departs from a segment node by delta distance. This algorithm doesn't work well and is outperformed by vector-based strategy when the structure of the road network is complex or the length of road segment in the road network is short [2].

III. System Model

We assume that moving clients are constrained by a road network and all of them are capable of obtaining their positions from an associated GPS receiver. Moving clients send their location information to a central database which is also named server via a wireless communication network.

After each update from a moving object, the database informs the moving object of the algorithm it will use to predict the object's position. The moving object then is always aware of where the server thinks it is located. The moving object issues an update when the predicted position deviates by the threshold from the real position obtained from the GPS device.

Fig. 1 shows the schematic diagram of how the system works.

The moving client initially obtains its location information from the GPS device. It then establishes a new connection with the server and issues an update, sending the GPS information to the server.

When the server receives the update, it firstly stores the information received from the moving client in the database and then sends its representation of the client's current and future position to the client.

When the client receives this information from the server, it obtains its actual, current location information from the GPS device. The client then calculates its predicted position using the algorithm received from the server and compares this position to the GPS position. If the distance between these two exceeds the given threshold, the client issues an update to the server. If not new comparison is made and this procedure continues until it is terminated by the client.

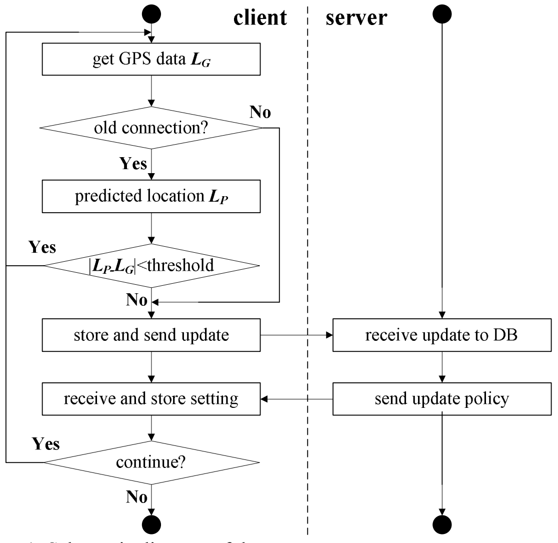


Figure 1. Schematic diagram of the system

This paper focuses on the updating algorithm i.e. how to predict the location of the moving clients.

IV. Algorithm

A. *Vector-based Algorithm*

Vector-based algorithm is widely used in prediction of an object's future position. It uses the velocity vector of the mobile object to make a prediction about its location. An update is only sent when the current location of the mobile client deviates from its predicted location by a predefined threshold. An example for the vector-based strategy is shown in Fig. 2.

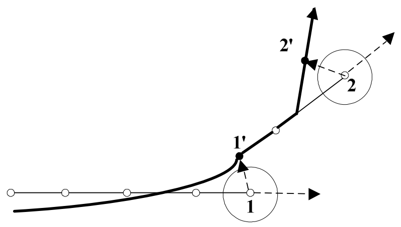


Figure 2. Vector-based algorithm

In Fig. 2, the circles indicate the threshold and the solid points (i.e. 1' and 2') indicate the locations that result from updates issued by the moving client. The bold line represents the road network and all the points except 1 and 2 indicate the trajectory of a moving client which the server side records.

As illustrated above, we can see that vector-based algorithm has some obvious shortcomings:

1. When there are many curved road segments in the road network, the vector-based algorithm doesn't perform well.
2. It doesn't take the regional restriction of moving clients' movement into account. In fact, many researches don't

consider the regional restrictions of the moving objects [1]. Without respect to the geographic restrictions of clients will lead to large predictive error. In real scenario, the motion of the moving objects is almost restricted all the time and a typical case is that the movement of clients is limited by the road network.

3. The vector-based algorithm can't detect the positioning errors of the GPS device.

B. Enhanced Vector-based Algorithm for Network-constrained Moving Clients

1) Dealing with Wrong GPS Position

Although GPS-equipped devices are widely used to locate clients' positions, as we all know, the positioning accuracy of GPS may be affected by many factors such as tall buildings and the weather conditions. As a consequence, sometimes positions of the moving objects obtained from GPS devices have large error and previous researches didn't propose appropriate solutions. In this paper, we give out a corresponding solution according to the road network.

As shown in Fig. 3, we assume that A indicates the starting point of two short road segments, B and C represent certain points on these two road segments respectively, so line section AB and AC can approximately denote the road segments in the road network. Actually, there can be many road segments with the starting point A, for simplicity, our example here only takes two road segments into account. L is the position we obtain from the GPS device, L' and L'' are the projections of L on the line segments AB and AC respectively. So we can get the distance from L to the road segment AB, i.e. d1:

$$d1 = \frac{|(Xb - Xa)Ly + (Ya - Yb)Lx + XaYb - XbYa|}{\sqrt{(Xb - Xa)^2 + (Ya - Yb)^2}}$$

(1)

In the above expression, (Xa, Ya), (Xb, Yb) and $Lx, Ly)$ represent the coordinates of A, B and L respectively. We assume the allowed positioning error of GPS is D (as shown in Fig. 3), in real life, the value of D is about 10 meters. If $d1$, $d2$ and D comply with the following inequality:

$$D < d1 \le d2$$

(2)

We will not issue an update and take the predicted location as the real position of the moving clients.

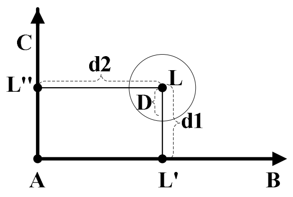


Figure 3. An example of wrong GPS positioning

2) Erect Correction

a) Minimum Position Correction

First, we describe the basic idea of the minimum correction. For simplicity, we present the schematic diagram of this algorithm in Fig. 4, there are some differences between the schematic diagram and the actual execution of this strategy. A simple road network is constructed by a series of nodes (i.e. 0, 1, 2, 3, 4, 5), the nodes (i.e. 1', 2', 3', 4' and 5') are generated by the vector-based algorithm.

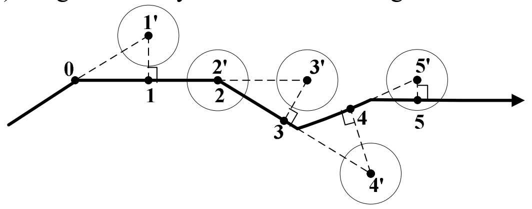


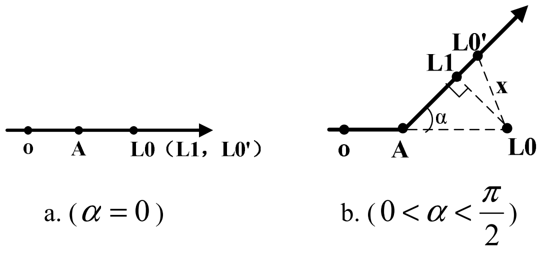
Figure 4. Minimum position correction algorithm

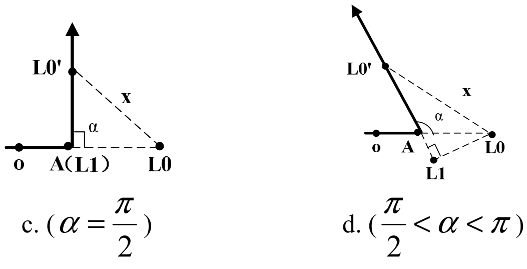
Every time we get a predictive location through vector-based strategy, we make a perpendicular line from the predictive point to the corresponding road segment. And then we use the revised position to compare with the location (here we assume that this location is the accurate position of the moving objects) obtained from the GPS device, if the distance between these two is out of the system-defined threshold, an update is sent by the moving client, otherwise the procedure goes on.

In this paper we assume that the velocity vector of the moving clients keep unchanged during a system-defined timestamp. As shown in Fig. 5, we give out typical situations to illustrate how the algorithm works. We will provide specific analysis of b, other situations in Fig. 5 can be analyzed by the same means used in b. In Fig. 5. b, velocity vector doesn't alter when the object travels on the road segment with the starting point o and ending point L0'. L0 indicates the predictive location by vector-based algorithm. We get a point L0' on the road segment and make AL0=AL0'. As the velocity vector is constant, L0' represents the accurate location of the moving client. L1 is the position obtained through our minimum position correction algorithm, α is the angle between the velocity vector and the road segment AL0', it is also the angle between the road segment AO and AL0'. We assume that the threshold in the system is x , i.e. the length of L0L0'is x, the length of L1L0' is the distance between the location obtained by our minimum position correction and the accurate position of moving client. Then we can get the length of L1L0' is $x \times \sin\frac{\alpha}{2}$, if the following inequality:

$$x \times \sin\frac{\alpha}{2} < x \qquad (3)$$

is tenable, we can get a better performance through minimum position correction algorithm than vector-based strategy. Inequality formula (3) always holds true when $0 < \alpha < \frac{\pi}{2}$, so we can say that minimum position correction algorithm is superior to the vector-based strategy in the most typical situation.




L0'
x
α
o
A(L1)
L0
c. ($\alpha = \frac{\pi}{2}$)
L0'
x
α
o
A
L0
L1
d. ($\frac{\pi}{2} < \alpha < \pi$)

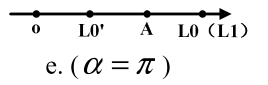


Figure 5. a-e Different situations in minimum position correction

b) Maximum Position Correction

The basic idea of the maximum position correction is described as follows: compared to minimum position correction, every time we obtain a predictive location by vector-based strategy, we make a perpendicular line of the velocity vector and we see the intersection point of the perpendicular line and the road segment as the revised location. The schematic diagram of this algorithm is shown in Fig. 6, the executive process of this algorithm is the same as minimum position correction.

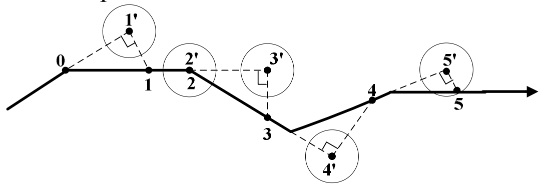


Figure 6. Maximum position correction algorithm

In Fig. 7, like the minimum position correction algorithm, we describe the most typical case happened in the prediction to analyze the maximum position correction algorithm. L1 is the location obtained through maximum position correction algorithm, the meanings of the remaining points are the same as defined in minimum position correction algorithm above.

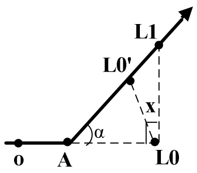


Figure 7. A typical case of maximum position correction

Here we define $Dis(A,B)$ as the distance between point A and point B. In Fig. 7, we can get the distance between the location obtained by our maximum position correction and the accurate position of moving client:

$$Dis(L0'L1) = \frac{x}{\sin\alpha} \times \cos\frac{\alpha}{2} \times (\frac{1}{\cos\alpha} - 1) \tag{4}$$

If the following inequality:

$$\frac{x}{\sin\alpha} \times \cos\frac{\alpha}{2} \times (\frac{1}{\cos\alpha} - 1) < x \tag{5}$$

is tenable, i.e. $\cos\frac{\alpha}{2} \times (1 - \cos\alpha) < \sin\alpha \times \cos\alpha$, maximum position correction is better than vector-based algorithm.

We have analyzed two erect correction algorithms and we will make a comparison between them. From these two algorithms, we have obtained the distance between the correction positions and the accurate location gained from GPS device respectively. In order to compare the performance of these two algorithms, we need to analyze the following inequality:

$$x \times \sin\frac{\alpha}{2} < \frac{x}{\sin\alpha} \times \cos\frac{\alpha}{2} \times (\frac{1}{\cos\alpha} - 1) \tag{6}$$

We can get that the minimum position correction is better than the maximum position correction algorithm under the most typical state, i.e. $0 < \alpha < \frac{\pi}{2}$.

3) Horizontal Correction

First, we depict the basic idea of horizontal correction. As shown in Fig. 8, the bold line represents the road network, C represents the predictive location through the vector-based strategy, $P_{\min}$ and $P_{\max}$ are respectively the positions corrected by the minimum and maximum correction algorithm, P_0 indicates the accurate location of moving client obtained from the GPS device, r stands for the threshold defined in the system.

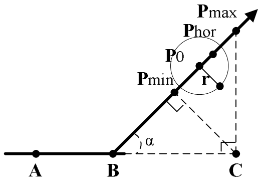


Figure 8. A typical case of midpoint position correction

In Fig. 8, we can see that although the position predicted by vector-based algorithm is revised by two erect correction algorithms, an update is still needed to be issued because the distance between the accurate position (i.e. P_0) of the moving client and the minimum correction position (i.e. $P_{\min}$) exceeds the system-defined threshold, and the distance between P_0 and the maximum correction position (i.e. $P_{\max}$) is also out of the system-defined threshold. According to the analysis of erect correction algorithm mentioned above, we know that the accurate position of the moving client must be on the right of point $P_{\min}$, what's more, it must be on the left of point $P_{\max}$, which means that the accurate location must be on the road segment $P_{\min}P_{\max}$, so we take the midpoint of the segment $P_{\min}P_{\max}$ as our horizontal correction point. Let us assume the coordinates of $P_{\min}$ and $P_{\max}$ are respectively $(P_{\min}x, P_{\min}y)$ and $(P_{\max}x, P_{\max}y)$, the coordinate of P_{hor} is $(P_{hor}x, P_{hor}y)$, we can obtain that:

$$P_{hor}x = \frac{1}{2}(P_{\min}x + P_{\max}x) \qquad (7)$$

$$P_{hor}y = \frac{1}{2}(P_{\min}y + P_{\max}y) \qquad (8)$$

If the location revised by the horizontal correction based on the erect correction algorithm is within the threshold as shown in Fig. 8, we can reduce an update between the moving client and the server, otherwise an update is needed.

4) Handling Predictive Position at Intersections

We have analyzed how to revise the position obtained from the vector-based algorithm on a single road segment, it means that we do not need to determine which road segment clients should be located on. However, in actual environment, the road network is considerably complex and there are many line segments at the connection point resulting in that we have to use the directed graph to represent it. So if we want to apply our proposed algorithm to the real road network, we must solve the problem that how to locate the moving objects to the right road segment at the intersections.

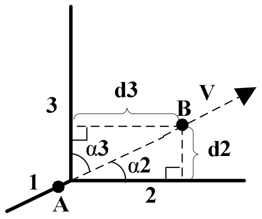


Figure 9. An example with predictive position at intersection

As shown in Fig. 9, for simplicity, let us assume that road segments 1, 2 and 3 compose a road network, B is the location predicted through the vector-based strategy, d_2 and d_3 are the distance from B to road segment 2 and 3 respectively, α_2 and α_3 are the angles between the velocity vector and the road segments 2 and 3. If $d_2 < d_3$, we will choose road segment 2 as the right segment on which the client are moving and vice versa.

The detailed enhanced vector-based algorithm is given in Algorithm 1.

In algorithm 1, functions GetMinCor(), GetMaxCor() and GetMidPos() are used to obtain the minimum correction, maximum correction and the midpoint correction position of the moving objects respectively. The function SENDUPDATE() generates the update message and sends it to the server.

Algorithm 1: The Enhanced Vector-Based Algorithm

```
General Arguments:
$D$;          //GPS positioning error;
$Threshold$ : // system-defined error threshold;
$P_{pre}$;    //predictive location by vector-based algorithm;
1.  Get GPS positioning location $P$;
2.  Calculate minimum distance $L_{\min}$ form $P$ to the road segments at the intersection;
3.  IF ( $L_{\min} > D$ ) THEN
4.    store $P_{pre}$;
5.  ELSE select the right road segment;
6.      $P_{\min}$ = GetMinCor(); //minimum correction;
7.      $P_{\max}$ = GetMaxCor();//maximum correction
8.      $P_{mid}$ = GetMidPos( $P_{\min}, P_{\max}$ );//get midpoint;
9.      IF( $Dis(P_{mid}, P) > Threshold$ ) THEN
10.       SENDUPDATE();
11.     ELSE CONTINUE;
12.     ENDIF;
13. ENDIF;
```

V. Experimental Evaluation

To evaluate the performance of our correction algorithms, we have implemented an experimental system and conducted a series of experiments. The system runs in a AMD Athlon (tm) 64 X2 Dual core Processor 5200+(2G RAM, 2.71GHz).

A. Experiment Setup

In the experiments, we use the Oldenburg data set for the road network which is available from [11]. The simulator developed by Brinkhoff [10, 11] is used to carry out our experiments. The moving clients generated by the simulator are randomly distributed on the road network. The simulator works following the system-defined timestamps and during every timestamp, a tuple of timestamp, location, speed etc is recorded. We limit all moving objects in a certain region (called "experiment region") with the size 5000×5000 during their movements as shown in Fig.10.

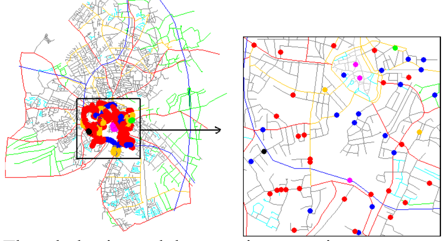


Figure 10. The whole city and the experiment region

B. *Number of Updates*

To verify the effectiveness of the algorithms we provide, update times issued by various location update algorithms mentioned above are compared; system-defined thresholds range from 10 to 100. Timestamps used in this experiment are set to 100. The experimental result is shown in Fig. 11.

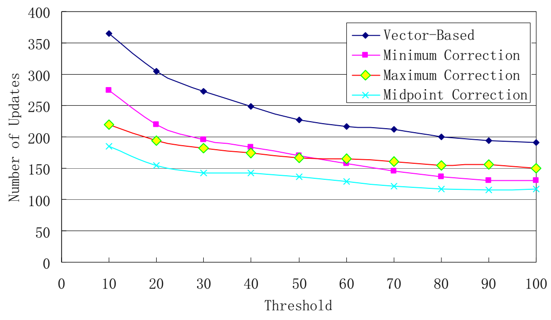


Figure 11. Comparison of different update algorithms

In Fig. 11, different thresholds are on the x axis. The y axis gives the number of updates sent from clients to the server in order to maintain the acquired accuracy. It is seen that all of our proposed algorithms in two steps performs better than vector-based strategy. Furthermore, when the threshold exceeds 50, the maximum correction is inferior to the minimum correction and the midpoint correcting algorithm is best of all. The smaller the threshold is, the better our proposed algorithms perform, for example, when the threshold is 10, the midpoint correction algorithm only needs half number of updates compared to vector-based strategy.

C. *Error Introduced by Different Algorithms*

We will see the difference between real trajectories of moving clients and trajectories resulting from different update algorithms, this can help us find the algorithm which can best record the trajectories of moving clients on the server side. We use the average error δ as the criterion to distinguish different algorithms, which is defined as follows:

$$\delta = \frac{1}{N\mu}\sum_{n=1}^{N}\sum_{t=1}^{\mu} Dis(P_{n(t)}, P_{n}'(t)) \qquad (9)$$

where N is the number of moving clients, μ is the timestamps each client moves, $P_{n(t)}$ is the actual location of moving client n at timestamp t, $P_{n}'(t)$ is the time-synchronous location recorded at the server side. So δ is the average error introduced by recording every position of all the moving clients at the server side.

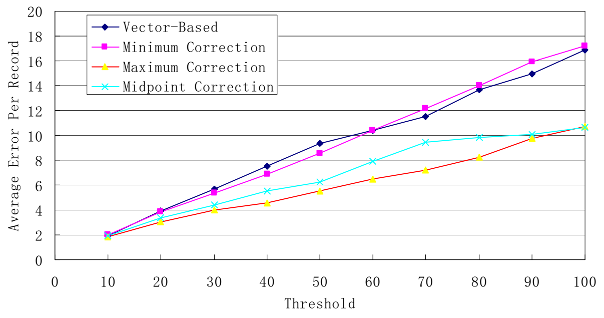


Figure 12. Comparison of different algorithms about average error

From Fig. 12 we can see that average error increases with the threshold, midpoint correction and maximum correction perform better than the vector-based and minimum correction, our update algorithm can approximate the actual trajectory of moving objects while generating less update messages.

D. *Scalability*

Fig. 13 shows the scalability with respect to the density of the moving clients. Four algorithms mentioned above are compared with the same number of moving clients. The y axis represents the ratio of update times issued by three different correction algorithms and the vector-based strategy, the unitary processing is carried out on the number of the updates sent by vector-based method. With a certain number of moving clients, the number of updates generated by different algorithms is the average value of updates with ten different thresholds as mentioned in Fig. 11. From Fig. 13, we can see that minimum correction always performs better that maximum correction with regard to four different numbers of moving objects. The ratio of updates issued by midpoint correction which is between 0.75 and 0.84 do not vary significantly, the average value is about 0.8. So the correction algorithm (i.e. midpoint correction algorithm) is not sensitive to the changes in the number of moving clients.

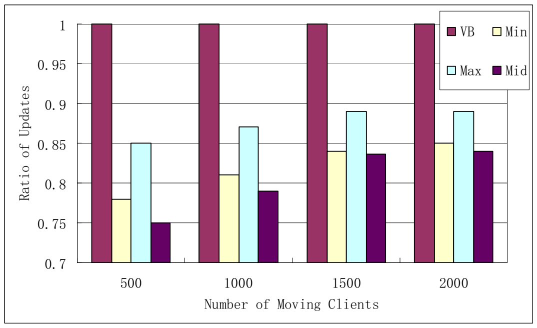


Figure 13. Updates with different number of moving clients

E. *Stability*

We take the stability of the algorithm into account from two aspects: average speed and the running timestamps of the moving clients.

In real life, the speed of moving clients may vary wildly, so in this experiment, we select four different average speed of moving clients. As each client in the road network has a random speed, so the average speed on x coordinate isn't evenly distributed. The meaning of y coordinate is the same as mentioned in Fig. 13.

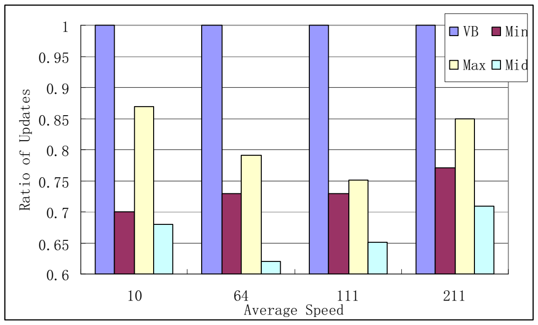


Figure 14. Updates with different average speed

As shown in Fig. 14, we can see that minimum correction always performs better that maximum correction with regard to four different average speeds. The ratio of updates sent by midpoint correction algorithm is from 0.62 to 0.71, the average value of this ratio is about 0.66, it means that under different average speed of moving clients, the midpoint correction algorithm can reduce about 34% of the updates issued by vector-based strategy.

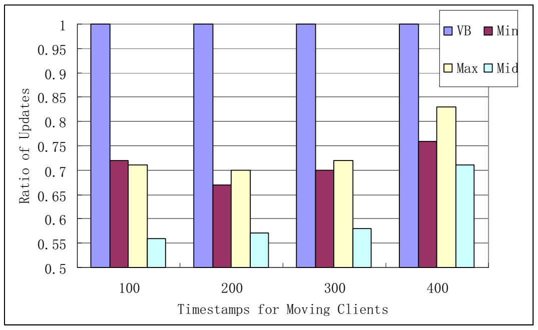


Figure 15. Updates with different timestamps for moving clients

Fig. 15 shows the comparison of four different algorithms on the condition that the clients move for different timestamps. As the timestamps increase, the structure of the road network each client traverses may become more complex. The ratio of updates issued by midpoint correction algorithm is from 0.56 to 0.71 which doesn't vary tempestuously. The average value of the ratio is about 0.61, it means that the number of the updates sent by vector-based algorithm can be decreased by 39%.

VI. CONCLUSION

With the rapid escalation of location based applications, services and the growing demand of being informed at all times, the problem of tremendous amounts of data resulting from the location updates of the mobile clients, if not addressed, will become a performance bottleneck for the mobile commerce and mobile service industry. Location update of mobile clients is a fundamental capability in mobile commerce and all types of LBS today. This paper proposes the correction of vector-based algorithm based on the road network. The correction is divided into two steps: erect correction and horizontal correction. The main idea of the two steps is to try to make the corrected position be close to the real location of the moving client so as to achieve the purpose of decreasing the number of updates while ensuring acquired accuracy. The experimental results show that the number of updates has a significant reduction while having lower error and our proposed algorithm is scalability and stability which can strongly support the proposition.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61003195.

REFERENCES

[1] Su Chen, Beng-Chin Ooi and Zhen-Jie Zhang, "An adaptive updating protocol for reducing moving object database workload", proceedings of the VLDB Endowment, vol. 3, pp. 735-746, 2010.

[2] A. Civilis, C. S. Jensen, and S. Pakalnis, "Techniques for efficient road-network-based tracking of moving objects," Proc. IEEE TKDE, vol. 17, no. 5, pp. 698–712, 2005.

[3] Skyhook Wireless, "Hybrid Positioning System (XPS)," http://www.skyhookwireless.com/howitworks/.

[4] Canalys. Worldwide Mobile Navigation Device Market More Than Doubles. Technical report, Canalys Research Release, 2007.

[5] Canalys. North America Overtakes EMEA as Largest Satellite Navigation Market. Technical report, Canalys Research Release, 2009.

[6] A. Bar-Noy and I. Kessler, "Mobile users: To update or not to update?" 1994.

[7] ROADTRACK: Scaling Location Updates for Mobile Clients on Road Networks with Query Awareness

[8] Shu Wang, Jungwon Min and Byung K. Yi, "Location Based Services for Mobiles: Technologies and Standards", IEEE International Conference on Communication (ICC) 2008, Beijing, China.

[9] A. Civilis, C.S. Jensen, J. Nenortaite, and S. Pakalnis, "Efficient Tracking of Moving Objects with Precision Guarantees," Proc. Int'l Conf. Mobile and Ubiquitous Systems: Networking and Services, pp. 164-173, 2004, extended version available as DB-TR-5, Dept. of Computer Science,AalborgUniv.,Denmark,http://www.cs.aau.dk/DBTR/DBPublications/DBTR-5.pdf.

[10] THOMAS B. Generating network-Based moving objects[C] //Proceedings of International Conference on Scientific and Statistical Database Management, Berlin:IEEE, 2000:253-256.

[11] Brinkhoff T. "Network-based Generator of Moving Objects" http://www.fhoow.de/institute/iapg/personen/brinkhoff/generator