E^3TP: A Novel Trajectory Prediction Algorithm in Moving Objects Databases*

Teng Long[1], Shaojie Qiao[2,1,3,**], Changjie Tang[1], Liangxu Liu[4,3], Taiyong Li[1, 5], and Jiang Wu[5]

[1] School of Computer Science, Sichuan University, Chengdu 610065, China
[2] School of Information Science and Technology, Southwest JiaoTong University, Chengdu 610031, China
qiaoshaojie@gmail.com
[3] School of Computing, National University of Singapore, Singapore, 117590, Singapore
[4] School of Electronic and Information Engineering, Ningbo University of Technology, Ningbo, 315016, China
[5] School of Economic Information Engineering, Southwest University of Finance and Economics, Chengdu 610074, China

Abstract. Prediction of uncertain trajectories in moving objects databases has recently become a new paradigm for tracking wireless and mobile devices in an accurate and efficient manner, and is critical in law enforcement applications such as criminal tracking analysis. However, existing approaches for prediction in spatio-temporal databases focus on either mining frequent sequential patterns at a certain geographical position, or constructing kinematical models to approximate real-world routes. The former overlooks the fact that movement patterns of objects are most likely to be local, and constrained in some certain region, while the later fails to take into consideration some important factors, e.g., population distribution, and the structure of traffic networks. To cope with those problems, we propose a general trajectory prediction algorithm called E^3TP (an **E**ffective, **E**fficient, and **E**asy **T**rajectory **P**rediction algorithm), which contains four main phases: (*i*) mining "hotspot" regions from moving objects databases; (*ii*) discovering frequent sequential routes in hotspot areas; (*iii*) computing the speed of a variety of moving objects; and (*iv*) predicting the dynamic motion behaviors of objects. Experimental results demonstrate that E^3TP is an efficient and effective algorithm for trajectory prediction, and the prediction accuracy is about 30% higher than the naive approach. In addition, it is easy-to-use in real-world scenarios.

Keywords: trajectory prediction, moving objects databases, criminal tracking analysis, hotspot regions, frequent sequential routes.

---

* This work is supported by the National Natural Science Foundation of China under Grant No. 60773169, the 11th Five Years Key Programs for Science and Technology Development of China under Grant No. 2006BAI05A01, the Youth Software Innovation Project of Sichuan Province under Grant No. 2007AA0032 and 2007AA0028.

** Corresponding author.

H. Chen et al. (Eds.): PAISI 2009, LNCS 5477, pp. 76–88, 2009.

1 Introduction

With the rapid development in the wireless and mobile techniques, law enforcement agencies are provided with a large volume of trajectory data in terms of "moving objects" in a crime database [1], such as the movement of vehicles and criminals. From these data, we can easily obtain the instant information of these objects, i.e., the current location and moving direction. Such information can be very helpful for law enforcement agencies in various applications such as criminal location analysis and border safety control. In general, there exist no apparent rules of moving objects that are stored and never used for research purpose. However, these data contain a lot of useful movement behaviors [2]. So, how to discover useful information from moving objects databases relevant to crime cases and use them to predict trajectories of fleeing criminals has become a hot research topic.

It is important to accurately predict the current position of a moving object at a given time instant. For instance, it can help law enforcement agencies accurately track criminals who have committed burglaries during a particular time interval. There exist several characteristics in terms of trajectory data. Firstly, the trajectories are often constrained by streets and highways, so it is impossible for them to move dynamically like hurricanes [3]. Secondly, the speed of objects is mainly determined by the current road condition and other natural factors (e.g., the weather). Thirdly, there are a variety of "frequent routes" which are used more frequently than others.

Recently, many manufactories equip their mobile products with GPS devices, which increases the demand of developing new trajectory prediction technologies [2]. Particularly, law enforcement agents can use such technologies to trace criminals. In addition, it can be applied to handling traffic planning problems. These requirements motivate us to develop an effective, efficient and easy trajectory prediction approach.

However, it is difficult to accurately and efficiently predict trajectories of moving objects due to the following reasons. Firstly, the position information is periodically delivered to the central databases, and the periodicity of position acknowledgement is apt to be affected by several factors, e.g., signal congestions. Whenever the position of an object is unknown, an effective trajectory prediction approach is necessarily required [2]. Secondly, the data of moving objects cannot precisely depict its real location due to continuous motions or network delays [4].

The original contributions of this paper include:

- We proposed to discover hotspot regions where frequent trajectory patterns clustered by processing historical trajectory data of moving objects.
- We proposed an FP-growth [5] based approach to discover frequent routes of moving objects within a certain range.
- We performed extensive experiments to demonstrate the effectiveness and efficiency of our purposed algorithm by comparing it with the naive approach.

2 Problem Statement

The 3D cylindrical model [4, 6] of trajectories is defined beyond temporally annotated sequences (*TAS*) [7, 8] which extended the sequential patterns by adding temporal information of locations. Formally, the definition is presented as follows.

Definition 1 (*Trajectory*). A trajectory of objects is composed of a series of triples:

$$S=\{(x_1, y_1, t_1), \ldots, (x_i, y_i, t_i), \ldots, (x_n, y_n, t_n)\} \quad (1)$$

where t_i represents time-stamps, $\forall i \in [1, n-1]$, $t_i<t_{i+1}$. (x_i, y_i) represents 2D coordinates.

However, it is difficult to accurately locate moving objects in real-life scenarios. So we use a disk area to replace the $\{xy\}$-coordinate in Equation 1. In Fig. 1, T_1 and T_2 are treated as the same trajectory if the following equation holds.

$$(x_i - x_i')^2 + (y_i - y_i')^2 \leq r^2 \quad (2)$$

where $i \in \{1,2,\ldots, n\}$, and r is the radius of the disk.

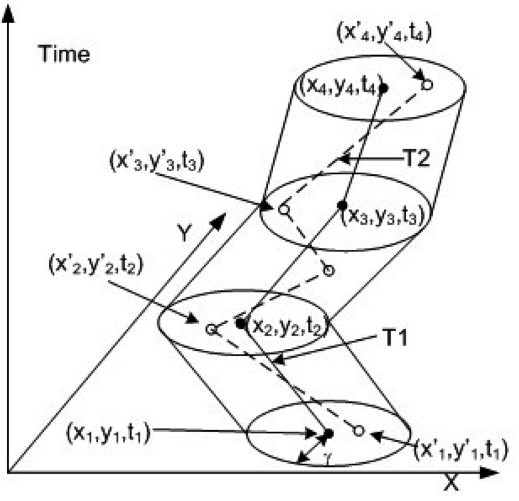


Fig. 1. The 3D cylindrical model of trajectories

When a trajectory is given, we can perform trajectory prediction. Generally, there are two kinds of information contained in a trajectory, i.e., spatial information and temporal information. The former is displayed by coordinates or disks, and the later is related to time-stamps. Our prediction algorithm takes into account these two aspects.

As mentioned in Section 1, the movement of objects is embraced by streets or highways, in other words, it is restricted within a map [9]. Once the geographical information of items (i.e, streets, and buildings) in a map is given, the position of a moving object can be predicted by exploring the streets and highways that can be visited in the future. Here, we firstly give two definitions.

Definition 2 (*Map*). A map is composed of streets, and each street is depicted by the following attributes:

- ID: the identifier number of a street;
- Str: a polygonal line between two ending points, which consists of several line-segments.

Figure 2 is an example of a map, where the number beside each street is the street ID, and each street is denoted as the path between two capital letters.

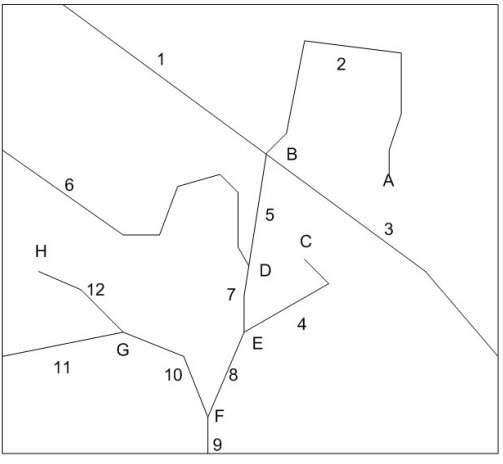


Fig. 2. Example of a map

Definition 3 (*Route*). A route is a sequence of streets:

$$R = \{s_1, s_2, \ldots, s_i, \ldots, s_n\} \tag{3}$$

where $i \in [1, n]$, and s_i has at least one common ending point with s_{i+1}.

For example, in Fig. 2, a route can be indicated as {11, 10, 8, 7, 5}, which means some object can move from street 11 to 5 by way of street 10, 8, 7.

The goal of trajectory prediction is to obtain trajectories as defined in Equation 1, so we should find the possible routes of objects and extract its temporal information as well. When a moving object arrives at a crossing road, it needs to choose which street to go next. Basically, the decision is made on where the object comes from, and where it is going. In this model, there exist some hidden rules in this map, i.e., traffic rules, population distributions. These rules can be mined from historical traffic data.

As for timestamps, they can be calculated as we know the approximate speed of moving objects. The speed is determined by several factors that can be divided in to two categories: external factors (i.e., weather, road condition) and internal factors. We should consider both categories of factors when designing prediction methods.

In addition, we should take into account another question as "*do the previous mentioned rules exist everywhere in a map*?" It is straight-forward that regions with heavy traffic most likely contain internal rules. So, in our algorithm, we firstly find hotspot areas in a map, and perform predicting in those hotspot areas.

In general, there are four phases to predict trajectories of moving objects:

1) Mining hotspot regions in a map from moving objects databases;
2) Discovering frequent routes from hotspot areas;
3) Computing the speed of a variety of moving objects in distinct streets;
4) Predicting the dynamic motion behaviors of moving objects.

3 Mining Hotspot Regions

Mining hotspot regions is an essential step for trajectory prediction, because the rules of choosing routes exist in those regions rather than the ones that are rarely visited. Historical traffic data in terms of these regions are processed in order to find rules. In addition, frequent routes are discovered in those regions. In summary, moving objects in hotspot areas are more predictable than the ones outside them.

Hotspot region mining is treated as a preprocessing phase in our approach. By calculating the visiting frequency of each street, we can find such streets that are frequently visited, i.e., they have higher visiting frequencies than a given threshold. Those connected streets are grouped into a hotspot region. Then, all historical data are scanned again, and segments of trajectories in those hotspot regions are stored.

The details of mining hotspot regions from trajectory data are shown in Algorithm 1. In Algorithm 1, we firstly calculate the visiting frequency of each street in E, and sort them in descending order (lines 1-3). Then, we compute the specified frequency threshold f (line 4), and delete such streets whose frequency is lower than f (lines 5-7). Next, the group of remaining streets forms a hotspot region (lines 9-12). After that, we iteratively scan each hotspot region, if two hotspot regions are connected, combine them into one region until no more hotspot regions can be amalgamated (lines 13-17). Finally, output all hotspot regions (line 18).

In Algorithm 1, p is the percentage of frequently visited streets, the Sort(·) function is used to sort all streets by frequency in descending order. The work mechanism of lines 9-18 is grouping all of the connected high-frequent streets.

Algorithm 1. Mining hotspot regions

Input: a trajectory data set $\boldsymbol{D}$, a set of streets $\boldsymbol{E}$, a threshold $\boldsymbol{p}$
Output: a set of hotspot region $O = \{O_1, \ldots, O_{num}\}$

```
1.  for each street e∈E do
2.      e.freq←ComputeFreq(e);
3.  Sort(E);
4.  f←E.size*p;
5.  for each street e∈E do
6.      if e.freq<f then
7.          delete e;
8.  ID←0;
9.  for each street e∈E do
10.     create a new hotspot region O_ID;
11.     O.add(O_ID);
12.     ID++;
13. while (O.size changes)
14.     for each O_i, O_j in O do
15.         if (O_i and O_j are connected) then
16.             O_i.add(O_j);
17.             delete(O_j);
18. output O;
```

After constructing the hotspot regions, we process historical trajectories by clearing out trajectory segments that are outside hotspot regions, while storing the segments in hotspot regions. In the following sections, we will introduce the approach of discovering moving rules from hotspot regions.

4 Mining Frequent Routes Based on FP-Tree

The main idea of our prediction approach is to discover moving rules of moving objects in those hotspot regions and use these rules to predict possible trajectories.

The FP-growth algorithm [5] is an efficient and scalable frequent itemset mining method. When treating each street in a hotspot region as an item, these items can be organized as a sequence by connecting their ending points. So, the problem of discovering frequent routes is equivalent to mining frequent trajectory patterns that are composed of streets in hotspot regions.

We modify FP-growth approach to suit the trajectory prediction problem, and we name the new method FR_mining (Frequent Route Mining), which includes FP-tree construction and mining phases. The FP-tree construction step is analogous to that applied in FP-growth, and is given in Algorithm 2.

Algorithm 2. FR_mining

Function 1: FP-tree construction

Input: *R*, a hotspot region contains a set of streets and a set of trajectory segments; ***min_sup***, the minimum support count

1. **for** each street s_i in R **do**
2. count $s_i.freq$;
3. **if** ($s_i.freq$<*min_sup*) **then**
4. delete s_i;
5. sort R by *freq* in descending order as L;
6. create a root node t of the FP-tree and label it as "*null*";
7. **for** each trajectory segment S in R **do**
8. select and sort streets in S and put them into S' by the order of nodes in L, $S'=[s|S^*]$, where s is the first street and S^* is the remaining nodes in L;
9. **insert_tree**($[s|S^*]$, t);
10. output t, L;

Procedure insert_tree($[s|S^*]$, t)

1. **if** (t has a child c **and** $c.sid$=$s.sid$)
2. $c.freq$←$c.freq$+1;
3. **else**
4. create a new node n;
5. $n.freq$←1;
6. $n.parent$←t;
7. find n' in L, where $n'.sid=n.sid$;
8. **if** $n'.next$=*null* **then**
9. $n'.next$←n;
10. **else**
11. **do**
12. n'←$n'.next$;
13. **while** ($n'.next$≠*null*)
14. $n'.next$←n;
15. **if** (S^* is not empty) **then**
16. **insert_tree**($[S^*]$, n);

Function 2: Mining frequent routes from FP-tree

Input: *t*, the root node of an FP-tree; ***L***, a list of streets in a hotspot region corresponding to t.
Output: a set of frequent routes in a hotspot region R.
Method:

1. Create a set of frequent routes R;

Procedure FP_mining(t, α)

1. **if** (t contains a single path P) **then**
2. generate a pattern $p=P\cup\alpha$ with *support_count* =the minimum *freq* of nodes in P;
3. $R.add(p)$;
4. **else**

2. **FP_mining**(*t*, *null*);	**5.** **for** each a_i in *L* **do**
3. Reorganize frequent routes in *R*;	**6.** generate pattern $\beta=a_i \cup \alpha$ with *support_count* = the minimum *freq* of nodes in β;
	7. construct β's conditional pattern base and conditional FP-tree *T';*
	8. **if** ($T' \neq \varnothing$) **then**
	9. FP_mining(T', β);
	10. **if** (R remains unchanged) **then**
	11. *R*.add(β);

In Algorithm 2, we firstly scan all streets in *R* to calculate how frequently they have been visited based on trajectory segments. If the frequency of a street is lower than *min_sup*, remove this street from the hotspot region (lines 1-4). Then, sort the remaining streets by frequency in descending order, save nodes' order in a head list *L* (line 5). Next, create a root node *t* of the FP-tree, label it as "*null*" (line 6), and insert all trajectory segments in *R* into *t* by calling the procedure of "**insert_tree**(·)" (lines 7-9). Finally, we output *t* and *L* (line 10).

When mining a FP-tree, however, the process is distinct from the FP-growth mining approach, as presented in Function 2. Basically, our objective is to find the longest frequent route whose *support_count* is higher than the given threshold. The difference lies in Step 2 and Step 6 of the procedure FP_mining(·), it simply generates the largest frequent pattern without obtaining their combinations. The purpose of Step 3 in Function 2 is to reorganize streets in each frequent pattern into a route.

By processing each hotspot region using Algorithm 2, we can obtain a set of frequent routes for hotspot regions. Given a previous moving route of an object, we can compare it with discovered frequent routes and find the most matching one, and then use this frequent route to predict trajectories of this object.

5 Time-Stamped Trajectory Prediction

In this section, we present and analyze the work mechanism of E^{3}TP, especially for the approach of extracting temporal information from trajectory data and obtaining timestamps of frequent routes. The E^{3}TP algorithm is detailed in Algorithm 3.

Algorithm 3. E^{3}TP- Effective&Efficient&Easy Trajectory Prediction

Input: a historical trajectory data set ***D***, the set of all streets ***E*** in a map ***m***, hotspot mining threshold ***p***, the minimum support count ***s*** for FR_mining, and the previous trajectory ***t*** of an aiming object
Output: a time-stamped trajectory ***t'*** of ***o***
1. Hotspot_mining(*D, E, p*);
2. FR_mining(*m, s*);
3. SpeedCalculation(*D, E*);
4. TrajPredict(*t, m, E*);

In Algorithm 3, Steps 1 and 3 are proposed to discover the movement rules from a certain map based on historical data. Steps 1 and 2 have already been introduced in Section 3 and Section 4, respectively. In Step 3, we obtain the possible speed for moving objects in each street by calculating their average speed. Ordinarily, an object moves at a constant speed, so we use the average speed to approximate it. Here, the category of moving objects represents its internal characters, and the *ID* of a street indicates its individual features, such as road condition and traffics. The obtained rules can be quickly retrieved when needed.

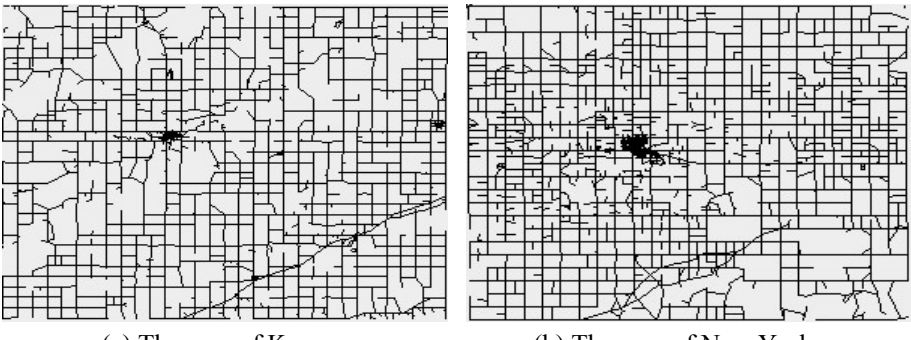
For predicting trajectories of moving objects, we firstly check whether an object is currently in a hotspot region. If so, we extract its previous trajectory, transform it into a route, and compare it with existing frequent routes in the hotspot region in order to find the most matched one based on Equation 2. Then, we employ kinematical formulations [10] to compute time intervals of visiting a street, and obtain the accurate position of objects at given timestamps. In this phase, we suppose that an object moves in a uniform speed, with the speed obtained by Step 3. Finally, we output timestamps and locations of the object in the form as given in Equation 1.

6 Experiments and Discussions

6.1 Experimental Setting and Definition

In this section, we perform experiments by comparing E^{3}TP with the naïve prediction method (called Naïve for short). In general, Naïve does not consider the hotspot mining and frequent pattern mining phases to predict possible routes. It only calculates the frequency of visiting each street and chooses the most frequently visited street to go when it arrives at a crossroad. Both algorithms were implemented in Java and the experiments were conducted on an AMD Athlon X2 5000+, 2.6GHz CPU with 2GB of main memory, running on Windows XP professional system.

All experiments were running on two distinct data sets generated by Brinkhoff's network-based generator [9]. They were generated by the network-based spatio-temporal data generating approach [9]. The moving objects are represented by its $\{xy\}$-coordinate in a real-world map. These two maps are shown in Fig. 3.



(a) The map of Kansas (b) The map of New York

Fig. 3. Maps for experiments

- A part area of Kansas State (denoted as Kansas).
- A part area of New York State (denoted as NY).

The parameter r in Equation 2 plays an essential role in determining whether two trajectories are equivalent. It has a great impact on the experimental results. Another important parameter is t, which decides the visiting time interval of an object's trajectory. In order to tune these two parameters, we gradually increase the value of r in a specified t value, aiming to find a proper r value.

In this section, we use *accuracy* defined in Equation 4 to estimate the effectiveness of prediction algorithms, and it is given as follows.

$$accuracy(N) = \frac{n_{hit}}{N} \tag{4}$$

In Equation 4, n_{hit} represents the number of predicted trajectories which match the real-world situation, and N is the number of all predicted trajectories.

6.2 Parameter Settings

As introduced in the previous sections, there are four important parameters that need to be specified first, including p (introduced in Algorithm 1), *min_sup* (see Algorithm 2), r (as shown in Equation 2) and t (the time interval during which prediction accuracy is evaluated). Due to the differences between these two data sets, these parameters should be specified separately. The properties and parameter settings of these two data sets are presented in Table 1**.**

Table 1. Properties of data sets and parameter settings

Parameter	*Kansas*	*NY*
Map width (pixel)	437,998	563,287
Map height (pixel)	348,693	435,186
Number of moving objects	8,000	8,000
p	0.2	0.2
min_sup	28	32
r (pixel)	2,000	2,000
Time interval t (time unit)	15	15

Table 1 indicates that each data set contains 8,000 moving objects. We use the historical data of 4,000 objects to build movement rules for each map, and the other 4,000 objects are employed to estimate the accuracy of our proposed algorithms. By trade-off, p is set to 0.2 for both data sets. When p equals 0.2, the minimum values of *min_sup* for both data sets are determined by f, because it is the minimum visiting frequency in terms of all streets in hotspot regions. We choose to use this minimum value in order to perform more accurate predictions. The setting of r depends on the scale of both maps (it is set 2,000 pixel based on these two maps), and t is set to 15 time units, which is large enough to compare both algorithms, i.e., Naïve and E^3TP.

6.3 Performance Analysis of Movement Rules Extraction

This section analyzes the time trend and memory cost in terms of hotspots region construction and frequent pattern mining phases as the number of trajectories grows.

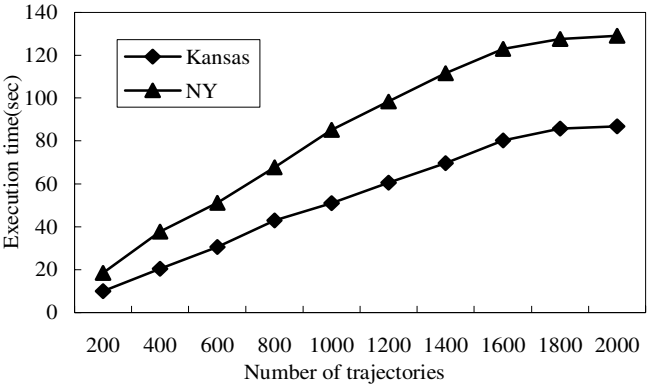


Fig. 4. Runtime comparison of hotspots region construction and frequent routes mining

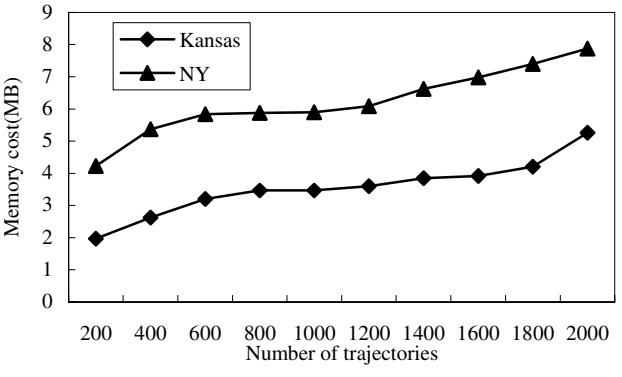


Fig. 5. Memory cost of hotspots region construction and frequent routes mining phases

Fig. 4-5 show the time performance comparison and memory cost of movement rules extraction and frequent pattern mining between Naïve and E^{3}TP, respectively, where the *x* axis is the number of trajectories, and the *y* axes represent the runtime and memory cost, respectively. According to Fig. 4, the execution time of hotspots region construction and frequent routes mining phases increases in an approximate linear manner with the number of trajectories growing. The reason is that most prediction time lies in trajectory processing and route searching, and the time complex approximates $O(n)$. As for Fig. 5, the memory cost does not increase drastically as the number of trajectories increases. Because redundant trajectory information is cut off when transforming historical trajectory data into frequent routes.

6.4 Prediction Accuracy Estimation

Prediction accuracy is used to estimate the performance of both Naive and E^{3}TP.

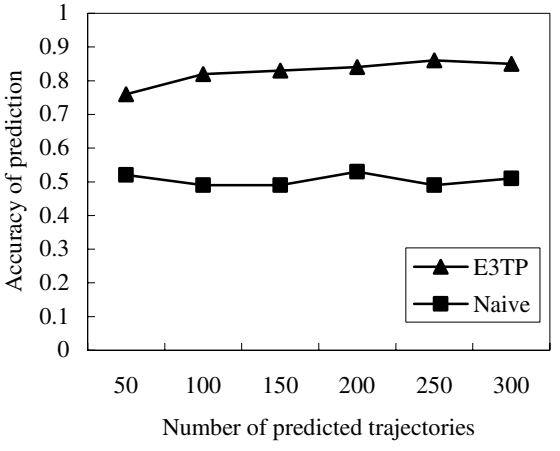


Fig. 6. Prediction accuracy comparison between E^{3}TP and Naïve on Kansas data set

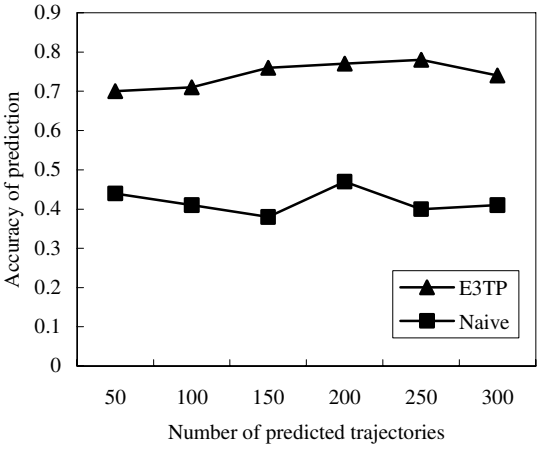


Fig. 7. Prediction accuracy comparison between E^{3}TP and Naïve on NY data set

In Fig. 6 and Fig. 7, the *x* axis represents the number of predicted trajectories, and the *y* axis is the percentage of trajectories that "hit" (match) the real-world situation by setting parameters as introduced in Section 6.2. By Fig. 6 and Fig. 7, E^{3}TP outperforms the naive method with a big gap of 32% on Kansas data set and 30% on NY data set, respectively. This is because E^{3}TP provides a better solution of finding possible routes of moving objects with high- frequent patterns in a certain region. In addition, we predict the motion curves of moving objects by taking into account both the road condition and the internal characteristics of moving objects, which helps improve the accuracy of calculating timestamps.

7 Conclusion and Further Work

Traditional trajectory prediction algorithms mainly focus on discovering frequent movement patterns of moving objects without constraints, which is far from the real-world situations. In this paper, we addressed the characteristics of motion behaviors of moving objects, including the key factors in route selection and the distribution rules of frequent trajectory patterns. By experiments, we compare the proposed trajectory prediction algorithm based on FP-tree, called E^{3}TP, with probability-based naïve method, and show the advantages of our solution. Most importantly, E^{3}TP can be used to prediction the fleeing criminals in order to help security agencies trace criminal suspects.

For further research, we will improve the proposed frequent routes discovery algorithm to be more suitable and effective for mining large scale of spatial and temporal data, and develop other data mining approaches, i.e., Genetic Algorithms [11, 12], neural networks [13], immune algorithms [14], to accurately find the most possible route. In addition, stochastic theories can be utilized to depict the distribution rules of moving objects' speed. Finally, trajectory prediction of moving objects in non-hotspot region will be seriously considered in our future study.

References

[1] Qiao, S., Tang, C., Jin, H., Dai, S., Chen, X.: Constrained K-Closest Pairs Query Processing Based on Growing Window in Crime Databases. In: 2008 IEEE International Conference on Intelligence and Security Informatics, ISI 2008, Taipei, pp. 58–63 (2008)

[2] Morzy, M.: Mining frequent trajectories of moving objects for location prediction. In: Perner, P. (ed.) MLDM 2007. LNCS, vol. 4571, pp. 667–680. Springer, Heidelberg (2007)

[3] Lee, J., Han, J., Whang, K.: Trajectory Clustering: A Partition-and-Group Framework. In: SIGMOD 2007, Beijing, China, pp. 593–604. ACM, New York (2007)

[4] Trajcevski, G., Wolfson, O., Zhang, F., Chamberlain, S.: The geometry of uncertainty in moving objects databases. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Bertino, E., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 233–250. Springer, Heidelberg (2002)

[5] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD 2000: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 1–12. ACM, New York (2000)

[6] Trajcevski, G., Wolfson, O., Hinrichs, K., Chamberlain, S.: Managing uncertainty in moving objects databases. ACM Trans. Database Syst. 29(3), 463–507 (2004)

[7] Giannotti, F., Nanni, M., Pedreschi, D.: Efficient mining of temporally annotated sequences. In: SDM 2006: Proceedings of the 6th SIAM International Conference on Data Mining, pp. 346–357. SIAM, Bethesda (2006)

[8] Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F.: Mining sequences with temporal annotations. In: SAC 2006: Proceedings of the 2006 ACM symposium on Applied computing, pp. 593–597. ACM, New York (2006)

[9] Brinkhoff, T.: A framework for generating network-based moving objects. Geoinformatica 6(2), 153–180 (2002)

[10] Halliday, D., Resnick, R., Walker, J.: Fundamentals of Physics, 8th edn. Wiley, Chichester (2007)
[11] Qiao, S., Tang, C., Peng, J., Fan, H., Xiang, Y.: VCCM Mining: Mining Virtual Community Core Members Based on Gene Expression Programming. In: Chen, H., Wang, F.-Y., Yang, C.C., Zeng, D., Chau, M., Chang, K. (eds.) WISI 2006. LNCS, vol. 3917, pp. 133–138. Springer, Heidelberg (2006)
[12] Qiao, S., Tang, C., Peng, J., Hu, J., Zhang, H.: BPGEP: Robot Path Planning based on Backtracking Parallel-Chromosome GEP. In: Proceedings of the International Conference on Sensing, Computing and Automation, ICSCA 2006, DCDIS series B: Application and Algorithm, vol. 13(e), pp. 439–444. Watam Press (2006)
[13] Qiao, S., Tang, C., Peng, J., Yu, Z., Jiang, Y., Han, N.: A Novel Prescription Function Reduction Algorithm based on Neural Network. In: Proceedings of the International Conference on Sensing, Computing and Automation, ICSCA 2006, DCDIS series B: Application and Algorithm, vol. 13(e), pp. 939–944. Watam Press (2006)
[14] Shao-jie, Q., Chang-jie, T., Shu-cheng, D., Chuan, L., Yu, C., Jiang-tao, Q.: SIGA: A novel self-adaptive immune genetic algorithm. Acta Scientiarum Natralium Universitatis Sunyatseni 47(3), 6–9 (2008)