Appl Intell (2010) 33: 370–386
DOI 10.1007/s10489-009-0173-z

PutMode: prediction of uncertain trajectories in moving objects databases

Shaojie Qiao · Changjie Tang · Huidong Jin · Teng Long · Shucheng Dai · Yungchang Ku · Michael Chau

Published online: 13 March 2009

Abstract *Objective*: Prediction of moving objects with uncertain motion patterns is emerging rapidly as a new exciting paradigm and is important for law enforcement applications such as criminal tracking analysis. However, existing algorithms for prediction in spatio-temporal databases focus on discovering frequent trajectory patterns from historical data. Moreover, these methods overlook the effect of some important factors, such as speed and moving direction. This lacks generality as moving objects may follow dynamic motion patterns in real life.

Methods: We propose a framework for predicating uncertain trajectories in moving objects databases. Based on Continuous Time Bayesian Networks (CTBNs), we develop a trajectory prediction algorithm, called PutMode (*P*rediction of *u*ncertain *t*rajectories in *M*oving *o*bjects *d*atabas*e*s). It comprises three phases: (i) construction of TCTBNs (Trajectory CTBNs) which obey the Markov property and consist of states combined by three important variables including street identifier, speed, and direction; (ii) trajectory clustering for clearing up outlying trajectories; (iii) predicting the motion behaviors of moving objects in order to obtain the possible trajectories based on TCTBNs.

Results: Experimental results show that PutMode can predict the possible motion curves of objects in an accurate and efficient manner in distinct trajectory data sets with an average accuracy higher than 80%. Furthermore, we illustrate the crucial role of trajectory clustering, which provides benefits on prediction time as well as prediction accuracy.

Keywords Trajectory prediction · CTBN · Trajectory clustering · Moving objects databases

1 Introduction

Information technologies have been increasingly applied to fight against criminals and terrorists in recent years [1–3]. With the rapid development in the wireless and mobile technology, law enforcement agencies are provided with a large volume of trajectory data of "moving objects" such as the movement of vehicles and criminals. From these data, we can easily obtain the instant information of these objects, such as 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, these data can be stored in the form of trajectories which hide large amounts of valuable knowledge describing their behaviors [4]. A trajectory of a moving object consists of several motions during a specified time interval, and it contains the following characteristics:

S. Qiao (✉)
School of Information Science and Technology, Southwest JiaoTong University, Chengdu 610031, China
e-mail: qiaoshaojie@gmail.com

S. Qiao · C. Tang · T. Long · S. Dai
School of Computer Science, Sichuan University, Chengdu 610065, China

H. Jin
Mathematical and Information Sciences, The Commonwealth Scientific and Industrial Research Organization, North Road, ANU Campus, Canberra, ACT 2601, Australia

Y. Ku
Department of Information Management, Yuan Ze University, 320 Taipei, Taiwan

M. Chau
School of Business, The University of Hong Kong, Pokfulam, Hong Kong

– It is often constrained in a network composed of several streets and buildings and it is not arbitrary like animal movement paths.
– The trajectory inside an electronic map has numerous distribution rules due to several factors, e.g., city planning, traffic planning, population distribution, etc.

It is important to accurately predict the current position of a moving object at a given time instant or in a given area. For instance, it can help law enforcement agencies to accurately track criminals who have committed car theft during a particular time interval. Usually, the position information is periodically delivered to the police center. However, the periodicity of position acknowledgement is apt to be affected by several unavoidable factors, i.e., signal congestions, signal losses due to natural phenomena, the power supply shortage of the mobile device, etc. [4]. Whenever the position of a moving object is unknown, an effective trajectory prediction approach is very useful in various applications.

However, accurate trajectory prediction of moving objects is difficult and challenging. Firstly, uncertainty is inherent to moving objects, and the spatio-temporal databases are utilized to store the positioning information of individual objects [5]. The spatio-temporal data of moving objects cannot precisely depict its real location due to continuous motions or network delays [6]. Secondly, the location prediction mechanism must guarantee to return accurate location of moving objects while not requiring extensive computation. Thirdly, the performance of prediction should not fall drastically as the number of objects explodes. Finally, prediction efficiency is as important as prediction accuracy, since the success of a location-based service depends on whether the service is delivered to a particular object at a particular location and on particular time [4]. Specifically, if the objects (i.e., fleeing criminals) often change speed or directions, the approach of prediction should give a quick response while the objects still reside at a certain location.

In general, the trajectory of a moving object is modelled as a polyline in 3D space, with two dimensions for geography and one for time [7]. The data collected from moving objects accumulate a large amount of useful knowledge that depicts the typical movement rules of objects [4]. In particular, these rules can be used to describe and predict the motion of objects. Existing trajectory prediction approaches [4, 7, 8] have several drawbacks such as: they do not utilize historical data, the cost of calculation is quite high, and they cannot scale up with the number of objects.

In order to predict the location of moving objects, we propose to construct CTBNs [9] based on trajectory data. A CTBN is a probability model that describes the underlying changing rules of these uncertain trajectories. For each candidate trajectory, we use an intensity matrix [10] to compute the state transition probability. More specifically, our proposed algorithm PutMode trains CTBNs offline due to its computation intensity, whereas the possible trajectory can be quickly predicted in an online manner. For example, a crime investigator can resort to it to find the possible path of a fleeing criminal instantaneously. In summary, the original contributions of this paper include the following:

– We propose a trajectory clustering method to filter outlying trajectories and perform prediction on trajectory clusters.
– We use TCTBNs to model uncertain trajectories and obtain the state transition rules of moving objects, and introduce a TCTBN construction approach.
– We propose a novel trajectory prediction algorithm based on TCTBNs, called PutMode, which takes into account the effect of moving speed as well as moving direction.
– We perform experiments to evaluate the efficiency and effectiveness of our proposed algorithms by comparing them with the naive solution in several aspects.

The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 briefly addresses the trajectory prediction problem. Section 4 presents a trajectory clustering approach for clearing up outlying trajectories. Section 5 reviews the TCTBN framework, introduces the background knowledge related to TCTBNs, and presents the construction method for TCTBNs. Section 6 proposes a TCTBN-based trajectory prediction algorithm. Section 7 describes performance studies and discusses experimental results. Finally, Sect. 8 gives the concluding remarks and outlines the directions for future work.

2 Related work

The problem of discovering uncertain trajectories in moving objects databases has recently received increasing attention. Existing work related to trajectory mining mainly focuses on mining frequent trajectory patterns [4, 11, 12] and trajectory queries for moving objects [6–8, 13]. However, most of these works assume that exact trajectory information was available during a certain time interval. Unfortunately, in real life situations, this assumption cannot be guaranteed where the uncertainty and incomplete information are inherent to trajectories in moving objects databases [13].

This study is relevant to two research problems. The first one is managing uncertain trajectories of moving objects. The second one is the prediction of trajectories with time annotations (stamps). Existing work on uncertainty management for moving objects pays attention to modelling uncertain trajectories based on historical object movements. A typical work was done by Mamoulis et al. [14]. They proposed a top-down technique, called STPMine2, for efficiently discovering periodic patterns from historical spatio-temporal data. In addition, the authors proposed an indexing

scheme that adopted discovered patterns to effectively manage the uncertain data.

In order to capture uncertainty of trajectories, Trajcevski et al. [6, 7] model a trajectory as a 3D cylindrical body, which incorporates uncertainty in a manner that enables efficient querying and is especially suitable for objects that are inside the region *sometime* during the time interval, or for the ones that are *always* inside during the time interval. Moreover, Mokhtar and Su represented each coordinate of a trajectory as a stochastic process with a time-parametric uniform distribution [13]. By experiments, they demonstrated that it was reasonable to use a uniform distribution to reason about in query evaluation. In particular, they validated the fact that "the state transitions of moving objects consist of a set of stochastic processes". This motivates us to represent each trajectory as a Markov process [15] whose state transition model is determined by its previous states.

Most of the above approaches for managing uncertainty of moving objects assume that the objects move according to linear functions. This limits their applicability, since the movement of objects is often unknown and may follow drastically changing motion patterns [8]. Aiming to cope with these problems, Tao et al. proposed a client-server architecture for querying spatio-temporal objects. Each client uses a recursive motion function to capture the motions of trajectories, while a server uses STP-tree (spatio-temporal prediction tree) to index the expected trajectories by a polynomial function. Another trajectory prediction approach is Traj-PrefixSpan [4], which combined the PrefixSpan [16] and the FP-tree [17] algorithms to predict unknown trajectories and moving rules of objects. However, the mining of frequent trajectory patterns needs very large amounts of computation, and the generation of indices for moving objects is quite high. This also pushes us to develop an efficient and effective trajectory prediction approach by utilizing the inherent properties of moving objects in order to mine unknown rules from trajectory data.

CTBN [9, 10] was proposed for structured stochastic processes with finite states that evolve over continuous time. It is particularly suitable for depicting uncertain trajectories of moving objects. Specifically, it can be used to approximately infer the successive state probabilities of objects in an instantaneous and effective manner. Therefore, we introduce CTBNs for managing and predicting trajectories of moving objects in this study. The experimental results shows that the average accuracy is higher than 80% which is considered to be a surprisingly good result by the policemen. To the best of our knowledge, there is no relevant work that has been done to predict uncertain trajectories of moving objects based on CTBNs despite its importance.

3 Problem statement

In this section, we present a formal definition of trajectory with temporal annotations and address the problem of predicting uncertain trajectories in moving objects databases.

The goal of trajectory prediction is to predict the possible motion behaviors of moving objects. The possible motion behaviors contain such information as "which street or direction will the moving object choose to go in the future, or whether it is going to accelerate or slow down in the street". Relying on these behaviors, we can finally obtain possible trajectories of moving objects based upon kinematic formulations.

In general, the position of a moving object is represented by Cartesian coordinates. A trajectory of a moving object is treated as a sequence of time-stamped locations, depicting the traces and instant positions collected by wireless devices [12] as formally defined below.

Definition 1 (Trajectory) A trajectory of a moving object is a sequence of triples:

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

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

The trajectory is defined beyond the concept of temporally annotated sequences (*TAS*) [18]. A typical example of *TAS* over the train travel route in China is denoted by latitude and longitude as: (126.7, 45.8, 13:00 PM)→(116.4, 39.9, 0:30 AM)→(104.1, 30.7, 2:30 AM), representing a path that starts from the city of *Harbin*, then after 11.5 hours arrives at *Beijing* and finally, after 26 hours ends with *Chengdu*.

Since it is difficult to locate exactly the positions of moving objects in real-life scenarios, we use the possible motion curves composed of line segments inside a trajectory volume [7] to approximate the trajectories. If an object falls in the disk of a trajectory volume, we use the xy-coordinate of the central point in the corresponding disk to represent its location. The graphical representation of a trajectory volume and the possible motion curve is illustrated in Fig. 1.

For each point (x, y, t) along a trajectory, the uncertain area is a disk with radius r. A possible motion curve P_T of trajectory T can be represented by a continuous function $f_{P_T} : t \to r^2$ such that for any time $t \in [t_1, t_m]$. Assuming a object moves at the constant speed of v_i, the 3D point (f_{P_T}, t) is inside an uncertain area of the expected location at a given time instant t satisfying:

$$(x - (x_i + v_i^x \cdot t))^2 + (y - (y_i + v_i^y \cdot t))^2 \leq r^2 \quad (2)$$

A trajectory can be modelled as Markov processes with a time-parametric uniform distribution. In particular, the location (x_{t_n}, y_{t_n}) of a moving object p at a time instant t obeys

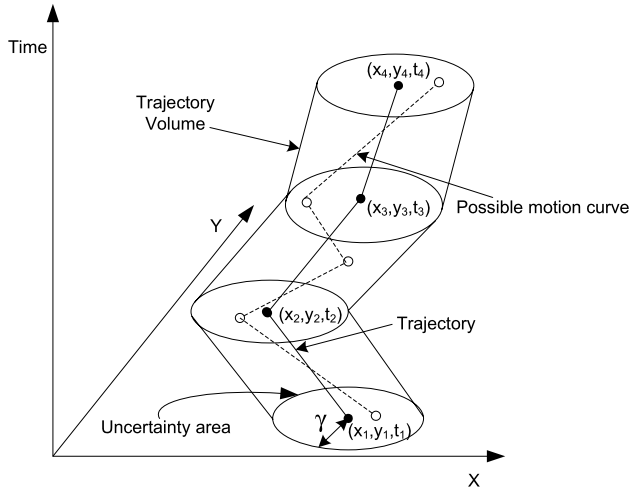
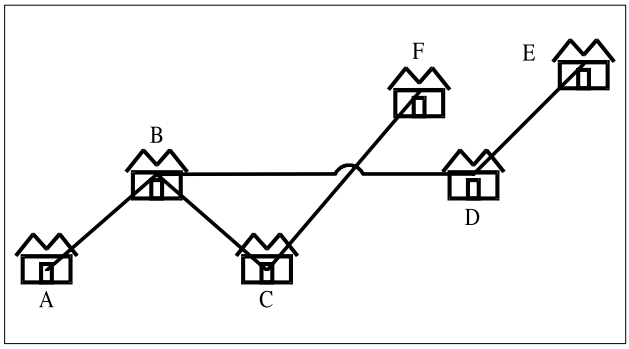


Fig. 1 Trajectory volume and possible motion curve



SId	Polyline	Length
AB	(1,1)(5,6)	6.4 mile
BC	(5,6)(8,1)	5.83 mile
BD	(5,6)(14,6)	9 mile
CF	(8,1)(12,10)	9.85 mile
DE	(14,6)(19,11)	7.07 mile

Fig. 2 Example of an electronic map

the Markov property which states that:

$$Pr\{X(t_{n+1}) = x_{n+1} | X(t_n) = x_n, \ldots, X(t_1) = x_1\} = Pr\{X(t_{n+1}) = x_{n+1} | X(t_n) = x_n\} \quad (3)$$

For any time instant t_i $(i = 1, \ldots, n)$ where $t_j > t_k$ for $j > k$, the above equation still holds for any y_{t_n}. In order to model a trajectory, we define an electronic map as follows.

Definition 2 (Electronic Map) An electronic map consists of streets, that are represented by triplets with the following attributes:

- *SId*: the identifier of a street;
- *Polyline*: a trajectory composed of streets between a starting point and an ending point;
- *Length*: the length of a street.

To illustrate this, consider an example of an electronic map in Fig. 2, which is composed of five streets. Here, three streets intersect at point B, which means there are three candidate streets to follow when residing at B.

Since the tragic event of the attacks from terrorists on the United States on September 11, 2001, both citizens and authorities have realized that knowledge about the terrorists or criminals and how to surveil them will be a key factor of safeguarding world peace [19]. We intend to develop a sub-system, which is a component of our CrimeMiner system and can be used to predict the path of fleeing criminals via mobile or traffic devices equipped with GPS in an accurate and instantaneous manner. The prediction task is to find the most likely next move (i.e., the most possible trajectory). The CrimeMiner system is an intelligent crime data mining system [20] that integrates the following capabilities: (1) classify criminals and predict crime events [21], (2) predict the possible trajectories of fleeing criminals, (3) analyze and visualize the criminal networks [22, 23] and discover the deep hierarchical structure of criminal gangs [24], and (4) discover crime hotspots.

We propose a general client-server framework for predicting uncertain trajectories of moving objects as illustrated in Fig. 3.

The main characteristic of this framework lies in the notion the possible trajectories can be forecasted in an online manner while the motion behaviors of objects are extracted from historical data offline. In terms of our prediction system, the online prediction is very fast and accurate, while the time spent on building the TCTBN with a large number of transition states is much costly than that for trajectory prediction.

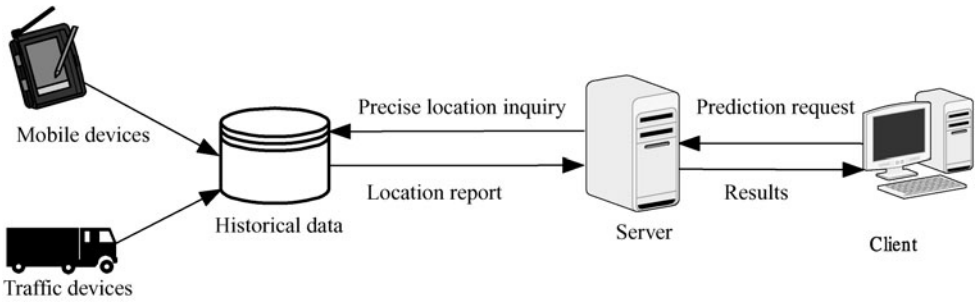
4 Trajectory clustering

Trajectory clustering is regarded as a preprocessing process, i.e., clearing up the outlying (noise) trajectories, in order to save calculation time. It plays an essential role in accurate trajectory prediction due to two reasons. Firstly, the phase of clustering can partition one data set into several subsets, and we only need to predict on a compressed TCTBNs due to state reduction. This can greatly help save time. Secondly, it can be used to filter such trajectories that are rarely visited, which benefits improving prediction accuracy. We use a density-based clustering method analogous to DBSCAN [25].

The key steps in this algorithm are outlined as follows.

1. Initialize a cluster identifier (*cId*) and label all candidate trajectories as 'UNCLASSIFIED' (lines 1–3).

Fig. 3 System framework



Algorithm 1 Trajectory clustering

Input: a trajectory set $\mathbb{T}$, a distance threshold ϵ, a minimum number threshold *minPts*.

Output: a set of trajectory clusters $\mathcal{O} = \{O_1, \ldots, O_{num}\}$.

1. $cId \leftarrow 0$;
2. **for** each $Tr \in \mathbb{T}$ **do**
3. $Tr.cLab \leftarrow$UNCLASSIFIED;
4. **for** each $Tr \in \mathbb{T}$ **do**
5. **if** $Tr.cLab ==$UNCLASSIFIED **then**
6. $\mathcal{S} \leftarrow Tr$.GetNeighbors($\epsilon$);
7. **if** $\mathcal{S}.size() < minPts$ **then**
8. $Tr.cLab \leftarrow$NOISE;
9. continue;
10. **else**
11. **for** each unclassified trajectory $L \in \mathcal{S}$ **do**
12. $L.cLab \leftarrow L.cId$;
13. $\mathcal{S}' \leftarrow L$.GetNeighbors($\epsilon$);
14. **if** $\mathcal{S}'.size() \geq minPts$ **then**
15. **for** each trajectory $L' \in \mathcal{S}'$ **do**
16. **if** $L'.cLab ==$UNCLASSIFIED **then**
17. $\mathcal{S}$.Add(L'); $L'.cLab \leftarrow cId$;
18. **if** $L'.cLab ==$NOISE **then**
19. $L'.cLab \leftarrow cId$;
20. cId++;
21. **for** $i := 0$ to $cId - 1$ **do**
22. output O_i;

2. Scan each trajectory *Tr*, if its label (*cLab*) is marked as 'UNCLASSIFIED' (lines 4–5), then perform the following operations.
   (a) Pick the neighbors whose distance to *Tr* is less than a distance threshold ϵ (lines 5–9).
   (b) In the ϵ-neighborhood, if the number of *Tr*'s neighbors is greater than a minimum number threshold *minPts*, then treat it as a core object.
   (c) Repeat the above two steps on each core object's neighbors until all trajectories are processed (lines 10–20).
3. Output each trajectory cluster O_i (lines 21–22).

If a trajectory is labeled as 'NOISE' before trajectory clusters are returned, it is treated as an outlier and is deleted. Note that we employ the function of GetNeighbors($\cdot$) to obtain *Tr*'s neighbors. In GetNeighbors($\cdot$), we use the starting point, the middle point, and the ending point of a trajectory to approximate the distance between two trajectories. If the geometric mean distance among these points is less than ϵ, these two trajectories are treated as neighborhoods. By our definition, a trajectory is regarded as its own neighbor.

By trajectory clustering, we partition trajectories based on their geographical information. As a consequence, we build a TCTBN for each cluster. Hereafter, a problem is raised as: how to choose appropriate TCTBNs corresponding to a given initial state of an object to perform prediction. A simple method is prediction on the TCTBNs where the initial state occurs.

5 Trajectory continuous time Bayesian networks

5.1 Motivation

Our solution is inspired by the following two observations.

1. There exist several unknown rules behind historical trajectory data and they can be applied in a wide variety of applications. A typical example is path planning. When a car moves along a street, the probabilities of selecting the next street are not similar in most cases. In practice, we can obtain the probabilities by analyzing historical data. For instance, assume that 300 objects have visited street A. Out of these 300 objects, 150 choose B as the next street to visit, 100 choose C, and 50 choose D. Then, it is easy for us to compute that the probability for objects to 'jump' from A to B is $1/2$, to C is $1/3$, and to D is $1/6$, respectively. So, it is suitable to use a Bayesian network to model an uncertain trajectory in which the transition intensities do not depend on time.
2. It is intuitive for us to cope with the path planning problem by defining a CTBN based on the stochastic variable *street identifier* (i.e., SId). However, it has two demerits: (i) in spite of getting a so-called possible path, we cannot describe the change of moving speed that can greatly affect the path selection of moving objects. Accordingly, it does not consider the effect of time that is an important

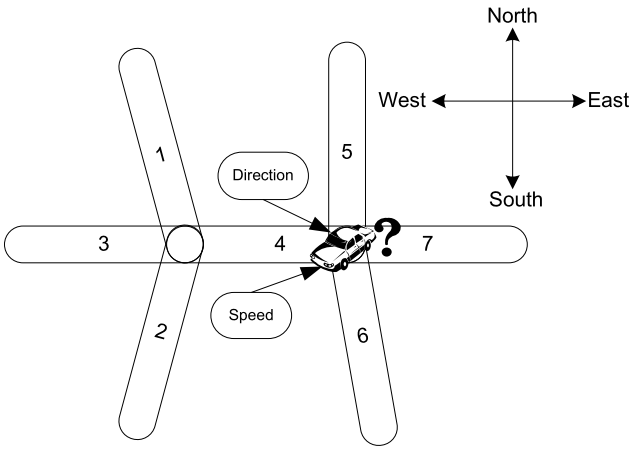
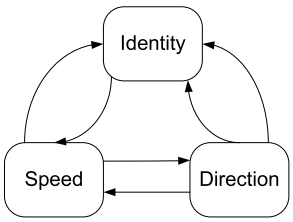


Fig. 4 Example of path planning

Fig. 5 Dependence relationship among three variables



factor for trajectory prediction, because each point in a trajectory is time-stamped; (ii) the *identifier* is not fully independent from other factors, such as speed or direction. Take Fig. 4 as an example, if an object is in street 4 and moving east at a high speed, it is more likely to go straight to street 7. In contrast, if it is moving slowly, it may turn to street 5 or 6. In addition, if it moves west, street 1, 2 and 3 will be its candidates.

From the above discussion, we conclude that street identifier, speed, and direction are dependent on the other two factors which builds in the illustration of Fig. 5.

As shown in Fig. 5, it is a complete graph among three variables, which means that the state transition of one variable is affected by other two variables. Based on their dependence relationship, we employ a hash function to map them into a unique value corresponding to one state in a TCTBN.

5.2 Basic definitions

Following [10], a CTBN is a graphical model whose nodes are random variables, whose states change continuously over time, and where the evolution of the variables depends on its previous states. In principle, the processes in a CTBN are defined as matrices of transition intensities [9]. The states in a CTBN are inhomogeneous Markov processes [10] where the intensities vary with time. As for the property of continuous time in terms of each variable in the same state, it is reasonable to model a trajectory as a CTBN. This network proposed in this study is called TCTBN due to its relevance to trajectories and is defined as follows.

Definition 3 (Trajectory continuous time Bayesian network) Let X be a set of three random variables $\{x_1, x_2, x_3\}$, representing street identifier, moving speed, and moving direction, respectively. Each x_i has a finite domain of values $D(x_i)$. A TCTBN consists of two components: an initial distribution ϕ_X^0 of three variables, denoted as a Bayesian network $\mathcal{N}$ over X, and a continuous transition model, specified as:

- A directed graph G whose nodes are x_1, x_2, x_3, $Pre(x_i)$ denotes the previous state (parent state) of x_i.
- A trajectory conditional intensity matrix $M_{x_i|Pre(x_i)}$, for each $x_i \in X$.

TCTBN is distinct from CTBN in the following two aspects.

1. CTBN relies on exponential distributions for modelling temporal distributions [26]. Unfortunately, the assumption of exponential distribution in terms of time is not consistent with the state transition in TCTBNs. Nevertheless, we use kinematic formulations in physics to compute the time interval spent on each edge (street). For example, for the case of uniform acceleration, the time interval can be calculated by the following formula.

$$\Delta t = 2\frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{v_i + v_{i+1}} \tag{4}$$

2. In order to combine the dynamic rules of all variables, several Conditional Intensity Matrices (CIMs) should be combined to produce a single CIM for the whole CTBN by the amalgamation operation [10]. However, we build an integral CIM without amalgamation. Because once we obtain the state of an object at a certain time instant, the values of all three variables are known. At the next timestamp, another three state values are retrieved. Based on this information, a CIM can be built by calculating the occurrence frequency of each state represented by three variables. Moreover, the values of anti-diagonal elements cannot be calculated by amalgamation and are simply set to 0 in a CTBN. Whereas, we can obtain these values from historical data in TCTBNs. Because, there is, however, one state transition, the transition probability of this shift does not equal 0.

We extend CIM and propose a new concept called Trajectory Conditional Intensity Matrix (TCIM) as follows.

Definition 4 (Trajectory conditional intensity matrix) Let L be a combination of three variables x_1, x_2, x_3 (street identifier, speed, and direction, respectively) whose domain is

$f(L) = \{l_1, l_2, \ldots, l_n\}$. Assume that L evolves as a TCTBN, $L(t)$ whose transition are conditioned on a set L' of variables evolve under time. A TCIM is defined as a matrix:

$$M_{L|L'} = \begin{pmatrix} -p^l_1(L') & p^l_{12}(L') & \cdots & p^l_{1n}(L') \\ p^l_{21}(L') & -p^l_2(L') & \cdots & p^l_{2n}(L') \\ \vdots & \vdots & \ddots & \vdots \\ p^l_{n1}(L') & p^l_{n2}(L') & \cdots & -p^l_n(L') \end{pmatrix} \quad (5)$$

In the above intensity matrix, the element $p^l_{ij}(L')$ in a TCIM represents the instantaneous probability of state transition from l_i to l_j, The transition probability can be computed by $p^l_{ij}(L') = f^l_{ij}(L')/\sum_{i\neq j} f^l_{ij}(L')$, and $f^l_{ij}(L')$ is the occurrence frequency of visiting a street in a trajectory where the state of a object shifts from l_i to l_j, and the frequency can be easily calculated from historical data. The intensity of the diagonal element $p^l_i(L')$=$\sum_{i\neq j} p^l_{ij}(L')$=1 is the probability of leaving state l_i. It is appropriate to set the diagonal elements to 1, since the objects move dynamically which makes it impossible for them to stay in one state.

TCIM is distinct from CIM, because each state in a TCIM is represented by three variables, i.e., $l_i = Hash(n_i, s_i, d_i)$. $Hash(\cdot)$ represents a hash function for mapping the identifier of the ith street, the speed level and moving direction into a unique state value of TCIM. For instance, $l_1 = Hash(n_1, s_1, d_1)$. Note that we use speed level to depict speed information of moving objects.

In summary, TCIM is more expressive and can be used to accurately describe the state transition possibility. Because it contains all state transition information depending on street identifier, speed, and direction. In addition, the state transition probability is obtained from historical data. In particular, the TCTBN $\mathcal{N}$ can be described by the joint intensity matrix in the following equation, where a single variable $x_i \in X$ shifts from state x_i to x_j with probability $p^x_{ij}(Pre(X))$ in Definition 5.

$$P_{\mathcal{N}} = \prod_{x_i \in X} P_{x_i|Pre(x_i)} \quad (6)$$

5.3 Naive solution for trajectory prediction

Figure 6 shows a typical map, where the arrows represent possible directions, the hollow circle is a node representing the road crossing. Each street is annotated by a character, and the real number with an edge is the transition probability. Note that A is an initial street where the object firstly traverses. Intuitively, the sum of transition intensities of the edges ejecting from one road crossing equals 1.

Take Fig. 6 as an example, a naive solution of trajectory prediction assumes that the object moves in a uniform speed and does not turn around. When the object shifts from one edge to another one, the state transition occurs. We can obtain the most possible route $Tr = A \rightarrow C \rightarrow E \rightarrow I$, since the product of probabilities Pr in Tr is highest, i.e., $Pr(Tr) = Pr(C|A) * Pr(E|A, C) * Pr(I|A, C, E) = 0.6 * 0.7 * 0.5 = 0.21$, which represents the most possible candidate trajectory.

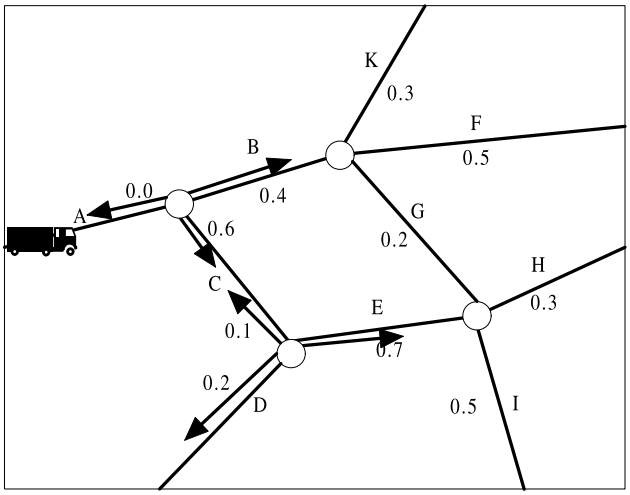


Fig. 6 A real-world scenario of trajectory prediction

5.4 Higher than first-order Markov chain in TCTBNs

According to [27], the naive approach constructs a first-order Markov chain with the random variable *street identifier*. However, this assumption does not obey the reality without considering other important factors.

- **Speed**. The change of speed can cause the moving object to shift from one street to another one as well as affecting the transition probability. For example, when the object traverses the crossing at high speed, it is more likely to go straight instead of turning around or shifting to a street that has a big turning angle from its current direction. Similarly, when moving at low speed, the object may sometimes turn around. In this study, the speed is categorized into six levels according to the speed criterion, represent by $\{s_1, s_2, \ldots, s_6\}$.
- **Direction**. In general, there exists two pairs of directions: North/South, East/West in a map. The object often goes forward along the path, whereas in some specific cases, it acts out of normal behaviors and moves back. Thus, we must take into account the affect of moving directions. For instance, assume that the object is currently running in street E in Fig. 6, if it decides to turn around, there is no opportunity for it to go forward to street I. Furthermore, the selection of directions can affect the transition probability of moving objects. Note that, in terms of streets stretching from Northeast to Southwest or from

Table 1 2nd-order probability matrix

Direction	E	W	S	N
EE	0.6	0.1	0.15	0.15
EW	0	0.6	0.2	0.2
ES	0.2	0.2	0.5	0.1
EN	0.15	0.15	0.3	0.4
WE	0.5	0	0.25	0.25
WW	0.1	0.6	0.15	0.15
WS	0.15	0.15	0.6	0.1
WN	0.1	0.1	0.2	0.6

Table 2 Parameter introduction

Parameter	Definition
s	state information of an object at some time instant
M	a trajectory conditional intensity matrix
Id	street identifier
Dir	moving direction
sLevel	speed level, e.g., low, normal, high
$\mathcal{H}$	a hash table used to store the state information
$\mathcal{E}$	a set of edges in a map
$\mathcal{N}$	a set of node in a map

Northwest to Southeast, we use the most approximate direction to represent its move trend, e.g., Northwest can be replaced by North.

The direction actually contains some second-order information which takes into account specific situations. For example, if we know the previous street, we could estimate roughly the direction at the current street. Therefore, our approach is more expressive than the naive method. Table 1 gives a simple second-order matrix in terms of direction, where E, W, S, N represent four distinct directions.

For the values of the first line in Table 1, the current and the previous state are "EE", it has 60% probability to move east (a object is more likely to keep its moving direction) and 10% probability to move west. For south and north direction, it has equivalent state transition probability. It is intuitive that the sum of the probability values of each line equals 1.

In terms of Markov process, we actually use the nth-order Markov chain to construct a TCTBN, where $n \in (1, 2)$. This is because our model is built based on the first-order variables (i.e., street identifier and speed) and the moving direction that contains second-order information. For space limitation, we omit the other eight lines of state transition probabilities of second-order state space in Table 1.

5.5 Construction of TCTBNs

In this section, we describe the approach of building TCTBNs and introduce a method of mapping three important variables into a unique state value in a TCIM. The important parameters and their meanings are given in Table 2.

In principle, a trajectory composed of several states at distinct time instants forms a state chain. The algorithm of constructing a TCTBN is shown in Algorithm 2. Without loss of generality, we do not only consider the state transition at each node of a street, but also the case occurs in the street.

In Algorithm 2, we firstly initialize a null intensity matrix M (line 1). Then, we obtain the initial state χ of a state chain corresponding to a trajectory composed of edges from $\mathcal{E}$ and nodes from $\mathcal{N}$ in $\mathcal{D}$ (lines 2–3). Next, we continue to check the next state χ' and compare it with χ until finding a transited state (lines 4–7). After that, we record the transition from one state to another by increasing the frequency (line 9) as well as increasing the sum of shifts by one w.r.t. s_0 (line 10), and set the current state as χ' (line 11). Finally, we compute the transition probability of each element in M except the diagonal elements (set to 1) (lines 12–18) and output M (line 19). Intuitively, the time complexity of Algorithm 2 is $O(m * n)$ where m is the number of state chains corresponding to trajectories and n is the number of states in each state chain.

The function of GetState(·) is crucial and used to transform the state information, i.e, street identifier, speed level and moving direction, into a **key** value corresponding to an

Algorithm 2 Construction of TCTBNs

Input: a trajectory cluster $\mathcal{D}$, a set of edges $\mathcal{E}$, a set of nodes $\mathcal{N}$.
Output: a trajectory conditional intensity matrix M.

```
1.  M ← ∅;
2.  for each state chain C ∈ < E, N > in D do
3.      χ = GetState(s_0);
4.      for k := 1 to Len(C) − 1 do
5.          χ' ← GetState(s_k);
6.          if χ == χ' then
7.              continue;
8.          else
9.              M.Set(χ,χ',n++);
10.             M.Set(χ,χ,n++);
11.             χ = χ';
12. for i := 1 to RowLen(M) do
13.     for j := 1 to ColumnLen(M) do
14.         if i == j then
15.             continue;
16.         else
17.             M.Set(i,j,M.Get(i,j) / M.Get(i,i));
18.     M.Set(i,i,1);
19. output M;
```

Algorithm 3 GetState(State s)

1. $\mathcal{H} = \emptyset$;
2. $Id \leftarrow s$.GetId();
3. $Dir \leftarrow s$.GetDirection();
4. $sLevel \leftarrow s$.GetSpeedLevel();
5. $F \leftarrow \mathcal{E}.size() * Id + 4 * Dir + sLevel$;
6. **if** $F \in \mathcal{H}$ **then**
7. return $\mathcal{H}$.GetKey(F);
8. **else**
9. $n = \mathcal{H}.size()$; n++;
10. $\mathcal{H}$.Put(F,n);
11. return n;

existing state in a TCIM. The details are shown in Algorithm 3.

In Algorithm 3, we firstly create a null hash table (line 1). Lines 2–4 are employed to obtain the value of street identifier, speed level and direction. Note that we use the number from 0 to 5 to represent distinct speed levels (0 to 3 to represent directions, respectively). In particular, we define a hash function as presented in line 5 to convert the state information that is represented by three key parameters into a unique integer F. If F exists in the hash table $\mathcal{H}$, then output its key value (lines 6–7). Otherwise, we increase the number n of elements in $\mathcal{H}$ (lines 8–9), treat n as the key of F and put it together with F into $\mathcal{H}$ (line 10). Finally, the algorithm returns the value of n (line 11). Here, the function $size()$ returns the number of elements in the set.

The design of the hash function is an important step in Algorithm 3, since it should guarantee to map three state variables into one unique state value in a TCIM. By extending this hash function, we propose a general hash function and provide a sufficient condition that makes this function unique.

Theorem 1 *Given a state $s(x, y, z)$ of a moving object, where $x \in [0, m]$, $y \in [0, n]$, and $z \in [0, p]$, for the hash function $f(x, y, z) = a * x + b * y + c * z$, the sufficient condition to make it unique is: $c > 0$, $b > c * p$, and $a > b * n + c * p$, where x, y, z, a, b, c are integers.*

Proof We prove the theorem by using the method of reduction to absurdity. Suppose this theorem does not hold when the sufficient condition is satisfied, and there exists two state $s_1 = (x_1, y_1, z_1)$ and $s_2 = (x_2, y_2, z_2)$ satisfying: $s_1 \neq s_2$, but $f(s_1) = f(s_2)$.

If $x_1 \neq x_2$, suppose $x_1 > x_2$ (the case is similar when $x_1 < x_2$), $x_1 - x_2 = r (r > 0)$.

$\because a * x_1 + b * y1 + c * z_1 = a * x_2 + b * y2 + c * z_2$

$\because y_1 - y_2 \leq n, z_1 - z_2 \leq p$

$\therefore a(x_1 - x_2) = b(y_2 - y_1) + c(z_2 - z_1) \leq b * n + c * p$

$\because a(x_1 - x_2) \geq a$

$\therefore a \leq b * n + c * p$, which is a contradiction.

$\therefore x_1 = x_2$.

Suppose $y_1 > y_2$, the case is the same when $y_1 < y_2$.

$\because a * x_1 + b * y1 + c * z_1 = a * x_2 + b * y2 + c * z_2$

$\therefore b * y_1 + c * z_1 = b * y_2 + c * z_2 \Longleftrightarrow b(y_1 - y_2) = c(z_1 - z_2)$

$\because b(y_1 - y_2) > b, c(z_1 - z_2) \leq p$

$\therefore b < c * p$, which is a contradiction.

$\therefore y_1 = y_2, z_1 = z_2$

$\therefore s_1 = s_2$, which is contradict to the assumption.

$\therefore f(x, y, z)$ is unique when $c > 0$, $b > c * p$, and $a > b * n + c * p$. □

6 Trajectory prediction

In this section, we present and analyze the trajectory prediction algorithm based on TCTBNs. The details are given in Algorithm 4.

Our approach of predicting uncertain trajectories can be partitioned into four phases. Firstly, we create a null trajectory sequence set $\mathcal{N}$ and a null state chain C_0 (line 1), then add the initial state s_0 to C_0 (line 2) and C_0 to $\mathcal{N}$ (line 3). For each state chain in $\mathcal{N}$, we firstly find the last state i of C_i (lines 4–5). Secondly, for each state j in the ith row of M and $\mathrm{M}(i, j) \neq 0$, if the result of its current transition probability multiplies the product of the previous transition probabilities is greater than ϵ, then we add j to the trajectory sequence, compute its new transition probability (lines 6–9), and add the extended state chain C'_p to $\mathcal{N}$ and remove C_p

Algorithm 4 Trajectory prediction based on TCTBNs

Input: A TCIM M, an initial state s_0 of a moving object, a probability threshold ϵ, a radius r of a trajectory volume.

Output: a set of possible trajectories $\mathbb{T} = \{T_1, \ldots, T_{num}\}$.

1. $\mathcal{N} \leftarrow \emptyset$; $C_0 \leftarrow \emptyset$;
2. C_0.Add(s_0); $C_0.prob = 1$;
3. $\mathcal{N}$.Add(C_0);
4. **for** each $C_p \in \mathcal{N}$ **do**
5. $i \leftarrow C_p$.LastIndex();
6. **for** each state j in the ith row of M & $\mathrm{M}(i, j) \neq 0$ **do**
7. **if** $C_p.prob * \mathrm{M}(i, j) \geq \epsilon$ **then**
8. $C'_p = C_p$; C'_p.Add(j);
9. $C'_p.prob \leftarrow C'_p.prob * \mathrm{M}(i, j)$;
10. $\mathcal{N}$.Add(C'_p);
11. **for** each $C_p \in \mathcal{N}$ **do**
12. Create a new trajectory T_p;
13. **for** each state κ in C_p **do**
14. $\tau \leftarrow$ComputeTime(κ);
15. T_p.Add(κ_s,κ_e,τ);
16. output T_p;

from $\mathcal{N}$ simultaneously (line 10). Thirdly, we create a trajectory for each state chain in $\mathcal{N}$ with an initial point (lines 11–12). Finally, we employ lines 14 to compute the time interval from κ_s to κ_e of passing a street by kinematic formulations like (4) (κ_s and κ_e represent the starting point and the ending point, respectively) corresponding to each state in $C_p \in \mathcal{N}$ and add the state information into T_i (line 15) in order to output the prediction results (line 16). In particular, the state information contains speed and direction, and it can be used to obtain more accurate time stamps and positions than the naive approach.

Here we give a straightforward example of this approach. Assume a car is located at point A in Fig. 2. We can easily obtain the state transition rules based on TCTBNs. Suppose the car has a 90% chance to go through path 'ABCF' at high speed and a 10% probability to visit path 'ABDE' at uniform speed. Then, we obtain the possible trajectory 'ABCF' with probabilities higher than a predefined threshold 20%. Then, we calculate the position of this object at certain time stamps using speed and direction information stored in a TCTBN, and finally output the trajectory represented by 2D points.

The time and space complexity of Algorithm 4 is $O(m * n)$ and $O(n)$, where m is the number of state chains, and n is the number of states in each state chain.

7 Experiments and discussions

7.1 Experimental setting

In this section, we report the experimental studies by comparing PutMode with the naive prediction algorithm (called Naive for short), and PutMode without the phase of trajectory clustering, namely PutMode-I. Basically, the naive approach does not take into consideration the effect of moving speed or direction when building TCTBNs or predicting trajectories. The transition probability from one state to another state is only determined by the visiting frequencies of streets. In order to evaluate the importance of clustering in PutMode, we also compare it with a reduced version of PutMode, i.e., PutMode-I. All algorithms are implemented in Java and the experiments are conducted on an AMD Athlon 5000+, 3.0 GHz CPU with 2.0 GHz of main memory, running on Ubuntu Linux 8.04.

All experiments were run on the following data sets generated by Brinkhoff's famous network-based generator.[1] The data were generated based on real-world maps by the network-based spatio-temporal data generating approach [28].

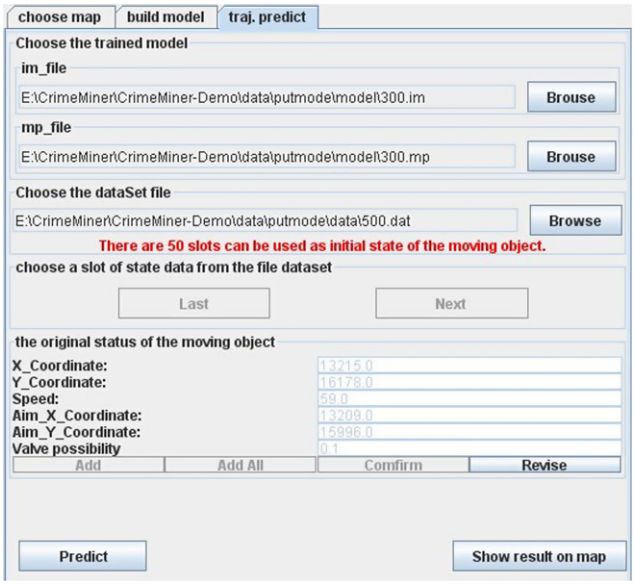


Fig. 7 The main interface of trajectory prediction modular

– The Oldenburg data set (denoted as Oldenburg) contains more than 100,000 trajectories with 6105 nodes and 7035 edges of one day movement over the road-network of the city of Oldenburg.
– The Tiger data set of one city in California (denoted as CA) consists of 1448 nodes and 1573 edges. The real-world map are obtained from U.S. Government Information and Maps Department[2] and its size is around two times larger than that of Oldenburg map.

This prediction modular is one part of our CrimeMiner system and can be used to predict the possible motion curve of fleeing criminals in an accurate and efficient manner. It consists of two components: the network training and trajectory predicting component as given in Fig. 7 and the trajectory visualization of moving objects at any time stamps as shown in Fig. 8.

The network training and trajectory predicting component consists of three distinct parts. The user chooses which component to view by selecting the tab corresponding to the desired component. When selecting the tab labelled "choose map", the user has to choose the edge and node file of a map. The format of these two data files are similar to the ones used in the network-based generator of moving objects proposed by Thomas Brinkhoff [28]. When selecting the "build model" tab, we have to choose the data set and set the maximum speed of moving objects first, than train this model and obtain the trained TCTBN. Finally, we select the "traj.predict" component and use it to predict the positions of different number of moving objects by adding the specified objects with distinct initial states. Specifically, the user

[1] http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/.

[2] http://www.census.gov/geo/www/tiger/tgrcd108/tgr108cd.html.

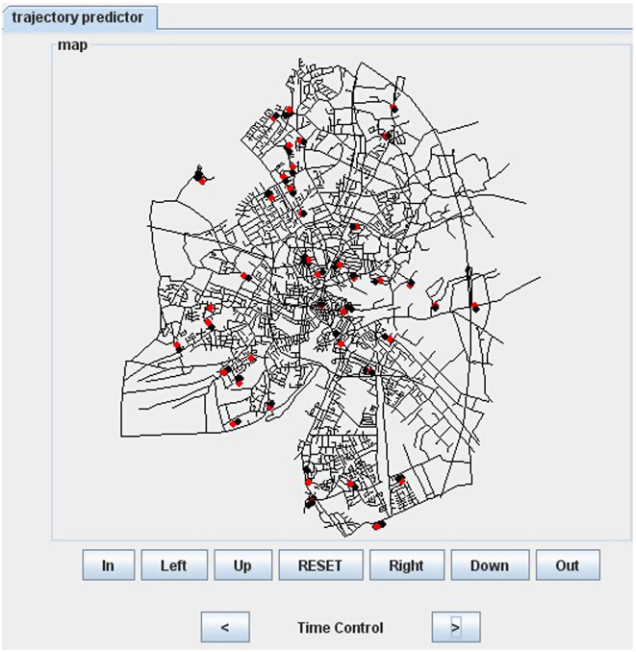


Fig. 8 The main interface of trajectory prediction modular

can set the position (denoted by the xy-coordinates) and the speed of objects. In addition, we provide a specific function to add all test objects by clicking on the button labelled "Add All". After selecting the "Predict" button, we can obtain the results shown in Fig. 8.

In the trajectory visualization component, the user can adjust the position and control the size of the specified map by clicking on distinct function buttons below it. For example, by hitting the "In" button, we can zoom in the map. Most importantly, trajectory visualization of objects at the specified time stamp is controlled by clicking on the arrows before and after the label denoted "Time Control". Note that, in Fig. 8, the initial objects are filled as red color and the positions of black objects are the predicted results at the specified time stamp.

7.2 Parameter tuning

There are two important parameters that need to be tuned first, i.e., the radius r of a disk in a trajectory volume and the probability threshold p. In this section, we conduct several experiments in order to determine values for parameters that are of interest in practical applications.

In principle, r as depicted in Fig. 1 is used to determine whether a trajectory is a possible motion curve, which directly impacts the accuracy of prediction. p is used to determine whether the product $\prod$ of state transition probabilities w.r.t. a trajectory T is large enough. If $\prod \geq p$, T is regarded as a candidate trajectory. Here, we increase r and p gradually in order to find an appropriate value with a high probability of obtaining a high prediction accuracy. The prediction accuracy is defined as follows by employing the basic idea of τ-containment [11].

Table 3 Parameter settings for turning

Parameter	Oldenburg	CA
Map width	23,572	492,826
Map height	26,915	60,2851
Number of moving objects	100,005	25,005
Time interval	15	7
Range of p	0.03∼0.1	0.01∼0.1
Range of r	200∼1200	200∼2000

$$Accuracy(Tr) = \frac{|Tr|}{|Tr^*|} \tag{7}$$

where Tr is the hits set of trajectories and Tr^* is the real-world trajectories during a given time interval. $|\cdot|$ is used to compute the number of trajectories.

For $o = (s_1, \tau_1) \in Tr$ and $o^* = (s_2, \tau_2) \in Tr^*$ where s is a set of streets and τ is a set of time stamps, o should obey the constrain $\preceq_\tau$ which means o is contained in o^* satisfying:

1. $\forall_{0 \leq k \leq n} s_{1,k} \subseteq s_{2,i_k}$
2. $\forall_{1 \leq k \leq n} |\tau_{1,k} - \tau_{*,k}| \leq \tau$

where $\tau_{*,k} = \sum_{j=i_{k-1}}^{i_k} \tau_{2,j}$, $\forall k \in [1, n]$. Here, τ is set to 1 (i.e., one time unit). The trajectory is time stamped, so we have to predict trajectories constrained at a specified time interval. In this study, we use time units to measure the time interval, i.e., $N_p = T_p/\tau_0$, where T_p is the ordinary prediction time and τ_0 is the cost per time unit. The experiments are conducted on both Oldenburg and CA data sets. The parameter settings are shown in Table 3.

To facilitate understanding, we explain the meaning of parameters given in Table 3. Take Oldenburg data set as an example, we firstly set r to 200 (in pixel) at the basic scale 1 (i.e. in basic coordinates) in a 23,572 × 26,915 electronic map, then gradually increase it by the step of 100 until reaching 1200.The product of state transition probabilities in a state chain will decay drastically by increasing the number of states, so it is appropriate to specify a small probability threshold. In this set of experiments, we observe the accuracy of prediction under distinct probability thresholds changing from 0.01 to 0.06. In this study, N_p is set to 15 (for Oldenburg data) and 7 (respectively, for CA data), which are empirical values large enough to obtain most complete trajectories and are comparable between PutMode and Naive without the bias of using relatively short time interval.

The results of two versions of PutMode are shown in Fig. 9 with the Oldenburg data, and in Fig. 10 with the CA data, respectively, where the x-axis represents the length of r, the y-axis is the accuracy of prediction.

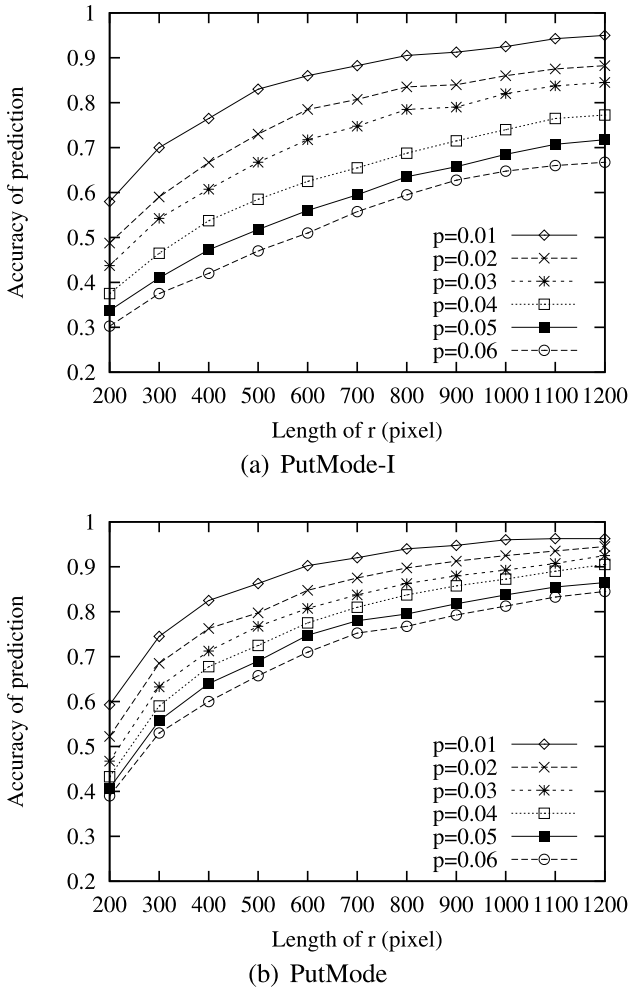


Fig. 9 Accuracy of prediction under distinct probability thresholds in Oldenburg data set: (**a**) PutMode-I and (**b**) PutMode

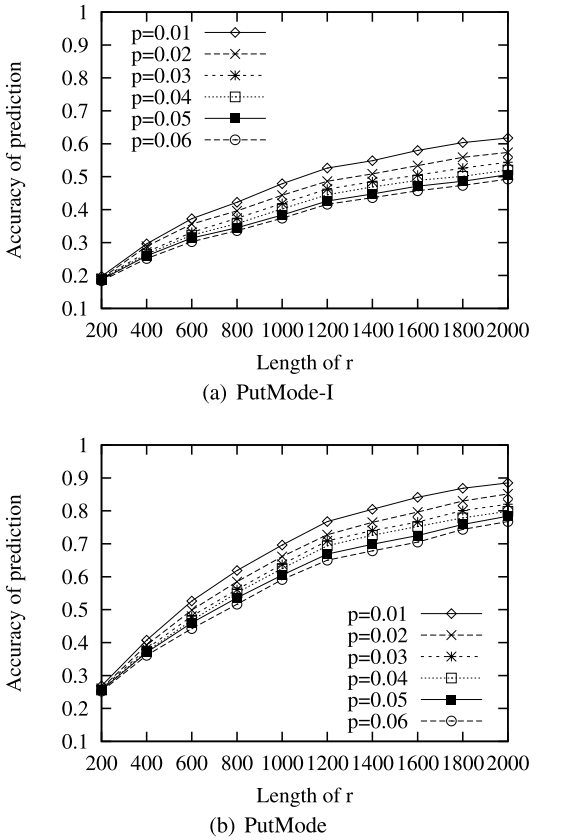


Fig. 10 Accuracy of prediction under distinct probability thresholds in CA data set: (**a**) PutMode-I and (**b**) PutMode

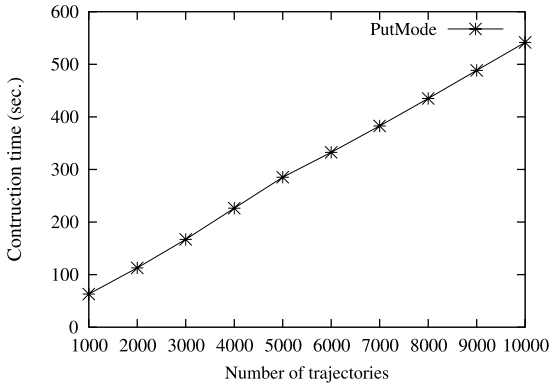
As we can see from Figs. 9 and 10, the higher a p value, the lower is the accuracy of prediction. The reason is straightforward: when the probability threshold is set to a higher value, PutMode has to find possible trajectories restricted by a further limit, i.e., such street in a trajectory is frequently visited, which leads to the decline of accuracy. For instance, in Fig. 9(a), the curve $p = 0.03$ is beyond the curve $p = 0.04$ as r grows.

Another observation does tell us that the accuracy corresponding to distinct p values increases with r. This is because PuteMode can find more possible trajectories with the increase of r. Most importantly, we must choose an appropriate r value that complies with the real-world cases. Since a quite high r value violates the principle that "it is appropriate to find possible motion curves in a disk with a small radius of a trajectory volume" in a real-world scenario. According to the map size of distinct data sets, r is set to 700 for Oldenburg data, and 1800 for CA data. As we can see in Fig. 9(a), even $p = 0.06$, the accuracy is higher than 50% that is an acceptable result when $r = 700$. For the consequent experiments, p is set to 0.03. If p is specified to a very small value (e.g., 0.01), our approach does not make sense since almost all trajectory are treated as candidates, which causes the accuracy to equal 1.

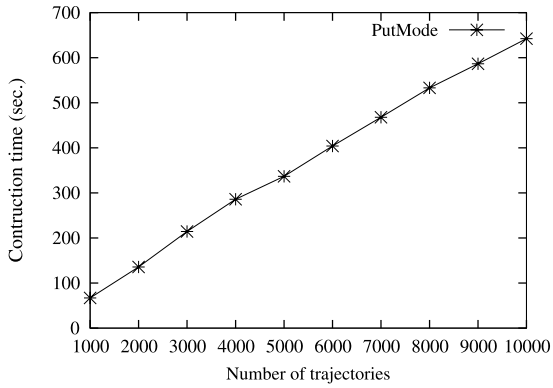
It is interesting to see that the accuracy improvement is not apparent when r reaches a large enough value in Figs. 9 and 10, which further prove that the accuracy is affected by the selection of streets instead of the error of position calculation. In addition, we find that the prediction accuracy of PutMode is higher than that of PutMode-I under similar situations. For instance, PutMode in Fig. 10(a) outperforms PutMode-I in Fig. 10(b) in the same settings of p value and r value. We will further compare the performance of two versions of PutMode.

7.3 Performance analysis of TCTBN construction

In this section, we analyze the time trend and the change of memory cost in terms of TCTBN construction as the number of trajectories grows in the Oldenburg and the CA data sets. The results are shown in Figs. 11 and 12, where the x-axis is the number of trajectories and the y-axes are the time and the memory cost of TCTBN construction, respectively.

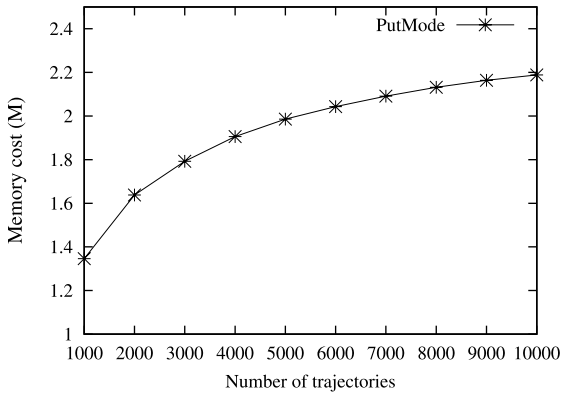


(a) Oldenburg data

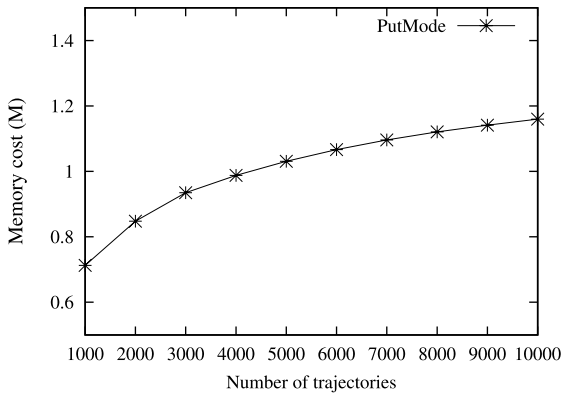


(b) CA data

Fig. 11 Execution time of constructing TCTBNs with (**a**) Oldenburg and (**b**) CA data



(a) Oldenburg data



(b) CA data

Fig. 12 Memory cost of constructing TCTBNs with (**a**) Oldenburg and (**b**) CA data

Empirically, the training time of a TCTBN with 10,000 trajectories is around 9 minutes.

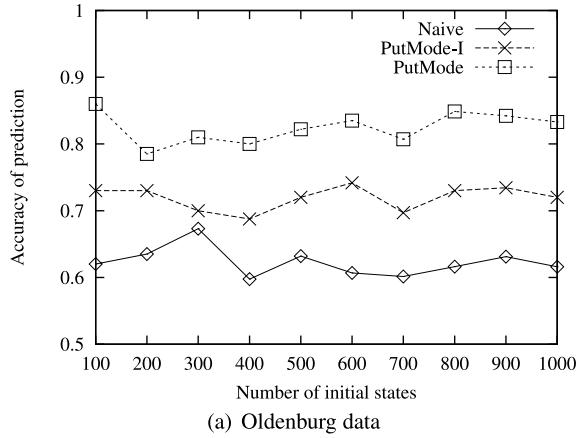
From Fig. 11, the construction time increases linearly with the number of trajectories. The reason is that the time complexity of TCTBN construction is $O(m * n)$. In real-life situations, $m \gg n$ where m is the number of state chains corresponding to trajectories and n is the number of states in each state chain. Thus, the time complexity is approximately equivalent to $O(m)$ that complies with our experimental results. In summary, we conclude that the time of TCTBN construction is mainly determined by the cardinality of trajectories. In general, when the number of trajectories grows to 100,000, the execution time of TCTBN construction is about one and half hours. Our approach yields the similar results when the number of trajectories changes from 2000 to 100,000, we omit these plots for brevity.

According to Fig. 12, the memory cost of TCTBN construction increases with the number of trajectories, but it does not change drastically when the number of trajectories is greater than 4000 for Oldenburg data and 2000 for CA data. This is because there exist several overlap states when building TCTBNs with a large number of trajectories, and the duplicate states can be represented by one state that provides benefits on memory cost of storing distinct states.

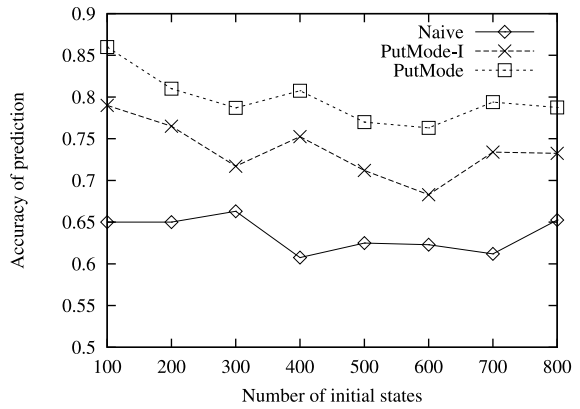
7.4 Prediction accuracy evaluation

In this section, we use **Accuracy** to evaluate these three prediction algorithms: Naive, PutMode-I, and PutMode in two distinct data sets. The results are shown in Fig. 13, where the x-axis is the number of initial states (i.e., the initial moving objects) and the y-axis is the accuracy of prediction.

In Fig. 13(a)–(b), PutMode outperforms PutMode-I in the accuracy of prediction in all cases with an average gap of 10.5% for Oldenburg data and 6.16% for CA data, respectively. The role of clustering in PutMode is of great importance, because it can filter noise data that, thus helping perform prediction on the concentrated trajectory clusters. In particular, the prediction accuracy of PutMode is higher than 82.4% and 80% on average for the Oldenburg and CA data, respectively. The results have been confirmed by law enforcement authorities, who stated that the results are surprisingly good for criminal escape route prediction. We observe that these two approaches outperform Naive by integrating more complex factors, i.e., speed and direction, in both TCTBN-building and trajectory-predicting processes. This is because Naive only uses first-order probability matrix to predict trajectories, whereas PutMode can obtain more

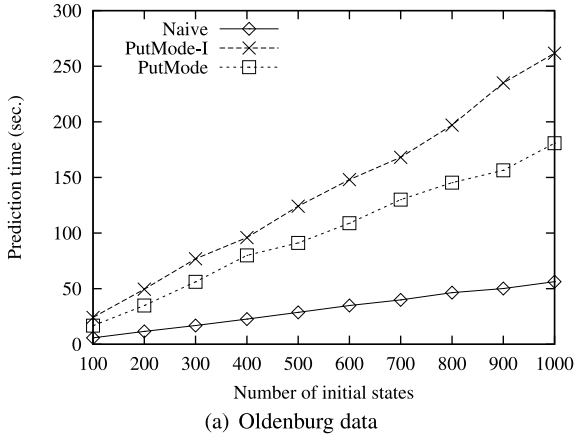


(a) Oldenburg data

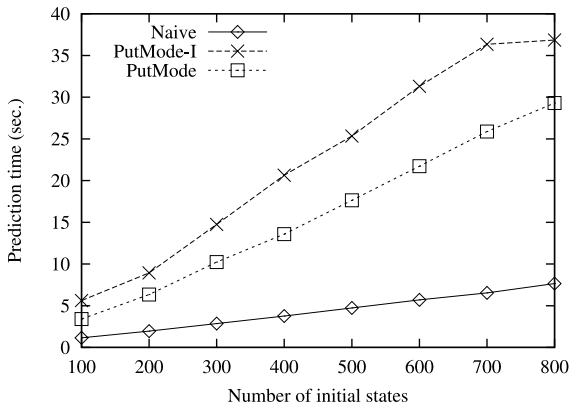


(b) CA data

Fig. 13 Prediction accuracy comparison of three algorithms with (**a**) Oldenburg and (**b**) CA data



(a) Oldenburg data



(b) CA data

Fig. 14 Prediction time comparison of three algorithms with (**a**) Oldenburg and (**b**) CA data

accurate results based on nth-order Markov chains where $n \in (1, 2)$. In other words, PutMode will benefit from more complicated consideration of real-world cases than the intuitive Markov processes. Empirically, PutMode outperforms Naive with an average gap of 20.1% in accuracy for the Oldenburg data set and 16.2% for the CA data set.

We also observe that the accuracies for two versions of PuteMode do not change drastically as the number of initial states grows, which shows that our approach is scalable with the number of initial moving objects. The reason is that we use the motion behaviors of objects discovered from historical data to obtain the possible trajectories, and the accuracy is determined by the TCTBN model without being affected by the number of initial states. As mentioned earlier, several outlying trajectories are eliminated by applying the clustering approach, which provides benefits for PutMode to use relatively less time to construct TCTBNs than PutMode-I. More details are given in the following subsection.

7.5 Prediction time comparison

In this section, we compare the performance of trajectory prediction among three algorithms by evaluating the prediction time as the number of initial states increases. Figure 14(a) and (b) show the prediction time comparison across three algorithms in the Oldenburg and the CA data sets, respectively, where x-axis is the number of initial states and y-axis represents the prediction time.

In Fig. 14(a) and (b), we can see that the prediction time of PutMode is less than that of PutMode-I in all cases and achieves an improvement of 17%–33% for the Oldenburg data and 20%–39% for the CA data, respectively. Overall, Naive is the winner in all cases, since it is simple and just uses the shift from one street to another street to represent one state transition. However, PutMode employs three variables to determine one state transition, and there are more candidate states than that of Naive. It is interesting to see that the prediction time of PutMode increases linearly with the number of initial states. This is because the time cost of predicting is mainly determined by the number of state chains (the reason is similar to the time trend analysis of TCTBN construction in Sect. 7.3), while the number of state chains depends on the number of initial states.

8 Conclusions and future work

Existing trajectory prediction approaches either predict possible trajectories by discovering frequent trajectory patterns, or fall short in concerning dynamic motion patterns. Specifically, this paper has provided a solution for the prediction of uncertain trajectories in moving objects databases.

To handle these limitations, the paper has proposed an efficient and effective trajectory prediction algorithm, called PutMode. It works by: (1) clustering trajectories in order to detect outlying trajectories; (2) modeling uncertain trajectories by constructing TCTBNs; and (3) combining three important variables (i.e., street identifier, moving speed, and moving direction) to predict possible motion trajectories. Experimental studies with data sets generated on real-world maps have demonstrated that PutMode is capable of providing high prediction accuracy with a guarantee of time efficiency.

In terms of future research directions, we will extend this trajectory prediction approach to handle more intricate and unpredictable phenomena by taking into account other important factors, such as weather and road conditions. Another challenging problem on our research agenda is extending PutMode towards tracing fleeing criminals by integrating the methodology of crime psychology, since it is difficult to predict the possible motion patterns of criminals who often break the common rules to avoid being hunted. Moreover, the optimization of TCTBNs is important. Since the more states one trajectory has, more time are spent on building this network. An intuitive solution is to use the popular data reduction approaches, e.g., PCA (Principle Component Analysis) and neural networks to optimize TCTBNs. Finally, the optimization of the storage space needed for trajectories is of practical interest as well.

Acknowledgements 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, and NICTA which is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Xiang Y, Chau M, Atabakhsh H, Chen H (2005) Visualizing criminal relationships: Comparison of a hyperbolic tree and a hierarchical list. Decis Support Syst 41(1):69–83
2. Zhao JL, Bi HH, Chen H, Zeng DD, Lin C, Chau M, Process-driven collaboration (2006) support for intra-agency crime analysis. Decis Support Syst 41(3):616–633
3. Kaza S, Wang Y, Chen H (2007) Enhancing border security: Mutual information analysis to identify suspect vehicles. Decis Support Syst 43(1):199–210
4. Morzy M (2007) Mining frequent trajectories of moving objects for location prediction. In: MLDM'07: Proceedings of the 5th international conference on machine learning and data mining in pattern recognition. Lecture notes in computer science, vol 4571. Springer, Berlin, pp 667–680
5. Chang W, Zeng D, Chen H (2008) Visualizing criminal relationships: Comparison of a hyperbolic tree and a hierarchical list. Decis Support Syst 45(6):697–713
6. Trajcevski G, Wolfson O, Zhang F, Chamberlain S (2002) The geometry of uncertainty in moving objects databases. In: EDBT'02: Proceedings of the 8th international conference on extending database technology. Springer, London, pp 233–250
7. Trajcevski G, Wolfson O, Hinrichs K, Chamberlain S (2004) Managing uncertainty in moving objects databases. ACM Trans Database Syst 29(3):463–507
8. Tao Y, Faloutsos C, Papadias D, Liu B (2004) Prediction and indexing of moving objects with unknown motion patterns. In: SIGMOD'04: Proceedings of the 2004 ACM SIGMOD international conference on management of data. ACM, New York, pp 611–622
9. Nodelman U, Shelton CR, Koller D (2003) Learning continuous time Bayesian networks. In: UAI'03: Proceedings of the 19th conference on uncertainty in artificial intelligence. Morgan Kaufmann, San Francisco, pp 451–458
10. Nodelman U, Shelton C, Koller D (2002) Continuous time Bayesian networks. In: UAI'02: Proceedings of the 18th conference in uncertainty in artificial intelligence. Morgan Kaufmann, San Francisco, pp 378–387
11. Giannotti F, Nanni M, Pedreschi D (2006) Efficient mining of temporally annotated sequences. In: SDM'06: Proceedings of the 6th SIAM international conference on data mining. SIAM, Bethesda, pp 346–357
12. Giannotti F, Nanni M, Pinelli F, Pedreschi D (2007) Trajectory pattern mining. In: KDD'07: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, San Jose, pp 330–339
13. Mokhtar HMO, Su J (2004) Universal trajectory queries for moving object databases. In: MDM'04: Proceedings of the 2004 IEEE international conference on mobile data management. IEEE Computer Society, Los Alamitos, pp 133–144
14. Mamoulis N, Cao H, Kollios G, Hadjieleftheriou M, Tao Y, Cheung DW (2004) Mining, indexing, and querying historical spatiotemporal data. In: KDD'04: Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 236–245
15. Bharucha-Reid AT (1960) Elements of the theory of Markov processes and their applications. McGraw-Hill, New York
16. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M (2001) Prefixspan: Mining sequential patterns by prefix-projected growth. In: ICDE'01: Proceedings of the 17th international conference on data engineering, Washington, DC, USA, pp 215–224
17. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: SIGMOD'00: Proceedings of the 2000 ACM SIGMOD international conference on management of data. ACM, New York, pp 1–12
18. Giannotti F, Nanni M, Pedreschi D, Pinelli F (2006) Mining sequences with temporal annotations. In: SAC'06: Proceedings of the 2006 ACM symposium on applied computing. ACM, New York, pp 593–597
19. Xu JJ, Chen H (2005) Crimenet explorer: a framework for criminal network knowledge discovery. ACM Trans Inf Syst 23(2):201–226
20. Chen H, Chung W, Xu JJ, Wang G, Qin Y, Chau M (2004) Crime data mining: A general framework and some examples. Computer 37(4):50–56
21. Qiao S, Tang C, Peng J, Fan H, Xiang Y (2006) Vccm mining: Mining virtual community core members based on gene expression programming. In: Chen H, Wang FY, Yang CC, Chau DZM,

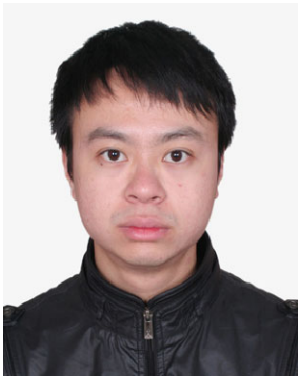
Chang K (eds) Proceedings of intelligence and security informatics: International workshop, WISI 2006, 9 April 2006. Lecture notes in computer science, vol 3917. Springer, Berlin, pp 133–138

22. Qiao S, Tang C, Peng J, Liu W, Wen F, Qiu J (2008) Mining key members of crime networks based on personality trait simulation email analysis system. Chin J Comput 31(10):1795–1803 (in Chinese)
23. Xu JJ, Chen H (2005) Criminal network analysis and visualization. Commun ACM 48(6):101–107
24. Qiao S, Tang C, Cheng Y, Peng J, Wen F (2007) A new hierarchical clustering algorithm based on tree edit distance. J Comput Sci Front 1(3):282–292 (in Chinese)
25. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD'96: Proceedings of the 2nd international conference on knowledge discovery and data mining. AAAI Press, Pittsburgh, pp 226–231
26. Gopalratnam K, Kautz HA, Weld DS (2005) Extending continuous time Bayesian networks. In: AAAI'05: Proceedings of the 20th national conference on artificial intelligence. AAAI Press, Pittsburgh, pp 981–986
27. Roads C (1996) The computer music tutorial. MIT Press, Cambridge
28. Brinkhoff T (2002) A framework for generating network-based moving objects. Geoinformatica 6(2):153–180

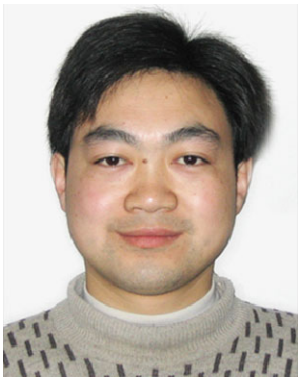
Shaojie Qiao received the BSc (Eng.) and PhD degree in the School of Computer Science in 2004 and 2009, respectively, both from Sichuan University, China. He is currently working in the School of Information Science and Technology at Southwest JiaoTong University, China. He worked as a visiting scholar in the School of Computing at the National University of Singapore in 2008. His research interests include crime data mining, spatio-temporal databases, digital forensics, and security informatics. He has authored or coauthored more than 50 papers in these areas. He is a member of the International Association of Computer Science and Information Technology (IACSIT).

Changjie Tang received the BSc and MSc degree in the Department of Mathematics in 1969 and 1981, respectively, both from Sichuan University, China. Prof. Changjie Tang is the vice director of Database Society of Chinese Computer Federation. He was a PC member of VLDB'97, DASFAA'99, ICSC'99, PAKDD'2001, DASFAA'2001, WAIM 2000–2008. His current research interests include data mining and databases. He has published 8 books and more than 100 research articles in international database conferences and computer science journals.

Huidong Jin received the BSc degree from the Department of Applied Mathematics in 1995, and the MSc degree from the Institute of Information and System Sciences in 1998, both from Xi'an Jiaotong University, China. In 2002, he received the PhD degree in computer science and engineering from the Chinese University of Hong Kong, Shatin, Hong Kong. He is currently with the Division of Mathematical and Information Sciences, CSIRO, Australia. His research interests are data mining, health informatics, and intelligent computation. He has authored or coauthored more than 15 papers in these areas. He is a member of the ACM, the IEEE Computer Society, and the IEEE.



Teng Long eceived the BSc (Eng.) degree from the School of Computer Science at Sichuan University, China, in 2009. His research interests are data mining and machine learning.



Shucheng Dai received the BSc and MSc degree in the School of Computer Science in 2004 and 2009, respectively, both from Sichuan University, China. He works as a lecturer in the School of Computer Science at Sichuan University. His research interests are data mining, wireless sensor networks, and intelligent computation. He has published more than 10 papers in these areas.

Yungchang Ku works as a police officer at the Curriculum Section, Academic Affairs of Central Police University, Taiwan. He received the BSc degree from Central Police University in 1996, the MSc degree from National Central University in 2001, both in the Department of Management Information. His research interests are data mining, criminal linguistics profiling, and criminal intelligence analysis.

Michael Chau is an Assistant Professor and the BBA(IS)/BEng(CS) Coordinator in the School of Business at the University of Hong Kong. He received his PhD degree in management information systems from the University of Arizona and a Bachelor Degree in Computer Science and Information Systems from the University of Hong Kong. His research interests include information retrieval, Web mining, data mining, knowledge management, electronic commerce, security informatics, and intelligent agents. He has published more than 60 research articles in leading journals and conferences, including IEEE Computer, Journal of the America Society for Information Science and Technology, Decision Support Systems, and Communications of the ACM.