MoveMine: Mining Moving Object Data for Discovery of Animal Movement Patterns

ZHENHUI LI, JIAWEI HAN, MING JI, LU-AN TANG, YINTAO YU,
and BOLIN DING, University of Illinois at Urbana-Champaign
JAE-GIL LEE, KAIST
ROLAND KAYS, New York State Museum

With the maturity and wide availability of GPS, wireless, telecommunication, and Web technologies, massive amounts of object movement data have been collected from various moving object targets, such as animals, mobile devices, vehicles, and climate radars. Analyzing such data has deep implications in many applications, such as, ecological study, traffic control, mobile communication management, and climatological forecast. In this article, we focus our study on animal movement data analysis and examine advanced data mining methods for discovery of various animal movement patterns. In particular, we introduce a moving object data mining system, MoveMine, which integrates multiple data mining functions, including sophisticated pattern mining and trajectory analysis. In this system, two interesting moving object pattern mining functions are newly developed: (1) *periodic behavior mining* and (2) *swarm pattern mining*. For mining periodic behaviors, a reference location-based method is developed, which first detects the reference locations, discovers the periods in complex movements, and then finds periodic patterns by hierarchical clustering. For mining swarm patterns, an efficient method is developed to uncover flexible moving object clusters by relaxing the popularly-enforced collective movement constraints.

In the MoveMine system, a set of commonly used moving object mining functions are built and a user-friendly interface is provided to facilitate interactive exploration of moving object data mining and flexible tuning of the mining constraints and parameters. MoveMine has been tested on multiple kinds of real datasets, especially for MoveBank applications and other moving object data analysis. The system will benefit scientists and other users to carry out versatile analysis tasks to analyze object movement regularities and anomalies. Moreover, it will benefit researchers to realize the importance and limitations of current techniques and promote future studies on moving object data mining. As expected, a mastery of animal movement patterns and trends will improve our understanding of the interactions between and the changes of the animal world and the ecosystem and therefore help ensure the sustainability of our ecosystem.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.4.0 [**Information Systems Applications**]: General

This is an extended and major-value added version of Li et al. [2010a, 2010b, 2010c]. Substantial new technical materials have been added to this journal submission. MoveMine system is online: http://dm.cs.uiuc.edu/movemine.
The work was supported in part by the Boeing company, NSF BDI-07-Movebank, NSF CCF-0905014 (Cyber-Physical Systems), U.S. Air Force Office of Scientific Research MURI award FA9550-08-1-0265, and by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (NS-CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.
Authors' addresses: Z. Li (corresponding author), J. Han, M. Ji, L.-A. Tang, Y. Yu, and B. Ding, Department of Computer Science, University of Illinois at Urbana-Champaign, 2132 Sibel Center, 201 N Goodwin, M/C 258, Urbana, IL 61801; email: zli28@uiuc.edu; J.-G. Lee, Department of Knowledge Service Engineering, KAIST, 291 Daehak-ro, Daejeon 305-701, Republic of Korea; jaegil@kaist.ac.kr.

DOI 10.1145/1989734.1989741 http://doi.acm.org/10.1145/1989734.1989741

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Moving objects, pattern mining, periodic behavior, swarm pattern, computational sustainability

ACM Reference Format:
Li, Z., Han, J., Ji, M., Tang, L.-A., Yu, Y., Ding, B., Lee, J.-G., and Kays, R. 2011. MoveMine: Mining moving object data for discovery of animal movement patterns. ACM Trans. Intell. Syst. Technol. 2, 4, Article 37 (July 2011), 32 pages.
DOI = 10.1145/1989734.1989741 http://doi.acm.org/10.1145/1989734.1989741

1. INTRODUCTION

Thanks to the wide adoption of GPS tracking system and other telemetry and telecommunication technologies, massive amounts of moving object data have been collected. Moving object data could be related to human, objects (e.g., airplanes, vehicles, and ships), animals, and/or natural forces (e.g., hurricanes and tornadoes). Although most human and man-made object movements are closely associated with social and economic behaviors of people and society, movements of animals and changes of natural phenomena are often related to ecological and climate studies and thus are related to the efforts towards a sustainable earth and ecosystem. A systematic development of computational methods and systems for moving object pattern analysis may have broad impacts to economic and ecological studies.

Based on the aforesaid motivation, a MoveMine system [Li et al. 2010a] is designed and developed in our recent research, which takes a spectrum of moving object data as inputs and aims for the discovery of various kinds of movement patterns and knowledge. Although moving objects can be referred to people and man-made objects, such as vehicles, airplanes, and ships, and their studies could benefit traffic planning, law enforcement, counter-terrorism, as well as many other applications, the MoveMine project is focused more on the study of animal and bird movements and movements of natural forces (such as hurricanes and tornadoes). This emphasis is due to two major factors: (1) the wide availability of animal/bird and natural force (e.g., weather) movement data, and their studies have less privacy concerns than studying human and vehicle movements, and (2) the movements of animals and natural forces provide a wide spectrum of sophisticated and versatile patterns because their movements often do not follow particular routes or tracks (thus relatively free movements) and pose challenges on mining many sophisticated patterns, such as clusters, leaders, followers, encounters, flocks, convoys, swarms, avoidance, chasing, and periodicity. Such studies lead to the generation of a set of sophisticated pattern mining methods, with rich semantics and interesting techniques.

In general, there are many data mining methods developed for analyzing moving objects data. Based on the nature of the problems, these methods can be categorized into three categories: (1) classification, where classification models are derived based on a set of object movement training data, labeled by domain experts, and people may use classification models to predict new object categories or movements; (2) outlier detection, where movements that are deviate substantially from regular moving patters are identified and can be used to single the abnormal events or environmental changes; and (3) moving object pattern analysis, where various kinds of patterns can be mined from moving object datasets. Moving object pattern analysis is the focused theme in this article, which can be further categorized into the following topics:

(1) *Repetitive pattern.* It is common that objects follow some regular movement patterns. For example, a bird could have daily behaviors between foraging areas and its nests. The periodic behaviors provide an insightful and concise explanation over a long moving history. However, it is difficult for humans to manually examine the periodicity in the movements and give a good assessment of such periodic behavior. If

Table I. Summary of Major Related Works in Pattern Analysis of Moving Objects

Category	Related works
Repetitive pattern	Periodic pattern [Mamoulis et al. 2004], periodic behavior [Li et al. 2010b, 2010c]
Relationship pattern	(1) moving clusters: flock [Gudmundsson and van Kreveld 2006], convoy [Jeung et al. 2008b], swarm [Li et al. 2010c]; (2) follower/leadership [Gudmundsson et al. 2008]
Frequent pattern	Trajectory-pattern [Giannotti et al. 2007]

one can automatically extract periodic behaviors from raw data (in the form of time and location), it would give biologists a semantic summarization of animal movements. Also, periodicity helps predict future movements and thus may give people better chances to protect endangered species. More interestingly, if we observe an animal starts changing its periodic behavior or route, it could imply a signal of abnormal environmental change or an accident. Mamoulis et al. [2004] work out an efficient algorithm to mine frequent periodic patterns with the period *given* beforehand. In our recent work [Li et al. 2010b], we develop algorithm Periodica, which detects the periods in the movement automatically and statistically summarizes the periodic behaviors.

(2) *Relationship pattern.* With a set of moving objects, one might want to know the relationships among the individuals. One of the most useful tasks is to find groups of objects that move together. By discovering such clusters, one can detect the communities of animals. There have been a lot of studies on clusters in terms of flock [Gudmundsson and van Kreveld 2006] and convoy [Jeung et al. 2008b]. While flock and convoy enforce the objects in one cluster are together for k consecutive times, in our recent work [Li et al. 2010c], we introduce a swarm pattern which relaxes consecutive time constraint. Swarm pattern will be able to find moving object clusters for many real applications.

Besides clusters, there are also moving object patterns that reflect interactions among movements, such as followers and leaders. For example, bears and coyotes are usually observed in the same area because of the food resources. But they always keep distance from each other. Some species could avoid or follow another by tracing their odor, sound, or footprints. It is interesting for animal scientists to detect such relationships automatically and study such relationships. Gudmundsson et al. [2008] propose several moving object patterns based on the movement directions and locations, such as flock, leadership, and convergence.

(3) *Frequent trajectory pattern.* Frequent trajectory patterns are the general moving trends of all the objects, in terms of both space (i.e., the regions of space visited during movements) and time (i.e., the duration of movements). For example, a frequent trajectory pattern could be a considerable number of objects starting from region A, reaching region B after 10 minutes, and then getting to region C after 20 minutes.

The research work on movement pattern analysis can be summarized in Table I.

Among all the object movement mining methods studied, we believe two issues are critical for effective animal movement pattern analysis: (1) finding clustered movements, and (2) finding periodic movements. However, for finding clustered movements, the previous works, such as flock [Gudmundsson and van Kreveld 2006] and convoy [Jeung et al. 2008b], assume that the objects in a cluster be close to each other for at least k *consecutive* times, which may not be true in most of the real applications. Thus we propose a new task, called *swarm pattern mining* [Li et al. 2010a]. This task relaxes the *consecutive time* constraint and allows a set of moving objects to be dispersed irregularly as long as they are close to each other for *many* of the timestamps for an extended

time period. This matches real applications and likely lead more fruitful finding the promising patterns.

For finding periodic movements, most previous work assumes periods are given and fixed. However, in real life, the periods of a moving object could be multiple and unknown, such as the moving wild animals. Thus, we propose to mining both unknown periods and periodic behavior [Li et al. 2010b]. We formulate such a periodic behavior mining problem and assume that the observed movement is generated from several periodic behaviors associated with some reference locations. A two-stage mining framework, Periodica, is developed to first detect the periods and then find the periodic behaviors. Comparing with Mamoulis et al. [2004], Periodica is more useful in real applications not only because it can detect periods but also because it gives a more concise summarization of the periodic patterns.

These two critical functions form the key innovations in our newly developed MoveMine system. In this study, we examine these two methods and study their effectiveness and efficiency at mining such patterns for animal and other object movements. Moreover, we discuss their possible extensions and impact on moving pattern analysis for animal studies and other similar applications. Further, we introduce the architecture of the MoveMine system and its various functions for mining moving object data. The major contributions of this study are outlined as follows.

(1) A system, MoveMine, is constructed that integrates various data mining functions. The system is tested on real datasets and the results are visualized in Google Map and Google Earth. MoveMine can facilitate users to analyze moving object data.
(2) A location-based method, Periodica, is developed that effectively mines multiple periodic behaviors of moving objects.
(3) The concept of swarm pattern is proposed that reveals real moving object clusters, and an ObjectGrowth method is developed for efficient mining swarm patterns.
(4) Comprehensive experiments are conducted on both real and synthetic datasets, including a set of animal movement data, deposited in MoveBank (MoveBank.org). The results demonstrate the usefulness of our system and the effectiveness of our new algorithms.
(5) We discussed the advantages of periodic behaviors and swarm patterns over other methods and examined their possible extensions to other problems.

The remainder of the article is organized as follows. Section 2 will give an overview of general system architecture. Periodic behavior mining and swarm pattern mining methods are introduced in Section 3 and Section 4 separately. The extensions of the two methods are discussed in Section 6. We report the experimental results in Section 5. Finally, we conclude our study in Section 7.

2. GENERAL SYSTEM ARCHITECTURE

Our system provides some interesting data mining functions for biologists to analyze the animal movement patterns. We focus on mining the repetitive pattern and the mutual relationship. There have been systems designed for answering spatio-temporal analysis queries. Stolorz et al. [1995] study the geophysical phenomenon. They basically extract two spatio-temporal features, cyclone and blocking features. Their analysis emphasizes on the general trends of climate changes. A system developed by Nanni et al. [2010] essentially mines the trajectory patterns. Trajectory patterns are the frequent moving trends over all the moving objects. It is useful for traffic analysis. However, those systems cannot mine the patterns that biologists are interested in, such as periodic pattern and swarm pattern. Our MoveMine system will provide these functions and be tested on the real animal movement data.

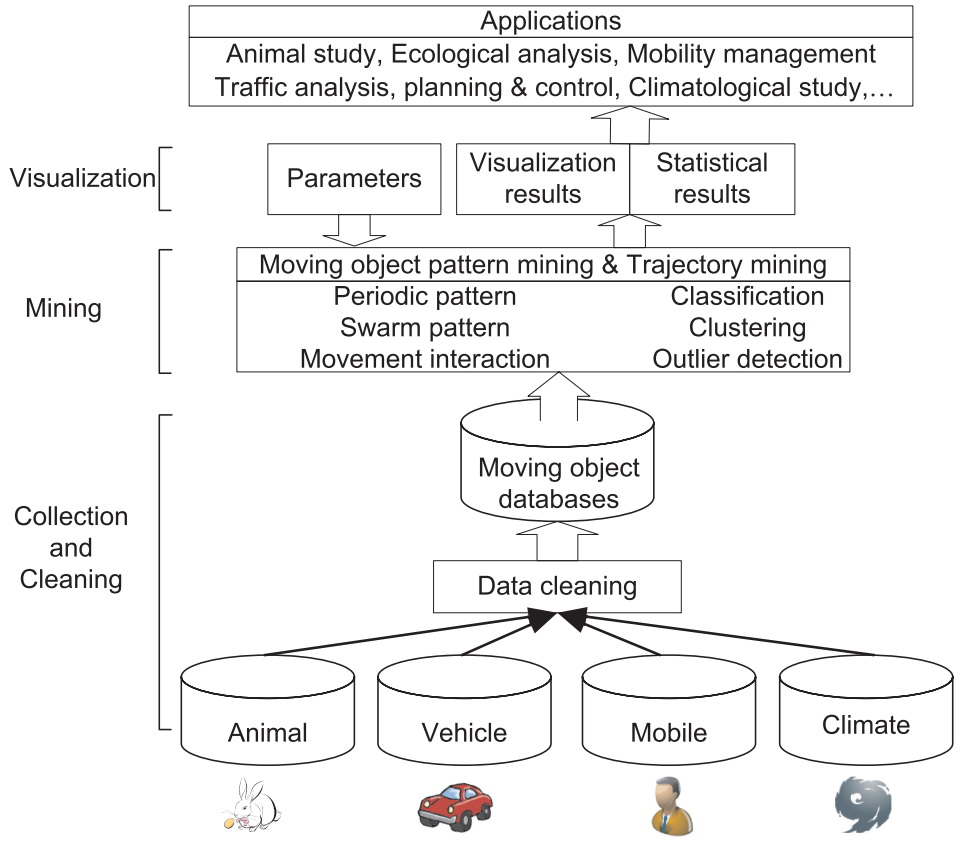


Fig. 1. System architecture.

Figure 1 depicts the system architecture of MoveMine that consists of three layers: (i) collection and cleaning, (ii) mining, and (iii) visualization. The lower layer is responsible for collection and cleaning of moving object data. Various moving object datasets are collected from different resources like animals, vehicles, mobile devices, and climate observations. Due to the limitations of technology, data could be inaccurate, inconsistent, and noisy. So preprocessing is needed to integrate and clean the raw data and to interpolate missing points. Mining is then performed on the preprocessed datasets stored in the moving object databases.

A rich set of data mining modules operate on top of the databases, enabling users to analyze data from different angles. The major functional modules we developed include periodic pattern mining, swarm pattern mining, trajectory clustering, and classification. The details of periodic pattern mining and swarm pattern mining are described in Section 3 and Section 4 separately.

The top layer shows the visualized results with some statistics. The visualized results can be plotted on 2D plane or embedded into other visualization tools (e.g., Google Map[1] and Google Earth[2]). Along with the visualized results, some statistics, if possible, are presented to provide users with more insights into these results.

[1]http://code.google.com/apis/maps/
[2]http://code.google.com/apis/earth/

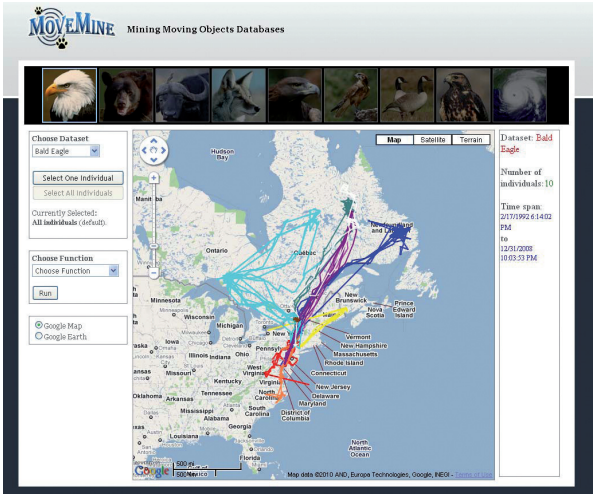


Fig. 2. Screenshot of the MoveMine system.

Figure 2 shows a screenshot of the MoveMine system. On the top of system, there are listed a bunch of real moving object datasets, most of them are animal movement obtained from MoveBank.org. The data can be visualized in Google Map or Google Earth. This screenshot shows the movement data of bald eagles. Each color represents the trajectory of each bald eagle. On the left side, people could select individual objects to analyze. After the dataset and moving objects in this dataset are selected, a user can choose the function to look into the data. Parameters for the selected function will be shown correspondingly. The default parameter values are set to the "optimal" ones derived by our heuristics. To better browse the results, outputs returned are visually displayed. Similarly, the results will be embedded in Google Map (as shown in Figure 2) and a user can zoom in/out or drag the map. Furthermore, a user can plot the results in Google Earth for 3D visualization of the results. There will be a text box on the right side to explain the results. We will show more system screenshots in Section 5.

3. MINING PERIODIC BEHAVIOR

3.1. Introduction

Periodic behaviors will provide people semantic understanding of the movement. For example, Figure 3 shows the raw movement data of a student David and the expected periodic behaviors. However, mining periodic behaviors is a challenging problem. Based on manual examination of the raw data (on the left), it is almost impossible to extract the periodic behaviors (on the right). And the periodic behaviors are actually quite complicated. There are multiple periods and periodic behaviors that may interleave with each other. Mining periodic behaviors can bridge the gap between raw data and semantic understanding of the data, which includes following two major issues.

First, the *periods* (i.e., the regular time intervals in a periodic behavior) are usually unknown. Even though there are many period detection techniques that are proposed in the signal processing area, such as Fourier transform and autocorrelation, these methods cannot be directly applied to the spatio-temporal data because the moving object will not repeat the movement by appearing at *exactly* the same point (in terms of (x, y)) on exactly the same time instance of a period. Besides, there could be multiple periods existing at the same time, such as David has one period as "day" and another as "week". If we consider the movement sequence as a whole, the longer period (i.e., week)

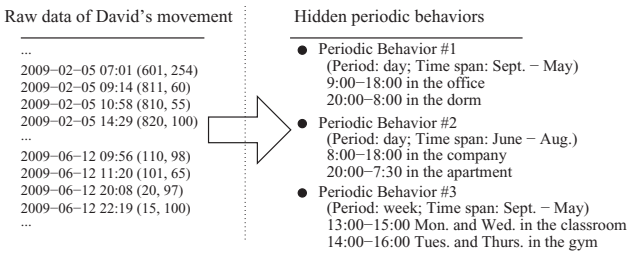


Fig. 3. Interleaving of multiple periodic behaviors.

will have fewer repeating times than the shorter period (i.e., day). So it is hard to select a threshold to find all periods. Surprisingly, there is no previous work that can handle the issue about how to detect multiple periods from the noisy moving object data. To the best of our knowledge, there is only one work [Bar-David et al. 2009] that addresses the detection of periods for moving objects. It directly applies the Fourier transform on moving object data by transforming a location onto a complex plane. However, as shown in the toy example we will show in Section 3.3, this method does not work in the presence of spatial noise.

Second, even if the periods are known, the periodic behaviors still need to be mined from the data because there could be several periodic behaviors with the same period. As we can see that, in David's movement, the same period (i.e., day) is associated with two different periodic behaviors, one from September to May and the other from June to August. In previous work, Mamoulis et al. [2004] studied the frequent periodic pattern mining problem for a moving object with a given period. However, the rigid definition of frequent periodic pattern does not encode the statistical information. It cannot describe the case such as "David has 0.8 probability to be in the office at 9:00 everyday." One may argue that these frequent periodic patterns can be further summarized using probabilistic modeling approach [Yan et al. 2005; Wang and Parthasarathy 2006]. But such models built on frequent periodic patterns do not truly reflect the real underlying periodic behaviors from the original movement, because frequent patterns are already a lossy summarization over the original data. Furthermore, if we can directly mine periodic behaviors on the original movement using polynomial-time complexity, it is unnecessary to mine frequent periodic patterns and then summarize over these patterns.

We formulate the periodic behavior mining problem and propose the assumption that the observed movement is generated from several periodic behaviors associated with some reference locations. We design a two-stage algorithm, Periodica, to detect the periods and further find the periodic behaviors.

At the first stage, we focus on detecting all the periods in the movement. Given the raw data as shown in Figure 3, we use the kernel method to discover those reference locations, namely reference spots. For each reference spot, the movement data is transformed from a spatial sequence to a binary sequence, which facilitates the detection of periods by filtering the spatial noise. Besides, based on our assumption, every period will be associated with at least one reference spot. All periods in the movement can be detected if we try to detect the periods in every reference spot. At the second stage, we statistically model the periodic behavior using a generative model. Based on this model, underlying periodic behaviors are generalized from the movement using a hierarchical clustering method and the number of periodic behaviors is automatically detected by measuring the representation error.

Table II. A Daily Periodic Behavior of David

	8:00	9:00	10:00	···	17:00	18:00	19:00
dorm	0.9	0.2	0.1	···	0.2	0.7	0.8
office	0.05	0.7	0.85	···	0.75	0.2	0.1
unknown	0.05	0.1	0.05	···	0.05	0.1	0.1

3.2. Problem Definition

Let $D = \{(x_1, y_1, time_1), (x_2, y_2, time_2), \ldots\}$ be the original movement database for a moving object. The raw data is linearly interpolated with constant time gap, such as hour or day. The interpolated sequence is denoted as $LOC = loc_1 loc_2 \cdots loc_n$, where loc_i is a spatial point represented as a pair $(loc_i.x, loc_i.y)$.

Given a location sequence LOC, our problem aims at mining all periodic behaviors. Before defining periodic behavior, we first define some concepts. A *reference spot* is a dense area that is frequently visited in the movement. The set of all reference spots is denoted as $O = \{o_1, o_2, \ldots, o_d\}$, where d is the number of reference spots. A *period* T is a regular time interval in the (partial) movement. Let t_i $(1 \leq i \leq T)$ denote the i-th *relative timestamp* in T.

A *periodic behavior* can be represented as a pair $\langle T, \mathbf{P} \rangle$, where $\mathbf{P}$ is a probability distribution matrix. Each entry $\mathbf{P}_{ik} (1 \leq i \leq d, 1 \leq k \leq T)$ of $\mathbf{P}$ is the probability that the moving object is at the reference spot o_i at relative timestamp t_k.

For example, for $T = 24$ (hours), David's daily periodic behavior (Figure 3 involved with 2 reference spots (i.e., "office" and "dorm") could be represented as $(2 + 1) \times 24$ probability distribution matrix, as shown Table II. This table is an intuitive explanation of formal output of periodic behaviors, which is not calculated according to specific data in Figure 3. The probability matrix encodes the noises and uncertainties in the movement. It statistically characterizes periodic behavior such as "David arrives at office around 9:00."

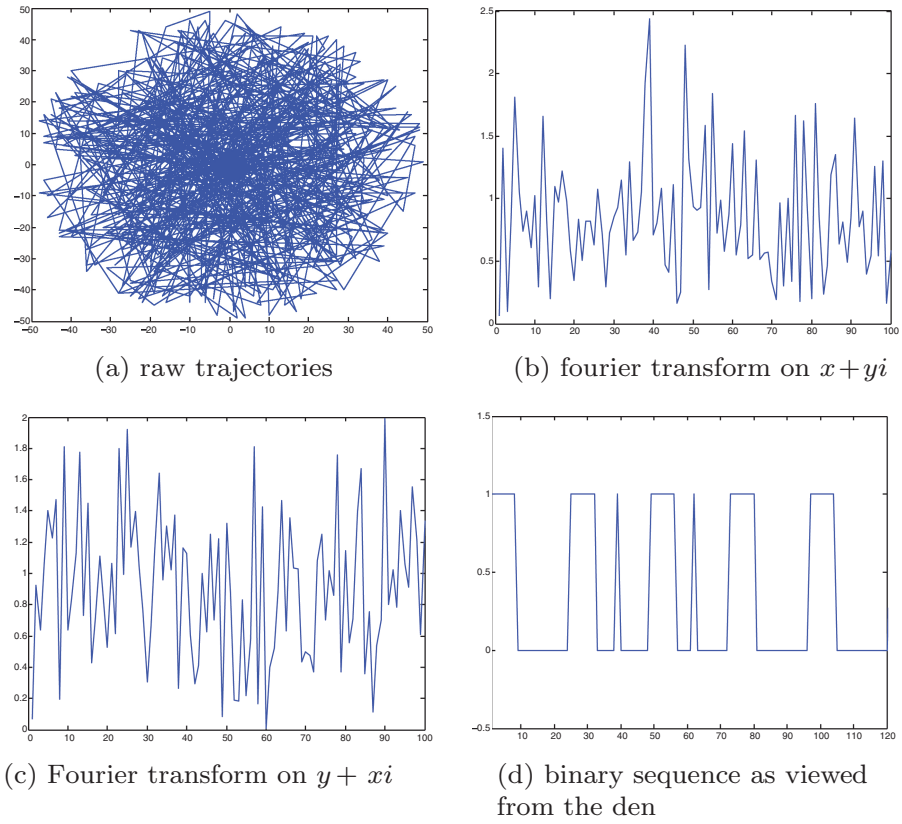
Definition 1 (*Periodic Behavior Mining*). Given a length-n movement sequence LOC, our goal is to mine all the periodic behaviors $\{\langle T, \mathbf{P} \rangle\}$.

There are two subtasks in the periodic behavior mining problem, detecting the periods and mining the periodic behaviors. We propose a two-stage algorithm Periodica, where the overall procedure of the algorithm is developed in two stages and each stage targets one subtask.

ALGORITHM 1: Periodica

INPUT: A movement sequence $LOC = loc_1 loc_2 \cdots loc_n$.
OUTPUT: A set of periodic behaviors.
ALGORITHM:

```
1: /* Stage 1: Detect periods (Section 3.3)*/
2: Find reference spots O = {o_1, o_2, ··· , o_d};
3: for each o_i ∈ O do
4:    Detect periods in o_i and store the periods in P_i;
5:    P_set ← P_set ∪ P_i;
6: end for
7: /* Stage 2: Mine periodic behaviors (Section 3.4) */
8: for each T ∈ P_set do
9:    O_T = {o_i | T ∈ P_i};
10:   Construct the symbolized sequence S using O_T;
11:   Mine periodic behaviors in S.
12: end for
```



(a) raw trajectories

(b) fourier transform on $x+yi$

(c) Fourier transform on $y+xi$

(d) binary sequence as viewed from the den

Fig. 4. Illustration of the importance to view movement from reference spots.

Algorithm 1 shows the general framework of Periodica. At the first stage, we first find all the reference spots (line 2) and for each reference spot, the periods are detected (line 3∼5). Then for every period T, we consider the reference spots with period T and further mine the corresponding periodic behaviors (line 7∼10).

3.3. Detecting Period

In this section, we will discuss how to detect periods in the movement data. This includes two subproblems, namely, finding reference spots and detecting periods on binary sequence generated by these spots. First of all, we want to show why the idea of reference spots is essential for period detection. Consider the following example.

We generate a movement dataset simulating an animal's daily activities. Every day, this animal has 8 hours staying at the den and the rest of the time going to some random places hunting for food. Figure 4(a) shows its trajectories. We first try the method introduced in Bar-David et al. [2009]. The method transforms locations (x, y) onto a complex plane and uses the Fourier transform to detect the periods. However, as shown in Figure 4(b) and Figure 4(c), there is no strong signal corresponding to the correct period because such a method is sensitive to spatial noise. If the object does not follow more or less the same hunting *route* every day, the period can hardly be detected. However, in real cases, few objects repeat the exactly same route in the periodic movement.

Our key observation is that, if we view the data from the den, the period is easier to detect. In Figure 4(d), we transform the movement into a binary sequence, where 1 represents the animal is at den and 0 when it goes out. It is easy to see the regularity in this binary sequence. Our idea is to find some important reference locations, namely reference spots, to view the movement. In this example, the den serves as our reference spot.

The notion of reference spots has several merits. First, it filters out the spatial noise and turns the period detection problem from a 2D space (i.e., spatial) to a 1D space (i.e., binary). As shown in Figure 4(d), we do not care where the animal goes when it is out of the den. As long as it follows a regular pattern going out and coming back to the den, there is a period associated with the den. Second, we can detect multiple periods in the movement. Consider the scenario that there is a daily period with one reference spot and a weekly period with another reference spot, it is possible that only period "day" is discovered because the shorter period will repeat more times. But if we view the movement from two reference spots separately, both periods can be individually detected. Third, based on the assumption that each periodic behavior is associated with some reference locations, all the periods can be found through reference spots.

The rest of this section will discuss in details how to find reference spots and detect the periods on the binary sequence for each reference spot.

3.3.1. Finding Reference Spots. Since an object with periodic movement will repeatedly visit some specific places, if we only consider the spatial information of the movement, reference spots are those dense regions containing more points than the other regions. Note that the reference spots are obtained for individual objects.

There are many methods could be applied to detect the reference spots, such as density-based clustering. The methods could vary according to different applications. We adapt a popular kernel method [Worton 1989] which is designed for the purpose of finding home ranges of animals. For human movement, we may use important location detection methods in Liao et al. [2005] and Zheng et al. [2010].

While computing the density for each location in a continuous space is computationally expensive, we discretize the space into a regular $w \times h$ grid and compute the density for each cell. The grid size is determined by the desired resolution to view the spatial data. If an animal has frequent activities at one place, this place will have higher probability to be its home. This actually aligns very well with our definition of reference spots.

For each grid cell c, the density is estimated using the bivariate normal density kernel

$$f(c) = \frac{1}{n\gamma^2} \sum_{i=1}^{n} \frac{1}{2\pi} \exp\left(-\frac{|c - loc_i|^2}{2\gamma^2} \right),$$

where $|c - loc_i|$ is the distance between cell c and location loc_i. In addition, γ is a smoothing parameter which is determined by the following heuristic method [Worton 1989]

$$\gamma = \frac{1}{2}\left(\sigma_x^2 + \sigma_y^2\right)^{\frac{1}{2}} n^{-\frac{1}{6}},$$

where σ_x and σ_y are the standard deviations of the whole sequence LOC in its x and y-coordinates, respectively. The time complexity for this method is $O(w \cdot h \cdot n)$.

After obtaining the density values, a reference spot can be defined by a contour line on the map, which joins the cells of the equal density value with some density threshold. The threshold can be determined as the top-p% density value among all the density values of all cells. The larger the value p is, the bigger the size of reference spot.

In practice, p can be chosen based on prior knowledge about the size of the reference spots. In many real applications, we can assume that the reference spots are usually very small on a large map (e.g., within 10% of whole area). So, by setting $p\% = 15\%$, most parts of reference spots should be detected with high probability.

3.3.2. Periods Detection on Binary Sequence. Given a set of reference spots, we further propose a method to obtain the potential periods within each spot separately. Viewed from a single reference spot, the movement sequence now can be transformed into a binary sequence $B = b_1 b_2 \dots b_n$, where $b_i = 1$ when this object is within the reference spot at timestamp i and 0 otherwise. In a discrete signal processing area, to detect periods in a sequence, the most popular methods are Fourier transform and autocorrelation, which essentially complement each other in the following sense, as discussed in Vlachos et al. [2005]. On one hand, Fourier transform often suffers the low-resolution problem in the low-frequency region, hence provides poor estimation of large periods. Also, the well-known spectral leakage problem of Fourier transform tends to generate a lot of false positives in the periodogram. On the other hand, autocorrelation offers accurate estimation for both short and large periods, but is more difficult to set the significance threshold for important periods. Consequently, Vlachos et al. [2005] proposed to combine Fourier transform and autocorrelation to find periods. Here, we adapt this approach to find periods in the binary sequence B.

In Discrete Fourier Transform (DFT), the sequence $B = b_1 b_2 \dots b_n$ is transformed into the sequence of n complex numbers $X_1, X_2, \dots, X_n$. Given coefficients X, the periodogram is defined as the squared length of each Fourier coefficient: $F_k = \|X_k\|^2$. Here, F_k is the power of frequency k. In order to specify which frequencies are important, we need to set a threshold and identify those higher frequencies than this threshold.

The threshold is determined using the following method. Let B' be a randomly permutated sequence from B. Since B' should not exhibit any periodicities, even the maximum power does not indicate the period in the sequence. Therefore, we record its maximum power as p_{max}, and only the frequencies in B that have higher power than p_{max} may correspond to real periods. To provide a 99% confidence level on what frequencies are important, we repeat the aforesaid random permutation experiment 100 times and record the maximum power of each permutated sequence. The 99-th largest value of these 100 experiments will serve as a good estimator of the power threshold.

Given that F_k is larger than the power threshold, we still need to determine the exact period in the time domain, because a single value k in *frequency domain* corresponds to a range of periods $[\frac{n}{k}, \frac{n}{k-1})$ in *time domain*. In order to do this, we use circular autocorrelation which examines how similar a sequence is to its previous values for different τ lags: $R(\tau) = \sum_{i=1}^{n} b_\tau b_{i+\tau}$.

Thus, for each period range $[l, r)$ given by the periodogram, we test whether there is a peak in $\{R(l), R(l+1), \dots, R(r-1)\}$ by fitting the data with a quadratic function. If the resulting function is concave in the period range, which indicates the existence of a peak, we return $t^* = \arg\max_{l \le t < r} R(t)$ as a detected period. Similarly, we employ a 99% confidence level to eliminate false positives caused by noise.

For example, if a binary sequence has a period as $T = 24$, the periodogram could be Figure 5(a). The red dashed line denotes the threshold of 99% confidence. There are two points P_1 and P_2 that are above the threshold. In Figure 5(b), P_1 and P_2 are mapped to a range of periods. We can see that there is only one peak, P_1, corresponding to $T = 24$ on the autocorrelation curve.

3.4. Mining Periodic Behaviors

After obtaining the periods for each reference spot, now we study the task how to mine periodic behaviors. We will consider the reference spots with the same period together

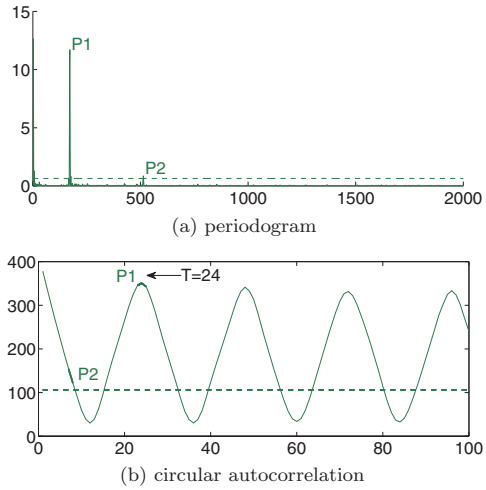


Fig. 5. Finding periods.

in order to obtain more concise and informative periodic behaviors. But, since a behavior may only exist in a partial movement, there could be several periodic behaviors with the same period. For example, there are two daily behaviors in David's movement. One corresponds to the school days and the other one occurs during the summer. However, given a long history of movement and a period as a "day," we actually do not know how many periodic behaviors exist in this movement and which days belong to which periodic behavior. This motivates us to use a clustering method. Because the "days" that belong to the same periodic behavior should have a similar temporal location pattern, we propose a generative model to measure the distance between two "days." Armed with such a distance measure, we can further group the "days" into several clusters and each cluster represents one periodic behavior. As in David's example, "school days" should be grouped into one cluster and "summer days" should be grouped into another one. Note that we assume that for each period, such as "day," one "day" will only belong to one behavior.

In this section, we will formally present the technique to mine periodic behaviors. Since every period in the movement will be considered separately, the rest of this section will focus on one specific period T.

3.4.1. Modeling Periodic Behaviors. First, we retrieve all the reference spots with period T. By combining the reference spots with the same period together, we will get more informative periodic behaviors associated with different reference spots. For example, we can summarize David's daily behavior as "9:00∼18:00 at office *and* 20:00∼8:00 in the dorm." We do not consider combining two different periods in the current work.

Let $O_T = \{o_1, o_2, \ldots, o_d\}$ denote reference spots with period T. For simplicity, we denote o_0 as any other locations outside the reference spots $o_1, o_2, \ldots, o_d$. Given $LOC = loc_1 loc_2 \cdots loc_n$, we generate the corresponding *symbolized movement sequence* $S = s_1 s_2 \ldots s_n$, where $s_i = j$ if loc_i is within o_j. S is further segmented into $m = \lfloor \frac{n}{T} \rfloor$ *segments*.[3] We use I^j to denote the j-th segment and t_k $(1 \leq k \leq T)$ to denote the k-th relative timestamp in a period. $I^j_k = i$ means that the object is within o_i at t_k in the j-th segment. For example, for $T = 24$ (hours), a segment represents a "day," t_9

[3]If n is not a multiple of T, then the last $(n \mod T)$ positions are truncated.

denotes 9:00 in a day, and $I_9^5 = 2$ means that the object is within o_2 at 9:00 in the 5-th day. Naturally, we may use the categorical distribution to model the probability of such events.

Definition 2 (*Categorical Distribution Matrix*). Let $\mathcal{T} = \{t_1, t_2, \ldots, t_T\}$ be a set of relative timestamps, x_k be the categorical random variable indicating the selection of reference spot at timestamp t_k. $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_T]$ is a categorical distribution matrix with each column $\mathbf{p}_k = [p(x_k = 0), p(x_k = 1), \ldots, p(x_k = d)]^T$ being an independent categorical distribution vector satisfying $\sum_{i=0}^{d} p(x_k = i) = 1$.

Now, suppose I^1, I^2, ..., I^l follow the same periodic behavior. The probability that the segment set $\mathcal{I} = \bigcup_{j=1}^{l} I^j$ is generated by some distribution matrix $\mathbf{P}$ is

$$P(\mathcal{I}|\mathbf{P}) = \prod_{I^j \in \mathcal{I}} \prod_{k=1}^{T} p\left(x_k = I_k^j\right).$$

Now, we formally define the concept of periodic behavior.

Definition 3 (*Periodic Behavior*). Let $\mathcal{I}$ be a set of segments. A periodic behavior over all the segments in $\mathcal{I}$, denoted as $\mathbf{H}(\mathcal{I})$, is a pair $\langle T, \mathbf{P} \rangle$. T is the period and $\mathbf{P}$ is a probability distribution matrix. We further let $|\mathcal{I}|$ denote the number of segments covered by this periodic behavior.

3.4.2. Discovery of Periodic Behaviors. With the definition of periodic behaviors, we are able to estimate periodic behaviors over a set of segments. Now given a set of segments $\{I^1, I^2, \ldots, I^m\}$, we need to discover which segments are generated by the same periodic behavior. Suppose there are K underlying periodic behaviors, each of which exists in a partial movement, the segments should be partitioned into K groups so that each group represents one periodic behavior.

A potential solution to this problem is to apply some clustering methods. In order to do this, a distance measure between two periodic behaviors needs to be defined. Since a behavior is represented as a pair $\langle T, \mathbf{P} \rangle$ and T is fixed, the distance should be determined by their probability distribution matrices. Further, a small distance between two periodic behaviors should indicate that the segments contained in each behavior are likely to be generated from the same periodic behavior.

Several measures between the two probability distribution matrices $\mathbf{P}$ and $\mathbf{Q}$ can be used to fulfill these requirements. Here, since we assume the independence of variables across different timestamps, we propose to use the well-known Kullback-Leibler divergence as our distance measure.

$$KL(\mathbf{P}\|\mathbf{Q}) = \sum_{k=1}^{T} \sum_{i=0}^{d} p(x_k = i) \log \frac{p(x_k = i)}{q(x_k = i)}$$

When $KL(\mathbf{P}\|\mathbf{Q})$ is small, it means that the two distribution matrices $\mathbf{P}$ and $\mathbf{Q}$ are similar, and vice versa.

Suppose there exist K underlying periodic behaviors, there are many ways to group the segments into K clusters with the distance measure defined. However, the number of underlying periodic behaviors (i.e., K) is usually unknown. So we propose a hierarchical agglomerative clustering method to group the segments while at the same time determining the optimal number of periodic behaviors. Ideally, during the hierarchical agglomerative clustering, the segments generated from the same behavior should be merged first because they have smaller KL-divergence distance. Thus, we judge a cluster as good if all the segments in the cluster are concentrated in one single reference

spot at a particular timestamp. Hence, a natural representation error measure to evaluate the representation quality of a cluster is as follows. Note that here we exclude the reference spot o_0 which essentially means the location is unknown.

Definition 4 (*Representation Error*). Given a set of segments $C = \{I^1, I^2, \ldots, I^l\}$ and its periodic behavior $\mathbf{H}(C) = \langle T, \mathbf{P} \rangle$, the representation error is

$$E(C) = \frac{\sum_{I^j \in C} \sum_{i=1}^{T} \mathbf{1}_{I_i^j \neq 0} \cdot \left(1 - p\left(x_i = I_i^j\right)\right)}{\sum_{I^j \in C} \sum_{i=1}^{T} \mathbf{1}_{I_i^j \neq 0}}.$$

At each iteration, all the segments are partitioned into k clusters $\{C_1, C_2, \ldots, C_k\}$. The overall representation error at current iteration is calculated as the mean over all clusters.

$$\mathcal{E}_k = \frac{1}{k} \sum_{i=1}^{k} E(C_i)$$

During the clustering process, we monitor the change of $\mathcal{E}_k$. If $\mathcal{E}_k$ exhibits a dramatic increase comparing with $\mathcal{E}_{k-1}$, it is a sign the newly merged cluster may contain two different behaviors and $k-1$ is likely to be a good choice of K. The degree of such change can be observed from the derivative of $\mathcal{E}$ over k, $\frac{\partial \mathcal{E}}{\partial k}$. Since a sudden increase of $\mathcal{E}$ will result in a peak in its derivative, we can find the optimal K as $K = \arg\max_k \frac{\partial \mathcal{E}}{\partial k}$.

4. MINING SWARM PATTERNS

4.1. Introduction

A moving object cluster can be defined in both spatial and temporal dimensions: (1) a group of moving objects should be geometrically close to each other, and (2) they should be together for at least some minimum time duration.

There have been many recent studies on mining moving object clusters. One line of study is to find moving object clusters including moving clusters [Kalnis et al. 2005], flocks [Gudmundsson et al. 2004; Gudmundsson and van Kreveld 2006; Benkert et al. 2008], and convoys [Jeung et al. 2008c, 2008b]. The common part of such patterns is that they require the group of moving objects to be together for at least k consecutive timestamps, which might not be practical in the real cases.

Another line of study of moving object clustering is trajectory clustering [Vlachos et al. 2002; Chen et al. 2005; Gaffney et al. 2006; Lee et al. 2007], which puts emphasis on geometric or spatial closeness of object trajectories. However, objects that are essentially moving together may not share similar geometric trajectories. In real life, there are often cases that a set of moving objects (e.g., birds, flies, and mammals) hardly stick together all the time; they do travel together, but only gather together at some timestamps.

We propose a new movement pattern, called *swarm*, which is a more general type of moving object clusters. More precisely, swarm is a group of moving objects containing at least min_o individuals who are in the same cluster for at least min_t timestamp snapshots. If we denote this group of moving objects as O and the set of these timestamps as T, a swarm is a pair (O, T) that satisfies the precedly constraints. Specially, the timestamps in T are not required to be consecutive, the detailed geometric trajectory of each object becomes unimportant, and clustering methods and/or measures can be flexible and application dependent (e.g., density-based clustering versus Euclidean distance-based clustering). To avoid finding redundant swarms, we further propose the *closed swarm* concept. The basic idea is that if (O, T) is a swarm, it is unnecessary to output any subset $O' \subseteq O$ and $T' \subseteq T$ even if (O', T') may also satisfy swarm requirements.

Efficient discovery of complete set of closed swarms in a large moving object database is a nontrivial task. First, the size of all the possible combinations is exponential (i.e., $2^{|O_{DB}|} \times 2^{|O_{TB}|}$) whereas the discovery of moving clusters, flocks, or convoys has polynomial solution due to stronger constraint posed by their definitions based on k consecutive timestamps. Second, although the problem is defined using the similar form of frequent pattern mining [Agrawal and Srikant 1994; Han et al. 2004], none of previous work [Agrawal and Srikant 1994; Han et al. 2004; Zaki and Hsiao 2002; Yan et al. 2003; Pei et al. 2000; Wang et al. 2003] solves exactly the same problem as finding swarms. In the typical frequent pattern mining problem, the input is a set of transactions and each transaction contains a set of items. However, the input of our problem is a sequence of timestamps and there is a collection of (overlapping) clusters at each timestamp. Thus, the discovery of swarms poses a new problem that needs to be solved by specifically designed techniques.

Facing the huge potential search space, we propose an efficient method, ObjectGrowth. In ObjectGrowth, besides the *Apriori Pruning Rule* which is commonly used, we design a novel *Backward Pruning Rule* which uses a simple checking step to stop unnecessary further search. Such a pruning rule could cover several redundant cases at the same time. After our pruning rules cut a great portion of unpromising candidates, the leftover number of candidate closed swarms could still be large. To avoid the time-consuming pairwise closure checking in the postprocessing step, we present a *Forward Closure Checking* step that can report the closed swarms on-the-fly. Using this checking rule, no space is needed to store candidates and no extra time is spent on postprocessing to check closure property.

4.2. Problem Definition

Let $O_{DB} = \{o_1, o_2, \ldots, o_n\}$ be the set of all moving objects and $T_{DB} = \{t_1, t_2, \ldots, t_m\}$ be the set of all timestamps in the database. A subset of O_{DB} is called an *objectset* O. A subset of T_{DB} is called a *timeset* T. The *size*, $|O|$ and $|T|$, is the number of objects and timestamps in O and T, respectively.

Database of clusters. A database of clusters, $C_{DB} = \{C_{t_1}, C_{t_2}, \ldots, C_{t_m}\}$, is the collection of snapshots of the moving object clusters at timestamps $\{t_1, t_2, \ldots, t_m\}$. We use $C_{t_i}(o_j)$ to denote the set of clusters that object o_j is in at timestamp t_i. Note that an object could belong to several clusters at one timestamp. In addition, for a given objectset O, we write $C_{t_i}(O) = \bigcap_{o_j \in O} C_{t_i}(o_j)$ for short. To make our framework more general, we take clustering as a preprocessing step. The clustering methods could be different based on various scenarios.

Swarm and closed swarm. A pair (O, T) is said to be a *swarm* if all objects in O are in the same cluster at any timestamp in T. Specifically, given two minimum thresholds min_o and min_t, for (O, T) to be a swarm, where $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_p}\} \subseteq O_{DB}$ and $T \subseteq T_{DB}$, it needs to satisfy three requirements.

(1) $|O| \geq min_o$: There should be at least min_o objects.
(2) $|T| \geq min_t$: Objects in O are in the same cluster for at least min_t timestamps.
(3) $C_{t_i}(o_{i_1}) \cap C_{t_i}(o_{i_2}) \cap \cdots \cap C_{t_i}(o_{i_p}) \neq \emptyset$ for any $t_i \in T$: there is at least one cluster containing all the objects in O at each timestamp in T.

To avoid mining redundant swarms, we further give the definition of *closed swarm*. A swarm (O, T) is *object-closed* if fixing T, O cannot be enlarged ($\nexists O'$ such that (O', T) is a swarm and $O \subsetneq O'$). Similarly, a swarm (O, T) is *time-closed* if fixing O, T cannot be enlarged ($\nexists T'$ such that (O, T') is a swarm and $T \subsetneq T'$). Finally, a swarm (O, T) is a closed swarm iff it is both object-closed and time-closed. Our goal is to find the complete set of closed swarms.

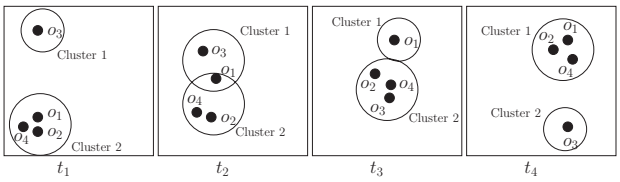


Fig. 6. Snapshots of object clusters at t_1 to t_4.

We use the following example as a running example in the remaining sections to give an intuitive explanation of our methods. We set $min_o = 2$ and $min_t = 2$ in this example.

Example 1 (*Running Example*). Figure 6 shows the input of our running example. There are 4 objects and 4 timestamps ($O_{DB} = \{o_1, o_2, o_3, o_4\}$, $T_{DB} = \{t_1, t_2, t_3, t_4\}$). Each subfigure is a snapshot of object clusters at each timestamp. It is easy to see that o_1, o_2, and o_4 travel together for most of the time, and o_2 and o_4 form an even more stable swarm since they are close to each other in the whole time span. Given $min_o = 2$ and $min_t = 2$, there are totally 15 swarms: ($\{o_1, o_2\}$, $\{t_1, t_2\}$), ($\{o_1, o_4\}$, $\{t_1, t_2\}$), ($\{o_2, o_4\}$, $\{t_1, t_3, t_4\}$), and so on.

But it is obviously redundant to output swarms like ($\{o_2, o_4\}$, $\{t_1, t_2\}$) and ($\{o_2, o_4\}$, $\{t_2, t_3, t_4\}$) (not time-closed) since both of them can be enlarged to form another swarm: ($\{o_2, o_4\}$, $\{t_1, t_2, t_3, t_4\}$). Similarly, ($\{o_1, o_2\}$, $\{t_1, t_2, t_4\}$) and ($\{o_2, o_4\}$, $\{t_1, t_2, t_4\}$) are redundant (not object-closed) for both of them can be enlarged as ($\{o_1, o_2, o_4\}$, $\{t_1, t_2, t_4\}$). There are only two closed swarms in this example: ($\{o_2, o_4\}$, $\{t_1, t_2, t_3, t_4\}$) and ($\{o_1, o_2, o_4\}$, $\{t_1, t_2, t_4\}$).

4.3. Algorithm Overview

The pattern we are interested in here, swarm, is a pair (O, T) of objectset O and timeset T. At the first glance, the number of different swarms could be $(2^{|O_{DB}|} \times 2^{|T_{DB}|})$, that is, the size of the search space. However, for a *closed* swarm, the following lemma shows that if the objectset is given, the corresponding maximal timeset can be uniquely determined.

LEMMA 1. *For any swarm (O, T), $O \neq \emptyset$, there is a unique time-closed swarm (O, T') such that $T \subseteq T'$.*

In the running example, if we set the objectset as $\{o_1, o_2\}$, its maximal corresponding timeset is $\{t_1, t_2, t_4\}$. Thus, we only need to search all subsets of O_{DB}. In this way, the search space shrinks from $(2^{|O_{DB}|} \times 2^{|T_{DB}|})$ to $2^{|O_{DB}|}$. Note that the time complexity in the worst case is $O(c \times 2^{|}O_{DB}|)$, where c is time spent at the each node in the search.

Basic idea of our algorithm. From the preceding analysis we see that, to find closed swarms, it suffices to only search all the subsets O of moving objects O_{DB}. For the search space of O_{DB}, we perform depth-first search of all subsets of O_{DB}, which is illustrated as preorder tree traversal in Figure 7: tree nodes are labeled with numbers denoting the depth-first search order (nodes without numbers are pruned).

Despite this, the search space is still huge for enumerating the objectsets in O_{DB} $(2^{|O_{DB}|})$. So efficient pruning rules are demanded to speed up the search process. We design two efficient pruning rules to further shrink the search space. The first pruning rule, called Apriori Pruning Rule, is to stop traversing the subtree when we find further traversal cannot satisfy min_t. The second pruning rule, called Backward Pruning Rule, is to make use of the closure property. It checks whether there is a superset of the current objectset which has the same maximal corresponding timeset as that of the current one. If so, the traversal of the subtree under the current objectset is meaningless. In previous works [Pei et al. 2000; Zaki and Hsiao 2002; Wang et al. 2003]

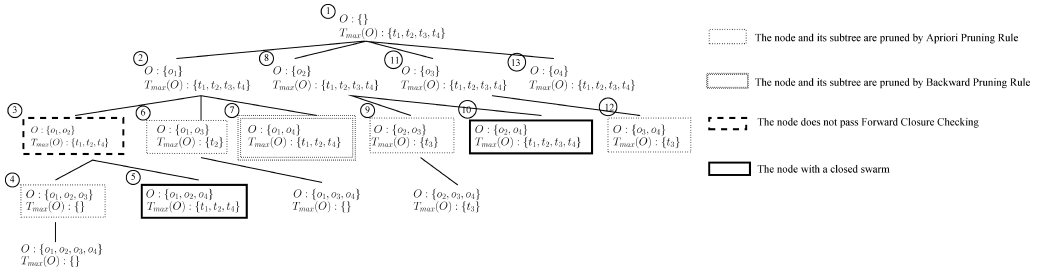


Fig. 7. ObjectGrowth search space ($min_o = 2, min_t = 2$).

on closed frequent pattern mining, there are three pruning rules (i.e., item-merging, sub-itemset pruning, and item skipping) to cover different redundant search cases. We simply use one pruning rule to cover all these cases and we will prove that we only need to examine each superset with one more object of the current objectset. Armed with these two pruning rules, the size of the search space can be significantly reduced.

After pruning the invalid candidates, the remaining ones may or may not be closed swarms. A brute-force solution is to check every pair of the candidates to see if one makes the other violate the closed swarm definition. But the time spent on this post-processing step is the square of the number of candidates, which is costly. Our proposal, Forward Closure Checking, is to embed a checking step in the search process. This checking step immediately determines whether a swarm is closed after the subtree under the swarm is traversed, and takes little extra time (actually, $O(1)$ additional time for each swarm in the search space). Thus, closed swarms are discovered on-the-fly and no extra postprocessing step is needed.

In the following subsections, we present the details of our ObjectGrowth algorithm.

4.4. The ObjectGrowth Method

ObjectGrowth method is a Depth-First Search (DFS) framework based on the *objectset search space* (i.e., the collection of all subsets of O_{DB}). First, we introduce the definition of maximal timeset. Intuitively, for an objectset O, the *maximal timeset* $T_{max}(O)$ is the one such that $(O, T_{max}(O))$ is a time-closed swarm. For an objectset O, the maximal timeset $T_{max}(O)$ is well-defined, because Lemma 1 shows the uniqueness of $T_{max}(O)$.

Definition 5 (*Maximal Timeset*). Timeset $T = \{t_j\}$ is a *maximal timeset* of objectset $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_m}\}$ if:

(1) $C_{t_j}(o_{i_1}) \cap C_{t_j}(o_{i_2}) \cap \cdots \cap C_{t_j}(o_{i_m}) \neq \emptyset$, $\forall t_j \in T$;

(2) $\nexists t_x \in T_{DB} \setminus T$, such that $C_{t_x}(o_{i_1}) \cap \cdots \cap C_{t_x}(o_{i_m}) \neq \emptyset$. We use $T_{max}(O)$ to denote the maximal timeset of objectset O.

In the running example, for $O = \{o_1, o_2\}$, $T_{max}(O) = \{t_1, t_2, t_4\}$ is the maximal timeset of O.

The objectset space is visited in a DFS order. When visiting each objectset O, we compute its maximal timeset. And three rules are further used to prune redundant search and detect the closed swarms on-the-fly.

4.4.1. Apriori Pruning Rule. The following lemma is from the definition of T_{max}.

LEMMA 2. *If $O \subseteq O'$, then $T_{max}(O') \subseteq T_{max}(O)$.*

This lemma is intuitive. When objectset grows bigger, the maximal timeset will shrink or at most keep the same. This further gives the following pruning rule.

Rule 1 (*Apriori Pruning Rule*). For an objectset O, if $|T_{max}(O)| < min_t$, then there is no strict superset O' of O ($O' \neq O$) such that $(O', T_{max}(O'))$ is a (closed) swarm.

In Figure 7, the nodes with objectset $O = \{o_1, o_3\}$ and its subtree are pruned by the Apriori Pruning Rule, because $T_{max}(O) < min_t$, and all objectsets in the subtree are strict supersets of O. Similarly, for the objectsets $\{o_2, o_3\}$, $\{o_3, o_4\}$ and $\{o_1, o_2, o_3\}$, the nodes with these objectsets and their subtrees are also pruned by the Apriori Pruning Rule.

4.4.2. Backward Pruning Rule. By using the Apriori Pruning Rule, we prune objectsets O with $T_{max}(O) < min_t$. However, the pruned search space could still be extremely huge as shown in the following example.

Suppose there are 100 objects which are all in the same cluster for the whole time span. Given $min_o = 1$ and $min_t = 1$, we can hardly prune any node using this the Apriori Pruning Rule. The number of objectsets we need to visit is 2^{100}! But it is easy to see that there is only one closed swarm: (O_{DB}, T_{DB}). We can get this closed swarm when we visit the objectset $O = O_{DB}$ in the DFS after 100 iterations. After that, we waste a lot of time searching objectsets which can never produce any closed swarms.

Since our goal is to mine only closed swarms, we can develop another stronger pruning rule to prune the subtrees which cannot produce closed swarms. Let us take some observations in the running example first.

In Figure 7, for the node with objectset $O = \{o_1, o_4\}$, we can insert o_2 into O and form a superset $O' = \{o_1, o_2, o_4\}$. O' has been visited and expanded before visiting O. And we can see that $T_{max}(O) = T_{max}(O') = \{t_1, t_2, t_4\}$. This indicates that for any timestamp when o_1 and o_4 are together, o_2 will also be in the same cluster as them. So for any superset of $\{o_1, o_4\}$ without o_2, it can never form a closed swarm. Meanwhile, o_2 will not be in O's subtree in the depth-first search order. Thus, the node with $\{o_1, o_4\}$ and its subtree can be pruned.

To formalize the Backward Pruning Rule, we first state the following lemma.

LEMMA 3. *Consider an objectset $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_m}\}$ ($i_1 < i_2 < \ldots < i_m$), if there exists an objectset O' such that O' is generated by adding an additional object $o_{i'}$ ($o_{i'} \notin O$ and $i' < i_m$) into O such that $C_{t_j}(O) \subseteq C_{t_j}(o_{i'})$, $\forall t_j \in T_{max}(O)$, then for any objectset O'' satisfying $O \subseteq O''$ but $O' \nsubseteq O''$, $(O'', T_{max}(O''))$ is not a closed swarm.*

Note that when overlapping is not allowed in the clusters, the condition $C_{t_j}(O) \subseteq C_{t_j}(o_{i'})$, $\forall t_j \in T_{max}(O)$ simply reduces to $T_{max}(O') = T_{max}(O)$. Armed with the preceding lemma, we have the following pruning rule.

Rule 2 (*Backward Pruning Rule*). Let $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_m}\}$ ($i_1 < i_2 < \ldots < i_m$) be an objectset. If there exists an objectset O' such that O' is generated by adding an additional object $o_{i'}$ ($o_{i'} \notin O$ and $i' < i_m$) into O such that $C_{t_j}(O) \subseteq C_{t_j}(o_{i'})$, $\forall t_j \in T_{max}(O)$, then O can be pruned in the objectset search space (stop growing from O in the depth-first search).

Backward Pruning Rule is efficient in the sense that it only needs to examine those supersets of O with one more object rather than all the supersets. This rule can prune a significant portion of the search space for mining closed swarms. Experimental results (see Figure 15) show that the speedup (compared with the algorithms for mining all swarms without this rule) is an exponential factor with respect to the dataset size.

4.4.3. Forward Closure Checking. To check whether a swarm $(O, T_{max}(O))$ is closed, from the definition of closed swarm, we need to check every superset O' of O and $T_{max}(O')$. But, actually, according to the following lemma, checking the superset O' of O with one more object suffices.

LEMMA 4. *The swarm* $(O, T_{max}(O))$ *is closed iff for any superset* O' *of* O *with exactly one more object, we have* $|T_{max}(O')| < |T_{max}(O)|$.

In Figure 7, the node with objectset $O = \{o_1, o_2\}$ is not pruned by any pruning rules. But it has a child node with objectset $\{o_1, o_2, o_4\}$ having same maximal timeset as $T_{max}(O)$. Thus $(\{o_1, o_2\}, \{t_1, t_2, t_4\})$ is not a closed swarm because of Lemma 4.

Consider a superset O' of objectset $O = \{o_{i_1}, \ldots, o_{i_m}\}$ s.t. $O' \setminus O = \{o_{i'}\}$. Rule 2 checks the case that $i' < i_m$. The following rule checks the case that $i' > i_m$.

Rule 3 (*Forward Closure Checking*). Let $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_m}\}$ $(i_1 < i_2 < \ldots < i_m)$ be an objectset. If there exists an objectset O' such that O' is generated by adding an additional object $o_{i'}$ ($o_{i'} \notin O$ and $i' > i_m$) into O, and $|T_{max}(O')| = |T_{max}(O)|$, then (O, T) is not a closed swarm.

Note, unlike Rule 2, Rule 3 does not prune the objectset O in the DFS. In other words, we cannot stop DFS from O. But this rule is useful for detecting nonclosed swarms.

4.4.4. ObjectGrowth *Algorithm.* Figure 7 shows the complete ObjectGrowth algorithm for our running example. We traverse the search space in DFS order. When visiting the node with $O = \{o_1, o_2, o_3\}$, it fails to pass the Apriori Pruning Rule. So we stop growing from it, trace back, and visit node $O = \{o_1, o_2, o_4\}$. O passes both pruning rules as well as Forward Closure Checking. By Theorem 1 that will be introduced immediately afterwards, O and its maximal timeset $T = \{t_1, t_2, t_4\}$ form a closed swarm. So we can output (O, T). When we trace back to node $\{o_1, o_2\}$, because its child contains a closed swarm with the same timeset as $\{o_1, o_2\}$'s maximal timeset, $\{o_1, o_2\}$ will not be a closed swarm by the Forward Closure Checking. We continue visiting the nodes until we finish the traversal of objectset-based DFS tree.

THEOREM 1 (IDENTIFICATION OF CLOSED SWARM IN OBJECTGROWTH). *For a node with objectset* O, $(O, T_{max}(O))$ *is a closed swarm* ***if and only if*** *it passes the Apriori Pruning Rule, Backward Pruning Rule, Forward Closure Checking, and* $|O| \geq min_o$.

Theorem 1 makes the discovery of closed swarms well embedded in the search process so that closed swarms can be reported on-the-fly.

5. EXPERIMENT

In this section, we present the experimental results along with the MoveMine system. We focus on the study of periodic behavior mining and swarm pattern mining, which are described in Section 5.1 and Section 5.2 separately. In each function study, we show the function in the MoveMine system, advanced effectiveness study on another dataset, and effectiveness/efficiency study with different parameters.

Most of the datasets are from MoveBank.org. The MoveMine was implemented in C#. All the efficiency experiments are carried out on a 2.8 GHz Intel Core 2 Duo system with 4GB memory.

5.1. Periodic Behavior Mining

5.1.1. Periodic Behavior Function in MoveMine. We now test our method on a real bald eagle movement. We pick this bald eagle data because this bald eagle has an obvious yearly migration pattern that has already been verified by biologists. We want to test our methods to see whether we can successfully detect such periodic behavior. The data contains a 3-year tracking (2006.1∼2008.12) of a bald eagle in North America. In the MoveMine system, people can select an individual moving object. Figure 8 shows the movement data of one bald eagle in Google Map. It is an enlarged area of Northeast in America and Quebec area in Canada. The data is preprocessed by linearly interpolation using the sampling rate as a day.

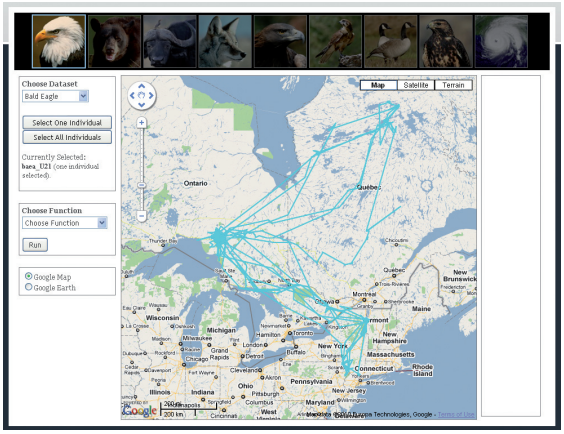
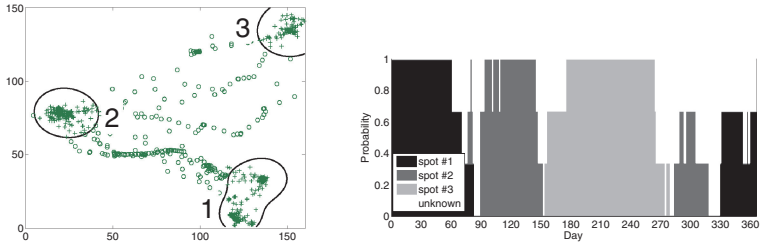


Fig. 8. Trajectory of one bald eagle.



(a) Reference spots (b) Periodic behavior

Fig. 9. Intermediate results of periodic behaviors

As shown in Figure 9(a), three reference spots are detected in areas of New York, Great Lakes, and Quebec. By applying period detection to each reference spot, we obtain the periods for each reference spots, which are 363, 363, and 364 days, respectively. The periods can be roughly explained as a year. It is a sign of yearly migration in the movement.

Now we check the periodic behaviors mined from the movement. Ideally, we want to consider three reference spots together because they all show the yearly period. However, we may discover that the periods are not exactly the same for all the reference spots. This is a very practical issue. In real cases, we can hardly get perfectly the same period for some reference spots. So, we should relax our constraint and consider the reference spots with *similar* periods together. If the difference of periods is within some tolerance threshold, we take the average of these periods and set it as the common period. Here, we take period T as 363 days, and the probability matrix is summarized in Figure 9(b). Using such probability matrix, we can well explain the yearly migration behavior as follows.

> This golden eagle stays in the New York area (i.e., reference spot # 1) from December to March. In March, it flies to the Great Lakes area (i.e., reference spot #2) and stays there until the end of May. It flies to the Quebec area (i.e., reference spot #3) in the summer and stays there until late September. Then it flies back to Great Lakes again staying there from mid-October to mid-November and goes back to New York in December.

In the MoveMine system, Figure 10 shows the periodic route by take the "average" locations over 3 years. This real example shows the periodic behaviors

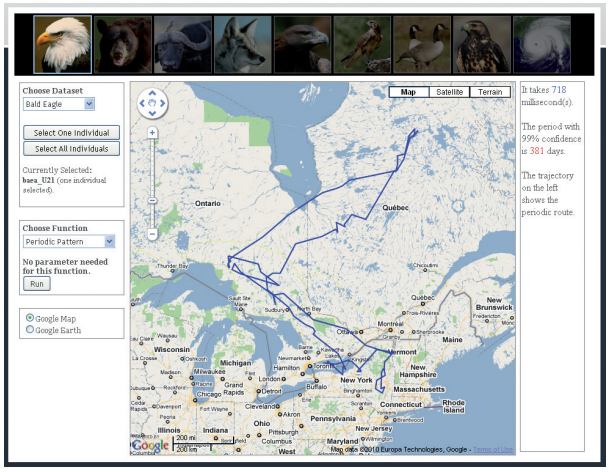


Fig. 10. Periodic route of one bald eagle.

mined from the movement and provides an insightful explanation for the movement data.

5.1.2. Synthetic Movement. In order to test the effectiveness under various scenarios, we design a generator for moving objects with periodicity according to a set of parameter values. These parameters are the length n of the time history (in timestamps), period T, the probability α for a periodic segment in the object's movement to comply with regular movement, the probability β for the noise for each timestamp in a regular periodic segment, and the variance σ of normal distribution to add temporal perturbations to the periodic segment.

Before generating the movement, we first create several reference spots. Each reference spot is a small circle with radius ranges from 1% to 5% of the map size. A standard segment seg_{std} with length T is the movement following the regular periodic pattern. For example, for $T = 24$ (hours), seg_{std} could be designed as 6:00pm∼8:00am at reference spot A (such as home) and 8:30am∼5:30pm hours at reference spot B (such as office). Then, the movement of the object is generated. For every segment *seg*, we first determine whether s should be a regular segment or not, given the probability α.

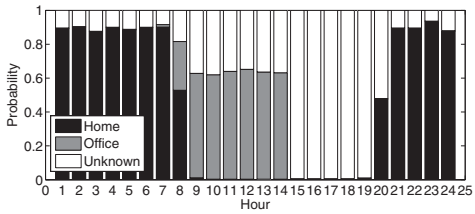
If *seg* is a regular segment, the object's movement is generated as follows. According to standard segment, suppose that from timestamp t_0 to t_1 the object is at reference spot A, we further perturb t_0 and t_1 with some normal distribution (i.e., $t_0' = N(t_0, \sigma^2)$, $t_1' = N(t_1, \sigma^2)$). For all the experiments, we fix $\sigma = 0.5$. Finally, with probability $1 - \beta$, the object is at a random location within the circle of reference spot A from t_0' to t_1'. For other timestamps that are not confined to any reference spot, a random location is generated. If *seg* is an irregular segment, for each timestamp, a random location is assigned.

Suppose that there are 4 reference spots. Imagine them as "home", "office", "gym", and "class". A standard movement segment is generated as 20:00∼8:00 at home every day; 9:00∼14:00 at office on weekdays; 15:00∼17:00 at gym on Tuesdays and Thursdays; 15:00∼17:00 at class on Mondays, Wednesdays, and Fridays. Furthermore, we choose $n = 8400$, $\alpha = 0.9$ and $\beta = 0.1$.

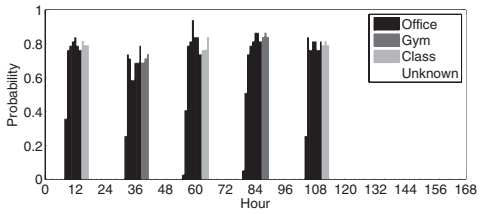
The periods detected for each reference spot are shown in Table III. There are two periods detected: 24 (i.e., day) and 168 (i.e., week). It is interesting to see that office

Table III. Periods Detected

Obs. Spot	Home	Office	Gym	Class
Periods (hours)	24	24, 168	168	168



(a) periodic behavior for $T = 24$



(b) periodic behavior for $T = 168$

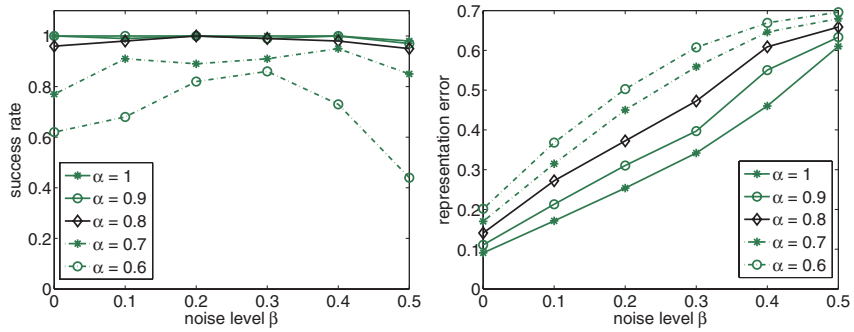
Fig. 11. Periodic behaviors.

has both 24 and 168 as the periods. This is because office is visited "almost" every day except weekends. So both day and week are reasonable periods.

There is one daily behavior and one weekly behavior. Their probability matrices are illustrated in Figure 11. In Figure 11(a), we can infer that this person leaves home *around* 8:00am because the probability starts to drop at 8:00am. In the weekly movement shown in Figure 11(b), 9:00∼14:00 weekdays, the person stays in the office with high probability. Gym is involved with Tuesday and Thursday afternoons and class is involved with Monday, Wednesday, and Friday afternoons. The behaviors on weekends are unknown.

5.1.3. Effectiveness Study with respect to Parameters. We further verify the effectiveness of our algorithms with respect to the two parameters we introduced at the beginning of this section, α and β, on synthetic datasets. Recall that α represents the proportion of regular segments in the whole sequence and β indicates the level of random noise. Again we use our running example to generate the synthetic data. This time, we vary α from 1 to 0.6, and simultaneously we choose β from 0 to 0.5. We test the effectiveness of the period detection algorithm and the summarization algorithm separately. All experiments are repeated 100 times and the results are averaged.

For the period detection algorithm, we report the success rates in Figure 12(a). Since we know the ground truth ($T = 24$), we judge a trial is successful if among all detected periods, the one with the large correlation value is within the range $[23, 25]$. When $\alpha = 0.8$ and $\beta = 0.5$, it means that 80% days the object follows the regular daily behavior. And in those 80% days, there is 50% probability that the object is not at its regular location as it is supposed to be. As we can see from Figure 12(a), our period detection algorithm is nearly perfect in all cases with $\alpha \geq 0.8$. This means that as long as 80% segments follow the periodic behavior, we can detect such a period successfully even if there is much noise in those regular segments. When α drops to 0.7, the success rate becomes much lower. It indicates that our method is more sensitive to the portion of irregular segments (i.e., α) but not that sensitive to random noise (i.e.,



(a) success rate of the period detection algorithm

(b) representation error of the summarization algorithm

Fig. 12. Performance evaluation.

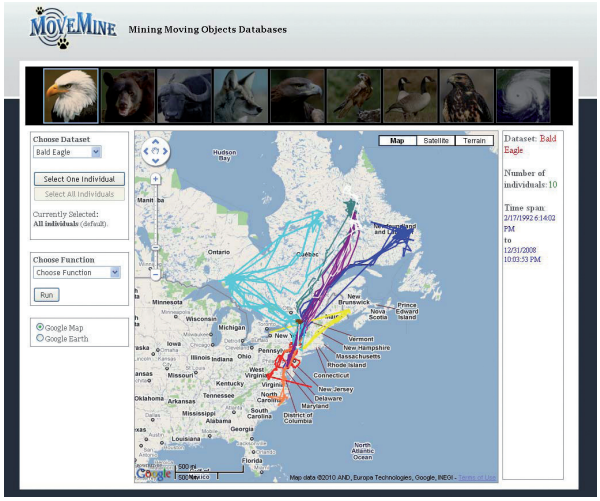


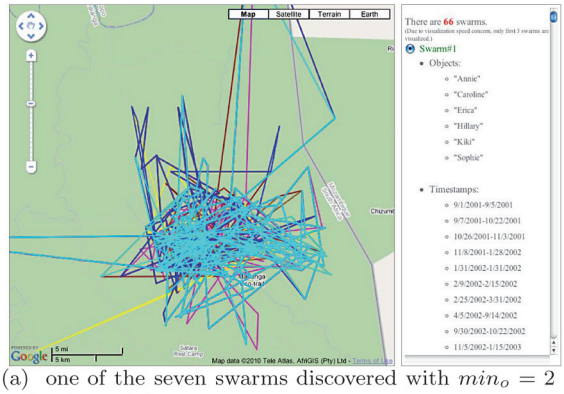
Fig. 13. Raw buffalo data.

β). Furthermore, since irregular segments often reflect the changes of behaviors in the movement, the sensitivity to the irregular segments is also desirable for our algorithm which is designed for mining periodic behaviors.

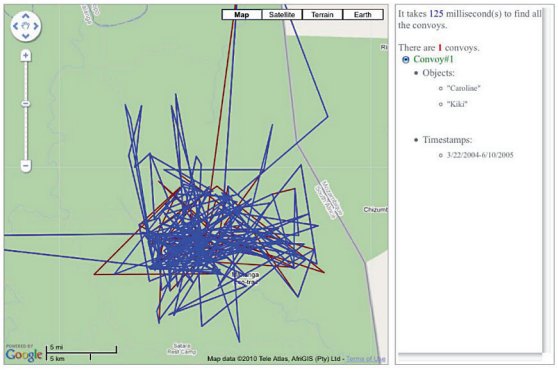
For the summarization algorithm, we show in Figure 12(b) the representation error for $K = 10$ as defined in Section 3.4.2. To see the significance of the result, observe that, for example, with $\alpha = 0.9$ and $\beta = 0.1$, if we use 10 clusters to summarize all the daily segments of one year, the representation error is about 0.2. This means that we can obtain compact high-quality summarization even with moderate amount of irregularity and noise. This further shows that our algorithm is indeed able to filter out redundancy between the segments which are generated by periodic behaviors and therefore reveals the true behaviors.

5.2. Swarm Pattern Mining

5.2.1. Swarm Pattern Mining in the MoveMine System. The effectiveness of swarm patterns can be demonstrated through our online demo system. Here, we use one dataset as an example to show the effectiveness. This dataset contains 165 buffalo with tracking time from year 2000 to year 2006. The original data has 26610 reported locations. Figure 13



(a) one of the seven swarms discovered with $min_o = 2$ and $min_t = 0.5$



(b) one convoy discovered with $min_o = m = 2$ and $min_t = k = 0.2$

Fig. 14. Effectiveness comparison between swarm and convoy.

shows the raw data plotted in Google Map. We pick this dataset because it contains a considerably large number of objects and also the tracking time is very long. Besides, we have biologists manually label data on the herds assignment, so we can better check the effectiveness of our swarm pattern.

For each buffalo, the locations are reported about every 3 or 4 days. We first use linear interpolation to fill in the missing data with time gap as one "day." Note that the first/last tracking days for each buffalo could be different. The buffalo movement with longest tracking time contains 2023 days and the one with shortest tracking time contains only 1 day. On average, each buffalo contains 901 days. We do not interpolate the data to enforce the same first/last tracking day. Instead, we require the objects that form a swarm should be together for at least min_t relative timestamps over their overlapping tracking timestamps. For example, by setting $min_t = 0.5$, o_1 and o_2 form a swarm if they are close for at least half of their overlapping tracking timestamps. Then, DBSCAN [Ester et al. 1996] with parameter $MinPts = 5$ and $Eps = 0.001$ is applied to generate clusters at each timestamp (i.e., C_{DB}). Note that, regarding users' specific requirements, different clustering methods and parameter settings can be applied to preprocess the raw data.

By setting $min_o = 2$ (i.e., at least 2 objects) and $min_t = 0.5$ (i.e., half of the overlapping time span), we can find 66 closed swarms. Figure 14(a) shows one swarm. Each color represents the raw trajectory of a buffalo. This swarm contains 5 buffalo. And the

timestamps that these buffalo are in within the same cluster are nonconsecutive. Interestingly, the swarms we detected from buffalo data are actually the herds that were manually labeled by biologists. When biologists were tracking these animals, they assigned animals to different herds for some timestamps. It is a time-consuming job for biologists to do manual labeling, especially when tracking lots of animals for long time. Our automatic discovery of the swarms can help them do this job and provide them useful information to further examine the relationship and habits of these buffalo. At the same time, the way that biologists assign the herd label also gives us motivation to further improve our swarm pattern method. When biologists identify the herds, they will identify big herds and small herds. That means there are different degrees of herds. Some have closer relationship. Therefore, it is interesting to rank our swarms based on such a degree. So we can give biologists more information on the degree of the relationship in one swarm. We consider this as a promising future work.

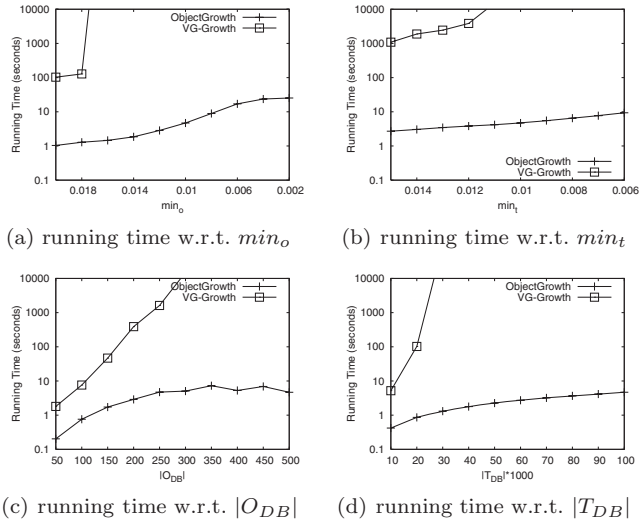
For comparison, we test convoy pattern mining on the same dataset. Note that there are two parameters in convoy definition, m (number of objects) and k (threshold of consecutive timestamps). So m actually equals to min_o and k is the same as min_t. We first use the same parameters (i.e., $min_o = 2$ and $min_t = 0.5$) to mine convoys. However, no convoy is discovered. This is because there is no group of buffalo that move together for consecutively half of the whole time span. By lowering the parameter min_t from 0.5 to 0.2, there is one convoy discovered as shown in Figure 14(b). But this convoy, containing 2 buffalo, is just a subset of one swarm pattern. The rigid definition of convoy makes it not practical to find potentially interesting patterns. The comparison shows that the concept of (closed) swarms is especially meaningful in revealing relaxed temporal moving object clusters.

5.2.2. Efficiency Study with respect to Parameters. To show the efficiency of our algorithms, we generate a larger synthetic dataset using Brinkhoff's network-based generator of moving objects.[4] We generate 500 objects ($|O_{DB}| = 500$) for 10^5 timestamps ($|T_{DB}| = 10^5$) using the generator's default map and parameter setting. There are $5 \cdot 10^7$ points in total. DBSCAN ($MinPts$= 3, $Eps = 300$) is applied to get clusters at each snapshot.

We will compare our algorithms with VG-Growth [Wang et al. 2006], which is the only previous work addressing the nonconsecutive timestamps issue. VG-Growth is developed in Wang et al. [2006] to mine group patterns. The definition of group pattern is similar to that of the swarm, which also addresses time relaxation issue. Group pattern is a set of moving objects that stay within a disc with max_dis radius for min_wei period and each consecutive time segment is no less than min_dur. Wang et al. [2006] develop the VG-Growth method whose general idea is depth-first search based on conditional VG-graph. Although the idea of group pattern is well-motivated, the problem is not well-defined. First, the "closeness" of moving objects is confined to be within a max_dis disk. A fixed max_dis for all group patterns could not produce natural cluster shapes. Second, since it does not consider the closure property of group patterns, it will produce an exponential number of redundant patterns that severely hinders efficiency. All these problems can be solved in our work by using density-based clustering to define "closeness" flexibly and introducing the closed swarm definition.

To make fair comparison on efficiency, we adapt VG-Growth to accommodate clusters as input. We set $min\_dur = 1$ and $min\_wei = min_t$. Since the search space of VG-Growth is the same as our methods to produce swarms, it is equivalent to compare the latter ones with our proposed closed swarm methods. To produce swarms, we can simply omit the Backward Pruning Rule and Forward Closure Checking in ObjectGrowth. So VG-Growth is essentially searching on objectset and using Apriori pruning rule only.

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



(a) running time w.r.t. min_o (b) running time w.r.t. min_t

(c) running time w.r.t. $|O_{DB}|$ (d) running time w.r.t. $|T_{DB}|$

Fig. 15. Running time on synthetic dataset.

The algorithms are compared with respect to two parameters (i.e., min_o and min_t) and the database size (i.e., O_{DB} and T_{DB}). By default, $|O_{DB}| = 500$, $|T_{DB}| = 10^5$, $\frac{min_o}{|O_{DB}|} = 0.01$, $\frac{min_t}{|T_{DB}|} = 0.01$. We carry out four experiments by varying one variable with the other three fixed. Note that in the following experiment part, we use min_o to denote the ratio of min_o over O_{DB} and min_t to denote the ratio of min_t over T_{DB}.

Efficiency with respect to min_o and min_t. Figure 15(a) shows the running time with respect to min_o. It is obvious that VG-Growth takes much longer time than ObjectGrowth. VG-Growth cannot even produce results within 5 hours when $min_o = 0.018$ in Figure 15(a). The reason is that VG-Growth tries to find all the swarms rather than closed swarms, and the number of swarms is exponentially larger than that of closed swarms as shown in Figure 16(a) and Figure 16(b).

Efficiency with respect to $|O_{DB}|$ and $|T_{DB}|$. Figure 15(c) and Figure 15(d) depict the running time when varying $|O_{DB}|$ and $|T_{DB}|$ respectively. In both figures, VG-Growth is much slower than ObjectGrowth. Comparing Figure 15(c) and Figure 15(d), we can see that ObjectGrowth is more sensitive to the change of O_{DB}. This is because its search space is enlarged with larger O_{DB} whereas the change of T_{DB} does not directly affect the running time of ObjectGrowth.

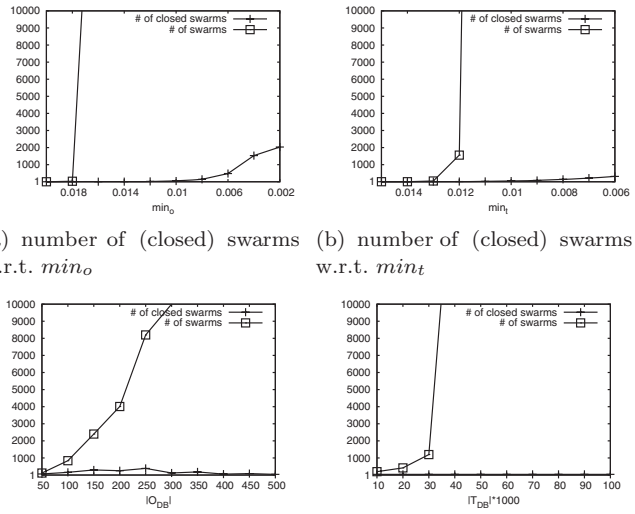
In summary, ObjectGrowth greatly outperforms VG-Growth since the number of swarms is exponential to the number of closed swarms. Besides, both ObjectGrowth are more sensitive to the size of O_{DB} rather than that of T_{DB} since the search space is based on the objectset.

6. DISCUSSION

In this section, we discuss the related works addressing the similar issues of periodic pattern and swarm pattern and point out how our methods can be further extended to solve other movement mining problems.

6.1. Periodic Behavior and its Extension

6.1.1. Frequent Periodic Patterns with a Given Period. Mamoulis et al. [2004] is the first work to study the periodic patterns of the moving objects. Give a movement sequence,



(a) number of (closed) swarms w.r.t. min_o

(b) number of (closed) swarms w.r.t. min_t

(c) number of (closed) swarms w.r.t. $|O_{DB}|$

(d) number of (closed) swarms w.r.t. $|T_{DB}|$

Fig. 16. Number of (closed) swarms in synthetic dataset.

a minimum support *min_sup*, and an integer T, called period, a periodic pattern P is a T-length sequence of the form $r_0 r_1 \ldots r_{T-1}$, where r_i is a spatial region or the sepcial character $*$, indicating the whole spatial universe. For example, with $T = 3$, a periodic pattern $AB * C$ implies that at the beginning of the cycle, the object is in region A, at the next timestamp, it is found in region B, then it moves irregularly (it can be anywhere), then it goes to region C. A periodic pattern P has to repeat itself for at least *min_sup* times in the movement sequence to become a frequent pattern.

The major problem of this method is that the period is given instead of automatically detected. Even though we could give period as one "day" or one "week" for human movement, animals do not follow human clocks. For example, when animals do yearly migration, they are not repeating it every 365 days. The yearly period for animals could be 363 days or 364 days. Similarly, animals having a daily period may not strictly follow the pattern every 24 hours. Their daily behaviors change according to seasons. In summer, they could have longer hours staying outside, and in winter, they may spend longer time sleeping. So the period actually changes at different times or at different locations. If we apply a frequent pattern mining method with one given period, the movement will be aligned incorrectly and the mined patterns become less meaningful. However, in our Periodica method, we can detect every possible approximate period hidden in the movement. Each reference spot could be associated with different periods. And we are only interested in those spots having periodic behaviors. This also filters out the spatial noise and randomness in the movement.

Furthermore, as we introduce in Section 3, periodic behaviors actually give a statistical summarization of the periodic patterns. However, frequent periodic patterns mined from Mamoulis et al. [2004] could produce redundancies which are not useful for people to get an overall understanding of the patterns, for example, if a bird usually wakes up around 6:00 and flies out of the nest around that time, as shown in Table IV, where A indicates the nest and B indicates the activity area. Since the time it wakes up is not fixed at 6:00 sharp, it flies out of the nest sometimes early and sometimes late. Periodic behaviors in Table V reflect such uncertainties in the movement. We can see

Table IV. An Example of a Possible Bird Movement (A indicates the nest and B indicates its activity area)

day	...	5:00	6:00	7:00	8:00	9:00	...
day 1	...	A	A	A	B	B	...
day 2	...	A	A	B	B	B	...
day 3	...	A	A	A	A	B	...
day 4	...	A	A	B	B	B	...
day 5	...	A	B	B	B	B	...

Table V. Periodic Behavior Summarized over Movement in Table IV

	...	5:00	6:00	7:00	8:00	9:00	...
A	...	1.0	0.8	0.4	0.2	0.0	...
B	...	0.0	0.2	0.6	0.8	1.0	...

Table VI. Frequent Periodic Patterns Mined from Movement Sequence in Table IV with $min\_sup = 3$

...	5:00	6:00	7:00	8:00	9:00	...	support
...	A	A	$*$	B	B	...	3
...	A	$*$	$*$	B	B	...	4
...	A	A	$*$	$*$	B	...	4

that at 5:00, it has 100% probability to be at nest, and at 6:00, the probability drops to 80%. Then, at 7:00, it only has 20% probability to be at the nest. Therefore, it indicates that the bird flies out of the nest around 6:00 to 7:00. However, there could be multiple frequent periodic patterns. Table VI shows three frequent periodic patterns. While these patterns are frequently repeated regular movement fractions, we can hardly get an overall summarization over the movement. Therefore, periodic behaviors should be more practical in real applications.

6.1.2. Application to Movement Prediction. The prediction for future movement is useful to better track and protect the animals. Jeung et al. [2008a] propose a method to predict future locations based on periodic patterns. To estimate an object's future locations, it combines periodic pattern information as well as existing motion functions using the object's recent movements. If it is to predict location for near time, such as the next minute, the method assigns more weight on motion functions for prediction. If the query time is distant time, such as the next day or next month, the method uses periodic patterns for prediction. Since this hybrid prediction method is based on frequent periodic patterns using the method from Mamoulis et al. [2004], it needs to choose one pattern from hundreds of patterns, which is the major challenge for prediction.

Instead of using frequent periodic patterns for prediction, we believe it is better to use periodic behaviors for prediction. Because the behaviors give a statistical summarization over the movement, they will give a better probability estimation for future location prediction. The prediction should be carried out in two steps. First, we need to decide which periodic behavior that recent movement belongs to. For example, there could be several daily periodic behaviors in different locations. We can take recent several days to see which behavior they are most similar to based on the KL-divergence measure. The smaller KL-divergence value between day d and behavior P means that day d is more likely to be generated from behavior P. We use the the behavior that has the smallest KL-divergence value with recent days. Next, based on this behavior, we can easily predict the future movement for the query time t. Since a behavior itself is a

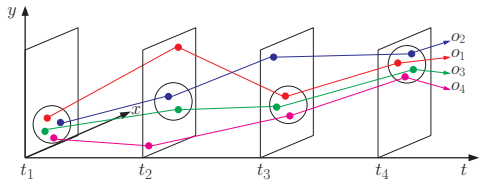


Fig. 17. Loss of interesting moving object clusters in the definition of moving cluster, flock, and convoy.

probability matrix, we can directly output the reference spots and their corresponding probability for query time t.

6.2. Swarm Pattern and its Extension

6.2.1. Swarm, Flock and Convoy Pattern. There have been many recent studies on mining moving object clusters. *Flock* is first introduced in Laube and Imfeld [2002] and further studied in Gudmundsson et al. [2004], Gudmundsson and van Kreveld [2006], and Al-Naymat et al. [2007]. Flock is defined as a group of moving objects moving in a disc of a fixed size for k *consecutive* timestamps. Another similar definition, *moving cluster* [Kalnis et al. 2005], tries to find a group of moving objects which have considerable portion of overlap at any two consecutive timestamps. A recent study by [Jeung et al. 2008b, 2008c] proposes *convoy*, an extension of flock, where spatial clustering is based on density. The common part of such patterns is that they require the group of moving objects to be together for at least k consecutive timestamps, which might not be practical in the real cases. For example, if we set $k = 3$ in Figure 17, no moving object cluster can be found. But intuitively, these four objects travel together even though some objects temporarily leave the cluster at some snapshots. If we relax the consecutive time constraint and still set $k = 3$, o_1, o_3 and o_4 actually form a moving object cluster. In other words, enforcing the consecutive time constraint may result in the loss of interesting moving object clusters.

6.2.2. Follower/Leadership Pattern Mining with Time Constraint Relaxed. It is interesting to see whether there is a leader in a herd of animals. The leader should sometimes lead the movement of the herd when they are moving. In Gudmundsson et al. [2004], the authors extend the flock pattern mining method to leadership mining. They give an additional parameter τ that prescribes during how many timesteps the leader was already moving in the specified direction. However, even if a moving object is leading a group of objects, it may not be leading the group for consecutively long times. It could move ahead for some time and join the group for some time. Therefore, it is also necessary to relax the consecutive time constraint.

Similarly, we can extend our swarm pattern mining method to leadership mining with a relaxed temporal constraint. With the parameter τ, we know that the object is leading τ timestamps ahead.Assume o is leading the herd. It should form a cluster with other objects if shifting its locations τ timestamps behind. After shifting the movement sequence of object o, we then do clustering at each timestamp. Here, we are only interested in those objects in the same cluster as o for each timestamp. Now, with a set of clusters in each timestamp, we can use our swarm pattern mining method to find those swarms. If object set O is a swarm, it means that object o is leading the group O for those timestamps.

7. CONCLUSION

In this article, we describe a MoveMine system that provides data mining functions to analyze moving object data for discovery of animal movement patterns. The system

embeds mainly four data mining functions, including periodic behavior mining, swarm pattern mining, trajectory clustering, and trajectory outlier detection. We test the system on various real animal movement datasets obtained from MoveBank.org and the results are visualized in Google Map and Google Earth.

Specifically, we introduce two interesting object pattern mining functions that are newly developed: periodic behavior mining and swarm pattern mining. In periodic behavior mining function, we propose a two-stage algorithm, Periodica. In the first stage, periods are detected through reference spots using Fourier transform and autocorrelation. In the second stage, periodic behaviors are statistically summarized using a hierarchical clustering method. In swarm pattern mining, the concept of swarm is different from the previous work and it enables the discovery of interesting moving object clusters with the temporal constraint relaxed. The ObjectGrowth method is proposed to efficiently discover closed swarms.

In experiments, we study these two functions in the MoveMine system. Furthermore, we conduct experiments on additional synthetic datasets to test the effectiveness and effiency of our methods. Experimental results demonstrate that our MoveMine system should benefit people to carry biological studies on the animal movement data. And such discoveries should improve the understanding of our ecosystem.

REFERENCES

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 487–499.

Al-Naymat, G., Chawla, S., and Gudmundsson, J. 2007. Dimensionality reduction for long duration and complex spatio-temporal queries. In *Proceedings of the ACM Symposium on Applied Computing (SAC'07)*. ACM, New York, NY, 393–397.

Bar-David, I., Cross, P. C., Ryan, S. J., and Getz, W. M. 2009. Methods for assessing movement path recursion with application to african buffalo in south africa. *Ecology*. 90.

Benkert, M., Gudmundsson, J., Hübner, F., and Wolle, T. 2008. Reporting flock patterns. *Comput. Geom. Theory Appl. 41,* 3, 111–125.

Chen, L., Özsu, M. T., and Oria, V. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*. ACM, New York, NY, 491–502.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases. In *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'96)*.

Gaffney, S., Robertson, A., Smyth, P., Camargo, S., and Ghil, M. 2006. Probabilistic clustering of extratropical cyclones using regression mixture models. *Tech. rep.* ICS 06-02. University of California, Irvino.

Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. 2007. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*. ACM, New York, NY, 330–339.

Gudmundsson, J., Laube, P., and Wolle, T. 2008. Movement patterns in spatio-temporal data. In *Encyclopedia of GIS*. 726–732.

Gudmundsson, J. and van Kreveld, M. 2006. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems (GIS'06)*. ACM, New York, 35–42.

Gudmundsson, J., van Kreveld, M., and Speckmann, B. 2004. Efficient detection of motion patterns in spatio-temporal data sets. In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems (GIS'04)*. ACM, New York, NY, 250–257.

Han, J., Pei, J., Yin, Y., and Mao, R. 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov. 8,* 1, 53–87.

Jeung, H., Liu, Q., Shen, H. T., and Zhou, X. 2008a. A hybrid prediction model for moving objects. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE'08)*. IEEE Computer Society, Los Alamitos, CA, 70–79.

JEUNG, H., SHEN, H. T., AND ZHOU, X. 2008b. Convoy queries in spatio-temporal databases. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE'08)*. IEEE Computer Society, Los Alamitos, CA, 1457–1459.

JEUNG, H., YIU, M. L., ZHOU, X., JENSEN, C. S., AND SHEN, H. T. 2008c. Discovery of convoys in trajectory databases. *Proc. VLDB Endow. 1,* 1, 1068–1080.

KALNIS, P., MAMOULIS, N., AND BAKIRAS, S. 2005. On discovering moving clusters in spatio-temporal data. In *Proceedings of 9th International Symposium on Spatial and Temporal Databases (SSTD'05)*. 364–381.

LAUBE, P. AND IMFELD, S. 2002. Analyzing relative motion within groups of trackable moving point objects. In *Proceedings of the 2nd International Conference on Geographic Information Science (GIScience'02)*. Springer-Verlag, Berlin, 132–144.

LEE, J.-G., HAN, J., AND WHANG, K.-Y. 2007. Trajectory clustering: A partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'07)*. ACM, New York, NY, 593–604.

LI, Z., DING, B., HAN, J., AND KAYS, R. 2010c. Swarm: Mining relaxed temporal moving object clusters. In *Proceedings of the Internation Conference on Very Large Data Bases (VLDB'10)*.

LI, Z., DING, B., HAN, J., KAYS, R., AND NYE, P. 2010b. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*. ACM, New York, NY, 1099–1108.

LI, Z., JI, M., LEE, J.-G., TANG, L.-A., YU, Y., HAN, J., AND KAYS, R. 2010a. Movemine: Mining moving object databases. In *Proceedings of the International Conference on Management of Data (SIGMOD'10)*. ACM, New York, NY, 1203–1206.

LIAO, L., FOX, D., AND KAUTZ, H. 2005. Location-based activity recognition using relational markov networks. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. Morgan Kaufmann Publishers Inc., San Francisco, 773–778.

MAMOULIS, N., CAO, H., KOLLIOS, G., HADJIELEFTHERIOU, M., TAO, Y., AND CHEUNG, D. W. 2004. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM, New York, NY, 236–245.

NANNI, M., TRASARTI, R., RENSO, C., GIANNOTTI, F., AND PEDRESCHI, D. 2010. Advanced knowledge discovery on movement data with the geopkdd system. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT'10)*. ACM, New York, NY, 693–696.

PEI, J., HAN, J., AND MAO, R. 2000. Closet: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of the ACM-SIGMOD International Workshop Data Mining and Knowledge Discovery (DMKDÕ00)*. 11–20.

STOLORZ, P., NAKAMURA, H., MESROBIAN, E., MUNTZ, R. R., SANTOS, J. R., YI, J., AND NG, K. 1995. Fast spatio-temporal data mining of large geophysical datasets. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 300–305.

VLACHOS, M., GUNOPOULOS, D., AND KOLLIOS, G. 2002. Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*. IEEE Computer Society, Los Alamitos, CA, 673.

VLACHOS, M., YU, P. S., AND CASTELLI, V. 2005. On periodicity detection and structural periodic similarity. In *Proceedings of the SIAM International Conference on Data Mining (SDM'05)*.

WANG, C. AND PARTHASARATHY, S. 2006. Summarizing itemset patterns using probabilistic models. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*. ACM, New York, NY, 730–735.

WANG, J., HAN, J., AND PEI, J. 2003. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*. ACM, New York, NY, 236–245.

WANG, Y., LIM, E.-P., AND HWANG, S.-Y. 2006. Efficient mining of group patterns from user movement data. *Data Knowl. Engin. 57,* 3, 240–282.

WORTON, B. J. 1989. Kernel methods for estimating the utilization distribution in home-range studies. *Ecology*. 70.

YAN, X., CHENG, H., HAN, J., AND XIN, D. 2005. Summarizing itemset patterns: a profile-based approach. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD'05)*. ACM, New York, NY, 314–323.

YAN, X., HAN, J., AND AFSHAR, R. 2003. CloSpan: Mining closed sequential patterns in large datasets. In *Proceedings of the SIAM International Conference on Data Mining (SDM'03)*. 166–177.

ZAKI, M. J. AND HSIAO, C. J. 2002. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the SIAM International Conference on Data Mining (SDM'02)*.

ZHENG, V. W., ZHENG, Y., XIE, X., AND YANG, Q. 2010. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*. ACM, New York, 1029–1038.

Received May 2010; revised August 2010; accepted August 2010