Mining Fuzzy Moving Object Clusters

Phan Nhat Hai[1,2], Dino Ienco[1,2], Pascal Poncelet[1,2], and Maguelonne Teisseire[1,2]

[1] IRSTEA Montpellier, UMR TETIS - 34093 Montpellier, France
{nhat-hai.phan,maguelonne.teisseire}@teledetection.fr
[2] LIRMM CNRS Montpellier - 34090 Montpellier, France pascal.poncelet@lirmm.fr

Abstract. Recent improvements in positioning technology have led to a much wider availability of massive moving object data. One of the objectives of spatio-temporal data mining is to analyze such datasets to exploit moving objects that travel together. Naturally, the moving objects in a cluster may actually diverge temporarily and congregate at certain timestamps. Thus, there are time gaps among moving object clusters. Existing approaches either put a strong constraint (i.e. no time gap) or completely relaxed (i.e. whatever the time gaps) in dealing with the gaps may result in the loss of interesting patterns or the extraction of huge amount of extraneous patterns. Thus it is difficult for analysts to understand the object movement behavior.
Motivated by this issue, we propose the concept of *fuzzy swarm* which softens the time gap constraint. The goal of our paper is to find all non-redundant fuzzy swarms, namely *fuzzy closed swarm*. As a contribution, we propose *fCS-Miner* algorithm which enables us to efficiently extract all the fuzzy closed swarms. Conducted experiments on real and large synthetic datasets demonstrate the effectiveness, parameter sensitiveness and efficiency of our methods.

Keywords: Fuzzy closed swarm, fuzzy time gap, frequent itemset.

1 Introduction

Nowadays, many electronic devices are used for real world applications. Telemetry attached on wildlife, GPS installed in cars, sensor networks, and mobile phones have enabled the tracking of almost any kind of data and has led to an increasingly large amount of data that contain moving object information. One of the objectives of spatio-temporal data mining [2] [4] [5] [10] [11] [12] [14] is to analyze such datasets for interesting patterns. For example, Buffaloes in South Africa (165 animals reported daily), Golden Eagles in Alaska (43 animals reported daily), and so on[3]. Analyzing this data gives insight into entity behavior, in particular and migration patterns [12]. The analysis of moving objects also has applications in transport analysis, route planning and vehicle control, socio-economic geography, location prediction, location-based services.

One of the crucial tasks for extracting patterns is to find moving object clusters (i.e. group of moving objects that are traveling together). A moving object cluster can be defined in both spatial and temporal dimensions: (1) a group of moving objects should be geometrically closed to each other, (2) they should be together for at least some

[3] http://www.movebank.org

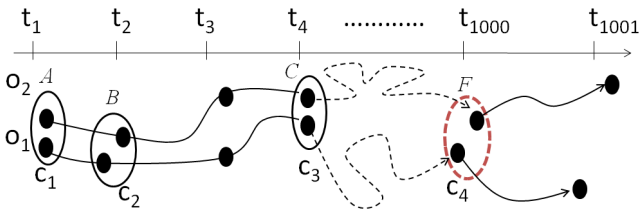


Fig. 1. An example of moving object clusters. o_1, o_2 are moving objects, $c_1, \ldots, c_4$ are clusters which are generated by applying some clustering techniques and *A, B, C, F* are spatial regions.

minimum number of certain timestamps. In this context, many recent studies have been defined to mine moving object clusters including flocks [2], moving clusters [9], convoy queries [10], closed swarms [12], group patterns [14], etc...

The common part of such patterns is that they require the group of moving objects to be together for at least min_t timestamps (i.e. could be consecutive or completely be non-consecutive), which might not be practical in the real cases. For instance, if we set $min_t = 3$ and the timestamps must be consecutive in Figure 1, no moving object cluster can be found. But essentially, these two objects, o_1 and o_2, travel together even though they temporarily leave the cluster at some snapshots. To address this issue, Zhenhui Li et al. [12] propose swarm in which moving objects are not required to be together in consecutive timestamps. Therefore, swarm can capture the movement pattern in Figure 1. The pattern is *"o_1, o_2 are moving together from A to B to C and to F at timestamps t_1, t_2, t_4 and t_{1000}"*. This pattern could be interesting since it expresses the relationship between o_1 and o_2. However, the issue here is that it is hard to say that o_1 and o_2 moving together to F since they only meet each other at F by chance after 996 timestamps. In other words, enforcing the consecutive time constraint may result in the loss of interesting moving object clusters, while completely relaxing this constraint may generate a large number of extraneous and useless patterns.

In this paper, we propose a new movement pattern, called *fuzzy closed swarm*, which softens the consecutive time constraint without generating extraneous patterns. The key challenge is to deal with the time gap between a pair of clusters since: 1) it is difficult to recognize which size of a time gap is relevant or not, 2) we need to know when the patterns should be ended to eliminate uninteresting ones. To address these issues, we present the definition of *fuzzy time gap* and *fuzzy time gap participation index*. Obtained patterns are of the type *"o_1, o_2 are moving together from A to B to C with 60% weak, 20% medium and 20% strong time gaps"*. These patterns are characterized by their time gap frequency (or support), which is by definition the proportion of time gaps involved in the patterns. As a contribution, we propose *fCS-Miner* algorithm to efficiently extract the complete set of fuzzy closed swarms. The approach shares the same spirit with the GeT_Move algorithm [5] [6] but is different in terms of goal and properties. The effectiveness as well as efficiency of our method are demonstrated on both real and large scale synthetic moving object databases.

This paper is structured as follows. Section 2 discusses the related work. The definitions of fuzzy time gap and fuzzy closed swarm are given in Section 3. fCS-Miner

algorithm will be clearly presented in Section 4. Experiments testing effectiveness and efficiency are shown in Section 5. Finally, we draw our conclusions in Section 6.

2 Related Work

As mentioned before, many approaches have been proposed to extract patterns. For instance, Gudmundsson and Van Kreveld [2] define a flock pattern, in which the same set of objects stay together in a circular region with a predefined radius, Kalnis et al. [11] propose the notion of *moving clusters*. Jeung et al. [10] define a convoy pattern and propose three algorithms $CMC, CuTS, CuTS^*$ that incorporate trajectory simplification techniques in the first step. Then, the authors proposed to interpolate the trajectories by creating virtual time points and by applying density measurements on trajectory segments. Additionally, the convoy is defined as a candidate when it has at least *k* clusters during *k* consecutive timestamps.

Recently, Zhenhui Li et al. [12] propose the concept of swarm and closed swarm and the *ObjectGrowth* algorithm to extract closed swarm patterns. The ObjectGrowth method is a depth-first-search of all subsets of O_{DB} through a pre-order tree traversal. To speed up the search process, they propose two pruning rules. *Apriori Pruning* and *Backward Pruning* are used to stop traversal the subtree when we find further traversal that cannot satisfy min_t and closure property. After pruning the invalid candidates, a *ForwardClosure checking* is used to determine whether a pattern is a closed swarm. In [14], Hwang et al. propose two algorithms to mine group patterns, known as the *Apriori-like Group Pattern mining* algorithm and *Valid Group-Growth* algorithm. The former explores the Apriori property of valid group patterns and the latter is based on idea similar to the FP-growth algorithm.

The interested readers may refer to [7] where short descriptions of the most efficient approaches and interesting patterns are proposed. Nevertheless, all the work above is not able to address the problem of capturing fuzzy closed swarms.

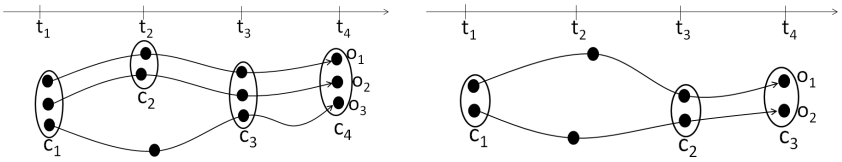
3 Problem Statement

3.1 Preliminarily Definitions

Let us assume that we have a set of moving objects $O_{DB} = \{o_1, o_2, \ldots, o_z\}$, a set of timestamps $T_{DB} = \{t_1, t_2, \ldots, t_m\}$.

Database of clusters. A database of clusters, $C_{DB} = \{C_1, C_2, \ldots, C_m\}$, is the collection of snapshots of the moving object clusters at timestamps $\{t_1, t_2, \ldots, t_m\}$. Note that an object could belong to several clusters at one timestamp (i.e. cluster overlapping). Given a cluster $c \in C_{DB}$ and $c \subseteq O_{DB}$, $|c|$ and $t(c)$ are respectively used to denote the number of objects belong to cluster c and the timestamp that c involved in. To make our framework more general, we take clustering as a preprocessing step. The clustering methods could be different based on various scenarios. We leave the details of this step in the Appendix *Obtaining Clusters*.

From now, $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_p}\} (O \subseteq O_{DB})$ stands for a group of objects, $T = \{t_{a_1}, t_{a_2}, \ldots, t_{a_m}\}$ $(T \subseteq T_{DB})$ is the set of timestamps within which objects stay together.



(a) Convoy (b) Swarm

Fig. 2. An example of convoy and swarm where c_1, c_2, c_3, c_4 are clusters.

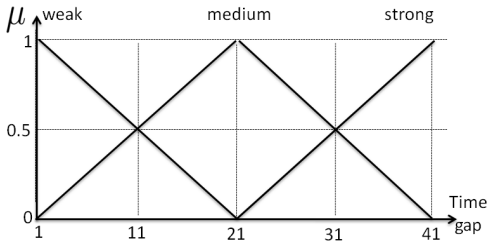


Fig. 3. Membership degree functions for fuzzy time gaps.

Convoys and closed swarms. Informally, a *convoy* (O, T) is a group of objects O containing at least min_o individuals which are closed each other during at least min_t consecutive time points T. While, consecutive time constraint is relaxed in *swarm* in which objects in O are closed each other for at least min_t timestamps. To avoid redundancy, Zhenhui Li et al. [12] propose the notion of *closed swarm* for grouping together both objects and time. A swarm (O, T) is *object-closed* if when fixing T, O cannot be enlarged. Similarly, a swarm (O, T) is *time-closed* if when fixing O, T cannot be enlarged. A swarm (O, T) is a closed swarm if it is both *object-closed* and *time-closed*.

For instance, in Figure 2a, with $min_o = 2, min_t = 2$ we have two convoys $(\{o_1, o_2\}, \{t_1, t_2, t_3, t_4\})$ and $(\{o_1, o_2, o_3\}, \{t_3, t_4\})$. While, in Figure 2b, if we set $min_o = 2$ and $min_t = 2$, we can find the following swarms $(\{o_1, o_2\}, \{t_1, t_3\})$, $(\{o_1, o_2\}, \{t_1, t_4\})$, $(\{o_1, o_2\}, \{t_3, t_4\})$, $(\{o_1, o_2\}, \{t_1, t_3, t_4\})$. We can note that these swarms are in fact redundant since they can be grouped together in the following closed swarm $(\{o_1, o_2\}, \{t_1, t_3, t_4\})$.

3.2 Fuzzy Closed Swarms

As illustrated before, enforcing the consecutive time constraint or completely relaxing this constraint may result in the loss of interesting patterns or the generation of uninteresting patterns. To deal with the issue, we propose the adaptation of fuzzy logic principle in which the strength of time gaps are evaluated with a membership degree function A (see Figure 3). Given two timestamps t_1 and t_2, a time gap x between t_1 and t_2 is computed as $x = |t_1 - t_2| - 1$ (i.e. $t_1 \neq t_2$). The fuzzy time gap is defined as follows.

Definition 1 *Fuzzy Time Gap. Given two timestamps t_1 and t_2, a pair of one time gap x and one corresponding fuzzy set a, denoted by $[x, a]$, is called a fuzzy time gap if $x = |t_1 - t_2| - 1$ is involved in membership function A.*

For instance, see Figure 4, there are totally four time gaps which are $x_1 = 3, x_2 = 18, x_3 = 34$ and $x_4 = 939$. The fuzzy time gap $[x_1, weak]$, $[x_1, medium]$ and $[x_1, strong]$ respectively are $\mu_{weak}(x_1) = 0.9$, $\mu_{medium}(x_1) = 0.1$ and $\mu_{strong}(x_1) = 0$. Since x_4 is out of function A, it cannot be considered as a fuzzy time gap.

Definition 2 *Fuzzy Time Gap Set. Given an ordered list of timestamps $T = \{t_{a_1}, t_{a_2}, \ldots, t_{a_m}\}$, a set of time gaps $X = \{x_1, \ldots, x_n\}, n = m - 1$. (X, A) is a fuzzy time gap set generated from T if $\forall i \in \{1, \ldots, n\} : x_i = |t_{a_i} - t_{a_{i+1}}| - 1$ and $\forall x \in X : x$ is involved in A. Note that for any $x \in X, x = 0$ then x will be excluded from X without any affection.*

For instance, see Figure 4, a proper pattern $(\{o_1, o_2\}, \{t_1, t_2, t_6, t_{25}, t_{60}\})$ and a fuzzy time gap set is $X = \{x_1, x_2, x_3\}$ and for each time gap $x_i \in X$, there are a corresponding fuzzy set including $strong, medium$ and $weak$. Note that x_4 is out of membership function and therefore it is not included in X and $(\{o_1, o_2\}, \{t_1, t_2, t_6, t_{25}, t_{60}, t_{1000}\})$ will not be considered as a valid pattern.

To highlight the participation of time gaps given by a fuzzy set a, we further propose an adaptation of the participation index [8] which is *fuzzy time gap participation index* proposed to take into account the fuzzy time gap occurrences in the pattern.

Definition 3 *Fuzzy Time Gap Participation Ratio. Let (X, A) be a set of fuzzy time gaps and a be an item of A, the fuzzy time gap participation ratio for a in X denoted $TGr(X, a)$ can be defined as follows.*

$$TGr(X, a) = \frac{\sum_{x \in X} \mu_a(x)}{|X|} \tag{1}$$

Definition 4 *Fuzzy Time Gap Participation Index. Let (X, A) be a set of fuzzy time gaps and a be an item of A, the fuzzy time gap participation index of (X, A) denoted $TGi(X)$ can be defined as follows.*

$$TGi(X) = Max_{\forall a \in A} TGr(X, a) \tag{2}$$

For instance, see Figure 4, a fuzzy time gap set $X = \{x_1, x_2, x_3\}$ and $TGr(X, weak) = \frac{0.1+0.9+0}{3} = 0.33$, $TGr(X, medium) = \frac{0.1+0.9+0.35}{3} = 0.45$, $TGr(X, strong) = \frac{0+0+0.65}{3} = 0.22$. Thus, the fuzzy time gap participation index of X, $TGi(X) = 0.45$.

Fuzzy swarm and fuzzy closed swarm. Given a group of objects O moving together in an ordered list of timestamps T and a set of fuzzy time gaps (X, A) generated from T. (O, T, X) is a fuzzy swarm that contains at least min_o objects (resp. $|O| \geq min_o$) during at least min_t timestamps (resp. $|T| \geq min_t$) and $TGi(X) \geq \varepsilon$. The fuzzy swarm can be defined as follows.

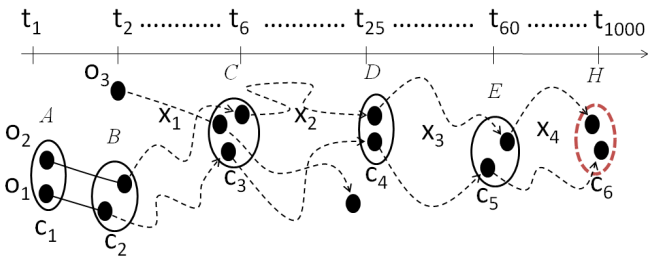


Fig. 4. A fuzzy closed swarm running example.

Definition 5 *Fuzzy Swarm. Given integers min_o, min_t and a user-defined threshold ε. (O, T, X) is a fuzzy swarm if and only if:*

$$\begin{cases} (1) : |O| \geq min_o. \\ (2) : |T| \geq min_t. \\ (3) : (X, A) \textit{ is a fuzzy time gap set.} \\ (4) : \forall i \in \{1, \ldots, n\}, TGi(\{x_1, \ldots, x_i\}) \geq \varepsilon. \end{cases} \tag{3}$$

Note that if $X = \{x_1, x_2, x_3\}$ then the condition (4) means that $TGi(\{x_1\}) \geq \varepsilon$, $TGi(\{x_1, x_2\}) \geq \varepsilon$ and $TGi(\{x_1, x_2, x_3\}) \geq \varepsilon$.

By definition, if we set $min_o = 2, min_t = 3$ and $\varepsilon = 0.2$ then there are totally 13 fuzzy swarms in Figure 4 such as $(\{o_1, o_2\}, \{t_1, t_2, t_6\}, \{x_1\})$, $(\{o_1, o_2\}, \{t_1, t_2, t_{25}\}, \{x = 22\})$, $(\{o_1, o_2\}, \{t_2, t_6, t_{25}, t_{60}\}, \{x_1, x_2, x_3\})$ and so on. However, it is obviously redundant to output fuzzy swarms like $(\{o_1, o_2\}, \{t_1, t_2, t_6, \})$ since it can be enlarged to $(\{o_1, o_2\}, \{t_1, t_2, \ t_6, t_{25}, t_{60}\}, \{x_1, x_2, x_3\})$. To avoid mining redundant fuzzy swarms, we further give the definition of fuzzy closed swarm. Essentially, a fuzzy swarm (O, T, X) is *time-closed* if fixing T, O cannot be enlarged ($\nexists O'$ s.t. (O', T, X) is a fuzzy swarm and $O \subset O'$). Similarly, a fuzzy swarm (O, T, X) is *object-closed* if fixing O then T cannot be enlarged. Finally, a fuzzy swarm (O, T, X) is a fuzzy closed swarm if it is both *time-closed* and *object-closed*. Our goal is to find the complete set of fuzzy closed swarms. The definition is formally presented as follows.

Definition 6 *Fuzzy Closed Swarm. Given a fuzzy swarm (O, T, X), it is a fuzzy closed swarm if and only if:*

$$\begin{cases} (1) : \nexists O', O \subset O' \wedge (O', T, X) \textit{ is a fuzzy swarm.} \\ (2) : \nexists T', T \subset T' \wedge (O, T', X') \textit{ is a fuzzy swarm.} \end{cases} \tag{4}$$

For instance (Figure 4), a closed swarm is $(\{o_1, o_2\}, \{t_1, t_2, t_6, t_{25}, t_{60}\}, \{x_1, x_2, x_3\})$.

Property 1. Anti-monotonic. For all patterns (O, T, X), if (O, T, X) is not a fuzzy swarm because of the condition (3) suffering then the following holds:

For all supersets of (O, T, X) by adding a later cluster and a fuzzy time gap in terms of time to T and X are not fuzzy swarms.

Table 1. Cluster Matrix corresponding to our running example in Figure 4.

T_{DB}		t_1	t_2	t_6	t_{25}	t_{60}	t_{1000}
Clusters C_{DB}		c_1	c_2	c_3	c_4	c_5	c_6
O_{DB}	o_1	1	1	1	1	1	1
	o_2	1	1	1	1	1	1
	o_3			1			

Proof. After construction, we have $\exists k \in \{1, \ldots, n\}$ s.t. $TGi(\{x_1, \ldots, x_k\}) < \varepsilon$. For any $X' = \{x_1, \ldots, x_n, x_m\}$, (O, T', X') is not a fuzzy swarm since $\exists k \in \{1, \ldots, m\}$ s.t. $TGi(\{x_1, \ldots, x_k\}) < \varepsilon$.

4 Discovering of Fuzzy Closed Swarms

The patterns we are interested in here, fuzzy closed swarms, is the association of a set of objects O, a set of timestamps T and a set of fuzzy time gaps X, denoted (O, T, X). As first glance, we can employ ObjectGrowth algorithm [12] to extract all closed swarms and then a post-processing step to obtain all the fuzzy closed swarms. However, moving object databases are naturally large and thus the search space of closed swarm extracting can be significantly increased (i.e. approximately $2^{|O_{DB}|} \times 2^{|T_{DB}|}$). Additionally, a huge amount of generated closed swarms (i.e. including extraneous patterns) can cause an expensive post-processing task. Furthermore, in real world applications (e.g. cars), object locations are continuously reported by using Global Positioning System (GPS). Thus, new data is always available and we need to execute again and again the algorithms on the whole database (i.e. including existing data and new data) to extract patterns. This is of course, cost-prohibitive and time consuming.

To deal with the issues, we propose *fCS-Miner* algorithm which is an adaptation of Incremental GeT_Move approach [5] [6] which has already been proved as being efficient in large moving object databases.

Basic idea of fCS-Miner algorithm. As in [5] [6], we first present C_{DB} in a cluster matrix (see Table 1) so that Incremental GeT_Move can be applied to extract all frequent closed itemsets (FCIs). Next, we propose an novel property which can be used to directly extract fuzzy closed swarms from generated FCIs without a post-processing step. The cluster matrix definition is as follows.

Definition 7 *Cluster Matrix [5] [6]. Given a set of clusters $C_{DB} = \{C_1, C_2, \ldots, C_m\}$ where $C_i = \{c_{i_1 t_i}, c_{i_2 t_i}, \ldots, c_{i_n t_i}\}$ is a set of clusters at timestamps t_i. A cluster matrix is thus a matrix of size $|O_{DB}| \times |C_{DB}|$. Each row represents an object and each column represents a cluster. The value of the cluster matrix cell, (o_i, c_j) is 1 (resp. empty) if o_i is in (resp. is not in) cluster c_j.*

For instance, see Table 1 and Figure 4, the matrix cell of (o_1, c_2) is 1 since $o_1 \in c_2$ and this is similar for c_1, c_3, c_4, c_6. While, the matrix cell of (o_3, c_1) is empty since $o_3 \notin c_1$.

By applying Incremental GeT_Move which mainly bases on LCM algorithm [13] on the cluster matrix, we are able to extract all FCIs. Let us denote a frequent itemset as $\Upsilon = \{c_1, c_2, \ldots, c_k\}$, O_Υ contains the corresponding group of moving objects

which are closed each other in a set of timestamps $T_\Upsilon = \{t(c_1), t(c_2), \ldots, t(c_k)\}$. We can recognize that $|O_\Upsilon| = \sigma(\Upsilon)$[4], $|\Upsilon| = |T_\Upsilon|$ and X_Υ is used to denote as a fuzzy time gap set generated from T_Υ. For instance, see Table 1, a proper frequent itemset is $\Upsilon = \{c_1, c_2, c_3, c_4, c_5\}$ with $O_\Upsilon = \{o_1, o_2\}$, $T_\Upsilon = \{t_1, t_2, t_6, t_{25}, t_{60}\}$ and $X_\Upsilon = \{x_1, x_2, x_3\}$.

The following property, *f-closed swarm*, is used to verify whenever a frequent itemset Υ can be a fuzzy closed swarm or not.

Property 2. f-Closed swarm. Given a frequent itemset $\Upsilon = \{c_1, c_2, \ldots, c_k\}$, $X_\Upsilon = \{x_1, \ldots, x_n\}$. $(O_\Upsilon, T_\Upsilon, X_\Upsilon)$ is a fuzzy closed swarm if and only if:

$$\begin{cases} (1) : \sigma(\Upsilon) \geq min_o. \\ (2) : |\Upsilon| \geq min_t. \\ (3) : \forall x \in X, x \textit{ is involved in } A. \\ (4) : \forall i \in \{1, \ldots, n\}, TGi(\{x_1, \ldots, x_i\}) \geq \varepsilon. \\ (5) : \nexists \Upsilon' \textit{ s.t } O_\Upsilon \subset O_{\Upsilon'}, T_{\Upsilon'} = T_\Upsilon \textit{ and } (O_{\Upsilon'}, T_\Upsilon, X_\Upsilon) \textit{ is a fuzzy swarm.} \\ (6) : \nexists \Upsilon' \textit{ s.t. } O_{\Upsilon'} = O_\Upsilon, T_\Upsilon \subset T_{\Upsilon'} \textit{ and } (O_\Upsilon, T_{\Upsilon'}, X_{\Upsilon'}) \textit{ is a fuzzy swarm.} \end{cases} \tag{5}$$

Proof. After construction, we have $\sigma(\Upsilon) \geq min_o$ and thus $|O_\Upsilon| \geq min_o$ since $|O_\Upsilon| = \sigma(\Upsilon)$. Additionally, $|\Upsilon| \geq min_t$ and therefore $|T_\Upsilon| \geq min_t$ since $|\Upsilon| = |T_\Upsilon|$. Furthermore, $\forall x \in X : x$ is involved in A and $\forall i \in \{1, \ldots, n\}, TGi(\{x_1, \ldots, x_i\}) \geq \varepsilon$.. Consequently, $(O_\Upsilon, T_\Upsilon, X_\Upsilon)$ is a fuzzy swarm (*Definition 5*). Moreover, if $\nexists \Upsilon'$ s.t $O_\Upsilon \subset O_{\Upsilon'}, T_{\Upsilon'} = T_\Upsilon$ and $(O_{\Upsilon'}, T_\Upsilon, X_\Upsilon)$ is a fuzzy swarm then $(O_\Upsilon, T_\Upsilon, X_\Upsilon)$ cannot be enlarged in terms of objects. Therefore, it satisfies the *object-closed* condition. Furthermore, if $\nexists \Upsilon'$ s.t. $O_{\Upsilon'} = O_\Upsilon, T_\Upsilon \subset T_{\Upsilon'}$ and $(O_\Upsilon, T_{\Upsilon'}, X_{\Upsilon'})$ is a fuzzy swarm then $(O_\Upsilon, T_\Upsilon, X_\Upsilon)$ cannot be enlarged in terms of lifetime. Therefore, it satisfies the *time-closed* condition. Consequently, $(O_\Upsilon, T_\Upsilon, X_\Upsilon)$ is a fuzzy swarm and it satisfies *object-closed* and *time-closed* conditions and therefore $(O_\Upsilon, T_\Upsilon, X_\Upsilon)$ is a fuzzy closed swarm according to the *Definition 6*.

To show the fact that from an itemset mining algorithm we are able to extract the set of all fuzzy closed swarms, we propose the following lemma.

Lemma 1. *Let $FI = \{\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_l\}$ be the set of frequent itemsets being mined from the cluster matrix with $minsup = min_o$. All fuzzy closed swarms (O, T, X) can be extracted from FI.*

Proof. Let us assume that (O, T, X) is a fuzzy closed swarm. Note, $T = \{t(c_1), \ldots, t(c_k)\}$. According to the *Definition 6* we have $|O| \geq min_o$. If (O, T, X) is a fuzzy closed swarm then $\forall t(c_i) \in T, \exists c_i$ s.t. $O \subseteq c_i$ therefore $\bigcap_{i=1}^{k} c_i = O$. Additionally, we have $\forall c_i$, c_i is an item so $\exists \Upsilon = \bigcup_{i=1}^{k} c_i$ is an itemset and $O_\Upsilon = \bigcap_{i=1}^{k} c_i = O, T_\Upsilon = \bigcup_{i=1}^{k} t(c_i) = T$. Furthermore, we also have $X_\Upsilon = X$ as well. Therefore, $(O_\Upsilon, T_\Upsilon, X_\Upsilon)$ is a fuzzy closed swarm since $O_\Upsilon = O, T_\Upsilon = T$ and $X_\Upsilon = X$. So, (O, T, X) is extracted from Υ. Furthermore, $\sigma(\Upsilon) = |O_\Upsilon| = |O| \geq min_o$ then Υ is a frequent itemset

[4] $\sigma(\Upsilon)$ is the support value of frequent itemset Υ.

Algorithm 1: fCS-Miner

```
   Input  : double ε, int min_o, int min_t, set of items C_DB
1  begin
2  |  Incremental GeT_Move(C_DB, min_o);
3  PatternMining(FCI, ε, min_t)
4  begin
5  |  f-CS := ∅;
6  |  if |FCI| ≥ min_t then
7  |  |  Υ := ∅;
8  |  |  for k := 1 to |FCI| do
9  |  |  |  Υ' := Υ ∪ c_k;
10 |  |  |  if fuzzy(X_Υ') = true ∧ TGi(Υ') ≥ ε then
11 |  |  |  |  Υ := Υ';
12 |  |  |  else
13 |  |  |  |  if O_Υ = O_FCI ∧ |Υ| ≥ min_t + 1 then
14 |  |  |  |  |  f-CS := f-CS ∪Υ;
15 |  |  |  |  Υ := ∅ ∪ c_k;
16 |  return f-CS;
17 where: fuzzy(X_Υ') returns true if X_Υ' is a fuzzy time gap set, otherwise returns false. In
   this function, we only need to verify that the last time gap is involved in A instead of all
   the time gaps in X_Υ'.
```

and $\Upsilon \in FI$. Finally, $\forall(O,T)$ s.t. if (O,T,X) is a fuzzy closed swarm then $\exists\Upsilon$ s.t. $\Upsilon \in FI$ and (O,T,X) can be extracted from Υ, we can conclude that $\forall$ fuzzy closed swarm (O,T,X), it can be mined from FI.

Essentially, by scanning the FCIs from the beginning to the end with the f-closed swarm property, we are able to extract the corresponding fuzzy closed swarms. The scanning process will be ended whenever one of conditions (3), (4) is suffered (*Property 1*), after that the current frequent itemset Υ (i.e. $\sigma(\Upsilon) \geq min_o$) only need to be verified the conditions $|\Upsilon| \geq min_t$ and Υ contains the same number of objects with the FCI. This is because Υ cannot be enlarged in terms of timestamps T_Υ (i.e. *Property 1*) and objects (i.e. FCI is closed). Thus, it satisfies all the requirements to be a fuzzy closed swarm completely.

The pseudo-code of fCS-Miner is presented in the Algorithm 1. We first apply Incremental GeT_Move on cluster matrix C_{DB} with $minsup = min_o$ (line 2). Then, for each generated FCI, we directly scan it with the *f-closed swarm* property as mentioned before (lines 4-16). By using fCS-Miner, we are able to extract all fuzzy closed swarms on-the-fly without a post-processing step.

5 Experimental Results

A comprehensive performance study has been conducted on real and synthetic datasets. All the algorithms are implemented in C++, and all the experiments are carried out on a 2.8GHz Intel Core i7 system with 4GB Memory. The system runs Ubuntu 11.10 and

g++ version 4.6.1. The implementation of our proposed algorithm is also integrated in a demonstration system available online[5]. As in [12] [2] [6], the following dataset[6] have been used during experiments: *Swainsoni dataset* includes 43 objects evolving over time and 764 different timestamps. It was generated from July 1995 to June 1998.

To the best of our knowledge, there is no previous work which addresses fuzzy closed swarms. Therefore, in the comparison, we employ the latest pattern mining algorithms such as $CuTS^*$[7] [10] (convoy mining) and *ObjectGrowth* [12] (closed swarm mining). As pointed out in [12], *ObjectGrowth* outperforms *VG-Growth* [14] (group pattern mining) in terms of performance and therefore we will only consider *ObjectGrowth* and not both.

Similarly to [10] [12], we first use linear interpolation to fill in the missing data. Furthermore, as [10] [11] [12], DBScan [1] ($MinPts = 2, Eps = 0.001$) is applied to generate clusters at each timestamp. To make fair comparison, we adapt all the algorithms to accommodate clusters as input but their time complexity will remain the same. Additionally, to retrieve all the patterns including fuzzy closed swarms, convoys and closed swarms, in the reported experiments the fuzzy function in Figure 3 is applied, the default value of min_t is 1, $min_o = 1$ and $\varepsilon = 0.001$. Note that the default values are the hardest conditions for examining all the algorithms.

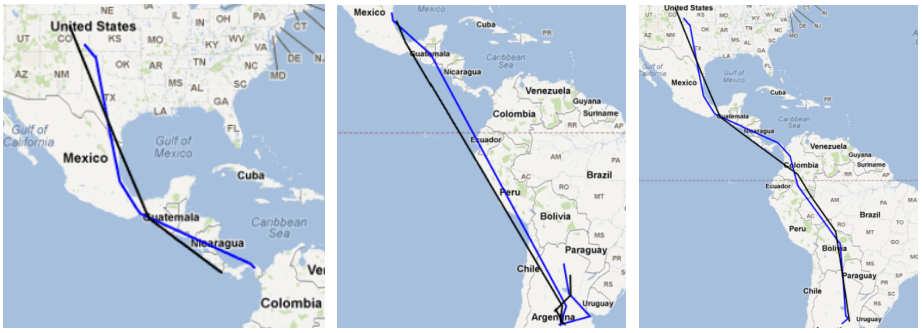
5.1 Effectiveness

The effectiveness of fuzzy closed swarms can be demonstrated through our online demo system. One of the extracted patterns from Swainsoni dataset is illustrated in Figure 5c. Each color represents a Swainsoni trajectory segment involved in the pattern.

To illustrate the feasibility of a fuzzy approach, we also show some of extracted closed swarms and convoys from our system[8] [3] in Figures 5a, b. We can consider that closed swarm is extraneous since the two objects meet each other at Mexico on October 1995 and after 5 months (i.e. to March 1996) for the next meeting location (i.e. Argentina). In fact, it is hard to say that they are moving together from Mexico to Argentina. While, the convoys are sensitive to time gaps and usually are short deal to the consecutive time constraint (see Figure 5a). Thus, they fail to describe the insightful relationship between objects. Either be too strict or too relaxed in dealing with time gaps may result in the loss of interesting patterns or reporting many uninteresting ones.

Distinguish from previous work, by proposing fuzzy closed swarms, we are able to reveal the relevant relationship between Swainsonies in a fuzzy point of view. Looking at the illustrated pattern in Figure 5c, we can consider that, from United States, the two objects are flying together along a narrow corridor through Central America and down to South America. Furthermore, they temporally diverge at Panama and congregate again at the Columbia central. The discovery of the fuzzy closed swarms on animal migration datasets provides useful information for biologists to better understand and examine the relationship and habits of these moving objects. Due to the space limitation,

[5] http://www.lirmm.fr/~phan/fcsminer.jsp
[6] http://www.movebank.org
[7] The source code of $CuTS^*$ is available at http://lsirpeople.epfl.ch/jeung/source_codes.htm
[8] http://www.lirmm.fr/~phan/index.jsp



(a) A convoy. (b) A closed swarm. (c) A fuzzy closed swarm.

Fig. 5. An example of extracted patterns from Swainsoni dataset. The two object names are 'SW22' and 'SW40'.

we do not provide experiments by varying the fuzzy membership function A. However, in real world context, users can express their expertise through the membership function for dealing with fuzzy approximate reasoning issues.

5.2 Parameter Sensitiveness

To show the parameter sensitiveness and efficiency of the proposed algorithm, as in [12], we also generate a large synthetic dataset using Brinkhoff's network[9]-based generator of moving objects. We generate 500 objects ($|O_{DB}| = 500$) for 10^4 timestamps ($|T_{DB}| = 10^4$) using the generator's default map with slow moving speed (5×10^6 points in total). DBScan ($MinPts = 3, Eps = 300$) is applied to obtain clusters.

Sensitiveness w.r.t ε. See Figure 6a, we can consider that fCS-Miner is not sensitive in ε. This is because ε is only used to scan the FCIs for fuzzy closed swarm extraction which is much less expensive than FCI mining task.

Sensitiveness w.r.t min_t. Figure 6b shows that ObjectGrowth is the most sensitive algorithm in min_t. This is because ObjectGrowth applies a min_t-based pruning rule, called *Apriori Pruning*, which is very sensitive in min_t. Since, it is used to limit approximately $2^{|T_{DB}|}$ candidates in total. Furthermore, with different values of min_t, there are great differences in terms of the number of extracted closed swarms (Figure 7b). Meanwhile, fCS-Miner and CuTS* only use min_t at the pattern reporting or verifying steps without any pruning rule for min_t. Additionally, as mentioned before the fuzzy closed swarm verifying task is less expensive than the FCI extraction. Consequently, be similar to CuTS*, the fCS-Miner sensitiveness in min_t is less sensitive than ObjectGrowth.

Sensitiveness w.r.t O_{DB}, T_{DB}. Figures 6c-d show the sensitiveness in the sizes of O_{DB} and T_{DB}. We can consider that all the algorithms are quite similar to each other. However, CuTS* is a little bit less sensitive than the others. This is because, in CuTS*: 1) the number of clusters at a certain timestamp is not exponentially increased due to the $|O_{DB}|$ and $|T_{DB}|$ increases, 2) for any cluster c, c can combine with the clusters

[9] http://iapg.jade-hs.de/personen/brinkhoff/generator/

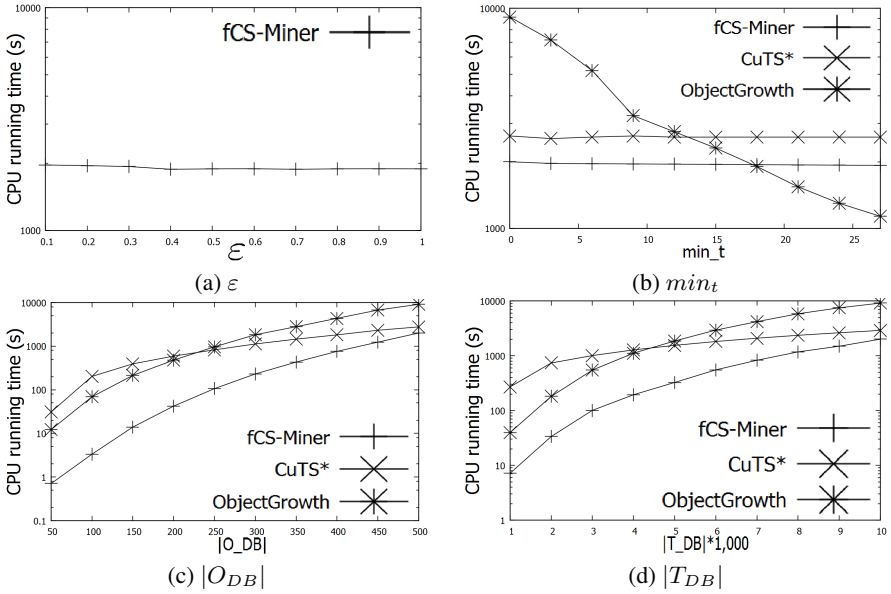
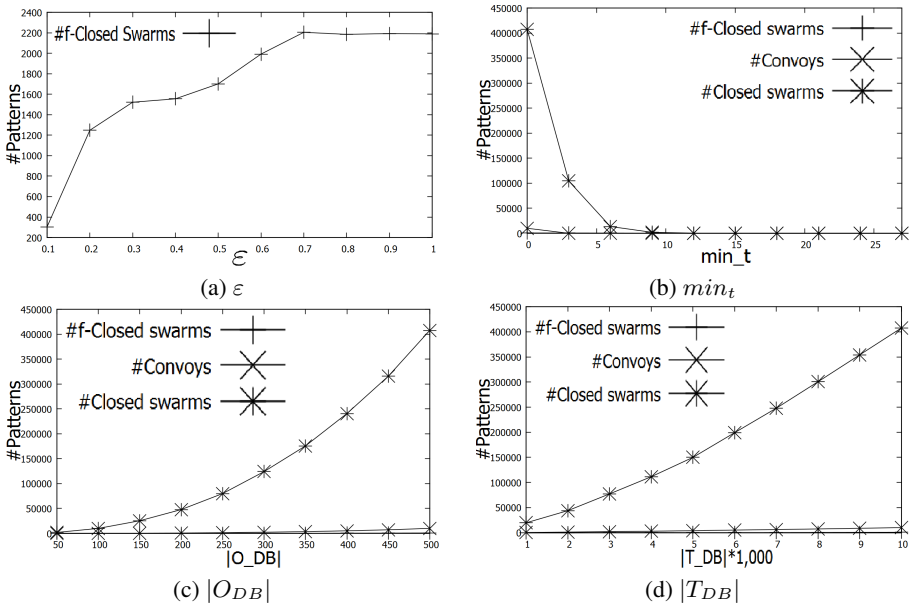


Fig. 6. Running time on Synthetic Dataset.

at the next timestamp. While, for ObjectGrowth, the number of candidates is greatly increased due to the size increase of $|O_{DB}|, |T_{DB}|$ (i.e. approximately $2^{|O_{DB}|} \times 2^{|T_{DB}|}$ candidates). As the results, the number of closed swarms is significantly increased (see Figures 7c-d). This behavior is similar in fCS-Miner since the number of FCIs can be large. However, thanks to the fuzzy approach, there are not huge amount of generated patterns compared to ObjectGrowth. Obviously, fCS-Miner is similar to ObjectGrowth and a little bit more sensitive than CuTS* in terms of $|O_{DB}|$ and $|T_{DB}|$.

Influence of $TGi(X)$ on #f-Closed swarms. Figure 8 shows the influence of the fuzzy time gap participation index on the number of patterns that contain weak, medium and strong time gaps. We can consider that the number of patterns which have $TGi(X)$ with weak fuzzy time gaps X, medium fuzzy time gaps X and strong fuzzy time gaps X are quite distinguished from each other. Since, the number of patterns with weak X is smallest, more number of patterns with medium X and the highest number of patterns with strong X. Therefore, the $TGi(X)$ enable us to rank the fuzzy closed swarms well corresponding with the membership degree function. Furthermore, if we ignore all the fuzzy closed swarms with strong (and medium) fuzzy time gaps, a number of uninteresting patterns can be eliminated.

To summarize, fCS-Miner is effective to extract fuzzy closed swarms which are novel and useful movement patterns. By applying fuzzy function, users can express their background knowledge in order to obtain interesting patterns without generating extraneous ones. Additionally, fCS-Miner parameter sensitiveness is quite acceptable compare to the other model algorithms. Moreover, with the purpose to extract the complete set of f-closed swarms, fCS-Miner is competitive in time efficiency to state-of-the-art approaches (see Figure 6).



(a) ε (b) min_t (c) $|O_{DB}|$ (d) $|T_{DB}|$

Fig. 7. Number of patterns on Synthetic Dataset.

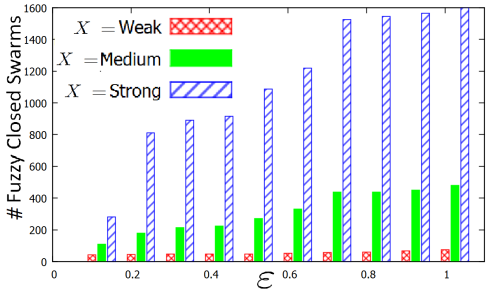


Fig. 8. Influence of $TGi(X)$ on #patterns through ε.

6 Conclusions and Future Directions

In this paper, to deal with the issue that enforcing the consecutive time constraint or completely relaxing may result in the loss of interesting patterns or the generation of uninteresting patterns, we propose the concept of fuzzy swarm which softens the time gap constraint. These concepts enable the discovery of insightful movement patterns and the elimination of extraneous patterns. A new method fCS-Miner is proposed to efficiently extract all the fuzzy closed swarms. The proposed algorithm's effectiveness, and parameter sensitiveness are demonstrated using real and large synthetic datasets.

In the future work, the proposed approaches can be applied on other kinds of patterns (e.g. gradual trajectory patterns [4]). Although the number of non-interesting patterns are significantly reduced, it is still difficult to analyze the results since number of

patterns is still large. Another future directions is to directly mining top-K informative fuzzy movement patterns to avoid extracting redundant information.

References

1. M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.
2. J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *ACM GIS 06*, 2006.
3. P. N. Hai, D. Ienco, P. Poncelet, and M. Teisseire. Extracting trajectories through an efficient and unifying spatio-temporal pattern mining system. In *ECML/PKDD*, pages 820–823, 2012.
4. P. N. Hai, D. Ienco, P. Poncelet, and M. Teisseire. Ming time relaxed gradual moving object clusters. In *ACM SIGSPATIAL GIS*, 2012.
5. P. N. Hai, P. Poncelet, and M. Teisseire. An efficient and unifying spatio-temporal pattern mining algorithm for moving objects. In *IDA*, pages 276–288, 2012.
6. P. N. Hai, P. Poncelet, and M. Teisseire. An efficient spatio-temporal mining approach to really know who travels with whom! In *BDA*, 2012.
7. J. Han, Z. Li, and L. A. Tang. Mining moving object, trajectory and traffic data. In *DASFAA*, 2010.
8. Y. Huang, S. Shekhar, and H. Xiong. Discovering colocation patterns from spatial data sets: a general approach. *Knowledge and Data Engineering, IEEE Transactions on*, 16(12):1472 – 1485, December 2004.
9. C.S. Jensen, L. Dan, and B. C. Ooi. Continuous clustering of moving objects. *Knowledge and Data Engineering, IEEE Transactions on*, 19(9):1161–1174, sept. 2007.
10. H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. *Proc. VLDB Endow.*, 1(1):1068–1080, August 2008.
11. P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *SSTD'05*, pages 364–381, 2005.
12. Z. Li, B. Ding, J. Han, and R. Kays. Swarm: mining relaxed temporal moving object clusters. *Proc. VLDB Endow.*, 3(1-2):723–734, September 2010.
13. T. Uno, M. Kiyomi, and H. Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *ICDM FIMI*, 2004.
14. Y. Wang, E. P. Lim, and S. Y. Hwang. Efficient mining of group patterns from user movement data. *Data Knowl. Eng.*, 57(3):240–282, 2006.

Obtaining clusters

The clustering method is not fixed in our system. Users can cluster cars along highways using a density-based method, or cluster birds in 3 dimension space using the k-means algorithm. Clustering methods that generate overlapping clusters are also applicable, such as EM algorithm or using a rigid definition of the radius to define a cluster. Moreover, clustering parameters are decided by users' requirements or can be indirectly controlled by setting the number of clusters at each timestamp.

Usually, most of clustering methods can be done in polynomial time. In our experiments, we used DBScan [1], which takes $O(|O_{DB}|log(|O_{DB}|) \times |T_{DB}|)$ in total to do clustering at every timestamp. To speed it up, there are also many incremental clustering methods for moving objects. Instead computing clusters at each timestamp, clusters can be incrementally updated from last timestamps.