On Discovery of Spatiotemporal Influence-Based Moving Clusters

DHAVAL PATEL, Indian Institute of Technology

A moving object cluster is a set of objects that move close to each other for a long time interval. Existing works have utilized object trajectories to discover moving object clusters efficiently. In this article, we define a spatiotemporal influence-based moving cluster that captures spatiotemporal influence spread over a set of spatial objects. A spatiotemporal influence-based moving cluster is a sequence of spatial clusters, where each cluster is a set of nearby objects, such that each object in a cluster influences at least one object in the next immediate cluster and is also influenced by an object from the immediate preceding cluster. Real-life examples of spatiotemporal influence-based moving clusters include diffusion of infectious diseases and spread of innovative ideas. We study the discovery of spatiotemporal influence-based moving clusters in a database of spatiotemporal events. While the search space for discovering all spatiotemporal influence-based moving clusters is prohibitively huge, we design a method, `STIMer`, to efficiently retrieve the maximal answer. The algorithm `STIMer` adopts a top-down recursive refinement method to generate the maximal spatiotemporal influence-based moving clusters directly. Empirical studies on the real data as well as large synthetic data demonstrate the effectiveness and efficiency of our method.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Data Mining

General Terms: Pattern Mining, Algorithms, Efficiency

Additional Key Words and Phrases: Spatiotemporal events, spatiotemporal influence-based moving clusters

ACM Reference Format:
Dhaval Patel. 2015. On discovery of spatiotemporal influence-based moving clusters. ACM Trans. Intell. Syst. Technol. 6, 1, Article 4 (March 2015), 23 pages.
DOI: http://dx.doi.org.10.1145/2631926

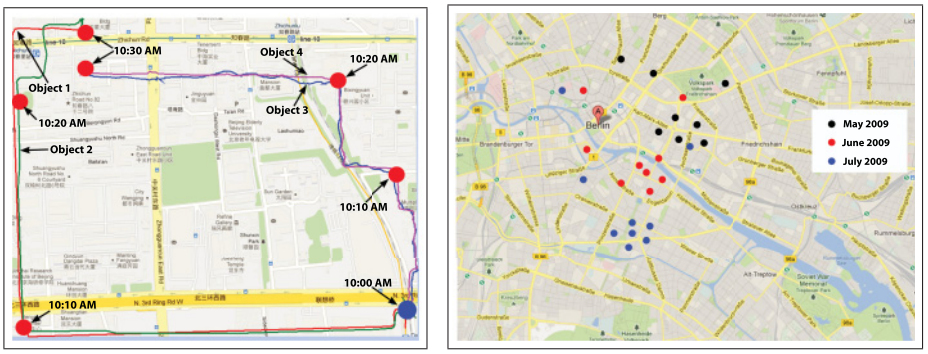
1. INTRODUCTION

Advances in sensing and satellite technologies and the rapid development of location-aware devices generate a large volume of spatiotemporal data in various domains. For example, the National Oceanic and Atmosphere Administration (NOAA) monitors and disseminates observations of hurricane tracks,[1] the GeoLife project tracks user movements using GPS-enabled mobile devices,[2] EpiscanGIS records the location and time information of infectious people in Germany,[3] social media services such as Twitter allows users to generate short messages with location and time information, and so forth. These scientific applications either monitor the location of moving objects in a continuous manner or record the occurrences of interesting events with location and

[1]http://weather.unisys.com/hurricane/.
[2]http://research.microsoft.com/en-us/projects/geolife/.
[3]http://episcangis.hygiene.uni-wuerzburg.de:8080/whatis.vm.

Author's address: D. Patel, S209, Department of Computer Science and Engineering, Indian Institute of Technology - Roorkee, Uttarakhand, India, PIN-247667; email: patelfec@iitr.ac.in.

DOI: http://dx.doi.org.10.1145/2631926



(a) GeoLife Project: Object's Movement in Beijing (Trajectory Data) (b) EpiScanGIS Project: Menignococcal Cases near Berlin (Spatio-temporal Data)

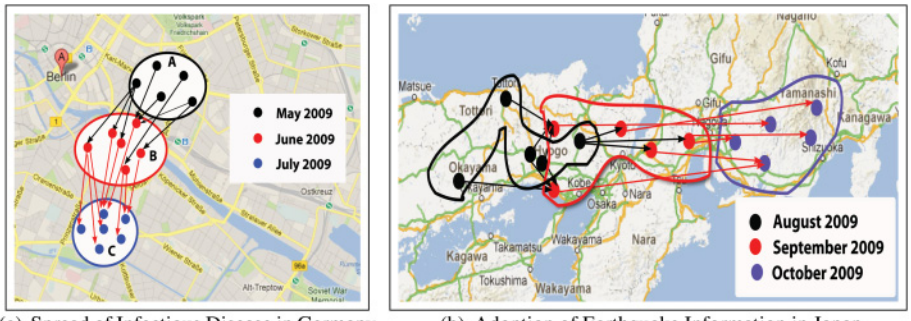
Fig. 1. Example of spatiotemporal dataset (best viewed in color).

time information. Accordingly, spatiotemporal data can be classified into **trajectory data** and **event data**.

In trajectory data, the movement of each object is maintained in space and time. For example, in Figure 1(a), we have visualized trajectories of four moving objects in Beijing from 10:00 a.m. to 10:30 a.m. These four objects start their journey from the same location and travel in different paths to reach their destinations. To date, a lot of research works have been dedicated to mining patterns from a trajectory database, such as clustering [Gaffney and Smyth 1999; Kriegel and Pfeifle 2005; Gonzalez et al. 2006; Kreveld and Luo 2007], frequent travel path [Gudmundsson et al. 2004; Monreale et al. 2009; Guo et al. 2010], and classification [Lee et al. 2007, 2008; Caon et al. 2010; Patel et al. 2012]. One interesting work is on discovering moving object clusters from trajectory data [Jeung et al. 2008; Li et al. 2010; Aung and Tan 2012; Tang et al. 2014]. A moving object cluster is a set of objects that move close to each other for a long time interval. For example, {Object 3, Object 4} is an example of a moving object cluster in Figure 1(a), since they move close to each other in space and time dimensions.

In spatiotemporal event data, each record describes what (event) happens where (location) and when (time). For example, in Figure 1(b), we have visualized a subset of meningococcal disease cases reported in Berlin over a period of 3 months. To date, a lot of research has been done for mining patterns from a spatiotemporal event database, such as colocation patterns [Celik et al. 2008; Shekhar and Huang 2001; Mohan et al. 2010], flow patterns [Wang et al. 2004], orientation patterns [Zhang et al. 2007; Wie and Shan 2006], spatiotemporal clusters [Birant and Kut 2007; Chang et al. 2008], and space-time hotspots [Neill et al. 2005; Kulldorff et al. 1998; Mohammadi et al. 2009]. A spatiotemporal cluster is a set of events that are close in space and time, whereas a space-time hotspot is a spatial region having a number of events higher than expected. Research in spatiotemporal cluster discovery or hotspot detection has been performed mostly in the field of epidemiology and crime analysis, where domain people are interested in identifying the dense or outlying regions for surveillance and monitoring purposes [Levine 2010; Zeng et al. 2010].

Up to this point, existing works in the area of spatiotemporal event analysis have analyzed the distribution of events irrespective of the order or time frame in which they have appeared. A number of events could occur within a short time period within a concentrated area. This type of effect is very common with motor vehicle thefts, flu cases, or dengue cases. A car thief gang may decide to attack a particular neighborhood. Similarly, infectious entities infect many people in a localized area. After a binge of car



(a) Spread of Infectious Disease in Germany (b) Adoption of Earthquake Information in Japan

Fig. 2. Example of spatiotemporal influence-based moving clusters in a real-world dataset.

thefts, the thief gang moves on to another neighborhood. Similarly, infected entities also move to another nearby region for a job, shopping, and so forth, and thus, entities from different regions get infected. In both situations, there are a number of events that occur within a limited period in a confined region. The clusters move from one location to another and there is a temporal interaction among these spatial clusters. The ability to detect this type of cluster shift in the temporal dimension is very important to capture the complex spatiotemporal evolution/diffusion, which concerns various domains such as agriculture, forest monitoring, ecology, hydrology, urbanization, and so forth.

In this article, we are interested in discovering spatiotemporal influence-based moving clusters from spatiotemporal events. The spatiotemporal influence-based moving cluster is a kind of spatiotemporal cluster that captures the spread of *influence* over a set of nearby objects. Influence can be in the form of infectious diseases, innovative ideas, or situation-aware information. Briefly, a spatiotemporal influence-based moving cluster is a sequence of spatial clusters, where each cluster is a set of nearby objects, such that each object in a cluster influences at least one object in the next immediate cluster and is also influenced by an object from the immediate preceding cluster. For example, in Figure 2(a), we have shown an example of a spatiotemporal influence-based moving cluster discovered from the EpiScanGIS dataset, $\{< A, May > \rightarrow < B, June > \rightarrow < C, July >\}$, where A, B, and C are clusters discovered in May, June, and July, respectively. Further, each case in cluster A infects at least one case in cluster B, and subsequently, each case in cluster B infects at least one case in cluster C. This pattern suggests that an infectious disease is moving from region A to region B and subsequently to region C. Another real-life example of spatiotemporal influence-based moving clusters is the adoption pattern of situation-aware information in a social network. For example, Figure 2(b) shows the spread of earthquake news among a set of Twitter users in Japan during the 2009 Japan earthquake [Sakaki et al. 2008].

Note that the spatiotemporal influence-based moving cluster proposed in this article is different from the moving object cluster and its variants [Kalnis et al. 2005]. In a spatiotemporal influence-based moving cluster, objects do not move; they influence other objects that are *spatially* and *temporarily* near. Further, a spatiotemporal influence-based moving cluster guarantees that there is a cluster at each time point and adjacent clusters are also connected by influence relationships. However, the traditional definition of a spatiotemporal cluster or hotspot does not enforce any of these requirements. As a result, existing approaches are not suitable for capturing the movements of phenomena over a distinct region. For example, three spatial clusters shown in Figure 2(b) are spatially far away and may not be detected as a single cluster. Compared to this, we use two distance thresholds, one to detect the spatial cluster and

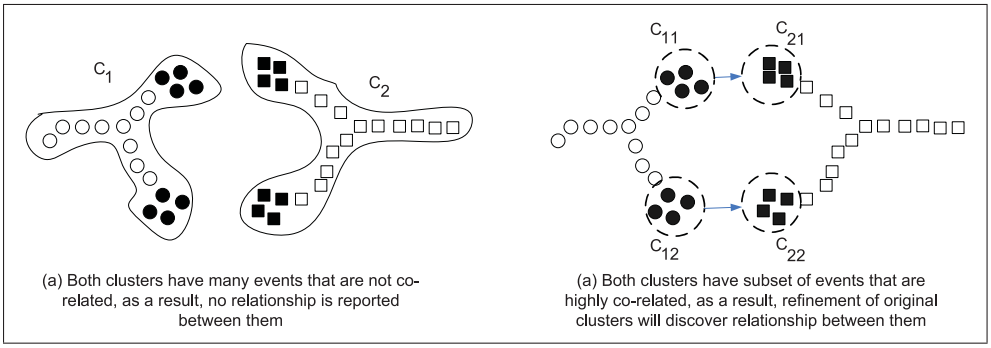


Fig. 3. Cluster C_1 and C_2 are discovered from time point t_i and t_{i+1}. Correlated events in both clusters are filled with black color.

another for establishing the influence relationship between the elements of adjacent clusters. Unlike existing approaches, use of two different distance thresholds allows analysts to specify the kinds of patterns they are looking for.

A simple approach to discover spatiotemporal influence-based moving clusters is to discover spatial clusters first and then establish relationships between the discovered clusters [Kalnis et al. 2005; Jeung et al. 2008; Tang et al. 2014]. Since cluster discovery and establishing relationships between them are performed independently, we argue that these methods miss patterns that are present at the subcluster level or capture misleading information. For example, consider two spatial clusters C_1 and C_2 discovered from different time points as shown in Figure 3. Since both clusters are spatially nonoverlapping and contain many events that are not related, existing approaches do not report any relationship between them. Careful observation of the example pattern reveals that there exists a relationship between their subclusters; that is, subcluster C_{11} is related to subcluster C_{21}. If the relationship is captured in between clusters C_1 and C_2, we cannot infer which sub-clusters are related in the pattern. The problem of missing patterns or misleading information can be addressed by generating all subclusters and working at the subcluster level. However, it is computationally impossible to generate all clusters and their subclusters, especially when a real-world spatiotemporal dataset gets bigger with millions of events. To address the scalability issue, we present a technique that generates clusters and then refines those clusters by incorporating the influence relationship information. Our key contributions in this article are as follows:

—A new concept, the spatiotemporal influence-based moving cluster, and its associated concept, **maximal** Spatiotemporal Influence-Based Moving Clusters, are introduced.
—We propose an algorithm, namely, spatiotemporal influence-based moving cluster Miner, in short, `STIMer`. The algorithm `STIMer` first generates spatiotemporal influence-based moving clusters of length 1, then length 2, and so on. While generating spatiotemporal influence-based moving clusters of a particular length, we follow a top-down refinement technique to generate the maximal pattern only.
—The effectiveness and efficiency of the `STIMer` algorithm have been demonstrated on both real and synthetic datasets. We have also discovered many interesting spatiotemporal influence-based moving clusters from the realworld datasets.

The remainder of the article is organized as follows. In Section 2, we discuss related work and then present the definition of the spatiotemporal influence-based moving cluster and its maximal patterns in Section 3. We provide a solution in Section 4 and present a naive algorithm in Section 5. Experiments testing the effectiveness and

efficiency of the proposed approach are shown in Section 6. Finally, we conclude our study in Section 7.

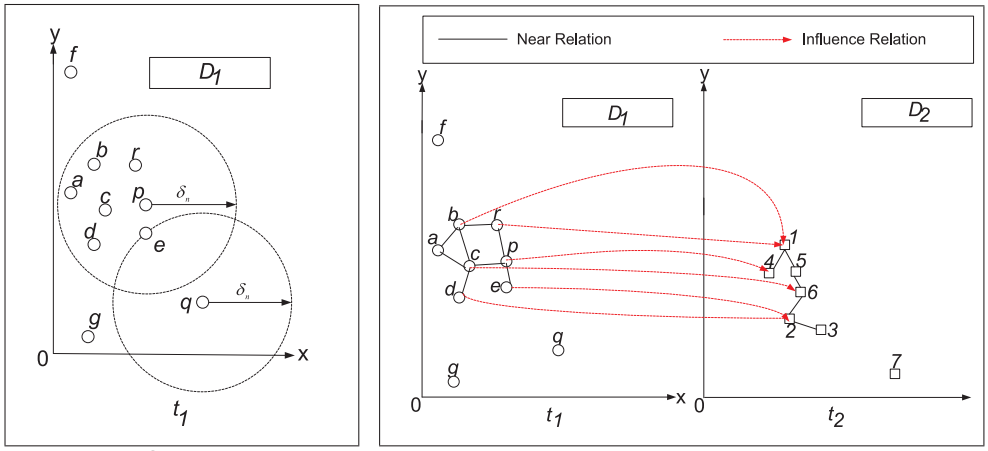
2. RELATED WORK

Clustering spatiotemporal data is the process of partitioning dataset into homogeneous subsets. Most of the research works in spatiotemporal data clustering are focused on the discovery of either trajectory clusters or moving object clusters. Gaffney and Smyth[1999] proposed a probabilistic model to group a set of trajectories. They adapted the density-based algorithm to trajectory data by using a distance measure between trajectories. Differently, Lee et al. [2007] proposed a partition-and-group framework for clustering trajectories, which partitions a trajectory into a set of segments and then groups similar segments into a trajectory cluster/network.

On the other hand, the moving object clusters group objects such that their identity remains unchanged, while cluster location and content may change over time [Li et al. 2004; Vieira et al. 2009; Kalnis et al. 2005; Spiliopoulou et al. 2006; Jeung et al. 2008]. The key difference is that a trajectory cluster has a constant set of objects throughout its lifetime, whereas the content of a moving cluster may change over time (i.e., one or more objects may leave the group or new objects may enter into it). A seminal method to discover moving clusters from the history of recorded trajectories has been proposed by Kalnis et al. [2005]. In this approach, spatial clusters are discovered at each snapshot by resorting to a static density-based clustering algorithm and results are then combined into a set of moving clusters. Note that combining clusters generated at different snapshots is easy because the object identified remains the same across different snapshots. Similar to moving object clusters, researchers have developed techniques to identify the cluster transitions. Spiliopoulou et al. [2006] categorized cluster transitions into internal and external transitions and described how to detect them. We note that almost all methods devised for discovering moving object clusters or detecting cluster transitions first discover clusters and then establish relationships between these discovered clusters.

Detecting clusters from spatiotemporal events is also an active area of research. The algorithm ST-DBSCAN [Birant and Kut 2007] was proposed to cluster events using spatial and temporal attributes. The authors utilized Quard tree implementation to speed up the point neighborhood query. However, as pointed out earlier, this approach does not consider the order in which events appear and thus is not suitable to detect a movement pattern.

Hotspot detection from spatiotemporal events is another active research area having applications in epidemiology and crime analysis. A hotspot refers to a geographical area where there is a higher concentration of events than expected. Note that a hotspot is not necessarily a dense region. Existing hotspot discovery methods in the statistical domain include spatial ellipse [Dong et al. 2012; Mohammadi et al. 2009], kernel density estimation [Maciejewski et al. 2011], and scan statistics [Kulldorff et al. 1998; Agarwal et al. 2006]. These techniques sweep a spatiotemporal scan window over all the locations to generate regions. For each generated region, statistical measures such as density, likelihood ratio, and so on are computed based on the observed number of events inside and outside that region. Regions whose measures are statistically significant are the hotspots. Apart from the high time complexity, these algorithms are sensitive to the size and shape of the spatiotemporal scan window. On the other hand, spatiotemporal data mining techniques for hotspot detection utilize clustering methods such as K-means, DBSCAN, nearest neighbor [Zeng et al. 2010], and support vector machine [Chang et al. 2008]. While these techniques discover irregularly shaped hotspots, they are not exhaustive and may miss some hotspots. In summary, none of



(a) A circle with δ_n radius is drawn on x-y plane with event p and q as a center

(b) Events connected by black solid line are spatially near and events connected by red dotted line represent influential relationships

Fig. 4. Datasets D_1 and D_2.

the existing techniques consider discovering a spatiotemporal influence-based moving cluster.

3. PRELIMINARIES

Let $T_{DB} = \{t_1, t_2, \ldots, t_n\}$ be the set of all time points under observation, and dataset $D_i, i = 1, 2, \ldots, n$, be the set of events recorded at time point t_i, $t_i \in T_{DB}$. Each event in dataset D_i is of the form $\langle x, y \rangle$, where $\langle x, y \rangle$ is the location information of the event. First, we present definitions involving concepts from the DBSCAN algorithm [Ester et al. 1996] regarding events from a single dataset D_i.

Definition 3.1. An event $p \in D_i$ is **near** another event $q \in D_i$ if the Euclidian distance between p and q is within the user-defined distance threshold δ_n.

Definition 3.2. An event $p \in D_i$ is a **core event** if the number of events *near* p is at least a minimum number, $MinPts$.

Definition 3.3. A core event $p \in D_i$ is **reachable** from another core event $q \in D_i$ if there is a chain of core events $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, $p_1 = q$ and $p_n = p$, such that p_{i+1} is *near* p_i with respect to δ_n, for $1 \leq i < n$, $p_i \in D_i$.

Definition 3.4. A **region** R in dataset D_i is a set of core events from D_i such that from any core event in R, all the other core events are reachable. The **size** of a region R, denoted as $size(R)$, is defined as the number of core events in it.

Definition 3.5. A region R in dataset D_i is **maximal** if its size ca not be increased. In short, there exists no other region that is a superset of region R.

We explain Definitions 3.1 to 3.5 using an example dataset D_1 given in Figure 4(a). In this example, event r is **near** event p, but event q is not **near** event p. Also, event p is a **core event** if $MinPts = 4$. The event set $\{a, b, c, d, e, r, p\}$ is a region with respect to the distance threshold δ_n and event p. Note that event q is not a part of this region, since it is not a core event. Further, $\{a, b, c, d, e, r, p\}$ is a maximal region, but $\{a, b, c\}$ is not a maximal region. Also, taking a subset of events from any region does not necessarily form a region. For example, $\{a, b, c, d, e, r, p\}$ is a region, but $\{b, e\}$ is

not a region as event b and event e are not reachable. Now, we present definitions that involve events from two adjacent datasets, that is, datasets D_i and D_{i+1}.

Definition 3.6. An event $q \in D_{i+1}$ is **influenced** by event $p \in D_i$ if their Euclidean distance is within the influence radius δ_i. We say that event p **influences** event q.

Definition 3.7. A **spatiotemporal influence-based moving cluster** is a pair $(\mathcal{R}, \mathcal{T})$, where $\mathcal{R} = \langle R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_m \rangle$ is a sequence of regions, and $\mathcal{T} = \langle t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_m \rangle$ is a sequence of consecutive time points, such that

—each time point t_i in $\mathcal{T}$ is from T_{DB}, and
—region R_i is discovered at time point t_i, and
—each event in R_i **influences** at least one event in R_{i+1}, $1 \leq i \leq m-1$, and
—each event in R_i is **influenced** by at least one event from R_{i-1}, $2 \leq i \leq m$.

A spatiotemporal influence-based moving cluster can also be represented as $P = \{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R_m, t_m)\}$. The *length* of a pattern P, denoted as $|P|$, is defined as the number of time points minus 1. The *size* of a pattern P, $size(P)$, is defined as the minimum $\{size(R_1), size(R_2), \ldots, size(R_m)\}$.

Definition 3.8. A spatiotemporal influence-based moving cluster $\{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R_m, t_m)\}$ is **maximal** if and only if both the *length* of the spatiotemporal influence-based moving cluster and the size of region R_k, $1 \leqslant k \leqslant m$, cannot be increased, that is,

— $\nexists$ (R_0, t_0) or (R_{m+1}, t_{m+1}) *s.t.* $\{(R_0, t_0) \rightarrow (R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R_m, t_m)\}$ or $\{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R_m, t_m) \rightarrow (R_{m+1}, t_{m+1})\}$ is a moving influence cluster, and
— $\nexists$ (R'_k, t_k) *s.t.* $\{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots (R'_k, t_k) \rightarrow \cdots \rightarrow (R_m, t_m)\}$ is a spatiotemporal influence-based moving cluster, and $R_k \subset R'_k$, $i \leqslant k \leqslant m$.

We illustrate Definitions 3.6 to 3.8 using an example given in Figure 4(b). We see that event p is near event r and event p influences event 4. Further, pattern $P_1 = \{(\{b, r, p\}, t_1) \rightarrow (\{1, 4\}, t_2)\}$ is a spatiotemporal influence-based moving cluster if we assume $MinPts$ is 2. Its size is 3 and length is 1. However, $P_2 = \{(\{b, r, p\}, t_1) \rightarrow (\{1, 4, 6\}, t_2)\}$ is not a valid pattern. This is due to the fact that no event from region $\{b, r, p\}$ influences event 6. Assuming we only have two datasets D_1 and D_2 under consideration, pattern P_1 is maximal as we cannot increase its size and length to form a new spatiotemporal influence-based moving cluster.

4. SOLUTION OVERVIEW

We outline the two-stage approach adopted by the proposed `STIMer` algorithm in Figure 5. In the **preprocessing stage**, we identify the subset of events from the input dataset that has the potential to generate results. Table I lists events that are not useful for generating spatiotemporal influence-based moving clusters and we get rid of these events in this stage. Next, we discover maximal regions using the remaining events. These maximal regions are the starting points of generating spatiotemporal influence-based moving clusters.

In the **pattern discovery stage**, we mine spatiotemporal influence-based moving clusters using regions generated in the previous stage. In this technique, we discover maximal patterns of length 1 and, subsequently, extend it further to discover the longer length patterns.

4.1. Preprocessing Stage

First, we index events from dataset D_i into grid index G_i. The size of each grid is $\frac{\delta_n}{2}$ x $\frac{\delta_n}{2}$, where δ_n is the user-defined distance threshold. An event $p = (x,y)$ from D_i is assigned

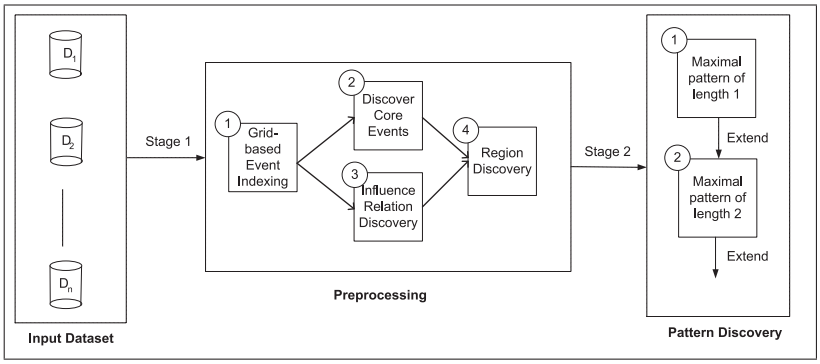


Fig. 5. Solution overview of mining spatiotemporal influence-based moving clusters by algorithm STIMer.

Table I. Events That Will Be Removed During Preprocessing Stage

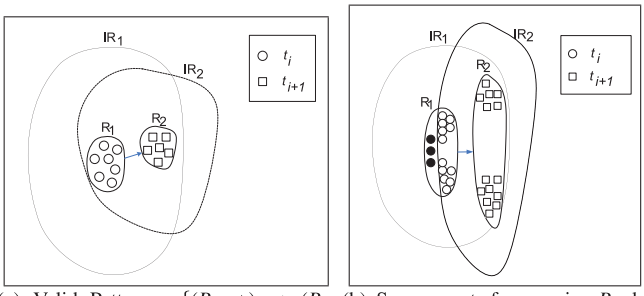
Sr. No.	Event Type	Remarks
1	Noncore event	Each region in proposed pattern contains only core events
2	An event that is neither influential nor influenced	Event in proposed pattern must influence or be influenced by other events

to grid $(\lfloor \frac{2*x}{\delta_n} \rfloor, \lfloor \frac{2*y}{\delta_n} \rfloor)$ in G_i. Once all events from D_i are indexed into G_i, each grid of G_i has a set of events that belong to it. Similarly, we prepare the grid index for the other dataset.

Once the grid index is prepared for each dataset D_i, we proceed to discover core events. The process of discovering core events from dataset D_i goes as follows: an event p from D_i is selected and the set of events that are *near* p is obtained using grid index G_i. Then, we count the number of events that are *near* event p. If this number is above the $MinPts$ threshold, event p is a core event. In this way, we process each event from dataset D_i and find all the core events. We also remove noncore events from D_i and update index G_i accordingly. Similarly, core events are discovered from the other datasets.

Next, we discover core events from dataset D_i that influence any event from dataset D_{i+1}. The process goes as follows: an event p from D_i is selected and the set of candidate events from dataset D_{i+1} that are influenced by p is obtained using grid index G_{i+1}. Next, we calculate the actual distance between event p and an event q from the candidate set. We say p influences q if the distance between them is less than δ_i. Similarly, other events from dataset D_i are processed. In this way, we obtain the set of core events from D_i that influence at least one event in dataset D_{i+1}. Once we process all datasets $D_1, D_2, \ldots, D_n$, we remove core events that are not influencing and influenced.

Now, we proceed to find regions from each dataset D_i. Recall, a region is a set of core events that are reachable. Initially, all core events in D_i are assigned to *tmpSet*. We randomly choose one event p from *tmpSet* and find all events from *tmpSet* that are near p. These events along with event p form an initial region R and are removed from *tmpSet*. Then, we check the remaining core events in *tmpSet*. If any core event q from *tmpSet* is near any event in region R, then we assign event q to region R and remove event q from *tmpSet*. We do this iteratively until no more core events from *tmpSet* can be assigned to region R. At this point, we have discovered a maximal region R. Next, we



(a) Valid Pattern : $\{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$

(b) Some events from region R_1 do not influence any event in R_2; events from R_2 do not form a single region

Fig. 6. Generating length 1 spatiotemporal influence-based moving cluster pattern using region R_1.

proceed to discover new regions from *tmpSet* until *tmpSet* is empty. Similarly, regions from another dataset D_i are discovered.

4.2. Discovery of Spatiotemporal Influence-Based Moving Clusters

We adopt a level-wise pattern growth approach to generate the spatiotemporal influence-based moving clusters. In particular, all maximal patterns of length 1 are discovered using regions generated in the previous stage. Then, we extend discovered maximal length 1 patterns to generate the length 2 patterns and so on.

4.2.1. Discover Length 1 *Pattern.* In this part, we process each region generated in the previous stage by considering it as a starting point. Assume we are processing a region R_1 discovered from dataset D_i. We first obtain a set of core events from dataset D_{i+1} that are influenced by any event from region R_1. Let R_2 be a set of such events. Now, we form an initial length 1 pattern $P = \{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$. If generated pattern P is a valid spatiotemporal influence-based moving cluster, we extend it further using the approach suggested in Section 4.2.2.

For example, consider a pattern $\{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$ shown in Figure 6(a). This pattern is obtained by extending the region R_1 from time point t_i. If R_2 is also a region (i.e., from any core event in R_2, all the other core events in R_2 are reachable), then pattern P is a valid spatiotemporal influence-based moving cluster. In detail, let IR_1 be a spatial region such that any event in this region is influenced by at least one event from R_1. Similarly, IR_2 represents a spatial region such that any event in this region influences at least one event in R_2. We can see that IR_1 contains R_2 and IR_2 contains R_1. Indirectly, each event in R_1 influences at least one event in R_2 and each event in R_2 is influenced by at least one event in R_1. Since R_1 and R_2 are regions, pattern $P = \{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$ is a valid spatiotemporal influence-based moving cluster. Note that the generated pattern is also a maximal pattern of length 1.

However, pattern P may not be a spatiotemporal influence-based moving cluster for the following two reasons: (1) some events in R_1 do not *influence* any event in R_2, and (2) core events in R_1 or R_2 do not form a *region*. For example, we present a case in Figure 6(b), where events shown with filled circles in region R_1 do not influence any event in region R_2, and events in R_2 do not form a single region. In this article, we present a **recursive top-down refinement** method to generate the maximal patterns from pattern P. This method works in three steps as follows:

(1) **Verify validity of pattern *P*.** We check whether pattern P is a valid spatiotemporal influence-based moving cluster or not. If pattern P is a valid pattern, we return P.

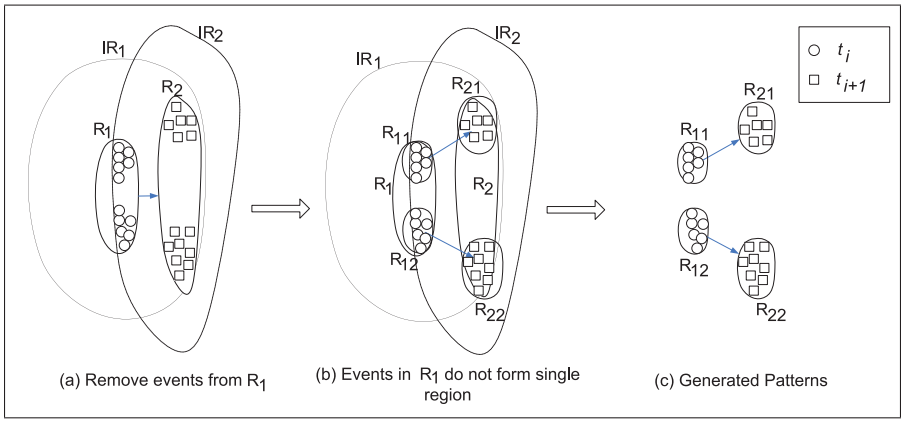


Fig. 7. Refinement of pattern $\{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$.

(2) **Remove events.** Assume some events in R_1 do not influence any event in R_2. Then, we remove all such events from R_1. Let R_1' be the set of remaining events in R_1. We modify pattern P as $\{(R_1', t_i) \rightarrow (R_2, t_{i+1})\}$ and start processing it from step 1. Similarly, we also remove some events in R_2 if they are not being influenced.

(3) **Refine region.** If the events in R_1 or R_2 do not form a single region:

—Assume the events from R_1 do not form a single region. Then, we first discover regions using the events from R_1. Assume we discover k regions $R_1^1, R_1^2, \ldots, R_1^k$ using the events from region R_1. Next, for each region R_1^i, we find events from R_2 that are influenced by any event from region R_1^i. Let R_2^i denote the set of such events for region R_1^i. Then, we form a new pattern $P_i = \{(R_1^i, t_i) \rightarrow (R_2^i, t_{i+1})\}$. In this way, we generate a total of k patterns and process each of them separately to identify valid patterns.

—Assume the events from R_2 do not form a single region and we discover a total of k regions $R_2^1, R_2^2, \ldots, R_2^k$ from R_2. Next, for each region R_2^i, we find events from R_1 that influence any event in region R_2^i. Let R_1^i denote the set of such events for region R_2^i. Then, we form pattern $P_i = \{(R_1^i, t_i) \rightarrow (R_2^i, t_{i+1})\}$. In this way, we generate a total of k patterns and follow a similar process to identify which of them are valid patterns.

The proposed recursive top-down refinement method is explained with an example. Again, consider a pattern P as shown in Figure 6(b). As mentioned, events with filled circles in region R_1 do not influence any event in R_2. Thus, we remove all such events from R_1. Figure 7(a) shows the resultant pattern. We observe that the resultant pattern is not a valid pattern as R_1 and R_2 do not form a single region. Thus, we refine regions. In this step, we (randomly) select R_1 and obtain its regions, denoted as $R_{1,1}$ and $R_{1,2}$. As explained, we use $R_{1,1}$ and $R_{1,2}$ and generate length 1 patterns $\{(R_{1,1}, t_i) \rightarrow (R_{2,1}, t_{i+1})\}$ and $\{(R_{1,2}, t_i) \rightarrow (R_{2,2}, t_{i+1})\}$, respectively. Both the generated patterns are valid maximal patterns, so no further refinement is required.

4.2.2. Discover Length $k+1$ Pattern. Now we have generated all length 1 patterns. Next, we extend the length $k\,(=1)$ pattern to discover length $k+1$ patterns, and so on. Assume pattern $P = \{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R_k, t_k)\}$ is a valid length k pattern. We select the last region, that is, R_k, from pattern P and find core events from time point t_{k+1} that are influenced by any event from R_k. Let R_{k+1} be the set of such events. Then, we form an initial length $k+1$ pattern, $Q = \{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R_k, t_k) \rightarrow (R_{k+1}, t_{k+1})\}$. Next,

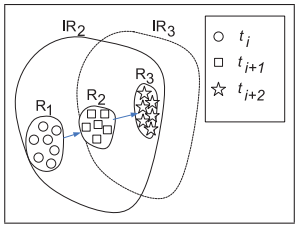


Fig. 8. Generate length 2 spatiotemporal influence-based moving cluster from length 1 pattern $\{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$: we select last region (i.e., R_2) for extension.

we perform top-down refinement of pattern Q to generate the maximal spatiotemporal influence-based moving clusters from Q. In the following example, we explain discovery of length 2 patterns from time point t_i.

For example, consider pattern $\{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$ and its extension as shown in Figure 8. Here, R_1 and R_2 are regions discovered at time points t_i and t_{i+1}, respectively. Let R_3 be a set of events from time point t_{i+2} that are influenced by any event from region R_2. Assume R_3 is a region (i.e., from any core event in R_3 all the other core events are reachable). Let IR_2 be a spatial region such that any event in this region is influenced by at least one event from R_2. Similarly, IR_3 represents a spatial region such that any event in this region influences at least one event in R_3. We can see that IR_2 contains R_3 and IR_3 contains R_2. Indirectly, $\{(R_1, t_i) \rightarrow (R_2, t_{i+1}) \rightarrow (R_3, t_{i+2})\}$ is a valid pattern of length 2.

However, if the extended pattern $Q = \{(R_1, t_i) \rightarrow (R_2, t_{i+1}) \rightarrow (R_3, t_{i+2})\}$ is not a valid pattern, we refine the pattern. This refinement method works in top-down fashion as follows:

(1) **Verify validity of pattern Q.** First, we check whether pattern Q is a valid spatiotemporal influence-based moving cluster or not. If pattern Q is a valid pattern, we do not process it further.
(2) **Remove noninfluential events.** If some events in R_i such that $i \in [1, k]$ do not influence any events in R_{i+1}, we remove all such events from R_i. Let R'_i be the set of events after removal. Finally, we modify the pattern $Q = \{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R'_i, t_i) \cdots \rightarrow (R_k, t_k)\}$. Now, we start processing Q from step 1.
(3) **Remove noninfluenced events.** If some events in R_i such that $i \in [2, k+1]$ are not influenced by any event from R_{i-1}, we remove all such events from R_i. Let R''_i be the set of events after removal. Finally, we form a new pattern $Q = \{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R''_i, t_i) \cdots \rightarrow (R_k, t_k)\}$. Now, we start processing Q from step 1.
(4) **Refine regions.** If the events from R_i such that $i \in [1, k+1]$ do not form a single region, we find regions using events from R_i. Assume, using events from R_i, we have generated k regions $R_i^1, R_i^2, \ldots, R_i^k$. Finally, we generate a total of k patterns, where the i^{th} pattern is $Q_i = \{(R_1, t_1) \rightarrow (R_2, t_2) \rightarrow \cdots \rightarrow (R^i_{i-1}, t_{i-1}) \rightarrow (R_i^i, t_i) \rightarrow (R^i_{i+1}, t_{i+1}) \cdots \rightarrow (R_k, t_k)\}$. Now, we process each generated pattern Q_i from step 1 that influences any event in R_i^i.

We explain this refinement process with an example. Consider the pattern shown in Figure 9(a). This pattern is obtained by extending the length 1 pattern $\{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$. Events with filled rectangles in region R_2 do not influence any event in R_3. Thus, we remove all such events from R_2. Figure 9(b) shows the resultant pattern.

Since we have removed some events from R_2, there are some events in R_1 that do not influence any event in R_2 (see Figure 9(b)). Figure 9(c) shows the resultant pattern after removing the noninfluential events from R_1.

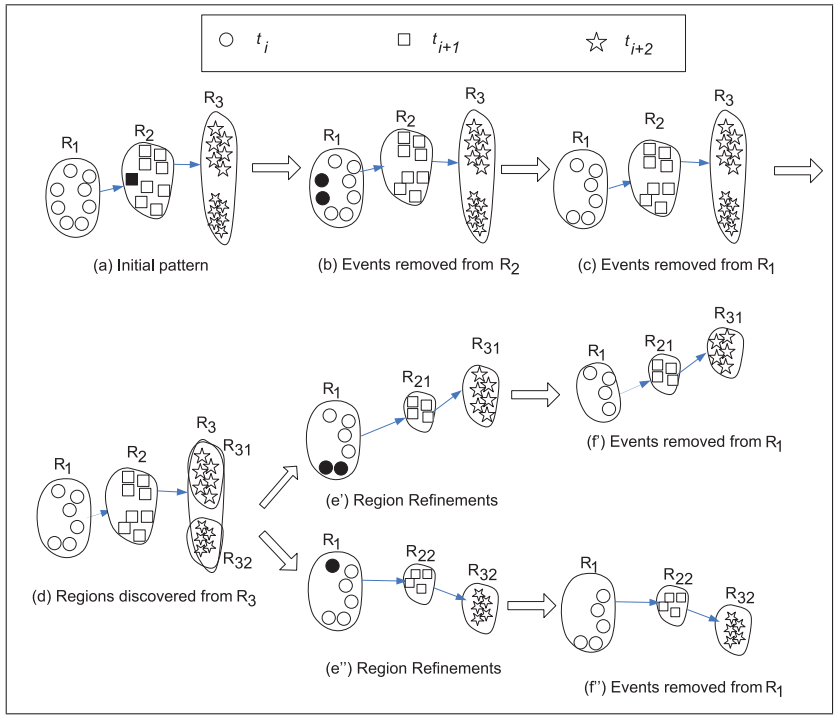


Fig. 9. Extending length 1 spatiotemporal influence-based moving cluster: $\{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$.

Now, we observe that the resultant pattern is not valid as events from R_3 do not form a single region. Thus, we refine region R_3. Let, $R_{3,1}$ and $R_{3,2}$ be the two regions obtained from region R_3. Next, we form two length 2 patterns, $\{(R_1, t_i) \rightarrow (R_{2,1}, t_{i+1}) \rightarrow (R_{3,1}, t_{i+2})\}$ and $\{(R_1, t_i) \rightarrow (R_{2,2}, t_{i+1}) \rightarrow (R_{3,2}, t_{i+2})\}$ (see Figure 9(e') and Figure 9(e")). Both of the generated patterns are processed separately to judge whether they are valid patterns or not.

Once all the valid patterns of length $k+1$ are generated, we perform an additional check to see whether they consume the previously generated length k maximal pattern. The algorithmic description of `STIMer` is available in Appendix A.

5. NAIVE ALGORITHM

In this section, we present a bottom-up algorithm motivated by the pattern growth approach [Huang et al. 2008]. This approach also mines length 1 patterns first, then length 2 patterns, and so on. However, while discovering spatiotemporal influence-based moving clusters of a particular length, the naive approach follows a bottom-up methodology to increase the size of the spatiotemporal influence-based moving cluster. In the following section, we explain the techniques to discover length 1 and length 2 spatiotemporal influence-based moving clusters.

5.1. Discover Length 1 Patterns

Assume event i influences event j, where event i is from time point t_i and event j is from time point t_{i+1}. Using this information, we form an initial pattern of size 1 and length 1 pattern, that is,

$$P = \{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}, \quad R_1 = \{i\}\ , R_2 = \{j\}.$$

Now, we **expand** pattern P to increase the size of any region in pattern P. For this, we find a set of events from time point t_i that are *near* any event from R_1, as well as a set of events from time point t_{i+1} that are *near* any event from R_2. Assume $\{i_1, i_2, \ldots, i_k\}$ and $\{j_1, j_2, \ldots, j_k\}$ are sets of such events for R_1 and R_2, respectively. Thus, the resultant pattern is

$$P' = \{(R'_1, t_i) \rightarrow (R'_2, t_{i+1})\},\ R'_1 = R_1 \cup \{i_1, i_2, \ldots, i_k\},\ R'_2 = R_2 \cup \{j_1, j_2, \ldots, j_k\}.$$

The pattern P' may not be a valid spatiotemporal influence-based moving cluster. Thus, we **refine** pattern P' by removing events from R'_1 that do not influence any event in R'_2. We also remove events from R'_2 that are not influenced by any event from R'_1.

After pattern refinement, the resultant pattern is a valid spatiotemporal influence-based moving cluster. But it may not be **maximal**. Thus, we iteratively follow the **expand** and **refine** paradigm until the size of any region in the pattern cannot be increased further. At the end, we have generated a maximal pattern of length 1. Similarly, we process other influence relations in our database and discover all maximal length 1 patterns.

5.2. Discover Length 2 Patterns

Now, we explain how length 2 patterns from time point t_i are discovered. Let events i, j, and k be recorded at time points t_i, t_{i+1}, and t_{i+2}, respectively. Assume event i influences event j and event j influences event k. With this information, we form an initial pattern of size 1 and length 2 that starts from time point t_i, that is,

$$P = \{(R_1, t_i) \rightarrow (R_2, t_{i+1}) \rightarrow (R_3, t_{i+2})\}, R_1 = \{i\}, R_2 = \{j\}, R_3 = \{k\}.$$

Next, we **expand** this pattern; that is, we find the set of events from time point t_i that are near any event from R_1, the set of events from time point t_{i+1} that are near any event from R_2, and the set of events from time point t_{i+2} that are near any event from R_3. Assume $\{i_1,i_2, \ldots, i_k\}$, $\{j_1, j_2, \ldots, j_k\}$, and $\{k_1, k_2, \ldots, k_k\}$ are the sets of such events for R_1, R_2, and R_3, respectively. Thus, the resultant pattern is

$$P' = \{(R'_1, t_i) \rightarrow (R'_2, t_{i+1}) \rightarrow (R'_3, t_{i+2})\},$$

where $R'_1 = R_1 \cup \{i_1, i_2, \ldots, i_k\}$, $R'_2 = R_2 \cup \{j_1, j_2, \ldots, j_k\}$, and $R'_3 = R_3 \cup \{k_1, k_2, \ldots, k_k\}$. Once the pattern is expanded, we **refine** P' as follows:

—If some events in $R'_1(R'_2)$ do not influence any event from $R'_2(R'_3)$, we remove such events from $R'_1(R'_2)$.
—If some events in $R'_2(R'_3)$ are not influenced by any event from $R'_2(R'_1)$, we remove such events from $R'_2(R'_3)$.

The resultant pattern is a valid pattern and will be expanded and refined further until the size of the region in the pattern cannot be increased. At the end, we have generated the maximal pattern of length 2.

6. EXPERIMENTAL EVALUATION

In this section, we present the results of the experiments conducted to evaluate the algorithm `STIMer`.[4] To compare the efficiency of the proposed algorithm, we selected two existing algorithms based on the `DBSCAN` algorithm, namely, `ST-DBSCAN` [Birant and Kut 2007] and `MONIC` [Spiliopoulou et al. 2006]. The algorithm `ST-DBSCAN` discovers spatiotemporal clusters using the spatial distance threshold δ_n and a temporal threshold. In this article, we set the temporal threshold to 1 for `ST-DBSCAN`. The algorithm `MONIC` [Spiliopoulou et al. 2006] discovers spatial clusters using the spatial distance

[4]http://people.iitr.ernet.in/facultywebsite/patelfec/Website/SourceCode/STIMer_code.rar.

Table II. User-Defined Parameters

(a) OutbreakSimSTevents database

Parameters	Range	Default Value
MinPts	[7,15]	7
δ_n	[0.01,0.02]	0.01
δ_i	[0.01,0.02]	0.01
Dataset Size	[10k,50k]	50k

(b) RandomSTevents database

Parameters	Range	Default Value
MinPts	[7,15]	7
δ_n	[0.01,0.02]	0.01
δ_i	[0.01,0.02]	0.01
Dataset Size	[10k,100k]	100k

threshold δ_n from each time point and connects two clusters if the maximum distance between them is less than an influence distance threshold δ_i. We have implemented `STIMer`, `NaiveMiner`, `ST-DBSCAN` [Birant and Kut 2007], and `MONIC` [Spiliopoulou et al. 2006] in C++ language. The experiments have been performed using an Intel Core 2 Quad CPU 2.83GHz system with 3GB of main memory and run on the Windows XP operating system.

6.1. Efficiency Experiments on Synthetic Datasets

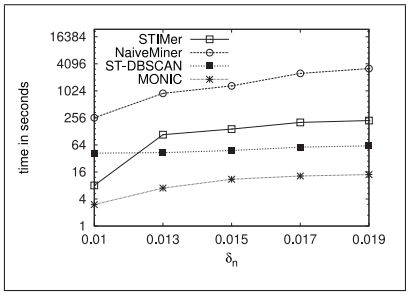
Synthetic datasets are used to evaluate the efficiency of the proposed algorithm. We use two different publicly available data generators, the OutbreakSim simulator [Watkins et al. 2007] and the random Spatio-Temporal Event Generator [Huang et al. 2008].

—The **OutbreakSim simulation model** [Watkins et al. 2007] mimics the real-world disease outbreak data for Western Australia. This simulation model uses the daily notifiable disease surveillance system to obtain the baseline information for disease spread. The 10% random sample of address points has been used to represent the population distribution of Western Australia (around 2 million individuals), and the postcode boundaries have been used for summary purposes. We have simulated endemic disease data using the Poisson endemic method with a reasonably low average case rate of 0.5 cases per day to avoid using confidential endemic data. We have generated a total of 50,000 outbreak cases over a period of 100 days. This dataset is referred to as OutbreakSimSTevents from now on.

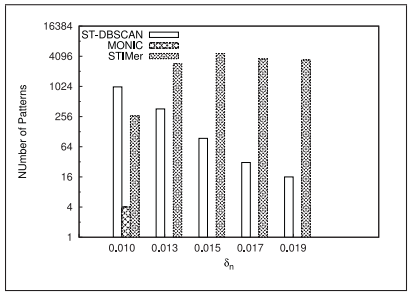
—The **Spatio-Temporal Event Generator model** [Huang et al. 2008] randomly generates spatiotemporal events in predefined grid regions. The input to this generator is the total number of events we want to generate. The events are distributed over a predefined square region of area $4cm^2$ with $1{,}000 \times 1{,}000$ grids for a 200-day period. This dataset is referred to as RandomSTevents from now on. By default, we have generated 100,000 spatiotemporal events for RandomSTevents.

We compare the runtime efficiency of the four algorithms on both synthetic datasets. Recall that `STIMer`, `NaiveMiner`, and `MONIC` have three user-defined parameters, namely, *MinPts*, δ_n, and δ_i. Thus, we evaluate them by varying these parameters one by one. Note that the algorithm `ST-DBSCAN` has two parameters, namely, *MinPts* and δ_n. Table II lists the range of these parameters for both synthetic datasets. Interested readers are advised to read the supplementary material provided on the website. Apart from evaluating algorithms on user-defined parameters, we vary the number of events (i.e., database size D) to study the scalability of all algorithms. While performing an experiment, we have captured the running time and number of patterns discovered by each algorithm.

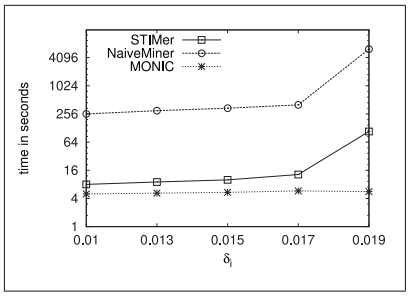
Figure 12 and Figure 15 show the experimental results of varying δ_n, δ_i, *MinPts*, and dataset size on the OutbreakSimSTevents and RandomSTevents datasets, respectively. We observe that, overall, the algorithm `STIMer` outperforms the algorithm `NaiveMiner` for both datasets. The algorithm `MONIC` converges quickly but discovers fewer patterns.



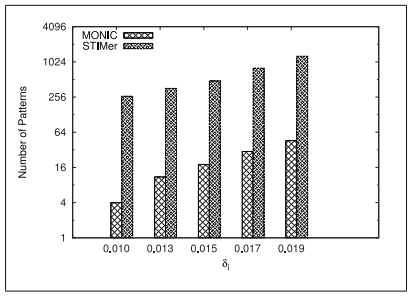
(a) Varying δ_n



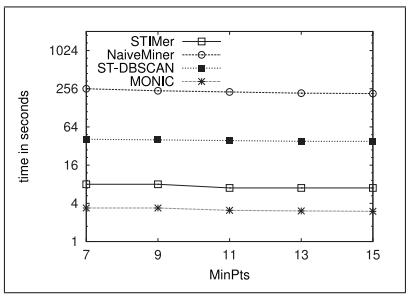
(a) Varying δ_n



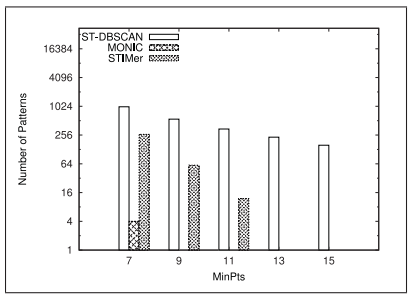
(b) Varying δ_i



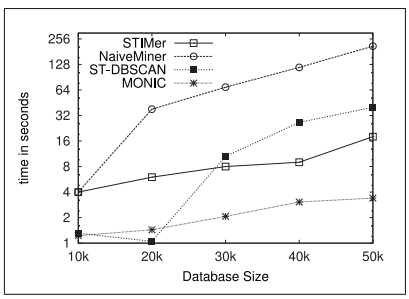
(b) Varying δ_i



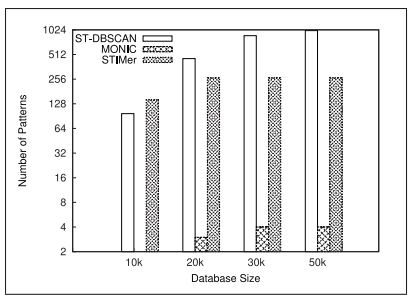
(c) Varying *MinPts*



(c) Varying *MinPts*



(d) OutbreakSimSTevents database

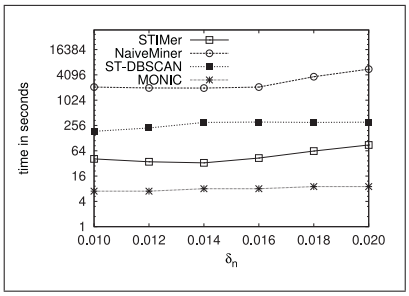


(d) Varying D

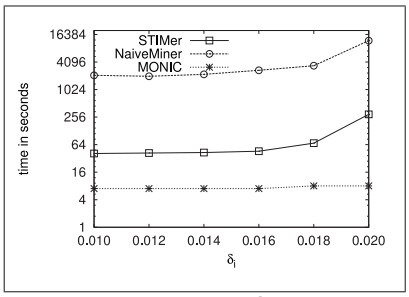
Fig. 10. Running time comparison

Fig. 11. Number of Generated Patterns

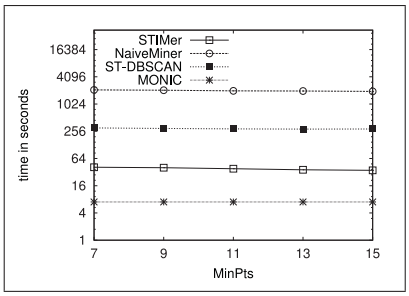
Fig. 12. Experiments on OutbreakSimSTevents database.



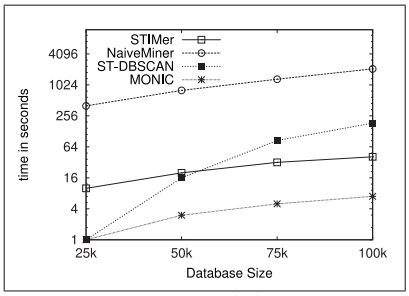
(a) Varying δ_n



(b) Varying δ_i

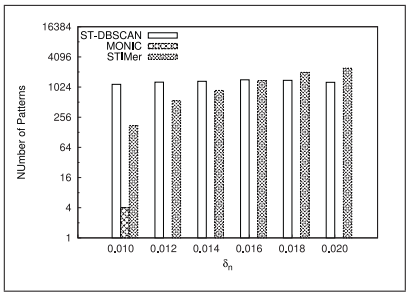


(c) Varying *MinPts*

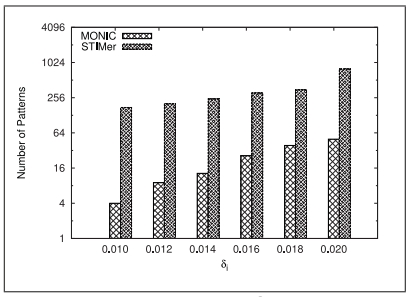


(d) Vary $|D|$

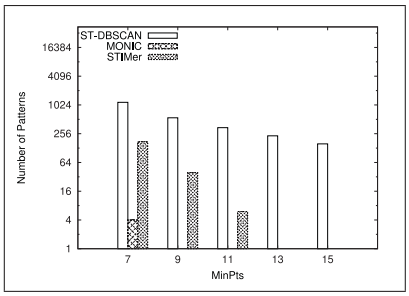
Fig. 13. Running time comparison



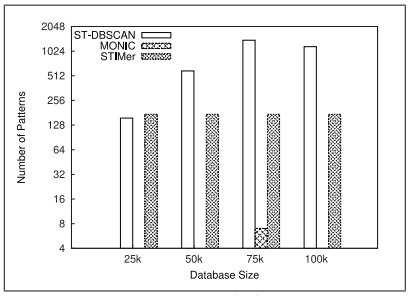
(a) Varying δ_n



(b) Varying δ_i



(c) Varying *MinPts*



(d) Vary $|D|$

Fig. 14. Number of Generated Patterns

Fig. 15. Experiments on RandomSTevents database.

The overall performance of STIMer and ST-DBSCAN is comparable. Overall, we have observed that the running time of the algorithm is proportional to the number of discovered patterns. Note that all algorithms report maximal patterns only and the algorithms STIMer and NaiveMiner generate an equal number of patterns.

We observe that varying the spatial near-distance threshold (i.e., δ_n) drastically increases the runtime of NaiveMiner (see Figure 10(a) and Figure 13(a)). Note that increasing δ_n increases the size of regions. Thus, the bottom-up methodology-based NaiveMiner algorithm takes a longer time to reach the maximal region. In case of ST-DBSCAN, the number of clusters consistently reduces as we increase the δ_n. However, the generated clusters are large in size as each cluster groups many events from different time points.

Similarly, increasing the influence relation threshold (i.e., δ_i) results in many influential events and thus generates many patterns. Recall that the running time of both algorithms depends on the number of patterns being generated. Since the top-down refinement approach directly mines the maximal pattern, it generates fewer candidate patterns than the bottom-up approach. As a result, the STIMer algorithm significantly outperforms NaiveMiner (see Figure 10(b) and Figure 13(b)). Recall that ST-DBSCAN does not have this parameter. In case of the algorithm MONIC, the total number of discovered patterns has slightly reduced, but the length of discovered patterns has increased. This is due to the fact that, as we increase δ_i, more clusters get connected. Increasing *MinPts* reduces the number of core events in the dataset and, in turn, generates fewer patterns. As a result, increasing *MinPts* reduces the running time of all the algorithms (see Figure 10(c) and Figure 13(c)).

Next, we evaluate the scalability of all algorithms by varying the number of events (i.e., database size). Figures 10(d) and 13(d) present the evaluation results for both datasets. We have randomly sampled events from the original dataset to get the subset of the data. Again, the algorithm STIMer works very fast for small-sized datasets. Also, as the database size increases, the running time of all the algorithms also increases, but the algorithm STIMer takes just less than a minute to process 100K events. We have observed that ST-DBSCAN is the worst affected as it starts generating many clusters.

6.2. Experiments on Real-World Datasets

We have discovered the spatiotemporal influence-based moving clusters from two real-world datasets as discussed next.

—**Meningococcal dataset.** This is a real-world dataset collected by EpiScanGIS[5] on meningococcal disease in Germany over a period of 9 years (from January 2003 to December 2011). This dataset contains 3,756 meningococcal disease cases. Meningococcal disease carries a high mortality rate if left untreated and is the major cause of illness, death, and disability in both developed and underdeveloped countries worldwide. Given that it is a contagious disease, the ability to predict where it will spread or not is important in preventing disease outbreak. We set the time granularity for this dataset to 1 month; that is, cases reported in the same month are grouped into a single database. In total, we have 108 months.

—**Crime dataset.** This dataset contains 19,418 crime (burglary) cases that have been reported in Washington D.C. over a period of 5 years[6] (from January 2008 to December 2012). We set the time granularity for this dataset to 1 week; that is, cases

[5]http://episcangis.hygiene.uni-wuerzburg.de/.
[6]http://data.octo.dc.gov//.

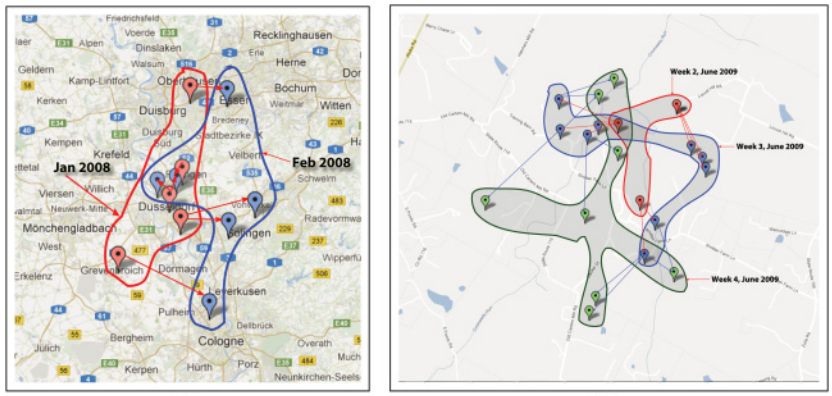


Fig. 16. Example of spatiotemporal influence-based moving clusters in (a) meningococcal dataset and (b) crime dataset. Red markers are the cases reported in January 2008 [in (a)] and the second week of June 2008 [in (b)]. Blue markers are the cases reported in February 2008 [in (a)] and the third week of July 2008 [in (b)]. Green markers in (b) are the cases reported in the fourth week of July 2008. Markers connected by an arrow represent an influence relationship.

reported in the same week are grouped into a single database. In total, we have around 260 weeks.

We set $MinPts = 3$, $\delta_n = 5$ kilometers and $\delta_i = 5$ kilometers to discover the spatiotemporal influence-based moving clusters from the meningococcal dataset. Recall, the cases in the meningococcal dataset have been partitioned monthly. This is due to the fact that a patient with this disease requires prolonged contact with a person to pass on the infection. In total, we have discovered seven spatiotemporal influence-based moving clusters. Figure 16(a) presents an example pattern, where a cluster discovered in January 2008 is shifted to the east direction.

We set $MinPts = 3$, $\delta_n = 1.5$ kilometers and $\delta_i = 1.5$ kilometers to discover the spatiotemporal influence-based moving clusters from the crime dataset. Recall, we have partitioned the cases in the crime dataset weekly. In total, we have discovered 180 spatiotemporal influence-based moving clusters. Figure 16(b) presents an example pattern. As expected, spatial locations of the cases reported in 1 week are very near that of the cases reported in the previous week but not exactly at the same location. Careful observation of these patterns may suggest the spatial regions where patrolling should be increased by police officials.

7. VISUAL ANALYSIS OF DISCOVERED PATTERNS

For the visual analysis of the discovered knowledge, instead of visualizing all the patterns as they are, we postprocess the discovered cluster patterns to infer the spatiotemporal movements. The spatiotemporal movements help us to visualize the evaluation of the underlying process. In particular, we obtain the spatiotemporal moving average from the (raw) event dataset. A spatiotemporal moving average is the moving mean center of $MinPts$ observations, where $MinPts$ is a subset of all the samples in the dataset. Here, the observations in the input dataset are sequences in the order of their occurrences before they are processed. Hence, there is a time dimension associated with the sequence.

A spatiotemporal moving average is simple but very much useful for detecting the changing behavior of the underlying disease/offenders. For example, Figure 17(a) shows

(a) Spatio-temporal moving average
(b) Movement paths generated by MONIC
(c) Movement paths generated by STIMer
(d) Movement paths generated by ST-DBSCAN

Fig. 17. Visual analysis of discovered patterns from meningococcal datasets.

the moving patterns for the meningococcal datasets. We observed that a path in meningococcal disease passes through the major metro cities. However, discovering the global path does not capture the localized movement that happens between nearby local areas frequently.

Similarly, each pattern discovered by the algorithms MONIC and STIMer represents one movement. Given a cluster pattern generated either by MONIC or by STIMer, we represent each cluster by its center and obtain a path that passes through the centers. Later, we plot all the paths for visualizations. Figure 17(b) and Figure 17(c) visualize the discovered paths from the meningococcal dataset by MONIC and STIMer, respectively. Compared to patterns discovered by MONIC, we observe that STIMer discovers many overlapping paths. This happens because the repeated disease spread happened over a period of time. These results indirectly infer the direction of virus spread and its frequency.

Since the ST-DBSCAN algorithm does not generate a cluster sequence, we obtain a movement by grouping all the events coming from the same time point and represent them by the cluster centers (see Figure 17(d)). We ignore the patterns generated by ST-DBSCAN if they do not contain movement. We observed that patterns generated by ST-DBSCAN are local and do not really capture the movement.

8. CONCLUSION

In this article, we introduce the problem of mining spatiotemporal influence-based moving clusters from spatiotemporal events. First, we define the spatiotemporal influence-based moving cluster that captures the spread of influence over a set of objects. Next, we introduce the concept of the maximal spatiotemporal influence-based moving cluster to address the problem of removing redundant patterns. We design a novel method, STIMer, to efficiently retrieve the maximal answer. The algorithm STIMer adopts a top-down recursive refinement method to generate the maximal spatiotemporal influence-based moving clusters directly. We have tested the efficiency of the STIMer algorithm on large synthetic datasets. We have also discovered patterns from two real-world datasets. In the future, we plan to attach semantic meaning to each spatiotemporal influence-based moving cluster, such as whether the spatiotemporal influence-based moving cluster is static, expanding or relocating, and so forth.

APPENDIX

Algorithm 1 outlines the details of the proposed algorithm STIMer. In Lines 2 to 9, we index events into a grid-based index structure and discover core events and events that are influential or being influenced. The third for loop (Lines 11–18) intends to discover length 1 maximal patterns and the later part of Algorithm 1 discovers length k ($>$1) maximal patterns (Lines 9–21).

Algorithm Complexity. The preprocessing stage visits each point of the given database to discover the core points. For the core point discovery from dataset D_i, however, the time complexity is mostly governed by the number of neighborhood query invocations. Our algorithm executes exactly one such query for each point in dataset D_i, and as we have used an indexing structure that executes such neighborhood query in $O(log|D_i|)$ for each [Birant and Kut 2007], an overall runtime complexity of $O(|D_i|log|D_i|)$ is obtained for the core point discovery from dataset D_i. Similarly, the runtime complexity is $O(|D_i|log|D_{i+1}|)$ for discovering the influence and influenced relation and $O(D_i)$ is for discovering regions from dataset D_i. Overall, the time complexity of the preprocessing stage is

$$O\left(\sum_{i=1}^{n}(|D_i|log|D_i|)+\sum_{i=1}^{n-1}(|D_i|log|D_{i+1}|)+\sum_{i=1}^{n-1}(|D_i|)\right).$$

Our experimental results highlight that the running time of the algorithm STIMer depends on the number of patterns it generates. Let n_i be the total number of regions that are generated from the snapshot dataset D_i; then the total number of spatiotemporal influence-based moving clusters in the worst case is $\Pi_{i=1}^{n} n_i$. Verifying whether a given spatiotemporal influence-based moving cluster is valid or not, we need to process each element in the cluster. Thus, the cost of evaluating the validity of each pattern is the size of that pattern. As a result, time complexity of the pattern discovery stage is exponential in terms of the number of generated regions.

ALGORITHM 1: Outline of Discovering Spatiotemporal Influence-based Moving Cluster

Data: $\{D_1, D_2, \ldots, D_n\}$, $TB = \{t_1, t_2, \ldots, t_n\}$, $MinPts$, δ_n, δ_i
Result: A set of spatio-temporal influence-based moving clusters $MICset$
/* Preprocessing Stage */
$RSet = \phi$;
for *each dataset* D_i *at time point* t_i **do**
  index all events from D_i into grid-based index G_i;
end
for *each dataset* D_i *at time point* t_i **do**
  discover ***core events*** from dataset D_i using index G_i;
  discover ***influential events*** from dataset D_i using index G_{i+1};
  $RSet \cup = \{$obtain maximal ***regions*** from dataset D_i using $G_i\}$;
end
/* Discover length 1 spatiotemporal influence-based moving cluster */
$extSet = \phi$;
for *each time point* $t_i \in T$ **do**
  for *each region* $R_1 \in RSet$ *generated from time points* t_i **do**
    $R_2 = \{$events from time point t_{i+1} that are influenced by any event from $R_1\}$;
    Generate initial pattern $P = \{(R_1, t_i) \rightarrow (R_2, t_{i+1})\}$;
    $Set_1 = \{$**top-down refinement** of pattern $P\}$;
    $extSet_1 = extSet_1 \cup Set_1$;
  end
end
/* Discover length $k+1$ spatiotemporal influence-based moving clusters */
$k = 1$;
repeat
  $canSet = extSet_k$;
  while $canSet \neq \phi$ **do**
    Select pattern P from $canSet$;
    Remove P from $canSet$;
    $Q =$ **extend** pattern P;
    $Set_{k+1} = \{$**top-down refinement** of pattern $Q\}$;
    $extSet_{k+1} = extSet_{k+1} \cup Set_{k+1}$;
  end
  remove patterns from $extSet_k$ that are subpattern of any pattern in $extSet_{k+1}$;
  $k = k + 1$;
until $extSet_{k+1} = \phi$;
$MICset = \bigcup_{i=1,\ldots k} extSet_i$;
return $MICset$;

REFERENCES

D. Agarwal, A. McGregor, J. Phillips, S. Venky, and Z. Zhu. 2006. Spatial scan statistics: Approximations and performance study. In *SIGKDD*. 24–33.

H. Aung and K. Tan. 2012. Mining multi-object spatial-temporal movement patterns. *SIGSPATIAL Special* 4, 3 (Nov. 2012), 14–19.

D. Birant and A. Kut. 2007. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data Knowl. Eng.* 60, 1 (2007), 208–221.

X. Caon, G. Cong, and C. Jensen. 2010. Mining significant semantic locations from GPS data. In *VLDB*. 320–332.

M. Celik, S. Shekar, J. Rogers, J. Shine, and J. Kang. 2008. Mining at most top-k mixed-drove spatio-temporal co-occurrence patterns: A summary of results. In *SSTD*. 187–192.

W. Chang, D. Zeng, and H. Chen. 2008. A spatio-temporal data analysis approach based on prospective support vector clustering. *Decision Support Systems* (2008).

W. Dong, X. Zhang, L. Li, C. Sun, L. Shi, and S. Wei. 2012. Detecting irregularly shaped significant spatial and spatio-temporal clusters. *SDM*. 732–743.

M. Ester, H. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD.* 226–231.

S. Gaffney and P. Smyth. 1999. Trajectory clustering with mixtures of regression models. In *KDD*. 63–72.

H. Gonzalez, J. Han, and X. Li. 2006. Mining compressed commodity workflows from massive RFID data sets. In *IKM*. 162–171.

J. Gudmundsson, M. van Kreveld, and B. Speckmann. 2004. Efficient detection of motion patterns in spatio-temporal data sets. In *GIS*. 20–25.

D. Guo, S. Liua, and H. Jina. 2010. A graph-based approach to vehicle trajectory analysis. In *JLBS*, 4. 183–199.

Y. Huang, L. Zhang, and P. Zhang. 2008. A framework for mining sequential patterns from spatio-temporal event data set. In *TKDE*. 433–448.

H. Jeung, M. Yiu, X. Zhou, C. Jensen, and H. Shen. 2008. Discovery of convoys in trajectory databases. In *PVLDB*. 1068–1080.

P. Kalnis, N. Mamoulis, and S. Bakiras. 2005. On discovering moving clusters in spatio-temporal data. In *SSTD*. 364–381.

M. Kreveld and J. Luo. 2007. The definition and computation of trajectory and subtrajectory similarity. In *GIS*. 1–4.

H.-P. Kriegel and M. Pfeifle. 2005. Clustering moving objects via medoid clusterings. In *SSDBM*. 153–162.

M. Kulldorff, W. Athas, E. Feuer, B. Miller, and C. Key. 1998. Evaluating cluster alarms: A space-time scan statistic and brain cancer in Los Alamos. In *AJPH*. 1377–1380.

J.-G. Lee, J. Han, and X. Li. 2008. Trajectory outlier detection: A partition-and-detect framework. In *ICDE* (2008), 140–149.

J.-G. Lee, J. Han, and K.-Y. Whang. 2007. Trajectory clustering: A partition-and-group framework. In *SIGMOD*. 10–22.

N. Levine. 2010. CrimeStat: A Spatial Statistics Program for the Analysis of Crime Incident Locations (v 3.3). Ned Levine and Associates, Houston, TX, and the National Institute of Justice, Washington, DC.

Y. Li, J. Han, and J. Yang. 2004. Clustering moving objects. In *KDD*. 617–622.

Z. Li, B. Ding, J. Han, and R. Kays. 2010. Swarm: Mining relaxed temporal moving object clusters. *VLDB Endow.* 3, 1–2 (2010), 723–734.

R. Maciejewski, R. Hafen, S. Rudolph, S. Larew, M. W. Cleveland, and D. Ebert. 2011. Forecasting hotspots: A predictive analytics approach. *IEEE Trans. Visual. Comput. Graphics* 17, 4 (2011), 440–453.

S. Mohammadi, V. Janeja, and A. Gangopadhyay. 2009. Discretized spatio-temporal scan window. In *SIAM*. 1195–1206.

P. Mohan, S. Shekhar, J. Shine, and J. Rogers. 2010. Cascading spatio-temporal pattern discovery: A summary of results. In *SDM*. 327–338.

A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. 2009. WhereNext: A location predictor on trajectory pattern mining. In *KDD*. 10–20.

D. Neill, A. Moore, M. Sabhnani, and K. Daniel. 2005. Detection of emerging space-time clusters. In *ACM SIGKDD*. 218–227.

D. Patel, C. Sheng, W. Hsu, and M. Lee. 2012. Incorporating duration information for trajectory classification. In *ICDE*. 1132–1143.

T. Sakaki, M. Okazaki, and Y. Matsuo. 2008. Earthquake shakes twitter users: Real-time event detection by social sensors. In *WWW*. 697–713.

S. Shekhar and Y. Huang. 2001. Discovering spatial co-location patterns: A summary of results. In *SSTD*. 236–256.

M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. 2006. MONIC: Modeling and monitoring cluster transitions. In *KDD*. 706–711.

L. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, W. Peng, and T. Porta. 2014. A framework of traveling companion discovery on trajectory data streams. *ACM Trans. Intell. Syst. Technol.* 5, 1 (Jan. 2014), 3:1–3:34.

M. Vieira, P. Bakalov, V. Tsotras, and J. Vassilis. 2009. On-line discovery of flock patterns in spatio-temporal data. In *GIS*. 286–295.

J. Wang, W. Hsu, M. L. Lee, and J. Wang. 2004. FlowMiner: Finding flow patterns in spatio-temporal databases. In *ICTAI*. 14–21.

R. Watkins, S. Eagleson, S. Beckett, G. Garner, B. Veenendaal, G. Wright, and A. Plant. 2007. Using GIS to create synthetic disease outbreaks. In *BMC Medical Informatics and Decision Making*.

L. Wie and M. Shan. 2006. Efficient mining of spatial co-orientation patterns from image databases. In *SMC*. 2982–2987.

D. Zeng, H. Chen, C. Chavez, W. Lober, and M. Thurmond. 2010. *Infectious Disease Informatics and Biosurveillance*. Springer (2010).

M. Zhang, W. Hsu, and M. L. Lee. 2007. Finding orientation-sensitive patterns in snapshot databases. In *ICTAI*. IEEE, 171–178.

Received April 2013; revised June 2014; accepted June 2014