Knowl Inf Syst (2013) 37:453–483
DOI 10.1007/s10115-013-0636-8

REGULAR PAPER

An incremental algorithm for clustering spatial data streams: exploring temporal locality

Ling-Yin Wei · Wen-Chih Peng

Received: 8 September 2011 / Revised: 17 February 2013 / Accepted: 5 April 2013 /
Published online: 26 April 2013

Abstract Clustering sensor data discovers useful information hidden in sensor networks. In sensor networks, a sensor has two types of attributes: a geographic attribute (i.e, its spatial location) and non-geographic attributes (e.g., sensed readings). Sensor data are periodically collected and viewed as spatial data streams, where a spatial data stream consists of a sequence of data points exhibiting attributes in both the geographic and non-geographic domains. Previous studies have developed a dual clustering problem for spatial data by considering similarity-connected relationships in both geographic and non-geographic domains. However, the clustering processes in stream environments are time-sensitive because of frequently updated sensor data. For sensor data, the readings from one sensor are similar for a period, and the readings refer to temporal locality features. Using the temporal locality features of the sensor data, this study proposes an incremental clustering (IC) algorithm to discover clusters efficiently. The IC algorithm comprises two phases: cluster prediction and cluster refinement. The first phase estimates the probability of two sensors belonging to a cluster from the previous clustering results. According to the estimation, a coarse clustering result is derived. The cluster refinement phase then refines the coarse result. This study evaluates the performance of the IC algorithm using synthetic and real datasets. Experimental results show that the IC algorithm outperforms exiting approaches confirming the scalability of the IC algorithm. In addition, the effect of temporal locality features on the IC algorithm is analyzed and thoroughly examined in the experiments.

Keywords Data mining · Dual clustering · Spatial data streams

L.-Y. Wei · W.-C. Peng (✉)
Department of Computer Science, National Chiao Tung University,
Hsinchu, Taiwan
e-mail: wcpeng@cs.nctu.edu.tw

L.-Y. Wei
Institute of Information Science, Academia Sinica, Taipei, Taiwan

1 Introduction

With the growth of sensor network applications, such as traffic surveillance and weather monitoring, clustering sensor data can reveal valuable insights hidden in the collected data. A sensor generally has two types of attributes: a geographic attribute (i.e., the spatial location of a sensor) and non-geographic attributes (e.g, sensing readings). Solving a traditional clustering problem involves partitioning sensors into clusters according to their readings [21,39]. To explore more informative clusters, researchers have widely investigated constrained clustering problems in recent years [6,13,14,16,17,24,40]. Furthermore, several studies have investigated dual clustering problems with considering constraints in both geographic and non-geographic domains [12,22,25,26,35,37,45]. A reading detected by a sensor is updated regularly, and a series of sensing readings from a sensor can be viewed as a spatial data stream. However, previous studies have focused on clustering sensors without considering data stream environments. Researchers have proposed several clustering algorithms for data stream environments [1,5,7,9,18,19,23,27,30,33,34,44,47], but they only consider non-geographic attributes. The authors of [41,46] investigated the spatial data stream clustering while considering only the geographic attribute. This study addresses a dual clustering problem for spatial data streams in which the attributes of both geographic and non-geographic domains are considered.

Previous research has presented a general dual clustering problem for non-stream environments [25,37] in which the number of clusters is pre-specified. However, for dual clustering problems in spatial data streams, the values in the non-geographic domain vary over time. General dual clustering problems are not suitable for data stream environments because the number of clusters usually changes over time according to the variation of data in the non-geographic domain. Hence, previous research has presented a dual clustering problem in spatial data streams [43]. That study presented a hierarchical-based clustering (HBC) algorithm for a dual clustering problem in spatial data streams without specifying the number of clusters.

The sensors discussed in this paper are fixed and therefore have no mobility. For example, to monitor the traffic status along a freeway, sensors are deployed and utilized to collect readings, such as vehicle speed and volume of traffic. Given a set of sensors with their locations, readings, and geographic and similarity constraints in both domains, the dual clustering problem in [43] clusters sensors into groups based on *similarity-connected relationships*, in which sensors have similar readings in a non-geographic domain under a geographic constraint. For example, considering the sensor data from a time stamp (i.e., with only one reading in the non-geographic domain) in Fig. 1a, the data of each sensor include a two-dimensional coordinate in the geographic domain and a reading in the non-geographic domain. Figure 1b shows the corresponding clustering result from the time stamp: with given geographic and similarity constraints in both domains, the sensors in the same cluster are connected by similar readings under the geographic constraint. Although the readings of the sensors in Cluster 3 and Cluster 4 are similar, these two clusters are not grouped together. This is because the sensors in these two clusters are far away from each other, failing to satisfy the geographic constraint.

Figure 2 shows an example of eight sensors, each of which has a two-dimensional coordinate in the geographic domain to detect a series of speed readings in the non-geographic domain. Given a time window size (e.g., $W = 5$), the readings of the sensors are divided into four non-overlapping time windows (i.e., $w_1,\ w_2,\ w_3$, and w_4). Exploring these clusters yields substantial benefits. First, the readings of the sensors in a cluster have similar speeds, and the clusters reveal traffic status. For example, sensors in a cluster with lower speeds

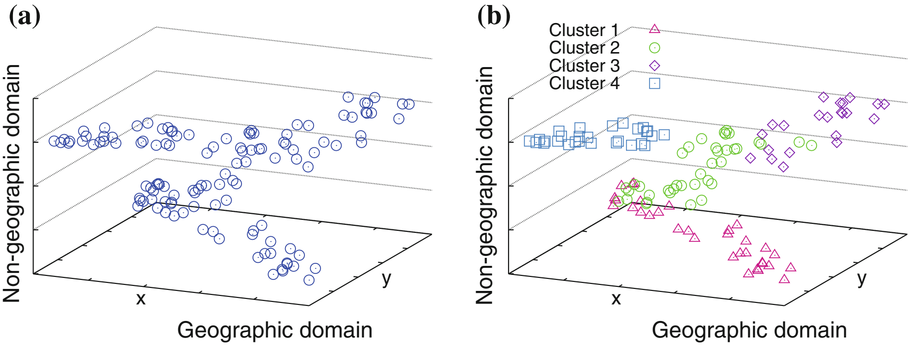


Fig. 1 An illustrative example of the dual clustering problem. **a** Sensor data distribution from a time stamp; **b** a clustering result from time stamp

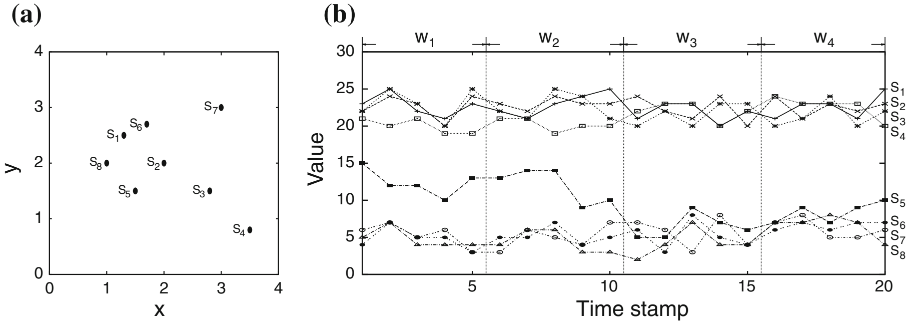


Fig. 2 An illustrative example of the dual clustering problem in spatial data streams. **a** Geographic domain; **b** non-geographic domain

typically indicate traffic jams near the sensors' locations. Second, information derived from clusters facilitates data recovery [29,42]. Because sensors deployed in an outdoor environment can easily malfunction, they may not report their sensing readings if they fail to work. If several sensors are frequently clustered in a group, they are likely to detect similar traffic statuses. Once a particular sensor S fails to work, its missing traffic status can be inferred from the readings of other sensors that are usually clustered with sensor S. Third, the cluster results can be used for outlier detection [2,36]. If a traffic status sensed by sensor S is different from the traffic status sensed by other sensors that are usually clustered with sensor S, the sensor may be reporting an abnormal event or the detected data may be an outlier. As such, utilizing dual clustering in spatial data streams could improve the aforementioned scenarios, justifying the motivation of this paper.

The authors in [43] proposed the HBC algorithm to solve the dual clustering problem in spatial data streams, and the algorithm is performed in each time window. Regarding runtime, the overhead might be high. To deal with dynamic data environments efficiently, researchers have developed incremental techniques for different clustering problems in [3,4,11,15,20,27,28,31,34,37]. The authors in [11] devised an incremental clustering approach to the reconciliation of textual entities, and their method can efficiently de-duplicate volumes of data. Another study [31] proposes an incremental hierarchical co-clustering algorithm for high-dimensional text datasets, whereas the study [4] introduces an incremental algorithm to cluster XML documents sharing similar structures. However, the incremental clustering algorithms of these three studies are developed for textual data or documents and do not cope with the data in dual domains.

On the other hand, researchers have developed some incremental algorithms for clustering data streams in the non-geographic domain or clustering data streams in the geographic domain. For a dynamic environment in which objects are inserted and deleted over time, previous studies [15,28] presented incremental algorithms for density-based clustering problems. The authors in [20] proposed a semi-supervised incremental algorithm for the density-based clustering problem by exploiting available background knowledge. However, the algorithms proposed in [15,20,28] only cope with geographic data without considering non-geographic attributes. Given time series data, the authors in [27] developed an incremental technique for traditional k-means clustering algorithms and EM clustering algorithms. Another study [34] presents an incremental system to discover clusters where time series in the same cluster behave similarly. The authors of [3] introduced an incremental fuzzy clustering for bank customers' transactions. However, these incremental clustering methods only deal with the data in the non-geographic domain without considering geographic attributes. Although the incremental algorithm in [37] is designed for a general dual clustering problem, it only deals with the data in non-stream environments. These incremental techniques were developed for clustering problems that are different from dual clustering in spatial data streams in this paper.

In the real world, the values of a sensor in the non-geographic domain usually have temporal locality features, meaning that the sensors' readings are similar for a period. For example, Fig. 2b shows that the readings of sensor S_4 in time window w_1 are similar to those in time window w_2. Therefore, based on the temporal locality feature, this study proposes an incremental clustering (IC) algorithm that clusters objects roughly by inducing clustering results from previous time windows. To explore the temporal locality features, this study proposes a probability matrix in which the values indicate the probability that two sensors will belong to the same cluster. Sensors are clustered roughly according to the estimation, and the coarse clustering results are refined to satisfy the required constraints. This study presents numerous experiments conducted on both a real dataset and a synthetic dataset. To evaluate the effects of temporal locality features on the proposed algorithm, this study proposes a simulator framework to generate synthetic datasets by effectively simulating real world data. Using the proposed simulation, synthetic datasets can be generated by controlling the degree of temporal locality. To use the synthetic dataset generated by the proposed simulation effectively, this study adopts a statistical approach to estimate accurately the constraint in the non-geographic domain for dual clustering problems in spatial data streams under an user-specified tolerance. Experimental results confirm the effectiveness of this approach. This study also proposes an approach to estimate the degree of temporal locality of sensor data and demonstrate its effectiveness by assessing the results of the experiments. Based on these results, it is possible to evaluate the performance of the proposed algorithm and compare its efficiency to that of existing algorithms using synthetic and real datasets. Experimental results show that the proposed algorithm outperforms exiting approaches, revealing the scalability of the algorithm. This study also analyzes the effects of temporal locality features on the algorithm. Based on the experimental results, this study presents guidelines for setting parameters in the proposed algorithm.

The remainder of this paper is organized as follows. Section 2 formally defines the dual clustering problem in spatial data streams. Section 3 presents the IC algorithm and analyze its complexity. Section 4 presents the performance of the IC algorithm. Section 5 concludes the paper.

Table 1 Non-geographic and geographic attributes for objects in Fig. 2

ID	Location	Data points	Time
S_1	(1.3,2.5)	(23,25,22,21,23,22,21,23,24,25,21,23,23,20,22,21,23,23,21,25)	[1, 20]
S_2	(2.0,2.0)	(22,24,23,20,24,23,22,24,23,23,24,22,21,24,20,24,21,23,22,23)	[1, 20]
S_3	(2.8,1.5)	(22,25,23,20,25,22,21,25,24,21,20,22,20,23,23,20,21,24,20,22)	[1, 20]
S_4	(3.5,0.8)	(21,20,21,19,19,21,21,19,20,20,22,23,23,20,22,24,23,23,23,20)	[1, 20]
S_5	(1.5,1.5)	(15,12,12,10,13,13,14,14,9,10,5,5,9,7,6,7,9,7,9,10)	[1, 20]
S_6	(1.7,2.7)	(6,7,5,6,3,3,6,5,4,7,7,6,3,8,4,7,8,5,5,6)	[1, 20]
S_7	(3.0,3.0)	(4,7,5,5,3,5,5,7,4,5,6,3,8,5,4,6,7,6,7,7)	[1, 20]
S_8	(1.0,2.0)	(5,7,4,4,4,4,6,6,3,3,2,4,7,4,4,7,7,8,7,4)	[1, 20]

2 Preliminaries

This section presents the notations and formulates the dual clustering problem in spatial data streams. An object consists of non-geographic attributes and a geographic attribute and is denoted as S_i. Throughout the rest of this study, an *object* refers to a sensor. Given a particular time interval, the values of non-geographic attributes of object S_i are represented as a vector $S_i.V_t$, where $S_i.V_t$ is the data point at time t in the time interval. In addition, the location of S_i is denoted as $S_i.L_x$, $S_i.L_y$, which represents the object's position in a two-dimensional space. Without loss of generality, an object's spatial position can be generalized to a high-dimensional space. In this paper, the locations of objects are fixed. Table 1 shows the values of the attributes in the geographic and the non-geographic domains based on the example in Fig. 2.

To describe the constraints of the dual clustering problem, this study first defines the dissimilarity between two objects in the non-geographic domain, and the physical distance between two objects in the geographic domain.

Definition 2.1 (*Dissimilarity in the non-geographic domain*) Given a time window $w = [t+1, t+W]$ and two objects S_i and S_j, the dissimilarity between S_i and S_j in time window w is defined as

$$diss(S_i, S_j, w) = \sqrt{\frac{1}{W} \cdot \sum_{k=1}^{W} (S_i.V_{t+k} - S_j.V_{t+k})^2}.$$

Definition 2.2 (*Physical distance in the geographic domain*) Given two objects S_i and S_j, the physical distance between S_i and S_j is defined as

$$ED(S_i, S_j) = \sqrt{(S_i.L_x - S_j.L_x)^2 + (S_i.L_y - S_j.L_y)^2}.$$

This study employs the most common dissimilarity measure (i.e., the average Euclidean distance) to focus on the dual clustering problem in spatial data streams. Depending on an application's requirements, other dissimilarity measures can be used. Based on these definitions, this study presents a concept called the *directly similarity-connected relationship* (i.e., directly SC-relationship), which indicates both the similarity and the connectivity between two objects.

Definition 2.3 (*Directly similarity-connected relationship*) Given a geographic constraint R and a similarity constraint ε, two objects S_i and S_j have a directly SC-relationship in time window w if $diss(S_i, S_j, w) \leq \varepsilon$ and $ED(S_i, S_j) \leq R$.

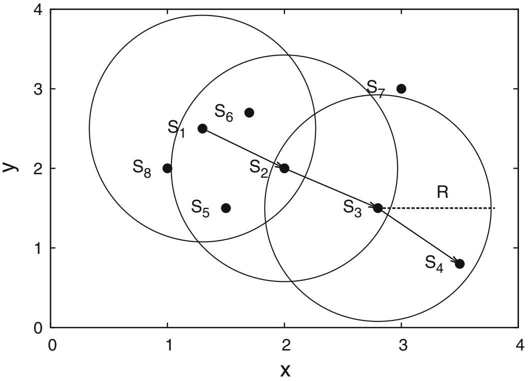


Fig. 3 A similarity-connected relationship between S_1 and S_4

Clearly, if two objects have a directly SC-relationship, they should have similar non-geographic attributes and the physical distance between them should satisfy the given geographic constraint. In the example of sensor networks used to monitor freeway traffic, sensors in proximity that have similar readings are likely to have a directly SC-relationship. Thus, based on the directly SC-relationship, this study defines a *similarity-connected relationship* (abbreviated as SC-relationship) as follows.

Definition 2.4 (*Similarity-connected relationship*) Given a geographic constraint R and a similarity constraint ε, two objects S_i and S_j have a similarity-connected relationship in time window w if there exists a chain of objects $S_i = S_{l_1}, S_{l_2}, \ldots, S_{l_{q-1}}, S_{l_q} = S_j$ such that, in time window w, the following conditions are satisfied: (1) for $1 \leq k, h \leq q$, $diss(S_{l_k}, S_{l_h}, w) \leq \varepsilon$; and (2) for $1 \leq k < q$, S_{l_k} and $S_{l_{k+1}}$ have a directly SC-relationship.

Given the objects in Table 1, geographic constraint $R = 1$, similarity constraint $\varepsilon = 10$, and time window size $W = 5$, the S_1 and S_4 in Fig. 3 have an SC-relationship in time window $w_1 = [1, 5]$, because their dissimilarity is within the given similarity constraint and a chain (i.e., S_1, S_2, S_3, S_4) satisfies the SC-relationship. Although the physical distance between S_1 and S_4 is larger than the geographic constraint R, S_1 and S_4 are similarity-connected. This reveals that two objects might have an SC-relationship even if the physical distance between them does not satisfy the geographic constraint. In this example, the clusters in w_1 are $\{S_1, S_2, S_3, S_4\}$, $\{S_5\}$, $\{S_6, S_8\}$, and $\{S_7\}$.

Based on the SC-relationship, the dual clustering problem for spatial data streams is formulated as follows.

A dual clustering problem in spatial data streams: Given a time window size W, a similarity constraint ε, and a geographic constraint R, the goal is to cluster the objects into several groups in each time window such that objects in the same group should have SC-relationships. Note that a series of values of an object's non-geographic attributes are partitioned into consecutive non-overlapping time windows, and the objects in each time window are clustered. A cluster in which the objects have SC-relationships is called an *SC-cluster*.

3 Dual clustering for spatial data streams

This section presents a graph structure to capture similarity relationships among objects. After exploring the temporal locality of attributes in the non-geographic domain, this study proposes an incremental clustering (IC) algorithm to improve the efficiency of dual clustering

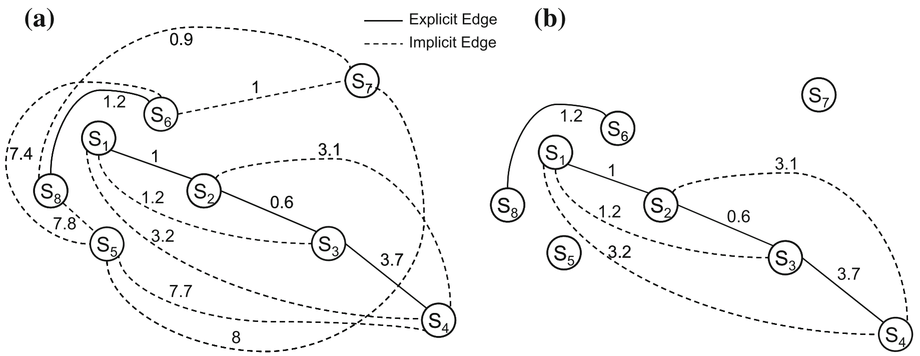


Fig. 4 An illustrative example of a graph representation. **a** Graph representation; **b** clustering result

in spatial data streams. This section also presents the derivation of the time complexity of the IC algorithm.

3.1 Graph representation

Given a set of objects, Similarity-Connected Graph (abbreviated as SC-graph) is utilized to capture the directly SC-relationships and SC-relationships among objects in each time window. Each vertex in an SC-graph represents an object with two types of edges: explicit and implicit edges. If two objects, S_i and S_j, have a directly SC-relationship, an explicit edge, denoted as $e_e(S_i, S_j)$, exists between them. Conversely, if S_i and S_j have similar non-geographic attributes but the physical distance between them is larger than the geographic constraint (i.e., $ED(S_i, S_j) > R$), an implicit edge, denoted as $e_i(S_i, S_j)$ exists between the objects. For example, given the objects' attributes in Table 1, if $R = 1$, $\varepsilon = 10$, and $W = 5$, the SC-graph in the first time window (i.e., w_1) is shown in Fig. 4a in which the solid line and the dotted line represent an explicit edge and an implicit edge, respectively. The edge between S_1 and S_2 is an explicit edge, meaning that S_1 and S_2 have a directly SC-relationship, because $diss(S_1, S_2, w_1) \approx 1 < \varepsilon$ and $ED(S_1, S_2) \approx 0.86 < R$. Conversely, the edge between S_1 and S_3 is an implicit edge because $ED(S_1, S_3) \approx 1.80 > R$ though $diss(S_1, S_3, w_1) \approx 1.2 < \varepsilon$. The weight of an edge represents the dissimilarity between two objects in the non-geographic domain.

According to an SC-graph, a clustering result consists of subgraphs that must satisfy two requirements: 1) the vertices of a subgraph must be connected through explicit edges; and 2) the subgraph must be complete through both explicit and implicit edges. Note that two vertices connected by an implicit edge do not imply an SC-relationship. However, when the two vertices are connected by a simple path comprised of explicit edges, they have an SC-relationship. Thus, the first requirement is that each subgraph must guarantee the connectivity of vertices through explicit edges. For example, Fig. 4b shows the clustering result for the graph in Fig. 4a. In Fig. 4b, the clustering result includes four subgraphs, $\{S_1, S_2, S_3, S_4\}$, $\{S_5\}$, $\{S_6, S_8\}$, and $\{S_7\}$, all of which fulfill the aforementioned two requirements. Note that a subgraph refers to an SC-cluster in this paper.

3.2 Incremental clustering algorithm

Because of the feature of streams (i.e., dynamic and rapid generation of data records), the time of clustering procedures should be as short as possible in stream environments. A previous

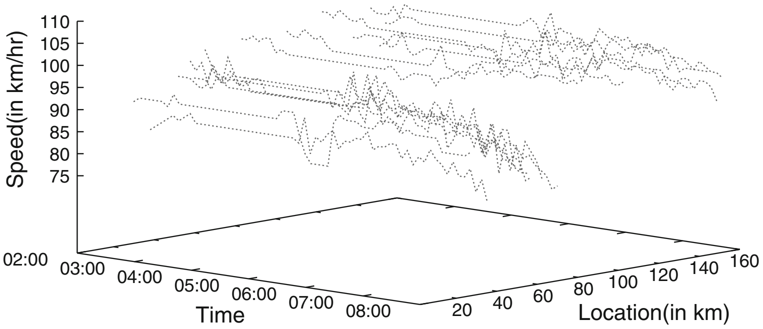


Fig. 5 The temporal locality feature in the real dataset

study [43] proposed a hierarchical-based algorithm (HBC) for dual clustering in spatial data streams. That study executed the HBC algorithm in each time window. Appendix A presents the HBC algorithm. This study proposes the IC algorithm, an incremental algorithm, to efficiently derive cluster results. To reiterate, the non-geographic attributes of objects usually have *temporal locality* features, meaning that their values are similar for a period. For example, Fig. 5 shows the speed readings of the sensors along a freeway [38], where the X-axis represents the time, the Y-axis represents sensors' locations deployed linearly, and the Z-axis represents sensors' speed readings. The speeds of a sensor are similar within a period, demonstrating the temporal locality features of the non-geographic attributes of objects. Consequently, the clustering results of adjacent time windows are similar. The derived clustering results reveal which objects are frequently clustered together. As Fig. 2 shows, a cluster $\{S_6, S_8\}$ is discovered in time windows w_1 and w_2, and a cluster $\{S_5, S_6, S_8\}$ is discovered in time windows w_3 and w_4. This phenomenon indicates that S_6 and S_8 are more likely to be in the same clusters with the given objects S_5, S_6, and S_8. Therefore, based on the temporal locality features, the IC algorithm uses the clustering results in the previous time windows to improve the efficiency of the clustering procedures.

Algorithm 1: Incremental Clustering (IC) Algorithm

```
input  : A set of objects $STD$, a similarity constraint $\varepsilon$, a geographic constraint $R$, and a window size
         $W$, a correlation factor $\alpha$, a *probability threshold* $\theta$, and the time interval $[t_s, t_e]$
output : A set of SC-clusters $C_{w_k}$ with respect to time window $w_k$
Do algorithm $HBC$ in $w_1$ and then generate $R_{w_1}$ and $P_{w_1}$;
for each time window $w_k = [t_s + k \cdot W, t_s + (k+1) \cdot W]$ where $1 \leq k \leq \lfloor \frac{t_e - t_s}{W} \rfloor$ do
    Generate coarse clusters $C_{w_k}$ by $P_{w_{k-1}}$ and $\theta$;
    Split each cluster of $C_{w_k}$ into SC-clusters if it is not an SC-cluster;
    Using the neighbor list to merge SC-clusters until the number of SC-clusters do not decrease;
    Generate $R_{w_k}$ by $C_{w_k}$;
    Generate $P_{w_{k+1}}$ by $P_{w_{k+1}} = (1-\alpha)P_{w_k} + \alpha R_{w_k}$;
end
return $R_{w_k}$;
```

The IC algorithm has two main phases: cluster prediction and cluster refinement. Specifically, this IC algorithm predicts which objects to cluster together using the previous clustering results and then refines the coarse clusters to derive SC-clusters by verifying the relationships among objects. To predict the clustering results of the subsequent time windows, two matrices record the clustering results and estimate the probability of being clustered together, respectively.

Let N denote the number of objects. For time window w_h, the clustering result of objects is recorded using an upper triangular matrix R_{w_h}, where R_{w_h} is an $N \times N$ matrix. The value of each element in the matrix is as follows:

$$r_{i,j,h} = \begin{cases} 1 & \text{if } i \leq j, \text{ and objects } S_i \text{ and } S_j \text{ are in the same cluster} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

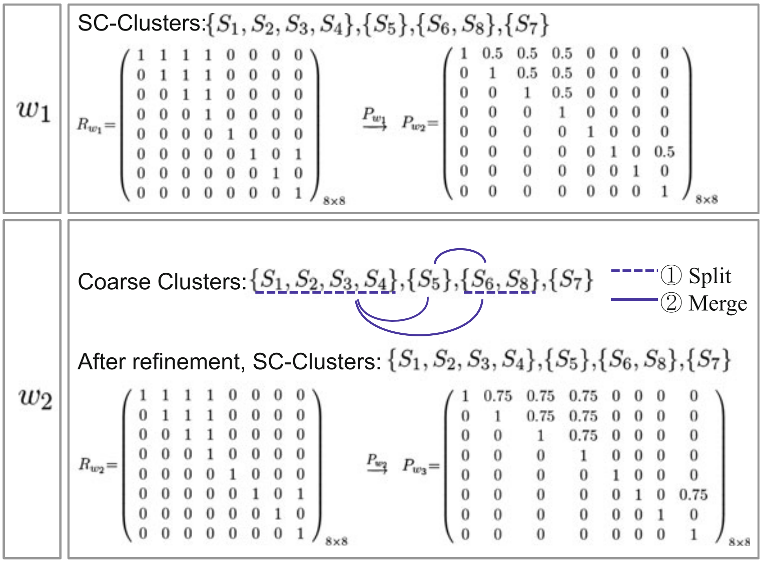
Because the non-geographic attributes of objects have temporal locality features, the objects that are clustered in previous time windows are likely to be in the same cluster in subsequent time windows. Therefore, based on the previous clustering results, a probability matrix can be used to predict future clustering results. The probability matrix for time window w_h is expressed by an upper triangular matrix P_{w_h}, where P_{w_h} is an $N \times N$ matrix. For time window w_h, an element $p_{i,j,h}$ in P_{w_h} represents the probability that S_i and S_j will be in the same cluster. The element $p_{i,j,h}$ can be estimated from the previous clustering results. The probability matrix P_{w_h} is defined as $P_{w_h} = (1 - \alpha)P_{w_{h-1}} + \alpha R_{w_{h-1}}$, where α represents a *temporal correlation* and $0 \leq \alpha \leq 1$. Thus, the values of elements in P_{w_h} can be determined as follows:

$$p_{i,j,h} = \begin{cases} (1 - \alpha)p_{i,j,h-1} + \alpha r_{i,j,h-1} & \text{if } i < j \\ 1 & \text{if } i = j \\ 0 & \text{if } i > j. \end{cases} \tag{2}$$

Initially, in P_{w_1}, $p_{i,j,1} = 1$ for $i = j$; otherwise, $p_{i,j,1} = 0$. The value of α determines the weights of the clustering results in the most recent time window.

After defining the two matrices, this study presents the IC algorithm in detail. Initially, this algorithm applies the HBC algorithm to derive SC-clusters in the first time window. The probability matrix in time window w_2 can be generated by R_{w_1} and P_{w_1}. As mentioned previously, in a subsequent time window w_h, the IC algorithm coarsely clusters objects based on the probability matrix P_{w_h} and then refines the coarse clusters to derive SC-clusters. In the matrix P_{w_h}, if $p_{i,j,h}$ is large, the corresponding objects (for example, S_i and S_j) are likely to have similar values in the non-geographic domain. Thus, if $p_{i,j,h}$ is larger than a given probability threshold θ, the objects form a cluster. These clusters are *coarse clusters* because the SC-relationships among objects in the same coarse cluster must be verified. As in hierarchical clustering techniques, if the probability value is greater than θ, iteratively merge the two objects with the maximal probability value. This procedure clusters objects in a bottom-up manner. The merging order is recorded and will be used to refine coarse clusters later.

To refine coarse clusters, the IC algorithm verifies whether the objects in each cluster have SC-relationships by assessing the non-geographic attributes of each pair of objects. It is only necessary to compute the similarity of the objects' non-geographic attributes because the location of each object does not change over time. The order for computing the similarity is the same as that used to derive coarse clusters, indicating the similarity degree of objects in the probability matrix. A higher value in the probability matrix means that corresponding objects are likely to have similar values in the non-geographic domain. Thus, the similarity of the non-geographic attributes of objects can be derived by following the order used to derive coarse clusters. An SC-graph is built for each coarse cluster in a similar manner. After computing the similarity between objects, it is easy to identify the corresponding edge type. Recall that SC-clusters must satisfy two requirements: 1) the vertices of a subgraph must be connected through explicit edges; and 2) a subgraph must be complete through explicit


w_1
SC-Clusters:$\{S_1, S_2, S_3, S_4\}$,$\{S_5\}$,$\{S_6, S_8\}$,$\{S_7\}$
w_2
Coarse Clusters:$\{S_1, S_2, S_3, S_4\}$,$\{S_5\}$,$\{S_6, S_8\}$,$\{S_7\}$
① Split
② Merge
After refinement, SC-Clusters: $\{S_1, S_2, S_3, S_4\}$,$\{S_5\}$,$\{S_6, S_8\}$,$\{S_7\}$

$$R_{w_1}=\begin{pmatrix} 1&1&1&1&0&0&0&0\\ 0&1&1&1&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}_{8\times 8} \xrightarrow{P_{w_1}} P_{w_2}=\begin{pmatrix} 1&0.5&0.5&0.5&0&0&0&0\\ 0&1&0.5&0.5&0&0&0&0\\ 0&0&1&0.5&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0.5\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}_{8\times 8}$$

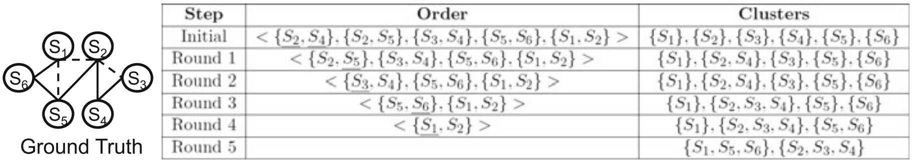
$$R_{w_2}=\begin{pmatrix} 1&1&1&1&0&0&0&0\\ 0&1&1&1&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}_{8\times 8} \xrightarrow{P_{w_2}} P_{w_3}=\begin{pmatrix} 1&0.75&0.75&0.75&0&0&0&0\\ 0&1&0.75&0.75&0&0&0&0\\ 0&0&1&0.75&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0.75\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}_{8\times 8}$$

Fig. 6 A running example of the IC algorithm from w_1 to w_2 in Fig. 2

and implicit edges. These two criteria are used to derive SC-clusters when computing the similarity of objects.

Given the data in Fig. 2, $W = 5$, $\alpha = 0.5$ and $\theta = 0.5$, Fig. 6 shows a running example of the IC algorithm. This algorithm applies the HBC algorithm to generate SC-clusters in the first time window. According to the clustering result in time window w_1, the matrix R_{w_1} is derived, and P_{w_2} is then calculated using Formula 2. In time window w_2, predict clusters by P_{w_2} and refine the coarse clusters to derive SC-clusters using a split-and-merge process. For example, in the cluster prediction phase, S_6 and S_8 form a coarse cluster because $p_{6,8,2} = 0.5 \geq \theta$. After generating coarse clusters, verify the SC-relationships among objects in each coarse cluster and split unqualified clusters into several SC-clusters. This example only assesses clusters $\{S_1, S_2, S_3, S_4\}$ and $\{S_6, S_8\}$ for the split steps, because other clusters only have one member. After splitting unqualified coarse clusters, perform bottom-up merging to derive a final clustering result. As Fig. 6 shows, except for cluster $\{S_7\}$, any two clusters are possibly merged in time window w_2. This is because the object S_7 is far away from other objects without satisfying the geographic constraint; therefore, the cluster $\{S_7\}$ does not form an SC-cluster with other objects. Because the geographic distances between any two objects were calculated in the first time window, cluster refinement in the following time windows can use the geographic information without an additional computation. The SC-clusters in time window w_2 are generated after merging clusters. Next, generate R_{w_2} and P_{w_3}. Similarly, the IC algorithm performs cluster prediction and cluster refinement in the remaining time windows (Fig. 6).

Regarding an example of refining a coarse cluster in detail, assume that a coarse cluster in time window w_h has a set of objects $\{S_1, S_2, S_3, S_4, S_5, S_6\}$ and the order of deriving the coarse cluster is $< \{S_2, S_4\}, \{S_2, S_5\}, \{S_3, S_4\}, \{S_5, S_6\}, \{S_1, S_2\} >$. Based on this order, calculate the similarity of the objects' non-geographic attributes. Figure 7 lists the rounds necessary for computing the similarity among objects in the same coarse cluster and shows the corresponding ground truth of the SC-graph. In Fig. 7, each object is initially viewed



Step	Order	Clusters
Initial	$<\{\underline{S_2}, \underline{S_4}\}, \{S_2, S_5\}, \{S_3, S_4\}, \{S_5, S_6\}, \{S_1, S_2\}>$	$\{S_1\}, \{S_2\}, \{S_3\}, \{S_4\}, \{S_5\}, \{S_6\}$
Round 1	$<\{S_2, \underline{S_5}\}, \{S_3, S_4\}, \{S_5, S_6\}, \{S_1, S_2\}>$	$\{S_1\}, \{S_2, S_4\}, \{S_3\}, \{S_5\}, \{S_6\}$
Round 2	$<\{\underline{S_3}, S_4\}, \{S_5, S_6\}, \{S_1, S_2\}>$	$\{S_1\}, \{S_2, S_4\}, \{S_3\}, \{S_5\}, \{S_6\}$
Round 3	$<\{S_5, \underline{S_6}\}, \{S_1, S_2\}>$	$\{S_1\}, \{S_2, S_3, S_4\}, \{S_5\}, \{S_6\}$
Round 4	$<\{\underline{S_1}, S_2\}>$	$\{S_1\}, \{S_2, S_3, S_4\}, \{S_5, S_6\}$
Round 5		$\{S_1, S_5, S_6\}, \{S_2, S_3, S_4\}$

Fig. 7 An example of refining a coarse cluster

as an individual cluster before calculating the similarity between object S_2 and object S_4 (that is, $diss(S_2, S_4, w_h)$). Based on the ground truth, an explicit edge (that is, $e_e(S_2, S_4)$) is generated and the SC-cluster $\{S_2, S_4\}$ is derived. Next, compute the similarity of objects S_2 and S_5. Although the explicit edge $e_e(S_2, S_5)$ can be identified, the similarity between S_5 and S_4 should still be verified. However, as shown by the ground truth, no edge exists between S_5 and S_4. Thus, S_5 cannot be merged with the SC-cluster that contains S_2 and S_4. In Round 3, choose and mark object S_3 and assess the relationships between objects S_3 and S_4. Because there is an edge $e_e(S_2, S_4)$ and the SC-cluster satisfies the aforementioned requirements, $\{S_2, S_3, S_4\}$ is an SC-cluster. This procedure generates SC-clusters similar to the example in Fig. 7, where a coarse cluster is split into two SC-clusters, $\{S_1, S_5, S_6\}$ and $\{S_2, S_3, S_4\}$.

After splitting unqualified coarse clusters, the IC algorithm performs merging processes to derive final clustering results by assessing the SC-relationships among objects in different clusters. The physical distances between objects were calculated in the first time window, and each object has a list of objects whose physical distances from it are less than the given geographic constraint. Therefore, each object in an SC-cluster can identify any explicit edges between itself and objects in different SC-clusters. Note that if two SC-clusters are split from the same coarse cluster, it is not necessary to assess them for this type of merging.

3.3 Analysis of the IC algorithm

This section first proves that the value of a probability matrix is between 0 and 1 in Theorem 3.1 and then derives the time complexity of the IC algorithm. This section also derives the time complexity of the HBC algorithm for comparison purposes.

Theorem 3.1 *The value of every element of a probability matrix P_{w_h} is between* 0 *and* 1.

Proof If $h = 1$, it is trivial to show that the value of every element of the probability matrix is between 0 and 1. For $h > 1$, divide elements in the probability matrix into three parts and prove that the value of every element in each case is between 0 and 1, respectively. Case 1: for each $i > j$, the value of the element $p_{i,j,h}$ of the probability matrix P_{w_h} is 0 by Formula 2. Case 2: for each $i = j$, the value of the element $p_{i,j,h}$ of the probability matrix P_{w_h} is 1 by Formula 2. Case 3: for each $i < j$, Formula 2 shows that the value of the element in the probability matrix P_{w_h} is $p_{i,j,h} = (1-\alpha)p_{i,j,h-1} + \alpha r_{i,j,h-1}$ for $i < j$. It is then possible to derive that $p_{i,j,h} = \alpha((1-\alpha)^{h-2}r_{i,j,1} + (1-\alpha)^{h-3}r_{i,j,2} + \cdots + r_{i,j,h-1}) \leq 1-(1-\alpha)^{h-1}$ because $0 \leq r_{i,j,k} \leq 1$ for each $1 \leq k \leq h-1$ by Formula 1. Because $0 \leq \alpha \leq 1$, $p_{i,j,h} \leq 1$. By contrast, $p_{i,j,h} \geq 0$ by Formula 2. Hence, the value of every element of a probability matrix is between 0 and 1. □

We next analyze the time complexity of the IC algorithm and that of the HBC algorithm. The time complexity of the HBC algorithm is $O(N^2 T_{ED} + F(N^2 W T_{diss} + |E| log |E| + |E| c_{max}^2))$, where F is the number of time windows; N is the number of objects; W is the window size; T_{diss} represents the time required to compute the dissimilarity between two objects; T_{ED} represents the time required to compute the physical distance between two objects; c_{max} denotes the number of objects in the cluster that has the maximal number of objects; and $|E|$ is the number of explicit edges among N objects. The cost of computing the dissimilarity and the physical distance between any two objects is $O(N^2(WT_{diss} + T_{ED}))$, and the cost of sorting the explicit edges in Q according to their dissimilarity values is $O(|E|log|E|)$. Therefore, the cost of merging clusters is $O(|E|c_{max}^2)$.

The IC algorithm derives SC-clusters by refining each coarse cluster, and R_{w_h} is updated. The IC algorithm utilizes the probability matrix to derive coarse clusters; therefore, the cost of refining each cluster is lower because the number of objects in coarse clusters is bounded.

The best case of the IC algorithm involves the predicted M coarse clusters fully satisfying the requirements of SC-clusters in every time window after the first window. Thus, the time complexity of the IC algorithm is $O(T_{HBC}+(F-1)(N^2+(Mc_{max}^2+M^2+M\tau+\tau^2)WT_{diss}))$, where T_{HBC} is the execution time of one round of the HBC algorithm; F is the number of time windows; M is the number of predicted coarse clusters; c_{max} represents the number of objects in the cluster that has the maximal number of objects; τ is the number of objects that do not belong to any predicted coarse cluster (τ approximates 0); W is the window size; and T_{diss} represents the time required to compute the dissimilarity of two objects. The cost of generating coarse clusters and computing matrices R and P is $O(N^2)$, and the cost of computing the dissimilarity values and assessing the SC-relationships between objects is $O((Mc_{max}^2 + M^2 + M\tau + \tau^2)WT_{diss})$.

This study presents a comparison of the complexity of the HBC algorithm with that of the IC algorithm after the first time window. For the HBC algorithm, the complexity in the time window w_h, where $h \geq 2$, is $O(N^2WT_{diss} + |E|log|E| + |E|c_{max}^2)$. For the IC algorithm, the complexity of the best case in the time window w_h, where $h \geq 2$, is $O(N^2 + (Mc_{max}^2 + M^2 + M\tau + \tau^2)WT_{diss})$. If all objects belong to coarse clusters, then $\tau \approx 0$. To prove that the time complexity of the IC algorithm is less than that of the HBC algorithm, we prove that $O(N^2WT_{diss}) > O((Mc_{max}^2 + M^2 + M\tau + \tau^2)WT_{diss})$. Because $\tau \approx 0$, $W > 0$ and $T_{diss} > 0$, we show that $O(N^2) > O(Mc_{max}^2 + M^2)$. Let $N = \sum_{i=1}^{M} c_i$ where c_i is the number of objects in the i-th cluster of M clusters and $c_i \leq c_{i+1}$ for each $1 \leq i < M$. The detailed time complexity of $O(Mc_{max})$ is $\sum_{i=1}^{M} c_i^2$ and the detailed time complexity of $O(M^2)$ is C_2^M. Then, if $M > 1$, $\sum_{i=1}^{M} c_i^2 + C_2^M < \sum_{i=1}^{M} c_i^2 + C_2^M c_1^2 < (c_1^2 + \cdots + c_M^2) + 2\sum_{i \neq j, 1 \leq i,j \leq M} c_i c_j = (c_1 + \cdots + c_M)^2 = N^2$. Thus, we derive that $O(N^2WT_{diss} + |E|log|E| + |E|c_{max}^2) > O(N^2 + (Mc_{max}^2 + M^2 + M\tau + \tau^2)WT_{diss}))$ if $M > 1$ and $\tau \approx 0$. Consequently, when $M > 1$ and $\tau \approx 0$, the IC algorithm is more efficient than the HBC algorithm in the best case.

4 Performance study and analysis

Because the IC algorithm is based on temporal locality, a framework is designed to generate a synthetic dataset from the observations of real dataset characteristics in Sect. 4.1 and then the effect of temporal locality on performance can be analyzed. Extensive experiments were conducted using the synthetic datasets generated by different temporal localities. Based on experiments using synthetic datasets, one could have some guidelines to set the parameters of the IC algorithm for experiments in a real dataset. These experiments compared the IC algorithm with existing approaches, the HBC algorithm [43], the CLS algorithm [25], and the CTS-ARMA algorithm [5]. Sections 4.2 and 4.3 present an analysis of the scalability of the algorithms using both synthetic and real datasets, respectively. Table 2 shows the notations

Table 2 The notations used in the experiments

Symbol	Description
W	Time window size
R	Geographic constraint
ε	Similarity constraint
α	Temporal correlation
θ	Probability threshold
N	Number of objects
L	Number of time stamps
F	Number of time windows
TL	Degree of temporal locality
a	Positive constant
Δt	Time gap

used in the experiments. All experiments were performed on a computer with a 2.80 GHz Intel CPU and 2 GB of memory.

Several previous studies [5,18,34] have presented different algorithms for clustering data streams. However, the method in [18] focuses on the k-median problem in stream environments. Although other methods in [5,34] can discover clusters where the data streams in the same cluster behave similarly, the non-geographic values of data streams from the same cluster are often quite different, and the clustering results do not meet the clustering criteria in this paper. Therefore, the approach in [5] was modified to derive SC-clusters from spatial data streams, creating the CTS-ARMA algorithm. At the beginning of the CTS-ARMA algorithm, the number of clusters should be specified, and we set it as the number of clusters derived in the IC algorithm. For the ARMA(p, q) model used in the CTS-ARMA algorithm, the experiments in this study chose parameters $p = 1$ and $q = 1$. On the other hand, for existing dual clustering algorithms, only the HBC and CLS algorithms were considered as competitors. This is because the performance of the incremental clustering algorithm in [37], which was developed for clustering data in dual domains, is worse than that of the CLS algorithm. In addition, the two traditional clustering algorithms (i.e., the k-means algorithm and the Jarvis-Patrick algorithm) modified in [25] also have worse performance than the CLS algorithm. As mentioned previously, the CLS algorithm was designed to solve general dual clustering problems from a single time stamp, and its constraints are different from the proposed algorithm. That is, the CLS algorithm requires users to pre-specify the number of clusters and has no similarity constraint for clusters. Therefore, the CLS algorithm was modified to be suitable for discovering SC-clusters from spatial data streams given a particular number of clusters. Similarly, for the CLS algorithm, the number of clusters was set as the number of clusters derived in the IC algorithm.

For this framework, the degree of temporal locality TL varies between 0 and 1, and synthetic datasets can be generated for different degrees of temporal locality. For the clustering problem in this paper, the window size and two constraints R and ε can be specified according to different application needs and domain knowledge. To use synthetic datasets for performance evaluation, Sect. 4.1.2 presents a guideline for setting similarity constraints and Sect. 4.2.3 shows how to evaluate their effectiveness. Sections 4.2.1 and 4.2.2 use synthetic datasets to investigate the effect of temporal locality on the efficiency and the scalability of algorithms. Section 4.1.3 presents an approach to estimate the temporal locality of spatial data streams and evaluate their effectiveness using synthetic datasets. If temporal locality of

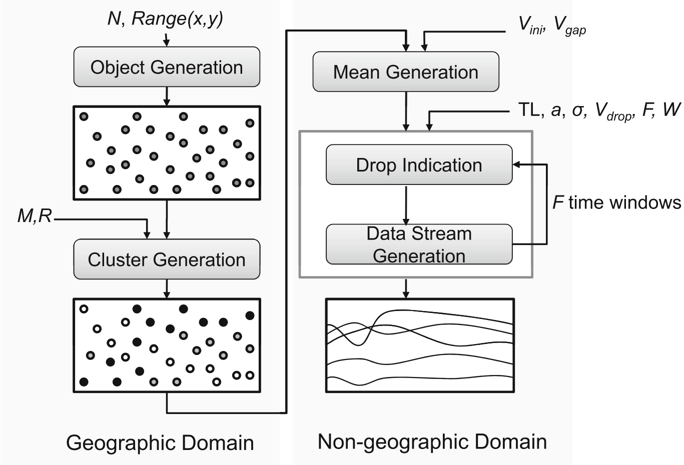


Fig. 8 The framework of the proposed simulation

spatial data streams can be estimated, it is possible to determine the parameters of the IC algorithm. Explicitly, to use the IC algorithm, temporal correlation α and probability threshold θ should be specified. Because these two parameters are related to temporal locality, Sect. 4.2.4 investigates the influence among the two parameters and temporal locality and provides guidelines for setting these two parameters for the IC algorithm.

4.1 Synthetic dataset generation and analysis

4.1.1 Framework for synthetic data generation

To simulate real data, this study presents a framework to generate synthetic datasets by controlling the features of objects. As Fig. 8 shows, this framework comprises two stages. The first stage generates the objects' geographic information, and the second stage generates the values of the objects' non-geographic attributes.

The left-hand side of Fig. 8 shows the flow of generating geographic information of objects. Assume that the number of objects is N and the objects are deployed in a specified range, $Range(x, y)$. The locations of objects are then uniformly distributed over $Range(x, y)$. Given a geographic constraint R and the desired number of clusters M, must determine the number of objects in each cluster. To generate a cluster, begin by selecting an object as the seed of the cluster. Note that each object has at least one neighbor from the same cluster in the range R. In other words, for each object S_i in a cluster C, an object $S_j \in C$ exists such that $ED(S_i, S_j) \leq R$. Therefore, use the location of the seed as the center and the geographic constraint R as the radius. For each cluster, choose objects within the radius R of the seed of the cluster at random as the members of the cluster. Next, choose the furthest object in the range, with respect to the current center, as the new center. Repeat these steps until the locations of the selected objects meet the boundary of the given geographic range $Range(x, y)$ and generate other clusters accordingly. Finally, mark the objects selected as cluster members and regard the unmarked objects as noisy objects.

This study uses a real dataset to generate the values of the objects' non-geographic attributes (that is, sensor data collected by monitoring traffic speeds on a freeway).

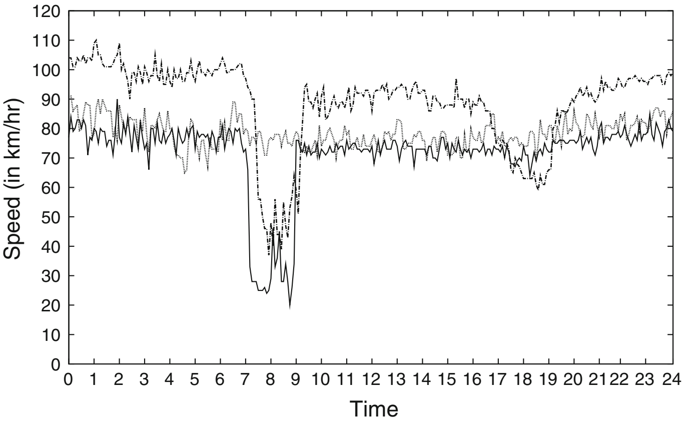


Fig. 9 Real freeway traffic data on a particular day

As Fig. 9 shows, most drivers drive close to the speed limit because of speed limits on the freeway. During peak periods, such as rush hour, driving speeds are drastically reduced. Traffic accidents also affect driving. Figure 9 shows that from approximately 7 a.m. to 9 a.m. (and from 5 p.m. to 7 p.m.), driving speeds are reduced because of high commuter volume. At other times, driving speeds are closer to the speed limit. Based on this observation, the characteristics of the real data indicate that: (1) the values of a sensor's non-geographic attributes are usually close to a certain value, and (2) the values sometimes decrease but increase again after a period. Thus, the proposed simulation model for the non-geographic domain adopts the concept of mean reverting jump diffusion [8,10].

The right-hand side of Fig. 8 shows the detailed flow generating the values of the non-geographic attributes of the objects. First, a value V_{ini} and a gap V_{gap} must be determined to generate the initial means of the non-geographic attributes of the objects. Based on V_{ini} and V_{gap}, generate M distinct initial means of the values in the non-geographic domain for different clusters. Note that the initial means are separated into different clusters. Thus, the initial means of clusters are defined as $\mu_i = V_{ini} + (i-1)V_{gap}$ for $i \in \mathbb{N}$ and $1 \le i \le M$, and the initial mean of the noisy set is defined as V_{ini}.

After generating the initial means of M clusters and the noisy set, iteratively generate the values of the non-geographic attributes of the objects with respect to the initial means and the temporal locality (TL) for each time window. To simulate jump diffusion for the non-geographic attributes of the objects in each cluster, determine whether the values decrease in time window w_h using the following indicator function:

$$\mathrm{I}_{w_h} = \begin{cases} 0 & \text{w.p. TL} \\ 1 & \text{w.p. 1-TL.} \end{cases}$$

In time window w_h, for object S_i in a cluster with initial mean μ, the value of the non-geographic attribute of S_i from time stamp $t \in w_h$, denoted as $S_i.V_t$, can be formulated as follows:

$$S_i.V_t = S_i.V_{t-1} - a(S_i.V_{t-1} - \mu)\Delta t + \omega_t - \mathrm{I}_{w_h} \cdot V_{w_h}, \tag{3}$$

where a is a positive constant, Δt represents the time gap between $S_i.V_{t-1}$ and $S_i.V_t$; ω_t follows a normal distribution with a mean of zero and variance σ^2, and V_{w_h} follows a uniform distribution within the range $(0, V_{drop})$.

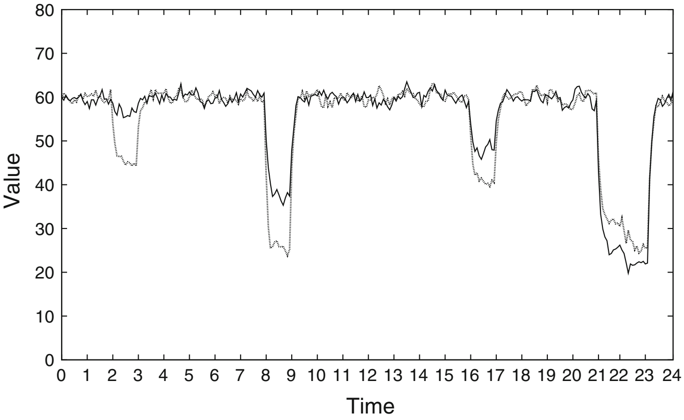


Fig. 10 Two synthetic spatial data streams in a cluster

The number of values for the non-geographic attribute of an object in each time window is W,with a total of F time windows. The proposed framework can generate a synthetic dataset. For example, Fig. 10 shows the values in the non-geographic domain of two objects in the same cluster when the parameters $\mu = 60$, $TL = 0.8$, $F = 24$, $a = 5$, $\sigma = 1$, $V_{drop} = 20$, $\Delta t = 0.1$, and $W = 12$. Compared with Fig. 9, the curves of the synthetic datasets capture the behaviors of the real datasets effectively.

4.1.2 Analysis of the similarity constraint

The previous section proposed a model to simulate real data. However, the clustering problem in this paper requires a time window size of W, a geographic constraint of R, and similarity constraint of ε. As mentioned previously, these parameters can be determined with respect to domain knowledge for real data. Although this model simulates real data based on user-specified parameters (for example, a geographic constraint), we do not specify a similarity constraint for simulation. To use the synthetic datasets generated by the model, this study presents guidelines for setting similarity constraint ε using a user-specified time window size. Based on this model, we first formulate the dissimilarity between two objects from the same cluster in Theorem 4.1 as follows. According to Theorem 4.1, it is then possible to statistically estimate the proper value of similarity constraint ε.

Theorem 4.1 *Given two objects S_i and S_j in the same cluster and a time window $w_h = [t+1, t+W]$ without a decrease (that is, $\mathrm{I}_{w_h} = 0$), the dissimilarity between S_i and S_j in time window w_h can be represented as*

$$\sqrt{\frac{1}{W}\sum_{k=1}^{W}(S_i.V_{t+k} - S_j.V_{t+k})^2},$$

where, for $k \in \mathbb{N}$ and $1 \le k \le W$, $(S_i.V_{t+k} - S_j.V_{t+k}) \sim \mathcal{N}(0, \frac{2\sigma^2(1-(1-a\Delta t)^{2k})}{1-(1-a\Delta t)^2})$.

Proof Given a time window $w_h = [t+1, t+W]$, where W is the size of the time window, the values of objects S_i and S_j in the non-geographic domain are $(S_i.V_{t+1}, S_i.V_{t+2}, \ldots, S_i.V_{t+W})$ and $(S_j.V_{t+1}, S_j.V_{t+2}, \ldots, S_j.V_{t+W})$, respectively. Based

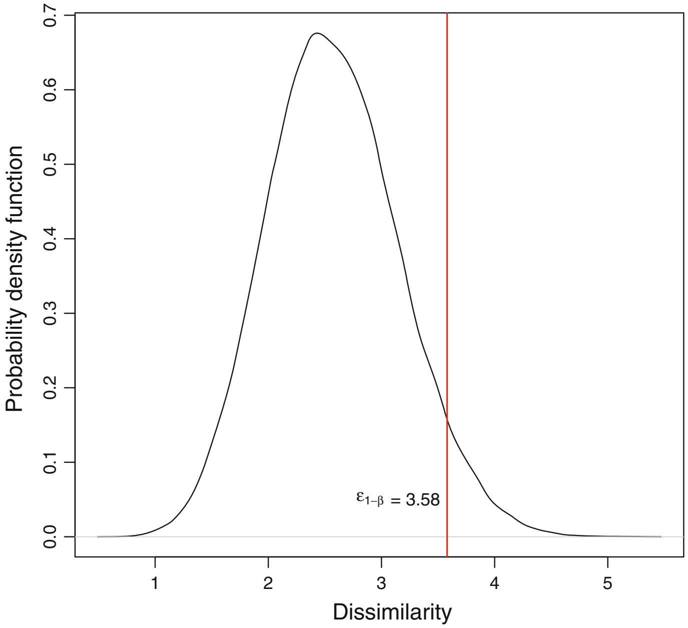


Fig. 11 The probability density function of the dissimilarity between two objects

on Formula 3 in the proposed framework, for $1 \leq k \leq W$, $S_i.V_{t+k} = (1-a\Delta t)S_i.V_{t+k-1} + a\mu\Delta t + \omega_{t+k}$ because $\mathrm{I}_{w_h} = 0$. Similarly, $S_j.V_{t+k} = (1-a\Delta t)S_j.V_{t+k-1} + a\mu'\Delta t + \omega'_{t+k}$ for $1 \leq k \leq W$. Because S_i and S_j are in the same cluster, they have the same initial mean (that is, $\mu = \mu'$). To simplify the derivation below, use μ instead of μ' for S_j. According to Definition 2.1, the dissimilarity between S_i and S_j in time window w_h can be derived by $diss(S_i, S_j, w_h) = \sqrt{\frac{1}{W}\sum_{k=1}^{W}(S_i.V_{t+k} - S_j.V_{t+k})^2}$.

Next, consider how to derive the distribution of the value $(S_i.V_{t+k} - S_j.V_{t+k})$ for $k \in \mathbb{N}$ and $1 \leq k \leq W$. Initially, when $k = 1$, $S_i.V_{t+1} - S_j.V_{t+1} = (1-a\Delta t)(S_i.V_t - S_j.V_t) + (\omega_{t+1} - \omega'_{t+1}) = \omega_{t+1} - \omega'_{t+1}$. Thus, $(S_i.V_{t+1} - S_j.V_{t+1}) \sim \mathcal{N}(0, 2\sigma^2)$ because ω_{t+1} and ω'_{t+1} are $i.i.d.\ \mathcal{N}(0, \sigma^2)$ according to Formula 3. When $k = 2$, we have $S_i.V_{t+2} - S_j.V_{t+2} = (1-a\Delta t)(S_i.V_{t+1} - S_j.V_{t+1}) + (\omega_{t+2} - \omega'_{t+2}) = (1-a\Delta t)(\omega_{t+1} - \omega'_{t+1}) + (\omega_{t+2} - \omega'_{t+2})$. Hence, $(S_i.V_{t+2} - S_j.V_{t+2}) \sim \mathcal{N}(0, 2\sigma^2(1 + (1-a\Delta t)^2))$. Consequently, for $k = W$, we have $S_i.V_{t+W} - S_j.V_{t+W} = (1-a\Delta t)^{W-1}(\omega_{t+1} - \omega'_{t+1}) + (1-a\Delta t)^{W-2}(\omega_{t+2} - \omega'_{t+2}) + \cdots + (\omega_{t+W} - \omega'_{t+W})$. Thus, $(S_i.V_{t+W} - S_j.V_{t+W}) \sim \mathcal{N}(0, \frac{2\sigma^2(1-(1-a\Delta t)^{2W})}{1-(1-a\Delta t)^2})$. □

Theorem 4.1 shows that there is no closed form for the probability density function of the dissimilarity between two objects in the same cluster. Given a set of objects in the same cluster, assume that a tolerance β represents the proportion of objects that are not in the same cluster. Therefore, clustering errors that occur because objects are not in their ground-truth clusters are bounded by the tolerance. Given the tolerance, the similarity constraint can be determined using the Monte Carlo method [32]. For example, assume that the synthetic dataset is generated using the parameter settings $a = 5$, $\sigma = 1$, $\Delta t = 0.1$ and $W = 10$. Using the Monte Carlo method, take 100,000 random samples to calculate the dissimilarity according to Theorem 4.1 and then derive the probability density function of the dissimilarity (Fig. 11). When $\beta = 0.05$, derive the similarity constraint as $\varepsilon = 3.58$. Table 3 shows the

Table 3 Similarity constraint ε with various time window size W settings and tolerance β settings

W	$\beta = 0.01$	$\beta = 0.025$	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.15$	$\beta = 0.2$
10	4.03	3.79	3.58	3.35	3.19	3.02
20	3.65	3.47	3.33	3.17	3.06	2.97
30	3.46	3.33	3.21	3.08	2.99	2.92
40	3.36	3.24	3.14	3.03	2.95	2.89
50	3.29	3.18	3.1	2.99	2.92	2.87

proper settings of the similarity constraint ε when the time window size W varies between 10 and 50 under different values of the tolerance β with $a = 5, \sigma = 1$ and $\Delta t = 0.1$. As Table 3 shows, given a fixed time window size W, the similarity constraint ε decreases slightly as the tolerance β increases. This is reasonable because random noise prevents the objects in a cluster from being grouped with a smaller similarity constraint. Table 3 also shows that, given a fixed tolerance β, the similarity constraint ε decreases sightly as the time window size W increases. This is because a larger time window increases the dissimilarity; therefore, the similarity constraint ε should be smaller as W increases to satisfy the fixed tolerance β. Section 4.2.3 validates the above derivation of the similarity constraint.

4.1.3 Temporal locality estimation

This subsection presents a method for estimating the temporal locality of objects in the non-geographic domain. Given an object and the similarity constraint ε determined by Theorem 4.1 with time window size W and tolerance β, the values of the object in the non-geographic domain are divided into multiple sequences based on the size of W. The number of sequences is denoted by F_{all}, and the multiple sequences represents a set of spatial data streams. The proposed method uses the HBC algorithm to cluster the data streams with the given similarity constraint ε, geographic constraint R and window size W. The mean of each cluster is computed by averaging the means of the spatial data streams in the cluster. For different applications, the relative interval of the general values of the spatial data stream in the non-geographic domain can be determined based on observations of the given values of a spatial data stream in the non-geographic domain. This approach uses the maximum likelihood estimation [8,10] to derive the estimated mean of a data stream and then selects the cluster whose mean is closest to the relative interval or the estimated mean. The term F_{reg} represents the number of objects in the selected cluster. Intuitively, the temporal locality is formulated as $\widehat{TL} = \frac{F_{reg}}{F_{all}}$. For example, Fig. 12a shows the values of a sensor in the non-geographic domain, where the ground truth for its temporal locality is 0.8. The values of this object in the non-geographic domain are partitioned into 12 sequences (i.e., $F_{all} = 12$). Assume that the clustering result for the 12 sequences is $\{S_1, S_2, S_3, S_4, S_6, S_7, S_8, S_9, S_{12}\}$, $\{S_5\}$ and $\{S_{10}, S_{11}\}$. According to the proposed synthetic dataset generation framework (that is, the general values of data streams are close to the maximal speed limits), the general values of data streams are larger. Thus, the mean of cluster $\{S_1, S_2, S_3, S_4, S_6, S_7, S_8, S_9, S_{12}\}$ is larger than that of other clusters, and $F_{reg} = 9$. As a result, the estimated temporal locality $\widehat{TL} = \frac{9}{12} = 0.75$.

To verify the effectiveness of this temporal locality estimation scheme, simulate the values of an object in the non-geographic domain with its parameters $\mu = 60$, $F = 50$, $a = 5, \sigma = 1$, $V_{drop} = 20$, $\Delta t = 0.1$, and $W = 10$. The temporal locality TL varies from 0.1 to 1.

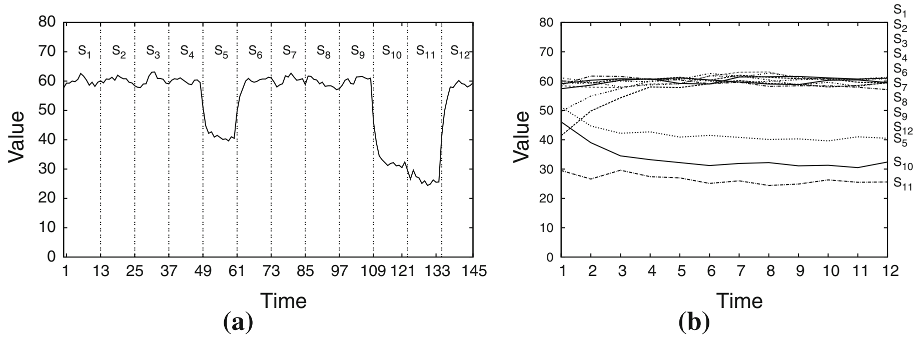


Fig. 12 An illustrative example of temporal locality estimation

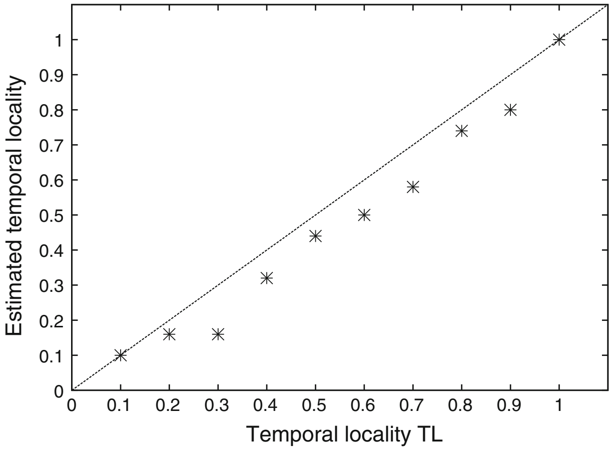


Fig. 13 Temporal locality estimation with various temporal locality TL settings

Figure 13 shows the effectiveness of this temporal locality estimation scheme. In this figure, the estimated temporal locality is less than or equal to the temporal locality of the ground truth, but the estimated temporal locality is close to the temporal locality of the ground truth.

4.2 Experimental results on synthetic datasets

Synthetic datasets were generated to evaluate the proposed algorithm and the existing algorithms. The default parameter settings for generating the synthetic datasets are that $N = 200$, $Range(8, 8)$, $M = 4$, $R = 1.3$, $V_{ini} = 60$, $V_{gap} = 40$, $a = 5$, $\Delta t = 0.1$, $\sigma = 1$, $V_{drop} = 20$, $F = 50$, $W = 10$. The default settings of parameters in the experiments are $\varepsilon = 110, R = 1.7, \alpha = 0.5$, and $\theta = 0.5$. The following subsections describe the experiments and present the experimental results.

4.2.1 Effect of the temporal locality

This section first investigates the effect of temporal locality on the efficiency of the IC algorithm and three existing approaches, the HBC algorithm [43], the CLS algorithm [25], and the CTS-ARMA algorithm [5]. In this experiment, to investigate the effect of TL, let

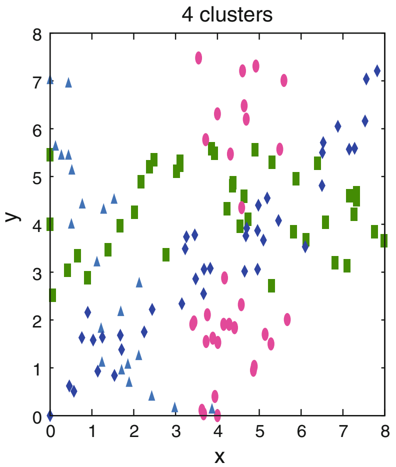


Fig. 14 The ground truth of clusters for the experiments in Sect. 4.2.1

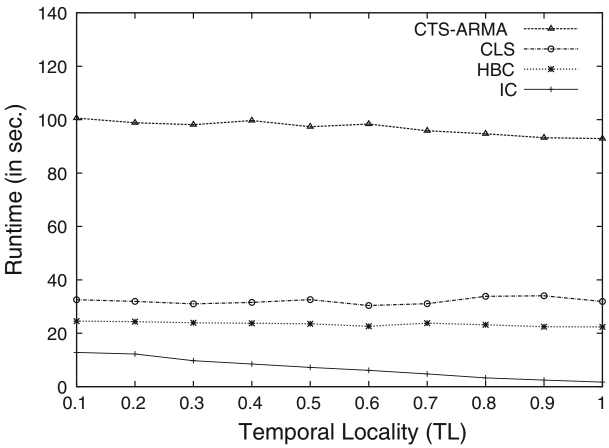


Fig. 15 The performance of the IC, HBC, CLS, and CTS-ARMA algorithms with various TL values

$N = 130$. Figure 14 shows the ground truth of clusters and shows the distribution of objects in the geographic domain, where each point represents an object and x (resp. y) refers to L_x (resp. L_y) of an object. Objects in the same cluster are drawn with the same symbol. Figure 15 shows the efficiency of the three algorithms when the temporal locality TL varies. This figure shows that using a larger temporal locality, the IC and HBC algorithms outperform the CTS-ARMA and CLS algorithms. The IC algorithm has the shortest runtime because it exploits the temporal locality feature, it has the shortest runtime. In particular, when the temporal locality is larger than 0.7, the runtime of the IC algorithm is reduced substantially. In summary, the IC and HBC algorithms are more efficient than the CTS-ARMA and CLS algorithms in stream environments. The remaining experiments focus on presenting a performance comparison of the IC and HBC algorithms.

To compare the clustering results of different algorithms, this study considers clustering results from the same time window. Figure 16 shows the clustering results of different algorithms with varied TL. This figure shows that the number of clusters of these algorithms

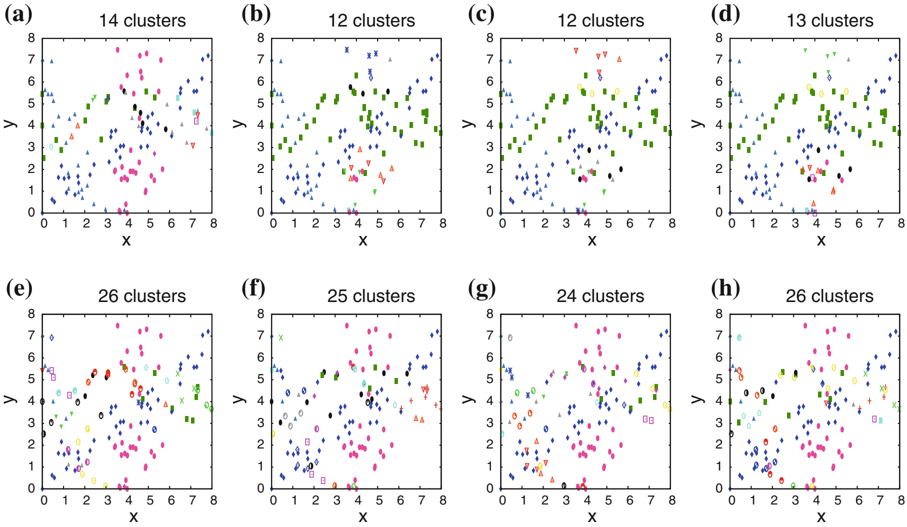


Fig. 16 The clustering results of the IC, HBC, CLS, and CTS-ARMA algorithms with various TL values. **a** CTS-ARMA-TL0.8; **b** CLS-TL0.8; **c** HBC-TL0.8; **d** IC-TL0.8; **e** CTS-ARMA-TL0.5; **f** CLS-TL0.5; **g** HBC-TL0.5; **h** IC-TL0.5;

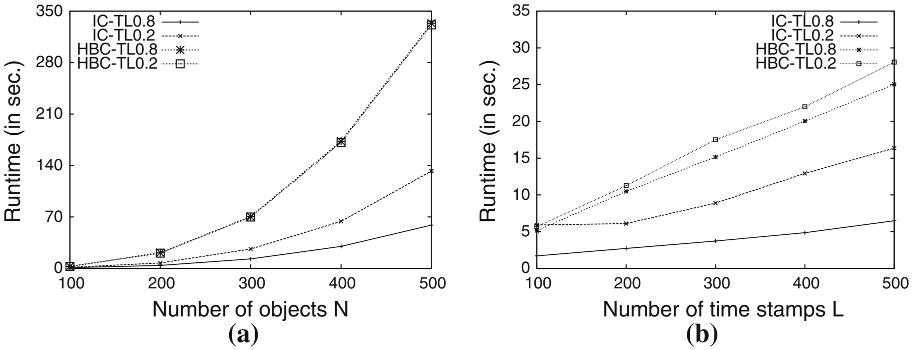


Fig. 17 Scalability analysis of the HBC and IC algorithms with various representations of N and L

increases as TL decreases. That is, for the three algorithms, the clustering result has 12–14 clusters when $TL = 0.8$ and has 24–26 clusters when $TL = 0.5$. This is because, as the value of TL is lower, the non-geographic values of objects varies frequently over time and the dissimilarity of objects in a non-geographic domain increases in the same time window. In other words, for a lower value of TL, only a few objects are clustered and the number of clusters increases.

4.2.2 Comparison of the scalability of algorithms

This experiment was designed to assess the scalability of the IC and HBC algorithms by increasing the number of objects and the number of time stamps using different temporal locality settings (that is, $TL = 0.2$ and $TL = 0.8$). Figure 17a shows that the runtime of the IC algorithm increases slightly as the number of objects increases. However, the algorithm's

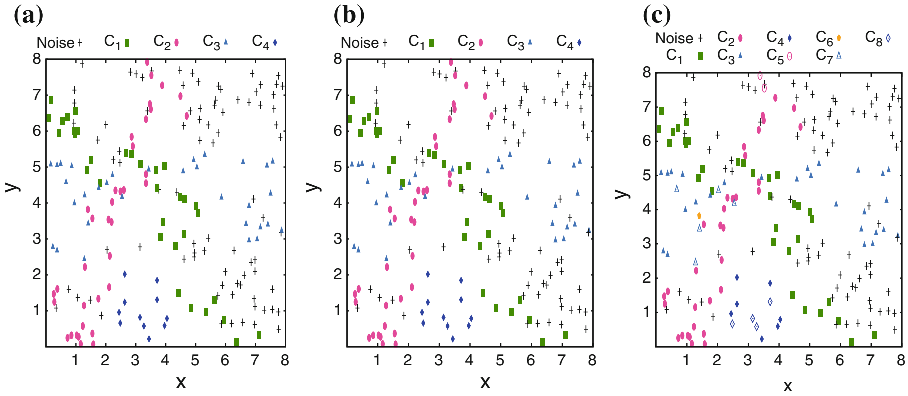


Fig. 18 Quality of the clustering results with various values of tolerance β. **a** Ground truth; **b** $\beta = 0 : \varepsilon = 5$; **c** $\beta = 0.2 : \varepsilon = 3$

runtime becomes shorter when using a larger temporal locality value. In contrast, the runtime of the HBC algorithm increases drastically as N increases. The runtime of 'HBC-TL0.8' and the runtime of 'HBC-TL0.2' are similar when N varies from 100 to 500. Figure 17b shows the effect of the number of time stamps on the performance of the IC and HBC algorithms. With a larger L, the runtimes of both algorithms tend to increase. However, the runtime of the IC algorithm is still shorter than that of the HBC algorithm. These experimental results demonstrate that the IC algorithm achieves favorable scalability for both a large number of objects and a large number of time stamps.

4.2.3 Validation of the similarity constraint

Given a tolerance β, Sect. 4.1.2 shows the derivation of an optimal setting for the similarity constraint based on Theorem 4.1. Because a synthetic dataset was generated for this experiment, Fig. 18a shows the ground truth for comparison. Note that Fig. 18 shows the distribution of objects in the geographic domain, where each point represents an object and x (resp. y) refers to the L_x (resp. L_y) of an object. The ground truth refers to the clusters generated by the proposed simulator. Clusters that have more than one member were used to evaluate the effectiveness of the proposed guidelines in Sect. 4.1.2, and the clusters that have only one member were regarded as noise. By setting different values of tolerance β (that is, $\beta = 0$ and $\beta = 0.2$), the similarity constraints were derived from Theorem 4.1. For $\beta = 0$ (resp. $\beta = 0.2$), the similarity constraint was set at 5 (resp. 3). The clustering results can be derived by applying the proposed algorithm. Figure 18b shows that with a tolerance β=0, the clustering result is extremely close to the ground truth in Fig. 18a. However, when β is set at 0.2, the clustering result is dissimilar to the ground truth. These experimental results validate the derivation in Theorem 4.1 and show that users can set their own tolerance values. These results also demonstrate that the similarity constraint is properly determined using Theorem 4.1.

4.2.4 Sensitivity analysis of the IC algorithm

Section 4.1.3 presented a method for estimating the temporal locality of the non-geographic attributes of objects. After estimating the temporal locality, set the temporal correlation α and

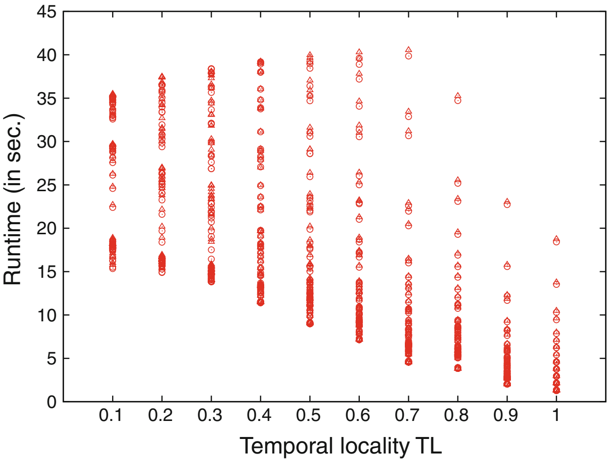


Fig. 19 The runtimes for different values of temporal localities TL with various settings of parameter α and probability threshold θ

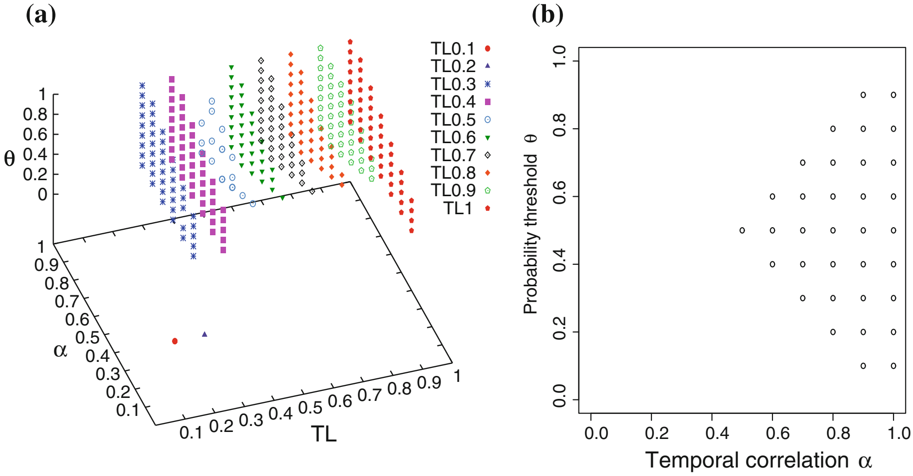


Fig. 20 A guideline for setting the temporal correlation α and probability threshold θ with various values of temporal locality TL. **a** TL = 0.1 ~1; **b** TL = 0.3 ~1

probability threshold θ for the IC algorithm as follows. The proposed simulation framework generated synthetic datasets with temporal locality TL varying between 0.1 and 1. For each synthetic dataset, the IC algorithm was then executed with different combinations of α and θ settings, where the range of the two parameters varies between 0.1 and 1. Each combination of parameters ten times was repeated to average the runtime of each combination. Figure 19 shows the experimental results. For each temporal locality, we have a range of runtimes with different combination settings for α and θ, and a lower bound of runtimes exists for different temporal locality values. The minimal runtime of each temporal locality decreases because the temporal locality tends to increase. Based on the shorter runtimes under different temporal localities, Fig. 20a shows the corresponding settings for α and θ.

To set the proper combinations of temporal correlation α and probability threshold θ for each temporal locality, choose the minimal runtime T_{min} and then select the combinations

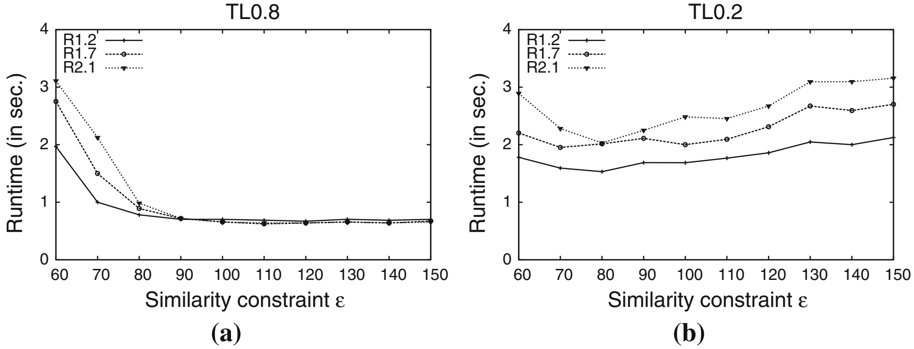


Fig. 21 Sensitivity analysis of the IC algorithm regarding TL with varying R and ε

of parameters whose runtimes are between $[T_{min}, (1 + \lambda)T_{min}]$ where $\lambda = 0.05$. Figure 20a shows a plot of these combinations with respect to different temporal locality values. This figure shows that the best combinations of temporal correlation α and probability threshold θ is 0.1 and 0.1, respectively, while $TL = 0.1$ and $TL = 0.2$. This clustering result is not closely related to that of the previous time window because the temporal locality is smaller. Therefore, the temporal correlation α should be set at a smaller value. The probability threshold should also be set at a smaller value, because the elements of a probability matrix do not easily exceed a larger probability threshold, and the IC algorithm does not generate coarse clusters to reduce the clustering runtime. For the temporal locality between 0.3 and 1, Fig. 20b shows the overlap of the combinations of the temporal correlation α and probability threshold θ. Both the temporal correlation and probability threshold can be selected from the range shown in Fig. 20b. Because the IC algorithm exploits the temporal locality for efficiency purposes, the temporal correlation α and probability threshold θ significantly influence the clustering results of the algorithm. Thus, these observations provide guidelines for determining the settings for temporal correlation α and probability threshold θ.

4.2.5 Effect of the constraints

This section presents the effect of the constraints on the IC algorithm at different TL values, and Fig. 21 shows the results. Given a higher value of TL (e.g., $TL = 0.8$), in Fig. 21a, the runtime of the IC algorithm dramatically decreases as ε varies from 60 to 90. The runtime of the IC algorithm becomes stable when $\varepsilon > 90$. This is because, when the similarity constraint is loosened (i.e., a higher value of ε), most objects are grouped together. For a higher value of TL (e.g., $TL = 0.8$), these objects still belong to the same cluster over time. Thus, the IC algorithm can effectively predict the clusters according to the previous clustering results. In addition, Fig. 21a shows that a higher value of R also loosens the geographic constraint. Thus, the runtime of the IC algorithm with a higher value of R (e.g., $R = 2.1$) is shorter than the runtime of the IC algorithm with a lower value of R (e.g., $R = 1.2$) given a fixed ε. When $\varepsilon > 90$, the effect of the geographic constraint is not obvious. This is because most objects can be clustered based on the similarity-constrained relationships given a higher value of ε.

Given a lower value of TL (e.g., $TL = 0.2$), Fig. 21b shows that the runtime of the IC algorithm decreases slightly as ε varies from 60 to 90, but increases slightly when $\varepsilon > 90$. This is because, for a lower value of TL, the non-geographic values of objects are not similar

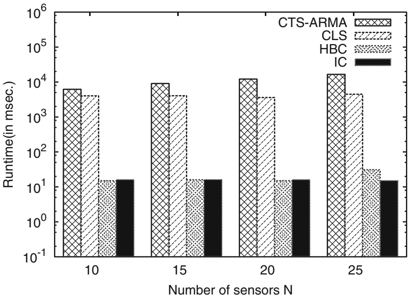


Fig. 22 The efficiency comparison of the IC, HBC, CLS, and CTS-ARMA algorithms using real datasets

with time, and the prediction of the IC algorithm would be destroyed. In addition, a more loosened constraint (e.g., a higher value of ε or a higher value of R) induces most of the objects to be grouped together, but these objects may not belong to the same cluster in the next time window given a lower TL. Similarly, Fig. 21b shows that, for the same ε, the runtime of the IC algorithm with a higher value of R (e.g., $R = 2.1$) is shorter than that with a lower value of R (e.g., $r = 1.2$).

4.3 Experimental results on real datasets

The following subsections present comparisons of the performance of the IC algorithm with that of existing approaches and analyze their scalability using a real dataset. This study also investigates the sensitivity of a geographic constraint and a similarity constraint.

4.3.1 Real dataset

The real dataset was obtained from the Taiwan Area National Freeway Bureau. We compiled a traffic database for Freeway No.1, which runs the length of the island (a distance of 372.7 kilometers). We collected data from 100 sensors positioned along the freeway. Each sensor has a specific location and reports the speeds of vehicles on the monitored segments every five minutes. The default settings for the real dataset include the number of objects (that is, sensors), $N = 79$; the number of time stamps, $L = 434$; the geographic constraint, $R = 10$ (Km); the time window size, $W = 10$; and the similarity constraint, $\varepsilon = 5$. According to Sect. 4.2.4, we set $\alpha = 0.5$ and the probability threshold $\theta = 0.5$. Note that $F = \lfloor \frac{L}{W} \rfloor$.

4.3.2 Efficiency comparison

As mentioned in Sect. 4.2.1, the HBC, CLS, and CTS-ARMA algorithms are regarded as competitors of the proposed algorithm. This experiment was designed to compare the IC algorithm with the HBC, CLS, and CTS-ARMA algorithms. Similarly, for the CLS and CTS-ARMA algorithms, the number of clusters was set as the number of clusters derived from the IC algorithm. We evaluate the efficiency of the three algorithms with different representations of N. Figure 22 shows that the runtimes of the CLS and CTS-ARMA algorithms are less favorable than the runtimes of the HBC and IC algorithms, revealing that the IC and HBC algorithms are more efficient than the CLS and CTS-ARMA algorithms when using real datasets. Hence, the remainder of the experiments only compare the performance of the IC algorithm and the competitor algorithm, HBC.

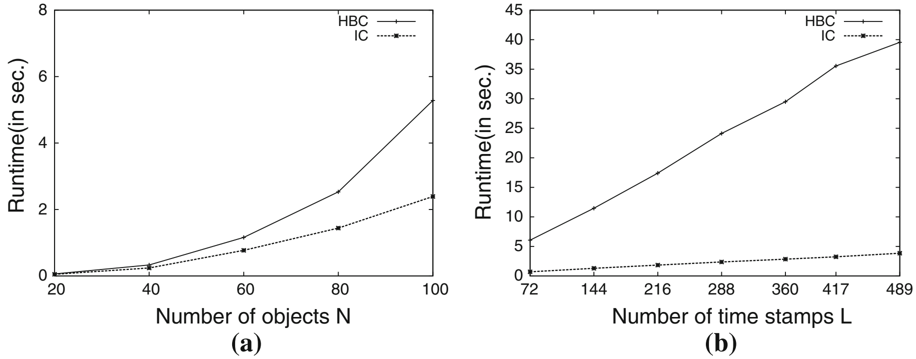


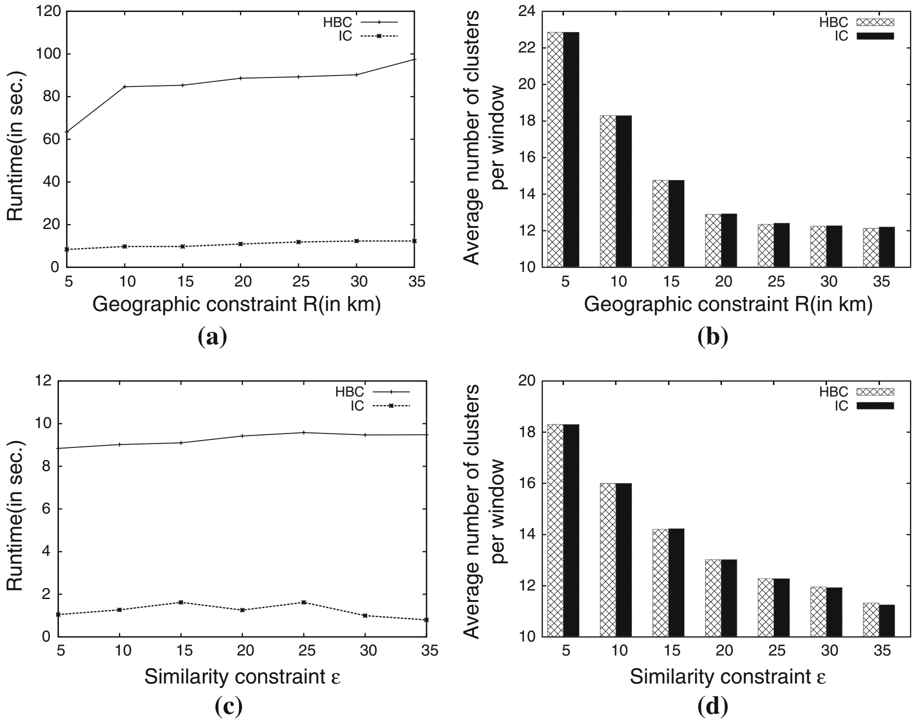
Fig. 23 Scalability analysis of the IC and HBC algorithms regarding N and L

4.3.3 Scalability comparison

These experiments show that the HBC and IC algorithms can cluster spatial data streams efficiently. To evaluate the scalability of the algorithms, the number of objects and the length of the data streams were increased. Figure 23 shows the experimental results when varying the number of objects and the length of the spatial data streams. Figure 23a shows that the runtimes of both algorithms tend to increase with the number of objects, but the IC algorithm is more efficient than the HBC algorithm. This is because the HBC algorithm must perform the clustering task in each time window, whereas the IC algorithm uses prior clustering results to derive coarse clusters. Figure 23b shows that the runtimes of both algorithms increase with the length of the spatial data streams. This is because longer data streams have more time windows. As a result, the HBC algorithm's runtime increases sharply. The IC algorithm's runtime is shorter because it exploits the temporal locality features to derive clusters.

4.3.4 Effect of the constraints

Next consider the effect of the constraints on the performance of the HBC and IC algorithms. Figure 24 shows the experimental results when varying the geographic constraint and the similarity constraint. Figure 24a shows that with a larger geographic constraint R, the runtimes of both algorithms increase. However, the increase in the runtime of the IC algorithm is smaller than that of the HBC algorithm. This is because, with a larger value of R, more objects are considered for clustering. Therefore, the runtime of the HBC algorithm increases. Although the runtime of the IC algorithm increases slightly as R increases, it is still smaller than that of the HBC algorithm because of the incremental feature in the IC algorithm. To investigate the effect of R on the clustering results, it is necessary to average the number of clusters of all time windows because the number of clusters varies over time. Figure 24b demonstrates the clustering results with various R, indicating that the average number of clusters per window decreases as R increases. This is because a higher value of R loosens the geographic constraint for clustering and more objects could be grouped together, which induces a low number of clusters. Figure 24c shows the performance of the algorithms when varying the similarity constraint ε. The effect of ε on the algorithms is not significant when ε varies between 5 and 35, because the runtime of each time window only increases slightly. In addition, Figure 24d shows the corresponding clustering results. This figure shows that the average number of clusters per window decreases as ε increases. Similar to the effect of



(a) (b)

(c) (d)

Fig. 24 Sensitivity analysis of the HBC and IC algorithms regarding R and ε

the geographic constraint, a higher value of ε loosens the similarity constraint for clustering, allowing more objects to be clustered together and decreasing the number of clusters.

5 Conclusion

The study presents a dual clustering problem for spatial data streams. However, the clustering processes in stream environments are time-sensitive because of frequently updated sensor data, and existing algorithms are not suitable for clustering time-sensitive spatial data streams. This study proposes an incremental clustering algorithm to discover clustering results efficiently using temporal locality features and proposes a framework for simulating real data effectively using various degrees of temporal locality. To use the synthetic datasets generated by the proposed simulation effectively, this study presents a statistical approach to estimate the similarity constraint for dual clustering problems in spatial data streams under a user-specified tolerance. The experiments in this study confirm the effectiveness of these clustering results. This study also proposes an approach to estimate the degree of temporal locality in sensor data and demonstrates the effectiveness of the approach by assessing the results of the experiment. Based on these results, this study presents a comparison of the performance of the proposed algorithm and existing algorithms using synthetic and real datasets. Results show that the IC algorithm is more efficient and more scalable than existing algorithms. This study also presents an analysis of the effect of temporal locality features on the proposed algorithm that was thoroughly examined in the experiments. This study presents the guidelines for setting parameters from the experimental results to apply the proposed algorithm effectively.

Acknowledgments We thank anonymous reviewers for their very useful comments and suggestions. Wen-Chih Peng was supported in part by the National Science Council, Project No.100-2218-E-009-016-MY3 and 100-2218-E-009-013-MY3, by TaiwanMoE ATU Program, by HTC and by Delta.

Appendix A. Hierarchical-based clustering algorithm

A hierarchical-based clustering (HBC) algorithm has been proposed for solving dual clustering problems in spatial data streams [43]. In each time window, the HBC algorithm first constructs an SC-graph and places explicit edges with their dissimilarity values in a priority queue Q, which is a data structure that returns the edge with the minimal dissimilarity value in the set of explicit edges in Q. As with a bottom-up hierarchical clustering algorithm, each vertex is initially regarded as a single cluster. The explicit edge with the minimal dissimilarity value is then removed from the priority queue Q. Let that edge be $e_e(S_i, S_j)$. If S_i and S_j belong to the same cluster, there is no need to cluster them. However, if they belong to different clusters (e.g., $S_i \in C_i$, $S_j \in C_j$, and $C_i \neq C_j$), both the connectivity requirement and the complete subgraph requirement (Section 3.1) should be verified. If these two requirements are met, clusters C_i and C_j are merged to form a new cluster. This procedure is performed iteratively until Q is empty.

Algorithm 2: Hierarchical-Based Clustering (HBC) Algorithm

```
input  : A set of objects $STD$, a similarity constraint $\varepsilon$, a geographic constraint $R$, a window size $W$,
         and the time interval $[t_s, t_e]$
output : A set of SC-clusters $R_{w_k}$ with respect to time window $w_k$
for each pair $S_i, S_j$ in $STD$ do
    Compute $ED(S_i, S_j)$;
end
for each time window $w_k = [t_s + k \cdot W, t_s + (k+1) \cdot W]$ where $0 \leq k \leq \lfloor \frac{t_e - t_s}{W} \rfloor$ do
    for each pair $S_i, S_j$ in $STD$ do
        Compute $diss(S_i, S_j, w_k)$;
        Generate trivial and hidden edges by $\varepsilon$ and $R$;
        Store all trivial edges in a priority queue $Q$ in increasing order by dissimilarity values;
    end
    $R_{w_k} = \bigcup_{\forall S_h \in STD} \{S_h\}$;
    while $Q$ is not empty do
        Delete $e_t(S_i, S_j)$ from $Q$;
        if $S_i \in C_i, S_j \in C_j$ where $C_i \neq C_j$ then
            If $C_i \bigcup C_j$ forms a SC-cluster by checking the complete subgraph requirement, and then
            remove $C_i, C_j$ from $R_{w_k}$ and add $C_i \bigcup C_j$ into $R_{w_k}$;
        end
    end
    return $R_{w_k}$;
end
```

References

1. Aggarwal CC, Han J, Wang J, Yu PS (2003) A framework for clustering evolving data streams. In: Proceedings of the 29th international conference on very large data bases (VLDB), pp 81–92
2. Aggarwal CC, Yu PS (2010) On clustering massive text and categorical data streams. Knowl Inf Syst 24(2):171–196

3. Aghabozorgi SR, Saybani MR, Wah TY (2012) Incremental clustering of time-series by fuzzy clustering. J Inf Sci Eng (JISE) 28(4):671–688
4. Alïtelhadj A, Boughanem M, Mezghiche M, Souam F (2012) Using structural similarity for clustering xml documents. Knowl Inf Syst (KAIS) 32(1):109–139
5. Bagnall AJ, Janacek GJ (2004) Clustering time series from arma models with clipped data. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 49–58
6. Banerjee A, Ghosh J (2006) Scalable clustering algorithms with balancing constraints. Data Min Knowl Discov 13(3):365–395
7. Beringer J, HLullermeier E (2006) Online clustering of parallel data streams. Data Knowl Eng (DKE) 58(2):180–204
8. Borovkova S, Permana FJ (2004) Modelling electricity prices by the potential jumpdiffusion. In: Proceedings of the autumn school and international conference on stochastic finance (StochFin), pp 239–263
9. Cao F, Ester M, Qian W, Zhou A (2006) Density-based clustering over an evolving data stream with noise. In: Proceedings of the sixth SIAM international conference on data mining (SDM), pp 326–337
10. Cartea A, Figueroa MG (2005) Pricing in electricity markets a mean reverting jump diffusion model with seasonality. Appl Math Finance 12(4):313–335
11. Costa G, Manco G, Ortale R (2010) Density-based clustering of data streams at multiple resolutions. Data Min Knowl Discov (DMKD) 20(1):152–187
12. Dai BR, Lin CR, Chen MS (2007) Constrained data clustering by depth control and progressive constraint relaxation. Int J Very Large Data Bases 16(2):201–217
13. Davidson I, Ester M, Ravi SS (2007) Efficient incremental constrained clustering. In: Proceedings of the 13th ACM international conference on knowledge discovery and data mining (SIGKDD), pp 240–249
14. Davidson I, Ravi SS (2005) Clustering with constraints: Feasibility issues and the k-means algorithm. In: Proceedings of the fifth SIAM international conference on data mining (SDM), pp 138–149
15. Ester M, Kriegel HP, Sander J, Wimmer M, Xu X (1998) Incremental clustering for mining in a data warehousing environment. In: Proceedings of the 24th international conference on very large data bases (VLDB), pp 323–333
16. Ge R, Ester M, Gao BJ, Hu Z, Bhattacharya B, Ben-Moshe B (2008) Joint cluster analysis of attribute data and relationship data: the connected k-center problem, algorithms and applications. ACM Trans Knowl Discov Data 2(2):7:1–7:35
17. Ge R, Ester M, Jin W, Davidson I (2007) Constraint-driven clustering. In: Proceedings of the 13th ACM international conference on knowledge discovery and data mining(SIGKDD), pp 320–329
18. Guha S, Meyerson A, Mishra N, Motwani R, OCallaghan L (2003) Clustering data streams: theory and practice. IEEE Trans Knowl Data Eng (TKDE) 15(3):515–528
19. Guo L, Ai C, Wang X, Cai Z, Li Y (2009) Real time clustering of sensory data in wireless sensor networks. In: Proceedings of the 28th IEEE international performance computing and communications conference (IPCCC), pp 33–40
20. Halkidi M, Spiliopoulou M, Pavlou A (2012) A semi-supervised incremental clustering algorithm for streaming data. In: Proceedings of the 16th Pacific-Asia conference on advances in knowledge discovery and data mining (PAKDD), pp 578–590
21. Han J, Kamber M (2000) Data mining: concepts and techniques. Morgan Kaufmann, New York
22. Huang J, Zhang J (2010) Distributed dual cluster algorithm based on grid for sensor streams. Int J Digit Content Technol Appl (JDCTA) 4(9):225–233
23. Kavitha V, Punithavalli M (2010) Clustering time series data stream—a literature survey. ACM Trans Knowl Discov Data (IJCSIS) 8(1):289–294
24. Klein D, Kamvar SD, Manning CD (2002) From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In: Proceedings of the 19th international conference on machine learning (ICML), pp 307–314
25. Liao ZX, Peng WC (2012) Clustering spatial data with a geographic constraint: exploring local search. Knowl Inf Syst 31(1):153–170
26. Lin CR, Liu KH, Chen MS (2005) Dual clustering: integrating data clustering over optimization and constraint domains. IEEE Trans Knowl Data Eng 17(5):628–637
27. Lin J, Vlachos M, Keogh EJ, Gunopulos D (2004) Iterative incremental clustering of time series. In: Proceedings of the ninth international conference on extending database technology (EDBT), pp 106V122
28. Lühr S, Lazarescu M (2009) Incremental clustering of dynamic data streams using connectivity based representative points. Data Knowl Eng (DKE) 68(1):1–27
29. Mirkin B (2011) Clustering for data mining: a data recovery approach. Taylor and Francis, London

30. O'Callaghan L, Meyerson A, Motwani R, Mishra N, Guha S (2002) Streaming-data algorithms for high-quality clustering. In: Proceedings of the 18th IEEE international conference on data engineering (ICDE), pp 685–694
31. Pensa RG, Ienco D, Meo R (2012) Hierarchical co-clustering: off-line and incremental approaches. Data Mining Knowl Discov (DMKD). doi:10.1007/s10618-012-0292-8
32. Robert CP, Casella G (2004) Monte carlo statistical methods. Springer, New York
33. Rodrigues PP, Gama J, Lopes L (2008) Clustering distributed sensor data streams. In: Proceedings of the European conference on machine learning and knowledge discovery in databases—part II (ECML PKDD), pp 282–297
34. Rodrigues PP, Gama J, Pedroso JP (2008) Hierarchical clustering of time series data streams. IEEE Trans Knowl Data Eng (TKDE) 20(5):615–627
35. Shi YB, Yuan CA, Huang Y, Wen YG (2010) A method of spatial clustering based on the combination of the spatial coordinate and attributes. In: Proceedings of the sixth international conference on information systems security (ICISS), pp 526–529
36. Shi Y, Zhang L (2010) Coid: A clustervoutlier iterative detection approach to multi-dimensional data analysis. Knowl Inf Sys. doi:10.1007/s10115-010-0323-y
37. Tai CH, Dai BR, Chen MS (2007) Incremental clustering in geography and optimization spaces. In: Proceedings of the 11th Pacific-Asia conference on knowledge discovery and data mining (PAKDD), pp 272–283
38. Taiwan area national freeway bureau. http://www.freeway.gov.tw/
39. Tan PN, Steinbach M, Kumar V (2006) Introduction to data mining. Addison-Wesley, New York
40. Wagstaff K, Cardie C (2000) Clustering with instance-level constraints. In: Proceedings of the 17th international conference on machine learning (ICML), pp 1103–1110
41. Wan L, Ng WK, Dang XH, Yu PS, Zhang K (2009) Density-based clustering of data streams at multiple resolutions. ACM Trans Knowl Discov Data (TKDD) 3(3):1–28
42. Wang JW, Cheng CH (2007) An efficient method for estimating null values in relational databases. Knowl Inf Syst 12(3):379–394
43. Wei LY, Peng WC (2009) Clustering data streams in optimization and geography domains. In: Proceedings of the 13th Pacific-Asia conference on knowledge discovery and data mining (PAKDD), pp 997–1005
44. Yang J (2003) Dynamic clustering of evolving streams with a single pass. In: Proceedings of the 19th IEEE international conference on data engineering (ICDE), pp 695–697
45. Zhou J, Guan J, Li P (2007c) Dcad: a dual clustering algorithm for distributed spatial databases. Geo-Spatial Inf Sci 10(2):137–144
46. Zhou A, Cao F, Yan Y, Sha C, He X (2007) Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 133–142
47. Zhou A, Cao F, Yan Y, Sha C, He X (2007) Distributed data stream clustering: a fast em-based approach. In: Proceedings of the 23rd international conference on data engineering (ICDE), pp 736–745

Author Biographies

Ling-Yin Wei received her Ph.D. in Computer Science from National Chiao Tung University, Taiwan, in 2012. Her research interests include data mining, location-based services, mobile social network, and cloud computing.

Wen-Chih Peng was born in Hsinchu, Taiwan, R.O.C. in 1973. He received the BS and MS degrees from the National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in Electrical Engineering from the National Taiwan University, Taiwan, R.O.C. in 2001. Currently, he is an assistant professor at the department of Computer Science, National Chiao Tung University, Taiwan. Prior to joining the department of Computer Science and Information Engineering, National Chiao Tung University, he was mainly involved in the projects related to mobile computing, data broadcasting, and network data management. Dr. Peng serves as a PC member in several prestigious conferences, such as IEEE International Conference on Data Engineering (ICDE), Pacic Asia Knowledge Discovering and Mining (PAKDD), and Mobile Data Management (MDM). His research interests include mobile computing, network data management, and data mining. He is a member of IEEE.