Distributed and Parallel Databases
https://doi.org/10.1007/s10619-020-07312-z

MISCELA: discovering simultaneous and time-delayed correlated attribute patterns

Kei Harada[1] · **Yuya Sasaki**[1] · **Makoto Onizuka**[1]

Accepted: 16 September 2020

Abstract
This article addresses a new pattern mining problem in time series sensor data, which we call ***correlated attribute pattern mining***. The correlated attribute patterns (CAPs for short) are the sets of attributes (e.g., temperature and traffic volume) on sensors that are spatially close to each other and temporally correlated in their measurements. Although the CAPs are useful to accurately analyze and understand spatio-temporal correlation between attributes, the existing mining methods are inefficient to discover CAPs because they extract unnecessary patterns. Therefore, we propose a mining method Miscela to efficiently discover CAPs. MISCELA can discover not only simultaneous correlated patterns but also time delayed correlated patterns. Furthermore, we extend MISCELA to automatically search for correlated patterns with any time delays. Through our experiments using three real sensor datasets, we show that the response time of MISCELA is up to 20.84 times faster compared with the state-of-the-art method. We show that MISCELA discovers meaningful patterns for urban managements and environmental studies.

Keywords Spatio-temporal data mining · Smart city · Co-evolving patterns · Correlated attribute patterns

1 Introduction

Many cities have installed a wide variety of sensors to continuously and cooperatively monitor urban conditions, such as the distribution of air pollution, the transition of traffic volume, and the change of temperature. Municipalities analyze the

✉ Kei Harada
harada.kei@ist.osaka-u.ac.jp

Yuya Sasaki
sasaki@ist.osaka-u.ac.jp

Makoto Onizuka
onizuka@ist.osaka-u.ac.jp

[1] Graduate School of Information Science and Technology, Osaka University, Suita 565-0871, Japan

Published online: 25 September 2020

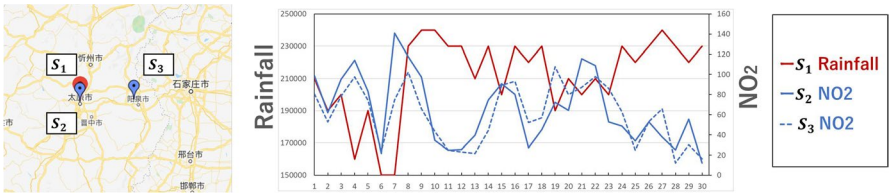


Fig. 1 Examples of simultaneous and time-delayed correlations in China

urban conditions and make a decision for the urban planning by using such sensor data. For example, Santander, Spain monitors the traffic volume within the city and informs people of the real-time traffic information [12, 13]. The accumulated traffic data are used to several urban managements such as the traffic prediction, the road extension, and the traffic signal control. In these services, it is useful to discover sets of roads which are spatially close and whose traffic volume increases or decreases during the same periods (i.e., co-evolve). The problem is called the *spatial co-evolving pattern mining* (for short, SCP mining), which discovers sensors that are spatially close to each other and temporally co-evolving in their measurements. Since the SCP mining is useful for many applications such as the air pollution analysis in an urban area, several SCP mining methods have been proposed [2, 17]. The SCP mining discovers meaningful patterns for analyzing urban environments.

1.1 Motivation

Many cities typically monitor multiple attributes, for example, Santander monitors temperature and traffic volume, and China monitors several types of particles such as NO_2 and meteorological data such as rainfall. Multiple attributes are useful to analyze urban environments from diversified viewpoints. To accurately analyze and understand the urban environments, it is expected to discover *correlated attributes* which are spatially close to each other and temporally co-evolving during both the same and different periods. Additionally, some attributes may co-evolve with time delay. That is, *time-delayed co-evolving* happens in many sensor data. We show examples of correlated attributes with time delay in the following.

Example Figure 1 shows an example of correlated attributes in a dataset of environmental sensors in China. There are two types of sensors; red (s_1) and blue (s_2 and s_3) points denote the sensors that measure rainfall and NO_2, respectively. The left figure shows the locations of sensors and these sensors are spatially close to each other. The right figure shows their measurement values for 30 days and each time step indicates each day. We here note that we choose 15 days out of 730 days for visualization so that we easily understand the changes of measured values.

s_1 and s_2 are placed at the same location and co-evolve simultaneously (i.e., the measured values often increase/decrease at the same time steps). For example, the values of s_1 and s_2 increase/decrease simultaneously at timestamps 1 and 2. On the other hand, s_1 and s_3 that are placed at different locations and co-evolve with 1 day

time delay (i.e., measured values of s_1 increase/decrease 1 day after increasing/decreasing measure values of s_3). For example, the value of s_1 increases 1 day after increasing the value of s_3 at timestamp 8. The co-evolution of 1 day delay can be computed by the same way of simultaneous co-evolution when we shift red lines left by one timestamp. It is interesting that s_1 and s_2 do not often co-evolve with 1 day time delay. These results indicate that rainfall affects the volume of NO_2 in the same location at the same time or in close locations with time delays.

The SCP mining cannot discover the above correlations because it assumes sensors with a single attribute and aims to discover patterns whose measurements simultaneously change. The correlated attribute provides more diverse knowledge than the patterns obtained by the SCP mining because the correlated attribute takes multiple attributes and time delays into account.

1.2 Contribution

Motivated by the above examples, we introduce a problem of discovering correlations among different attributes, which is called *correlated attribute pattern* (*CAP*) *mining*. A CAP is a set of multiple attributes measured by a set of sensors which are close to each other and whose measurements co-evolve. Although we can naively extend the existing SCP mining methods to discover CAPs, they are inefficient because they extract unnecessary sets of sensors that all the sensors measure a single attribute data. Therefore, we propose a novel CAP mining method, called Miscela, which can efficiently discover CAPs in a set of sensors whose measurements contain multiple attributes. Miscela can discover not only simultaneous correlated patterns but also time-delayed correlated patterns. Time-delayed correlated patterns are effective to discover diverse knowledge. However, it is difficult to find all the correlated patterns with time delays because the search space is significantly large. To discover all time-delayed correlated patterns with any time delays, we need repeatedly execute a significantly large number of Miscela with changing the time delays. For efficiently discovering all time-delayed correlated patterns with any time delay, we extend Miscela to automatically search for time-delayed CAPs to reduce the number of executing Miscela. This method leverages an apriori-based approach that if finds CAPs consisting of a small number of attributes and then combine multiple CAPs to make new candidates consisting of a large number of attributes. This method is efficient because it can prune unnecessary time delay parameters that do not generate CAPs.

We summarize the main contributions of this article in the following:

- We introduce a new problem, CAP mining, which aims to discover correlations among different attributes.
- We propose Miscela that efficiently discovers CAPs. Miscela accelerates the CAP mining with a novel data structure called the *CAP search tree*, which conceptually organizes all CAPs based on the spatial constraint and combinations of attributes.

- We propose MISCELA with automatic time-delay search to efficiently discover all CAPs with any time-delay patterns.
- We conduct experiments with three real sensor datasets measured in Santander and China. The experimental results demonstrate that MISCELA is up to 20.84 times faster to that of the state-of-the-art SCP mining method [17]. MISCELA with automatic time-delay search significantly reduces the computation cost compared with a naive method that repeats executing MISCELA. Furthermore, we discovered meaningful CAPs for urban managements and environmental studies.

This article is an extended version of our previous work [5]. New contents are as follows. The previous work discovers only simultaneous correlation patterns. We redefine the CAP mining and extend MISCELA to discover time-delayed correlation patterns. Additionally, we develop MISCELA with automatic time-delay search. In experimental studies, we show meaningful CAPs both with and without time delays.

1.3 Organization

The rest of paper is organized as follows. We formulate the CAP mining problem in Sect. 2 and present a novel CAP mining method MISCELA in Sect. 3. In Sect. 4, we present MISCELA with automatic time-delay search. We conduct CAP mining experiments to evaluate the performance of MISCELA and show meaningful CAPs in Sect. 5. After that, we summarize the past typical works related to our work in Sect. 6, followed by the conclusion in Sect. 7.

2 Problem description

We first explain an overview of CAP mining and then details of the problem definition that we solve in this paper.

2.1 Overview

We define the CAP mining as a problem of discovering spatially and temporally correlated environmental properties (correlated attributes) such that multiple sensors measure on those attributes that satisfy the following conditions: (1) the sensors are deployed at spatially close locations, (2) the measurements of the attributes co-evolve frequently, and (3) the measurements of a certain number of attributes are temporally similar with permitting time-delay. We introduce Definitions 1 and 2 for spatial connectivity, Definitions 3–5 for temporal co-evolution on a single attribute, and Definitions 6 and 7 for co-evolution among multiple attributes, respectively. Finally, we define CAP in Definition 8. We summarize the description of variables in Table 1.

Table 1 The description of the variables

Variables	Description
$\boldsymbol{S} = \{s_1, \ldots, s_n\}$	Set of n sensors
$\boldsymbol{A} = \{a_1, \ldots, a_m\}$	Set of m attributes
$\boldsymbol{T} = \langle t_1, t_2, \ldots, t_T \rangle$	Time domain
$s_i[t_j]$	Measurement of s_i at t_j
l_i	Location of s_i
a_i	Attribute of s_i
η	Distance threshold
$\boldsymbol{G} \subseteq \boldsymbol{S}$	Spatially connected set
$\boldsymbol{G}_a \subseteq \boldsymbol{S}$	Spatially connected congeneric set of attribute a
$r_i[t_j]$	Change rate of s_i at t_j
ε	Evolving rate
$\boldsymbol{\Theta}_a = (\theta_a{}^+, \theta_a{}^-)$	Evolving threshold of threshold a
$E(\boldsymbol{G}_a)$	Co-evolution on $\mathbf{G}_a$
μ	The maximum number of CAP attributes
$\boldsymbol{\tau} = \{\tau_{a_1}, \ldots, \tau_{a_\mu}\}$	Set of time-delay offsets
$C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}$	Time-delayed co-evolution among μ attributes
ψ	Minimum support

2.2 Details of the definitions

Let $\boldsymbol{S} = \{s_1, s_2, \ldots, s_n\}$ be a sensor set in a geographical region. Each sensor $s_i \in \boldsymbol{S}$ $(1 \leq i \leq n)$ is deployed at location l_i and has attribute $a_i \in \boldsymbol{A}, \boldsymbol{A} = \{a_1, a_2, \ldots, a_m\}$, where m indicates the number of attributes of the deployed sensors in a city. Each attribute represents the type of data such as temperature, traffic volume, and PM2.5. The sensor s_i has synchronized measurements $s_i[t_j]$ $(1 \leq j \leq T)$ over the time domain $\boldsymbol{T} = \langle t_1, t_2, \ldots, t_T \rangle$, where each t_j is a timestamp.

Our goal is to discover sets of attributes which are spatially and temporally correlated measured by sets of sensors in $\boldsymbol{S}$. The correlation among attributes is treated as the correlation among sensors which measure different attributes. The spatial correlation among sensors is evaluated based on the spatial closeness of the sensors. To introduce the concept of the spatial correlation among sensors, we define *spatially connected set* and *spatially connected congeneric set* as follows.

Definition 1 (*Spatially Connected Set*) Given distance threshold η and subset of a sensor set $\boldsymbol{G} \subseteq \boldsymbol{S}$, $\boldsymbol{G}$ is a spatially connected set if for any subset $\boldsymbol{G}'$ of $\boldsymbol{G}$, there are $s \in \boldsymbol{G}'$ and $s' \in \boldsymbol{G} \backslash \boldsymbol{G}'$ that $dist(s, s') \leq \eta$, where $dist(s, s')$ is the geographical distance between s and s'.

Definition 2 (*Spatially Connected Congeneric Set*) Given spatially connected set $\boldsymbol{G}$ and attribute a, $\boldsymbol{G}$ is a spatially connected congeneric set if all sensors in $\boldsymbol{G}$ have the same attribute a. We denote it by $\boldsymbol{G}_a$.

The temporal correlation among sensors is evaluated based on the number of timestamps of sensors whose measurements change similarly. The time domain $\boldsymbol{T}$ typically includes many uninterested intervals in which the measurements have random and small fluctuations. To obtain the meaningful correlations, we only compare timestamps at which measurements of sensors change significantly in $\boldsymbol{T}$. Thus, we define *change rate* and *evolving timestamp*.

Definition 3 (*Change Rate*) Given a sensor s_i and a timestamp t_j, change rate $r_i[t_j]$ of s_i at timestamp t_j is defined as follows:

$$r_i[t_j] = \frac{s_i[t_{j+1}] - s_i[t_j]}{t_{j+1} - t_j}.$$

Definition 4 (*Evolving Timestamp*) Given evolving rate ε and attribute a, Evolving timestamps for a is defined as the timestamps with the top-ε% (absolute) change rate in the whole sensor data with a. Let ε_a be the top-ε% (absolute) change rate of a, timestamp t_j is called positive and negative evolving timestamp if $r_i[t_j] \geq \varepsilon_a$ and $r_i[t_j] \leq -\varepsilon_a$, respectively.

Next, we define *co-evolution* on a spatially connected congeneric set.

Definition 5 (*Co-evolution on Spatially Connected Congeneric Set*) Let $\boldsymbol{G}_a$ be a spatially connected congeneric set for attribute a. Given evolving threshold $\boldsymbol{\Theta}_a = (\theta_a{}^+, \theta_a{}^-)$, timestamp t_j positively co-evolves in regard to $\boldsymbol{\Theta}_a$ if $\forall s_i \in \boldsymbol{G}_a$, $r_i[t_j] \geq \theta^+$, denoted as $t_j \xrightarrow{+} \boldsymbol{\Theta}_a$. As well, timestamp t_j negatively co-evolves as for $\boldsymbol{\Theta}_a$ if $\forall s_i \in \boldsymbol{G}_a$, $r_i[t_j] \leq \theta^-$, which is denoted as $t_j \xrightarrow{-} \boldsymbol{\Theta}_a$. The set of timestamps positively or negatively co-evolving is called co-evolution on $\boldsymbol{G}_a$, denote as $E(\boldsymbol{G}_a) = \{t_j \in \boldsymbol{T} | t_j \xrightarrow{+} \boldsymbol{\Theta}_a \vee t_j \xrightarrow{-} \boldsymbol{\Theta}_a\}$.

The purpose of our research is to discover correlations among different attributes. The correlation among different attributes is defined as a co-evolution among a variety of spatially connected congeneric sets. We are not only interested in the simultaneous correlations but also the correlations with time delay (i.e. the time-delayed correlation). Therefor, we firstly define *simultaneous co-evolution* among attributes, followed by *time-delayed co-evolution* among attributes.

Definition 6 (*Simultaneous Co-evolution among μ Attributes*) Let $*_i$ be a symbol which represents either $+$ or $-$ and $\bar{*}_i$ be an inverse of $*_i$. Let $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$ be μ spatially connected congeneric sets ($a_1 \cdots a_\mu$ are different each other). Given μ evolving thresholds $\boldsymbol{\Theta}_{a_1}, \ldots, \boldsymbol{\Theta}_{a_\mu}$, if $\boldsymbol{G}_{a_1} \cup \cdots \cup \boldsymbol{G}_{a_\mu}$ is a spatially connected set, we call the set of timestamps $C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu} = \{t_j \in E(\boldsymbol{G}_{a_1}) \cap \cdots \cap E(\boldsymbol{G}_{a_\mu}) | (t_j \xrightarrow{*_1} \boldsymbol{\Theta}_{a_1} \wedge \cdots \wedge t_j \xrightarrow{*_\mu} \boldsymbol{\Theta}_{a_\mu}) \vee (t_j \xrightarrow{\bar{*}_1} \boldsymbol{\Theta}_{a_1} \wedge \cdots \wedge t_j \xrightarrow{\bar{*}_\mu} \boldsymbol{\Theta}_{a_\mu})\}$ simultaneous co-evolution among the attributes on $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$.

Definition 7 (*Time-delayed Co-evolution among μ Attributes*) Let $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$ be μ spatially connected congeneric sets. Given μ evolving thresholds $\Theta_{a_1}, \ldots, \Theta_{a_\mu}$ and set of time-delay offsets $\boldsymbol{\tau} = \{\tau_{a_1}, \ldots, \tau_{a_\mu}\}$, if $\boldsymbol{G}_{a_1} \cup \cdots \cup \boldsymbol{G}_{a_\mu}$ is a spatially connected set, we call the set of timestamps $C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu} = \{t_j \in E(\boldsymbol{G}_{a_1}) \cap \cdots \cap E(\boldsymbol{G}_{a_\mu}) | ((t_j + \tau_{a_1}) \xrightarrow{*_1} \Theta_{a_1} \wedge \cdots \wedge (t_j + \tau_{a_\mu}) \xrightarrow{*_\mu} \Theta_{a_\mu}) \vee ((t_j + \tau_{a_1}) \xrightarrow{\bar{*}_1} \Theta_{a_1} \wedge \cdots \wedge (t_j + \tau_{a_\mu}) \xrightarrow{\bar{*}_\mu} \Theta_{a_\mu})\}$ time-delayed co-evolution among the attributes on $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$.

We specify each time-delay offset $\{\tau_{a_1}, \ldots, \tau_{a_\mu}\}$ to each attribute. In Fig. 1a, for instance, $\{\tau_{Rain}, \tau_{NO_2}\} = \{0, 0\}$ which means that the CAP is a simultaneous co-evolution between the rainfall and the NO_2. In Fig. 1b, on the other hand, $\{\tau_{Rain}, \tau_{NO_2}\} = \{+1, 0\}$ which means that the CAP is a time-delayed co-evolution between the rainfall and the NO_2. Note that the definition of the time-delayed co-evolution is equivalent to the definition of the simultaneous co-evolution when all time-delay offsets are equal to zero.

Time-delayed co-evolution among attributes includes any patterns without considering their frequency (i.e., patterns with less number of co-evolving timestamps). If a time-delayed co-evolution among attributes appears frequently, we call it *correlated attribute pattern*.

Definition 8 (*Correlated Attribute Pattern*) Let $C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}$ be a time-delayed co-evolution among $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$. Given minimum support ψ, $C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}$ is a correlated attribute pattern of μ attributes $a_1, \ldots, a_\mu$ on $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$ if $|C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}| \geq \psi$.

Based on the above definitions, we define our problem *CAP mining* as follows.

Problem Definition (CAP Mining) *Given sensor set $\boldsymbol{S}$ over time domain $\boldsymbol{T}$, minimum support ψ, evolving rate ε, distance threshold η, the maximum number of CAP attributes μ, and set of time delay offsets $\boldsymbol{\tau}$, the CAP mining discovers all the correlated attribute patterns which contain two to μ attributes.*

In our previous work [5], we defined the CAP mining as discovering the simultaneous co-evolution among attributes. In this article, we redefine the CAP mining as discovering the time-delayed co-evolution among attributes, which is a more generalized problem than the problem in the previous work.

We here summarize parameters and their impacts to the number of CAPs to be discovered.

- Evolving rate ε: ε is for deciding evolving timestamps of each sensor. If ε is large, many time stamps are evaluated as evolving timestamps and then the number of CAPs likely becomes large. Otherwise, the number of CAPs likely becomes small.
- Distance threshold η: η is for deciding connectivity of sensors. If η is large, many sensors are spatially connected and then the number of CAPs likely becomes large. Otherwise, the number of CAPs likely becomes small.

- The maximum number of CAP attributes μ: μ restricts the number of attributes in CAPs. Thus, if μ is small, the number of CAPs likely becomes small. Otherwise, the number of CAPs likely becomes large.
- The minimum support ψ: ψ is the minimum support for restricting CAPs which are not frequently co-evolving. If ψ is small, many co-evolution among μ attributes becomes CAPs since the number of co-evolving timestamps is larger than ψ. Thus, the number of CAPs likely becomes large. Otherwise, the number of CAPs likely becomes small.

Example We show an example of CAP mining by using Fig. 1. We set five parameters $\varepsilon, \eta, \mu, \boldsymbol{\tau}$, and ψ. We set 70% as ε to reduce effects of small changes of measured values. Small changes such as timestamps 6 and 9 of s_1 and timestamps 10 and 11 of s_2 and s_3 are not included in evolving timestamp (Definition 4). We set 200 km as η because each sensor is distant from each other. s_1, s_2, and s_3 are within 200 km. Thus, s_1, s_2, and s_3 constitute spatially connected set (Definition 1). s_2 and s_3 constitute spatially connected congeneric set G_{NO_2} because s_2 and s_3 have the same attribute NO_2 based on Definition 2. Co-evolution on spatially connected congeneric set $E(G_{\mathrm{NO}_2})$ on s_2 and s_3 includes the set of timestamps, for example timestamps from 1 to 7 (Definition 5). We set two as μ and $\{\tau_{Rain}, \tau_{\mathrm{NO}_2}\} = \{+1, 0\}$ as $\boldsymbol{\tau}$ to find CAPs such that rainfall increases/decreases 1 day after NO_2 increases/decrease (Definition 7).

Since s_1 and s_3 are spatially connected, have different attributes, and often increase and decrease together, the set of attributes $\{rainfall, \mathrm{NO}_2\}$ on the set of sensors $\{s_1, s_3\}$ is found as the CAP (Definition 8). Of course, if we set small ψ as the minimum support, the set of attributes $\{rainfall, \mathrm{NO}_2\}$ on the set of sensors $\{s_1, s_2\}$ is also found as the CAP. We can control the number of found CAPs by configuring the parameters ε, η, μ, and ψ.

3 MISCELA

In this section, we present our CAP mining method Miscela. Firstly, we describe an outline of Miscela. Then we explain the detail of each component of Miscela. After that, we show the algorithm of Miscela, followed by a discussion of the time complexity.

3.1 Outline of Miscela

According to Definition 8, the attributes on a sensor sets are a CAP if (1) the sensors are spatially connected, (2) the sensors contain two to μ attributes, and (3) the cardinality of the co-evolution is larger than ψ. We can naively discover all CAPs by searching all the spatially connected sets within a given sensor set, evaluating the number of attributes, and evaluating the cardinality

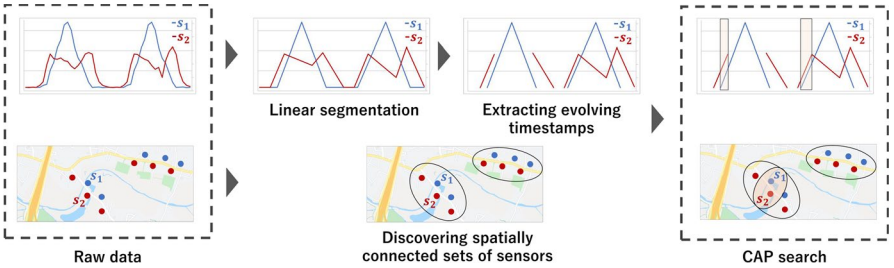


Fig. 2 An overview of workflow of MISCELA

of co-evolution. However, separately conducting the procedures is inefficient because the spatially connected sets often have uninteresting attribute patterns which do not satisfy the CAP conditions; do not contain two to μ attributes (i.e., contains only single or more than μ attributes) and/or the attributes do not co-evolve. Hence, we structurally search only the spatially connected sets whose attribute patterns satisfy the CAP conditions. For this purpose, we take an expansion-based search, which gradually expands a spatially connected set so that the expanded sensor set would also be spatially connected and contain two to μ attributes. In addition, we propose a tree structure, called *CAP search tree*, for effectively expanding the spatially connected set and stopping the expansion. We can avoid evaluating unnecessary attribute patterns which definitely do not satisfy CAP definition by using the CAP search tree.

MISCELA comprises the following four steps.

1. *Linear segmentation* we filter uninteresting data fluctuation by applying a linear segmentation algorithm to time series data.
2. *Extracting evolving timestamps* we extract evolving timestamps in the measurements of all sensors by using given evolving rate ε.
3. *Discovering spatially connected sets of sensors* since CAPs are discovered only from spatially connected sets, we divide a given sensor set into spatially connected sets to restrict the search space.
4. *CAP search* for each spatially connected set, we search for CAPs. We recursively conduct the CAP search with gradually expanding a spatially connected set according to the CAP search tree.

Figure 2 shows the overview of workflow of MISCELA. Given temporal sensor values and sensor locations, MISCELA first takes linear segmentation to filter small fluctuation and then extracts evolving time stamps in which sensor values largely increase/decrease. Next, it discovers spatially connected sets of sensors from sensor locations. After extracting evolving time stamps and spatially connected sets of sensors, MISCELA finds CAPs by CAP search process.

3.2 Linear segmentation

As for the first step, we approximate the time series data to filter uninteresting fluctuations because the time series data often includes noises. To approximate the time series data, we employ a simple and effective linear segmentation algorithm, the bottom-up algorithm in [7]. The bottom-up algorithm first merges successive two measurements to approximate the T-length time series data by $T/2$-length one. That is, we compute $\langle (s[t_1] + s[t_2])/2, (s[t_3] + s[t_4])/2, \ldots, (s[t_{T-1}] + s[t_T])/2 \rangle$. Then, we iteratively merge successive two measurements that have the smallest difference between them among any successive measurements. If the smallest difference is larger than a given threshold, we stop the procedure. The linear segmentation reduces unexpected effects on CAPs caused by small fluctuations. Note that any algorithms can be used in this step instead of the bottom-up algorithm.

3.3 Extracting evolving timestamps

To discover CAPs, we need to extract evolving timestamps. We permit users to specify the evolving rate ε instead of directly specifying the evolving threshold in Definition 4. The detail of the extraction is as follows. First, we calculate the change rate for each time series data of each attribute. Next, we compute the change rate with the top-ε% absolute value as an evolving threshold by which evolving timestamps are extracted. Then, we extract both positive and negative evolving timestamps whose absolute change rate is larger than the evolving thresholds.

3.4 Discovering spatially connected sensors

As a CAP is discovered on a spatially connected set, we divide a given sensor set into spatially connected sets so that we restrict the search space only inside of each maximal spatially connected set. Here, *maximal* means that the spatially connected set is not contained in any larger connected set. In MISCELA, we model the spatially connected sensor sets as graphs. Then, we firstly introduce concepts of *sensor graph*, *connected sub-graph*, and *connected component*.

Definition 9 (*Sensor Graph*) Given sensor set $\boldsymbol{S}$ and distance threshold η, sensor graph $\boldsymbol{G}_S$ is a graph where each vertex in $\boldsymbol{G}_S$ corresponds to a sensor in $\boldsymbol{S}$ and there is an edge between two vertices if their corresponding sensors are located within η.

Definition 10 (*Connected Sub-graph*) Given sensor graph $\boldsymbol{G}_S$, let $\boldsymbol{G}_S{}' = (\boldsymbol{V}', \boldsymbol{E}')$ be a sub-graph of $\boldsymbol{G}_S$. $\boldsymbol{G}_S{}'$ is a connected sub-graph if there is a path in $\boldsymbol{G}_S{}'$ from u to v for every $u, v \in \boldsymbol{V}'$. Let λ be the number of vertices of $\boldsymbol{G}_S{}'$, $\boldsymbol{G}_S{}'$ is called size-λ connected sub-graph.

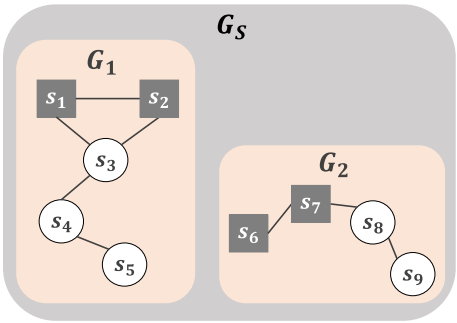


Fig. 3 The sensor graph of the sensor set $\boldsymbol{S}$

Definition 11 (*Connected Component*) Given sensor graph $\boldsymbol{G}_S$, let $\boldsymbol{G}_S{}' = (\boldsymbol{V}', \boldsymbol{E}')$ be a sub-graph of $\boldsymbol{G}_S$. $\boldsymbol{G}_S{}'$ is a connected component if $\boldsymbol{G}_S{}'$ is not contained any larger connected sub-graph in $\boldsymbol{G}_S$.

To discover spatially connected sensors, we construct a sensor graph and find connected components of the sensor graph. For efficiently computing them, we use Dbscan [4]. Dbscan is one of the clustering algorithms, which groups points surrounded with many nearby neighbors. It has two input parameters; distance threshold and MinPts. When we set η as distance threshold and 2 as MinPts, Dbscan can identify edges between s and s' if the distance between two sensors is less than or equal to η. The clustering results of Dbscan can be considered as connected components. Dbscan simultaneously constructs sensor graphs and find connected components.

Example Figure 3 shows an example of sensor graph. Let the square and the circle symbols be sensors and the different symbols mean to measure different attributes. Given sensor set $\boldsymbol{S} = \{s_1, s_2, \ldots, s_9\}$ and distance threshold η, we transform it as the sensor graph $\boldsymbol{G}_S$. There is no edge between s_5 and s_6 because the distance between them is larger than η. We identify two connected components $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ in $\boldsymbol{G}_S$.

3.5 CAP search

We can naively discover all CAPs by searching all connected sub-graphs of each connected component. However, it is quite inefficient because the number of connected sub-graphs exponentially increases as the number of sensors increases. Therefore, we gradually expand a connected sub-graph based on the connectivity. Additionally, it is unnecessary to compute the intersection of evolving timestamps on connected sub-graph which contain only a single attribute or more than μ attributes for the CAP mining because the connected sub-graphs do not have any CAPs. Thus, we expand a connected sub-graph so as to make the explored connected sub-graphs can have CAPs.

For the efficient expansion, we develop a tree structure called the *CAP search tree*. We construct a CAP search tree for every connected component. For each connected component, the CAP search tree effectively organizes all the connected

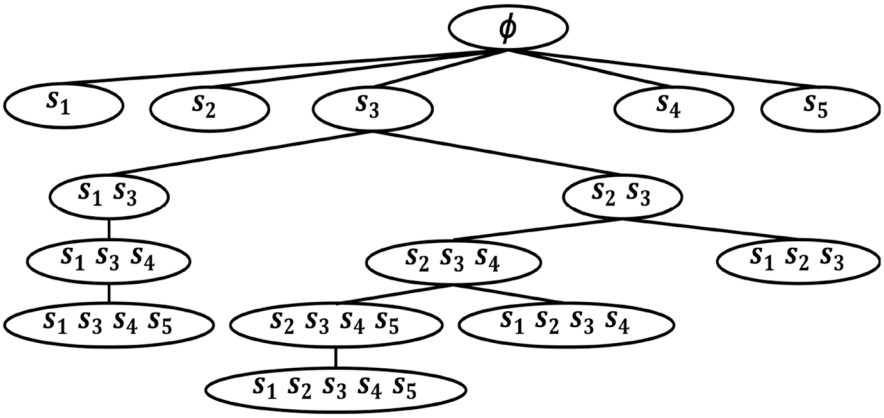


Fig. 4 The CAP search tree of $\boldsymbol{G}_1$ in Fig. 3

sub-graphs which contain at most μ attributes into a tree structure based on the spatial connectivity. Each tree node in the CAP search tree uniquely corresponds to a connected sub-graph of a connected component. In the CAP search tree, we compute the intersection of evolving timestamps only for connected sub-graphs corresponded by tree nodes. If the number of intersections in tree nodes is larger than ψ, the tree nodes contains CAPs. We call the computation of the intersection *CAP computation*. The CAP search tree effectively reduces the computation cost because it reduces the number of CAP computations.

To construct the CAP search tree based on the spatial connectivity, we introduce *parent* relation between two connected sub-graphs.

Definition 12 (*Parent*) Given size-λ connected sub-graph $\boldsymbol{Y}$ in sensor graph $\boldsymbol{G}_S$, vertex ordering ν of $\boldsymbol{G}_S$, and the maximum number of CAP attributes μ. Let s be the first possible vertex in ν satisfying the condition such that $\boldsymbol{X} = \boldsymbol{Y} \backslash \{s\}$ is a connected sub-graph containing less than or equal to μ attributes. At this time, $\boldsymbol{X}$ is called a parent of $\boldsymbol{Y}$.

We use the parent relation to construct a CAP search tree of each connected component. Each tree node in the CAP search tree has one unique parent, and the nodes containing a single sensor are connected the root node ϕ. In short, all the connected sub-graphs in the connected component form a tree structure with the empty set ϕ as the root. To discover all CAPs, we explore all the tree node from the root to the leaves. The construction of the entire tree structure takes large costs. Thus, we do not construct the entire tree structure beforehand for the efficient discovery of CAPs. Instead, we perform depth-first construction from the root node and only visit the tree nodes that have CAPs.

Here, vertex ordering ν is used for deciding parents which affect the order of searching CAPs. Our algorithm can handle any order of vertices and the vertex order does not affect the time complexity.

Example Consider the connected component $\boldsymbol{G}_1$ in Fig. 3. Suppose a vertex ordering $\nu = \{s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5\}$ and the number of maximum attributes$\mu = 2$. Figure 4 shows the CAP search tree for the connected component $\boldsymbol{G}_1$, Each tree node in the CAP search tree corresponds to a connected sub-graph in $\boldsymbol{G}_1$. Any connected sub-graphs in the tree contain two to μ attributes (except for a single sensor). The parent of the connected sub-graph $\{s_2, s_3, s_4\}$ is the set $\{s_2, s_3\}$ because sensor s_4 is the first possible one in ν that ensures the remaining sensors are still connected and it contains 2 attributes. We can discover all CAPs in $\boldsymbol{G}_1$ by exploring all the tree nodes in the CAP search tree.

For any tree node in the CAP search tree, if a node does not have any CAPs, no descendants of the node have CAPs. Hence, we can safely prune the sub-tree rooted at the node by the following theorem.

Theorem 1 *Given spatially connected set* $\boldsymbol{G} = \boldsymbol{G}_{a_1} \cup \cdots \cup \boldsymbol{G}_{a_\mu}$, *where* $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$ *are spatially connected congeneric sets. Let* $\boldsymbol{G}'$ *be a connected sub-graph of* $\boldsymbol{G}$, $\boldsymbol{G}$ *has no CAPs if* $\boldsymbol{G}'$ *has no CAPs.*

Proof Let $C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}$ be a co-evolution among $\boldsymbol{G}_{a_1}, \ldots, \boldsymbol{G}_{a_\mu}$. All the timestamps in $C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}$ is also contained in $C'^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}$ because $\boldsymbol{G}'$ is a subset of $\boldsymbol{G}$. Therefore, $|C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}| < \psi$ if $|C'^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}| < \psi$. Thus, $\boldsymbol{G}$ has no CAPs if $\boldsymbol{G}'$ has no CAPs. □

3.6 Algorithm of Miscela

We can efficiently discover CAPs by using four steps in Miscela. The pseudo-code of our proposal, Miscela, is given in Algorithms 1, 2, and 3. Algorithm 1 contains first, second, and third steps. Miscela applies the bottom-up segmentation algorithm to the given sensor data in order to filter uninteresting fluctuations (lines 1–3). Next, it computes the evolving thresholds for all the attributes (line 5) and extracts evolving timestamps (lines 6–15). Then, it identifies connected components by using Dbscan (line 17). Finally, it conducts the CAP search starting from every size-1 connected sub-graph (lines 18–20).

Algorithm 2 sketches the fourth step, that is the CAP search. Given connected component $\boldsymbol{X}$, the algorithm starts the depth-first search from $\boldsymbol{X}$. First, we select all connected sub-graphs whose parent is $\boldsymbol{X}$ in the CAP search tree (line 1). Second, for each connected sub-graph $\boldsymbol{Y}$, the algorithm conducts the CAP computations (i.e. the intersection of the evolving timestamps) on $\boldsymbol{Y}$ (line 3). If $\boldsymbol{Y}$ contains CAPs, the algorithm outputs the CAPs on $\boldsymbol{Y}$ (line 5). Then, we recursively conduct depth-first search on $\boldsymbol{Y}$ (line 6). If $\boldsymbol{Y}$ does not contain any CAPs, we prune all the subtree rooted at $\boldsymbol{Y}$.

Algorithm 1 MISCELA

Input: sensor set $\boldsymbol{S}$, minimum support ψ, evolving rate ε, distance threshold η, the maximum number of CAP attributes μ, set of time delay offsets $\boldsymbol{\tau}$
Output: All CAPs on $\boldsymbol{S}$

ForEach $s_i \in \boldsymbol{S}$ **do**
  BOTTOM-UP LINEAR SEGMENTATION(s_i)
end for
ForEach attribute a that $\boldsymbol{S}$ contains **do**
  $\Theta_a \leftarrow$ EVOLVING THRESHOLD ESTIMATION($\boldsymbol{G}_a$,ε)
  ForEach $s_i \in \boldsymbol{G}_a$ **do**
    for $j = 0 \rightarrow$ length of $\boldsymbol{T}$ **do**
      if $r_i[t_j] \geq \theta_a^+$ **then**
        t_j is a positive evolving timestamp
      end if
      if $r_i[t_j] \leq \theta_a^-$ **then**
        t_j is a negative evolving timestamp
      end if
    end for
  end for
end for
$\mathbb{G}_S \leftarrow$ DBSCAN($\boldsymbol{S}$,η,$MinPts = 2$)
ForEach $\boldsymbol{G} \in \mathbb{G}_S$ **do**
  ForEach $s \in \boldsymbol{G}$ **do**
    CAP SEARCH($\boldsymbol{G}$,$\{s\}$,ψ,τ)
  end for
end for

Algorithm 2 CAP SEARCH

Input: connected component $\boldsymbol{G}$, connected sub-graph $\boldsymbol{X} \subseteq \boldsymbol{G}$, minimum support ψ, set of time delay offsets $\boldsymbol{\tau}$
Output: CAPs on $\boldsymbol{X}$

$\mathbb{Y} \leftarrow$ the sets of sensors whose *parent* is $\boldsymbol{X}$
ForEach $\boldsymbol{Y} \in \mathbb{Y}$ **do**
  $C_y \leftarrow$ CAP COMPUTATION($\boldsymbol{Y}$,ψ,τ)
  if C_y is not empty **then**
    output C_y
    CAP SEARCH($\boldsymbol{G}$, $\boldsymbol{Y}$,ψ,τ)
  end if
end for

Algorithm 3 CAP COMPUTATION

Input: connected sub-graph $\boldsymbol{Y}$, minimum support ψ, set of time delay offsets $\boldsymbol{\tau}$
Output: CAPs on $\boldsymbol{Y}$

ForEach $s_i \in \boldsymbol{Y}$ **do**
  All the positive and negative evolving timestamps of s_i are shifted to $t_j + \tau_{a_i}$
end for
$C_y \leftarrow \phi$
ForAny combinations of $*_1, \ldots, *_{\tilde{\mu}}$ **do** ▷ $\tilde{\mu}$: the number of attributes included in $\boldsymbol{Y}$
  if $|C^{*_1,\ldots,*_{\tilde{\mu}}}_{a_1,\ldots,a_{\tilde{\mu}}}| \geq \psi$ **then**
    add $C^{*_1,\ldots,*_{\tilde{\mu}}}_{a_1,\ldots,a_{\tilde{\mu}}}$ to C_y
  end if
end for
Output C_y

Algorithm 3 describes the computation process of CAPs. First, the algorithm calculates the time delay by adding the time delay offset to all the positive and negative evolving timestamps of all the sensors in $\boldsymbol{Y}$, where the added time delay offset is selected for each attribute (lines 1–3). Then, a list of CAPs is initialized as an empty set (line 4). After that, the algorithm computes CAP with ψ for each change pattern among different attributes (lines 5–9). Note that since there are some equivalent combinations of $*_1, \ldots, *_{\tilde{\mu}}$ (e.g. $\{*_1 = +, *_2 = -\}$ is equivalent to $\{*_1 = -, *_2 = +\}$), we skip the computations of either combination. Finally, the CAP computation returns all CAPs on $\boldsymbol{Y}$ as the output (line 10).

3.7 Time complexity of Miscela

We analyze the time complexity of MISCELA.

Theorem 2 *Given the number of sensors n, the length of time domain T, the number of connected components $|\mathbb{G}_S|$, the maximum number of CAP attributes μ, the maximum size of sensor sets in tree nodes λ, the height of CAP search tree h, and the average degree of sensor graph d,* MISCELA *incurs time complexity of* $O(nT + n \log n + |\mathbb{G}_S| hd 2^{\lambda} + |\mathbb{G}_S| hd 2^{\mu})$.

Proof Since MISCELA contains four steps, we describe the time complexity of each step. Then, We describe the total time complexity of MISCELA.

(1) The first step of MISCELA is the linear segmentation. The time complexity of this step follows the bottom-up algorithm. Let L be the average number of final segments, the bottom-up algorithm takes $O(LT)$ for n sensors [7]. Since basically L is much smaller than T, the time complexity of first step is $O(nT)$.
(2) The second step of MISCELA is extracting evolving timestamps. This involves two parts: (1) estimating the evolving threshold for each attribute, (2) extracting evolving timestamps for the sensors. Estimating the evolving threshold takes $O(nT)$ because we calculate the change rate for all the timestamps in the whole of time series data. While extracting evolving timestamps takes $O(nT)$ because we compare all the change rate of timestamps to the thresholds. Thus, the time complexity of second step is $O(nT)$.
(3) The third step of MISCELA discovers spatially connected sensor sets. The time complexity of this step follows that of DBSCAN, and thus it takes $O(n \log n)$.
(4) The fourth step of MISCELA is the CAP search. The CAP search is executed for all tree nodes in the CAP search tree. The number of tree nodes is $h \cdot d$ because each tree node has at most d children nodes. For each node, it checks if the sets of sensors have CAPs. It takes $O(2^{\mu})$ because the number of combinations on time-delayed co-evaluations among μ attributes (Definition 7) is 2^{μ}. If the number of sensor sets in tree nodes is at most λ, the number of the combinations is 2^{λ}. Since the CAP search is executed for all $\boldsymbol{G}_{\sigma} \in \mathbb{G}_S$, the time complexity is $O(|\mathbb{G}_S| hd 2^{\lambda} + |\mathbb{G}_S| hd 2^{\mu})$.

Finally, the total time complexity of MISCELA has been calculated by adding all the steps, as $O(nT + n \log n + |\mathbb{G}_S|hd2^{\lambda} + |\mathbb{G}_S|hd2^{\mu})$. □

We here note that a set of time-delay offsets $\boldsymbol{\tau}$ affects the height of CAP search tree h because it affects the number of CAPs.

4 Automatic time-delay search

Time-delayed CAP are helpful to analyze and understand phenomena that affect environments of places close to each other at different time. Although we can find all time-delayed CAPs by setting arbitrary time delays, it is difficult to thoroughly find time-delayed CAPs because users have to repeatedly execute MISCELA with changing the sets of time-delay offsets. Its computational cost is large, so it is better to automatically search for sets of time-delay offsets so that users can find time-delayed CAPs with any sets of time-delayed offsets. We here define our problem of CAP mining with automatic time-delay search.

Problem Definition (CAP Mining with Automatic Time-delay Search) *Given sensor set $\boldsymbol{S}$ over time domain $\boldsymbol{T}$, minimum support ψ, evolving rate ε, distance threshold η, the maximum number of CAP attributes μ, and the maximum time-delay offset τ_M, the CAP mining with automatic time-delay search discovers all the correlated attribute patterns which contain two to μ attributes with any patterns $\boldsymbol{Y}$ of the set of time-delay offsets whose values are between $-\tau_M$ and τ_M.*

Given the maximum time-delay offset τ_M, the number of patterns of the sets of time-delayed offsets is $(2\tau_M + 1)^m$ (recall that m is the number of attributes), so a naive approach takes a significantly large time. That is, it takes $(2\tau_M + 1)^m$ multiplied by the average execution time of MISCELA. Therefore, we need an efficient search method for the sets of time-delay offsets that generate time-delayed CAPs.

We develop MISCELA with automatic time-delay search based on apriori manner [1]. Our method executes MISCELA for all patterns of two attributes and sets of time-delayed offsets, and then combines these attributes and the sets of time-delayed offsets if these parameters generate the time-delayed CAPs. The apriori manner can prune unnecessary parameters of attributes and time-delay offsets that do not generate the time-delayed CAPs. Furthermore, we initially reduce the number of sets of time-delayed offsets by pruning the sets that generate the same time-delay CAPs. For example, the sets of time delay offsets $\{0, 0\}$ and $\{-1, -1\}$, and $\{1, 0\}$ and $\{0, -1\}$ generate the same results, respectively, because they extract the same co-evolving timestamps. We prune set of time-delayed offsets $\{\tau_0, \ldots, \tau_j\}$ if $\{\tau_0 + i, \ldots, \tau_j + i\} \in \boldsymbol{Y} \backslash \{\tau_0, \ldots, \tau_j\}$ for $i = -\tau_M, \ldots, \tau_M$. After pruning the sets of time-delay offsets, the number of patterns is $\tau_M^i - (\tau_M - 1)^i$ where i is the number of attributes in the candidates.

The pseudo-code of MISCELA with automatic time-delay search, is given in Algorithm 4. Algorithm 4 first initializes candidates of input parameters (lines 1–9). Next, it conducts MISCELA for all candidates, and then generates new candidates

based on CAPs found by MISCELA (lines 10–19). Then, we generate new candidates that includes $i+1$ attributes (line 20). If the new candidates are empty, it terminates its process (lines 21–23)

Algorithm 4 MISCELA with automatic time-deay search

Input: sensor set $\boldsymbol{S}$, minimum support ψ, evolving rate ε, distance threshold η, the maximum number of CAP attributes μ, the maximum time delay offset τ_M
Output: All CAPs with any sets of time-delay offsets on $\boldsymbol{S}$

```
1: A2, Υ2, C2 ← φ
2: A2 ← any pair of attributes
3: Υ2 ← any pair of time-delay offsets between −τM and τM
4: Delete the sets of time-delay offsets that generate duplicate CAPs from Υ2
5: ForEach τ ∈ Υ2 do
6:     ForEach A ∈ A2 do
7:         C2 ← C2 ∪ {A, τ}
8:     end for
9: end for
10: for i = 2 → μ do
11:     Ci+1, C' ← φ
12:     ForEach c ∈ Ci do
13:         τ ← the set of time-delay offsets in c
14:         S' ← sensors that have attributes in c
15:         MISCELA(S', ψ, ε, η, i, τ)
16:         if find CAPs then
17:             Ci ← Ci ∪ c
18:         end if
19:     end for
20:     Combines any pair of Ci that includes i + 1 attributes to generate Ci+1
21:     if Ci+1 = φ then
22:         break.
23:     end if
24: end for
```

Example Let us assume that we have four attributes $\{a_1, a_2, a_3, a_4\}$ and the maximum time-delay offset τ_M is one. There are six patterns of two attributes; $\{a_1, a_2\}$, $\{a_1, a_3\}$, $\{a_1, a_4\}$, $\{a_2, a_3\}$, $\{a_2, a_4\}$, $\{a_3, a_4\}$, and nine sets of two time-delay offsets; $\{-1, -1\}$, $\{-1, 0\}$, $\{-1, 1\}$, $\{0, -1\}$, $\{0, 0\}$, $\{0, 1\}$, $\{1, -1\}$, $\{1, 0\}$, and $\{1, 1\}$. We delete the sets of time-delay offsets that generate the same CAPs. For example, we delete $\{-1, -1\}$ and $\{1, 1\}$ because they generate the same CAPs with $\{0, 0\}$. After deleting unnecessary sets of time-delay offsets, we have five sets of time-delay offsets; $\{-1, 1\}, \{0, -1\}, \{0, 0\}, \{0, 1\}, \{1, -1\}$. Then, we generate the initial candidates, for example, $\{\{a_1, a_2\}, \{0, 0\}\}$, $\{\{a_1, a_2\}, \{1, 0\}\}$, and $\{\{a_1, a_3\}, \{0, 0\}\}$. We execute MISCELA for all the candidates, and then we can output all time-delayed CAPs with two attributes. We generate new candidates that include three attributes. For example, if we find CAPs for $\{\{a_1, a_2\}, \{0, -1\}\}$ and $\{\{a_2, a_3\}, \{-1, 1\}\}$ that are included two attributes, we generate $\{\{a_1, a_2, a_3\}, \{0, -1, 1\}\}$ as new candidates that are included three attributes. We then execute MISCELA for the new candidates. We repeat these procedures until no candidates are generated or the number of attributes exceed μ.

We describe the time complexity of MISCELA with automatic time-delay search. The worst case of its computation cost is $O((\tau_M{}^{\mu} - (\tau_M - 1)^{\mu})(nT + n \log n + |\mathbb{G}_S| hd 2^{\lambda} + |\mathbb{G}_S| hd 2^{\mu})$, which is the case in which it cannot prune any patterns (i.e., repeating executions of MISCELA

$({\tau_M}^{\mu} - (\tau_M - 1)^{\mu})$ times). It is the same computation cost with the naive method. The computation cost is significantly large theoretically, but empirically MISCELA with automatic time-delay search efficiently computes CAPs by pruning a large number of candidates.

5 Experiments

In this section, we evaluate the efficiency of MISCELA and the usefulness of the CAP mining. To the best of our knowledge, no existing methods can directly apply the CAP mining when a given sensor set contains multiple attributes. Hence, we compare MISCELA with an SCP mining method *Assembler* [17] which is the state-of-the-art algorithm for the SCP mining. Assembler discovers SCPs which contains all necessary sensor sets for the CAP mining but also contains unnecessary ones. Then, we filter the SCPs to remove unnecessary sensor sets during the search process. Since the difference between MISCELA and Assembler is their CAP search, we compare the response time of the CAP search of MISCELA with that of Assembler by changing the following parameters:

- The maximum number of CAP attributes μ,
- The evolving rate ε,
- The minimum support ψ,
- The set of time-delay offsets $\boldsymbol{\tau}$.

We additionally evaluate the efficiency of automatic searches.

The algorithms of MISCELA and Assembler are implemented in Python. The experiments are conducted on a computer with Intel Xenon E7-8860v4 2.20GHz CPU.

5.1 Experimental setup

Our experiments use three real sensor datasets; (1) 5 attributes daily sensor data collected in Santander, Spain from March 1, 2016 to September 30, 2016, (2) 6 attributes daily sensor data collected in China from September 1, 2016 to August 31, 2018, and (3) 13 attributes daily sensor data collected in China from September 1, 2016 to August 31, 2018. We obtained the Santander dataset from *FESTIVAL*[1]. and the two China datasets from emphenvicloud.cn.[2]. Table 2 shows the number of sensors, the number of timestamps, and the type of attributes in three datasets. We set distance threshold η as 80 m for the Santander dataset and as 200 km for the China datasets, respectively. We chose these distance thresholds so as to divide sensors

[1] http://www.festival-project.eu/.
[2] http://www.envicloud.cn/.

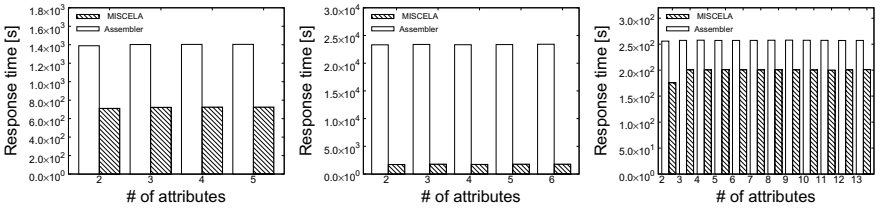
Table 2 The detail of datasets

Name	# of timestamps	Attribute	# of sensors
Santander	5136	Temperature	297
		Light	181
		Noise	32
		Traffic volume	31
		Humidity	10
$China_6$	730	PM2.5	1573
		PM10	1573
		SO_2	1573
		NO_2	1573
		CO	1573
		O_3	1573
$China_{13}$	730	PM2.5	370
		PM10	370
		SO_2	370
		NO_2	370
		CO	370
		O_3	370
		Sunny-percent	370
		Rainy-percent	370
		Rainfall	370
		Temperature	370
		Air pressure	370
		Humidity	370
		Wind speed	370

in each dataset into around 20 connected components. For all the datasets, we use $\varepsilon = 50\%$, $\mu = 2$, $\psi = 500$, and $\boldsymbol{\tau} = \{0, \ldots, 0\}$ (i.e. no time delay) as default parameters. Vertex ordering ν is calculated based on sensor identifiers which are assigned in the datasets (i.e., almost random order).

5.2 Efficiency of Miscela

In this section, we evaluate the efficiency of Miscela. We show the response time of Miscela and Assembler by changing three parameters μ, ε, and ψ. In addition, we show the response time of Miscela by changing $\boldsymbol{\tau}$. In summery, Miscela is always faster than Assembler throughout the experiments. Miscela is up to 20.84 times faster than Assembler. The results of the experiments for the time-delayed setting show that the response time of Miscela does not depend on $\boldsymbol{\tau}$ but depends on the number of found CAPs.



(a) Santander (b) China_6 (c) China_{13}

Fig. 5 The maximum number of CAP attributes μ

Table 3 # of CAP computations varying μ in Santander

μ	2	3	4	5
MISCELAis	423,958	424,510	424,510	424,510
Assembler	813,824	813,824	813,824	813,824

5.2.1 The maximum number of CAP attributes μ

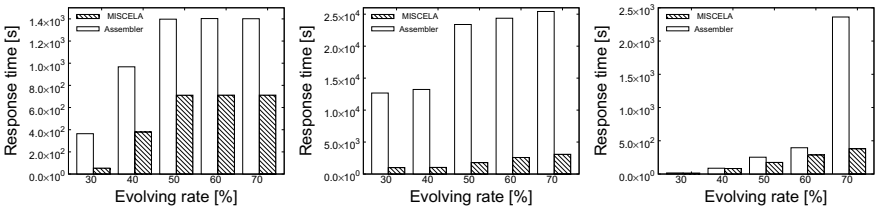
We describe the experimental results by changing the maximum number of CAP attributes μ. We vary μ from 2 to the number of attributes in each dataset (e.g., from 2 to 5 for the Santander dataset). Figure 5 shows the result of response time in the three datasets. We observe that MISCELA is always faster than Assembler in all the datasets for any μ. Moreover, the result of China_{13} dataset indicates that MISCELA is more efficient with smaller μ. This is because MISCELA more frequently skips processing the connected sub-graphs which have no CAPs when we set smaller μ.

From the viewpoint of the time complexity of the CAP search (see Sect. 3.7), the computation cost of the CAP search increases as μ increases. Table 3 shows the number of CAP computations for Santander dataset with varying μ. We confirm that the time complexity of CAP search depends on μ.

5.2.2 Evolving rate ε

We describe the experimental result by changing the evolving rate ε. We measure the response time of both methods with $\varepsilon = 30\%, 40\%, 50\%, 60\%$, and 70%. Figure 6 shows that MISCELAis is faster than Assembler in all the datasets for any ε. The response time increases as the evolving rate ε increases. The setting of the large ε makes the number of extracted evolving timestamps to be large. As a result of this, the height of CAP search tree h becomes large, which makes the computation cost of the CAP search large, because more CAPs are discovered with a larger number of evolving timestamps.

Table 4 shows the number of CAP computations for Santander dataset with varying ε. We observe the number of CAP computations increases as ε increases, and this result confirms the correctness of the time complexity of the CAP search given by Theorem 2.

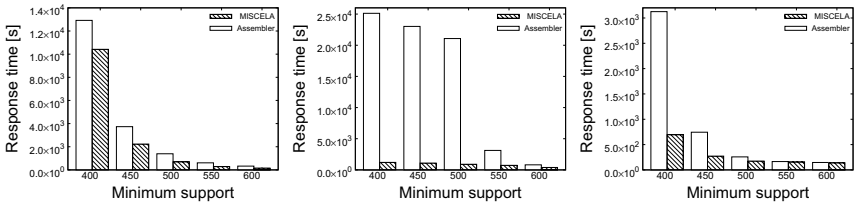


(a) Santander (b) China$_6$ (c) China$_{13}$

Fig. 6 Evolving rate ε

Table 4 # of CAP computations varying ε in Santander

ε	30	40	50	60	70
MISCELA	32,191	230,950	423,958	423,989	423,989
Assembler	229,853	576,878	813,824	813,921	813,921



(a) Santander (b) China$_6$ (c) China$_{13}$

Fig. 7 Minimum support ψ

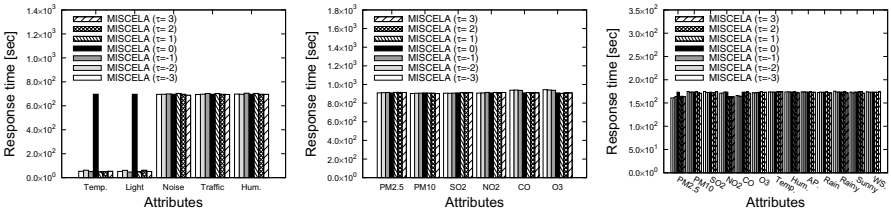
5.2.3 Minimum support ψ

We describe the experimental result by changing the minimum support ψ. We measure the response time of both methods with $\psi = 400$, 450, 500, 550, and 600. Figure 7 shows the result of the response time. It shows that MISCELA is faster than Assembler in all the cases. In particular, the response time of MISCELA is 20.84 times smaller compared with that of Assembler in China$_6$ dataset with $\psi = 400$. We can see in the result that the response time of CAP search increases with decreasing the minimum support. This is because the CAP search with smaller minimum support discovers more CAPs, which makes the computation cost of the CAP search larger because CAP search trees become high.

Table 5 shows the number of CAP computations for Santander dataset with varying ψ. We observe the number of CAP computations decrease as ψ. This directly

Table 5 # of CAP computations varying ψ in Santander

ψ	400	450	500	550	600
MISCELA	6,546,244	1,412,515	423,958	162,822	84,216
Assembler	8,090,479	2,305,839	813,824	334,223	165,449



(a) Santander (b) China$_6$ (c) China$_{13}$

Fig. 8 Time delay offset τ

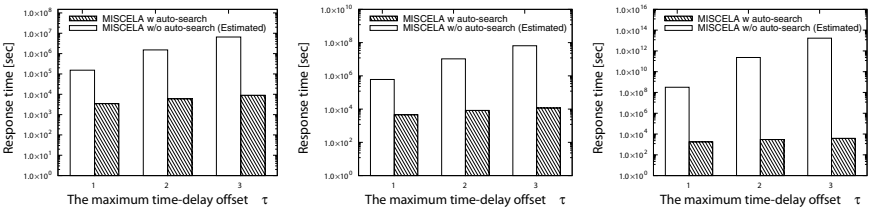
Table 6 # of found CAPs with varying τ for each attribute in Santander

	Temperature	Light	Noise	Traffic volume	Humidity
$\tau = -1$	1423	1403	27,961	27,966	27,971
$\tau = 0$	27,971				
$\tau = +1$	1399	1423	27,960	27,971	27,971

indicates that the smaller minimum support causes the smaller computation cost of the CAP search.

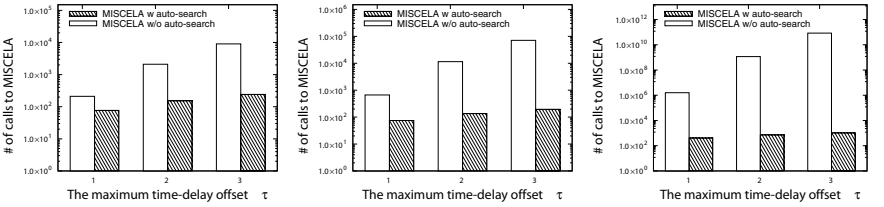
5.2.4 Time delay offset τ

We describe the experimental result by changing the set of time delay offsets τ. We measure the response time of MISCELA with $\tau_a = -1, 0,$ and $+1$ for each attribute. More concretely, we set -3 to $+3$ to a single attribute and set zero to the other attributes. Figure 8 shows the response time of MISCELA depends on the number of founded CAPs, so it does not change unless the number of found CAPs becomes large. We can see in the result of Santander that the response time of MISCELA is very small when $\tau_{Temp.} \neq 0$ compared with $\tau_{Temp.} = 0$. As shown in Table 6, this is because the number of found CAPs is 1423 and 1399 when we set $\tau_{Temp.} = -1$ and $+1$, respectively, while the number of found CAPs is 27,971 when $\tau_{Temp.} = 0$. In the other experiments, we observe that the response times of MISCELA are almost same when we set any time delay offset τ because the numbers of found CAPs are also almost same. We confirm from these result that the response time of MISCELA does not change much for the time delay unless the number of found CAPs significantly changes.



(a) Santander (b) China$_6$ (c) China$_{13}$

Fig. 9 Response time of MISCELA with automatic time-delay search varying the maximum time-delay offset τ_M



(a) Santander (b) China$_6$ (c) China$_{13}$

Fig. 10 The number of executions of MISCELA varying the maximum time-delay offset τ_M

5.3 Efficiency of Miscela with automatic time-delay search

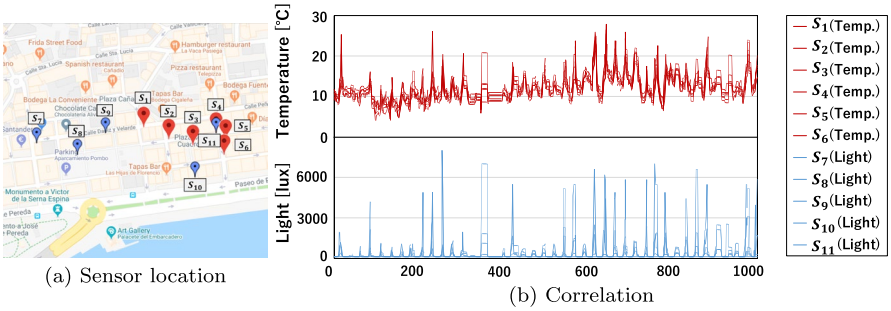
We evaluate the efficiency of MISCELA extended with automatic time-delay search. Figure 9 shows response time of MISCELA with automatic time-delay search and the naive approach, varying with the maximum time-delay offsets. In the response time of the naive approach, we show the estimated response time since it did not finish over 24 h. The estimated response time is computed as the average response time of MISCELA times $(2\tau_M + 1)^\mu - (2\tau_M)^\mu$.

From this result, we can see that our algorithm is much more efficient than the naive approach. In particular, as μ increases, the response time of our algorithm does not increase exponentially. This is because our algorithm can effectively prune sets of time-delay offsets that do not generate time-delayed CAPs. Figure 10 shows the number of executions of MISCELA varying with the maximum time-delay offset τ_M. We can see that the automatic time-delay search drastically reduces the number of executions of MISCELA.

In summery, MISCELA with automatic time-delay search enables us to efficiently discover all time-delay CAPs with any time delays.

5.4 Examples of meaningful CAPs

We here show meaningful CAPs that are found in our experiments. We describe three examples; (1) simultaneous CAPs in Santander, (2) simultaneous and



(a) Sensor location (b) Correlation

Fig. 11 The CAP in Santander

time-delayed CAPs in China, and (3) the difference between the amounts of simultaneous and time-delayed CAPs in China.

5.4.1 Simultaneous CAPs in Santander

We first show a CAP between temperature and light in Santander. Figure 11 shows the locations of the sensors and the measurements of the sensors. The CAP is represented as $C^{+_1,+_2}_{a_1,a_2}$, where $a_1 = temperature$ and $a_2 = light$, $\boldsymbol{G}_{a_1} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ and $\boldsymbol{G}_{a_2} = \{s_7, s_8, s_9, s_{10}, s_{11}\}$. The light and temperature sensors are spatially close and their measurements increase/decrease simultaneously, which means there is no time delay between the changes of the values of temperature and light on these sensors. These sensors are located in a downtown area, and this pattern indicates that the downtown receives a lot of sunshine during the day time. Conversely, if some sensors that are not found in CAPs, this indicates that these sensors are located in the shade area.

5.4.2 Simultaneous and time-delayed CAPs in China

Next, we show a CAP between rainfall and NO_2 in $China_{13}$ dataset, which is described in Sect. 1 (see Fig. 1).

The CAP for s_1 and s_2 is a simultaneous CAP and a negative correlation pattern [i.e., if the measurement value of s_1 increases (resp. decreases), the measurement value of s_2 decreases (resp. increases)]. This indicates that the volume of NO_2 decreases/increases while the amount of rainfall increases/decreases. This is reasonable because meteorological particles are affected by environmental conditions (e.g., rain and wind) at the same location.

On the other hand, the CAP for s_1 and s_3 is a time-delayed CAP and a negative correlation pattern. This indicates that the volume of NO_2 decreases/increases a day later the amount of rainfall increases/decreases. Besides, we did not find any CAPs between the NO_2 and rainfall sensors located at the same location with 1 day time delay. These indicate that the rain affects the reduction of NO_2 at the location 80 km east 1 day later. This is an interesting result because our

Table 7 # of CAPs in China that the PM2.5 sensors correlate with the other attributes sensors in cases of $\tau_{PM2.5} = 0$ and $\tau_{PM2.5} = +1$

	PM10	SO_2	NO_2	CO	O_3	Temp.	Hum.	AP	Rain	Rainy-%	Sunny-%	WS
PM2.5 ($\tau = 0$)	1	511	165	999	89	0	0	0	0	0	0	0
PM2.5 ($\tau = +1$)	0	3	21	3	0	0	0	0	0	0	0	0

algorithm found the phenomena based on data science without using physical simulations or other meteorological techniques and knowledge.

5.4.3 The difference between amounts of simultaneous and time-delayed CAPs in China

We finally show the amounts of simultaneous and time-delayed CAPs that PM2.5 correlates with the other attributes in $China_{13}$ dataset. Table 7 shows the number of CAPs in cases of $\tau_{PM2.5} = 0$ and $\tau_{PM2.5} = +1$ and $\tau_a = 0$ for the other attributes a. We observe that the PM2.5 are often correlated with the other attributes when $\tau_{PM2.5} = 0$ (e.g. there are 999 CAPs among PM2.5 and CO) but did not correlate with the other meteorological sensors. Since meteorological data does not co-evolve with PM2.5, the volume of PM2.5 does not highly depend on meteorology such as weather. We here have to note that *no or rare CAPs* are also meaningful for environmental studies. In case of $\tau_{PM2.5} = +1$, the number of CAPs becomes much small. This indicates that the volume of particles typically increase/decrease simultaneously. Specially, there is no CAPs among PM2.5 and O_3 on $\tau_{PM2.5} = +1$ even though there are many CAPs on $\tau_{PM2.5} = 0$. Some attributes related to air quality (e.g. SO_2, NO_2, and CO) correlate with PM2.5 when $\tau_{PM2.5} = +1$, that is, the amounts of particles increases/decreases 1 day later after the value of the PM2.5 increases/decreases like the CAP in Sect. 5.4.2.

6 Related work

The CAP mining is one of the pattern mining tasks which aim to extract similar and frequent patterns in the time series data. We review two similar tasks; motif discovery and co-evolving mining task.

Motif discovery in time series data extracts a pair of subsequences whose distance is smaller than a given threshold δ. The subsequence is called motif. Lonardi and Patel [8] first introduced the top-K motif discovery task that discovers the K subsequences that have the largest numbers of matches among time series data. Chiu et al. [3] developed an algorithm that discovers approximate motifs in linear time. Mueen et al. [11] proposed an algorithm that efficiently discovers exact motifs with the linear ordering heuristic and the early abandoning strategy. Motif discovery in multi-dimensional time series has also been studied. Tanaka et al. [14, 15] proposed

an algorithm that transforms multi-dimensional time series data into a sequence of symbols using principal component analysis. Minnen et al. [10] studied the problem of mining sub-dimensional motifs that across only a subset of the dimension. These techniques are not suitable for the CAP mining because they do not consider the locations of sensors and the various patterns of attributes.

Co-evolving mining task aims at discovering sets of sensors whose measurements co-evolve frequently. Trasarti et al. [16] studied the problem of discovering regions which show a similar deviation of population density by using mobile phone data. Their method extracts vertical changes by calculating the same hour of different days. In contrast, CAP search extracts horizontal frequent changes in different sensors. Matsubara et al. [9] proposed a spatially co-evolving framework FUNNEL to discover both the county-level and the state-level properties of different diseases. However, it is designed specifically for epidemic data instead of general urban sensor data. Zhang et al. [17] proposed a problem called the *SCP mining*, which aims to discover sensors which are spatially close each other and frequently co-evolving in their measurements. They proposed an efficient algorithm called *Assembler* which is the state-of-the-art of the SCP mining. Although Assembler can discover CAPs, there is large redundancy in their processing because they extract unnecessary correlated sensors that do not have CAPs. Here, to define the correlation among time series sensor data, they adopted the co-evolution instead of the standard Pearson correlation. The Pearson correlation is only for two variables while the co-evolution can be used for multiple variables. Since the SCP mining aims to find the correlated patterns among multiple sensors, the co-evolution is more suitable compared to the Pearson correlation. The CAP mining also targets multiple variables, we also adopt the co-evolution to define the correlation. Cheng et al. [2] studied discovering dynamic co-evolving zones in time series data. They proposed the divide-and-conquer strategy to discover the relationship between the co-evolving zones of the different time period. Hassani et al. [6] proposed a method for constructing physical clusters of sensor nodes based on both spatial and measurement similarities to make groups which record similar measurement over a time period. These algorithms do not target the CAP mining. We show that MISCELA is more efficient than Assembler which is the state-of-the-art for the SCP mining.

7 Conclusion

In this paper, we introduced a problem called correlated attribute pattern (CAP) mining, which discovers the correlated patterns among attributes in multi-attributes sensor set. We are motivated to discover not only the simultaneous correlated patterns among attributes but also the correlated patterns with time delay. Hence, we defined the CAP mining as discovering time-delayed correlated patterns among attributes. We proposed a efficient method, MISCELA, for the CAP mining. MISCELA effectively prunes the unnecessary computations for the CAP mining. Furthermore, we developed MISCELA with automatic time-delay search to efficiently find all CAPs with any time delay. We conducted experiments using three real sensor datasets. The experiments proved that MISCELA can more efficiently discover CAPs in multi-attribute

sensor sets compared with the state-of-the-art the SCP mining method. The results showed that the CAP mining obtains several meaningful patterns.

There are several major directions for further investigation. First, CAPs highly depend on parameters, so users may repeat MISCELA several times to discover meaningful CAPs. Thus, it is interesting to re-use prior results or compute reusable intermediate results. Second, CAPs can extract regional characteristics in certain areas. For example, two attributes do not co-evolve in an area though the two attributes often co-evolve in other areas. This indicates CAPs can be useful to discover outliers of areas. Third, we discover CAPs only for spatially close sensors, but attributes on distinct sensors may often co-evolve. Thus, we would like to extend the CAP mining for discovering co-evolution among attributes on distinct sensors.

Acknowledgements This work was supported by JSPS KAKENHI Grant Numbers JP15K21069, JP16H01722, and 20H00584.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB, pp. 487–499 (1994)
2. Cheng, Y., Li, X., Li, Y.: Finding dynamic co-evolving zones in spatial-temporal time series data. In: Proceedings of the ECML PKDD, pp. 129–144 (2016)
3. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: Proceedings of the ACM SIGKDD, pp. 493–498 (2003)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the ACM SIGKDD, pp. 226–231 (1996)
5. Harada, K., Sasaki, Y., Onizuka, M.: Miscela: Discovering correlated attribute patterns in time series sensor data. In: Proceedings of the IEEE MDM, pp. 72–80 (2019)
6. Hassani, M., Müller, E., Spaus, P., Faqolli, A., Palpanas, T., Seidl, T.: Self-organizing energy aware clustering of nodes in sensor networks using relevant attributes (2010)
7. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings of the IEEE ICDM, pp. 289–296 (2001)
8. Lonardi, J., Patel, P.: Finding motifs in time series. In: Proceedings of the ACM SIGKDD, pp. 53–68 (2002)
9. Matsubara, Y., Sakurai, Y., Van Panhuis, W.G., Faloutsos, C.: FUNNEL: automatic mining of spatially coevolving epidemics. In: Proceedings of the ACM SIGKDD, pp. 105–114 (2014)
10. Minnen, D., Isbell, C., Essa, I., Starner, T.: Detecting subdimensional motifs: an efficient algorithm for generalized multivariate pattern discovery. In: Proceedings of the IEEE ICDM, pp. 601–606 (2007)
11. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, B.: Exact discovery of time series motifs. In: Proceedings of the SDM, pp. 473–484 (2009)

12. Sanchez, L., Muñoz, L., Galache, J.A., Sotres, P., Santana, J.R., Gutierrez, V., Ramdhany, R., Gluhak, A., Krco, S., Theodoridis, E.: Smartsantander: IoT experimentation over a smart city testbed. Comput. Netw. **61**, 217–238 (2014)
13. Sasaki, Y., Ishikawa, Y., Fujiwara, Y., Onizuka, M.: Sequenced route query with semantic hierarchy. In: EDBT, pp. 37–48 (2018)
14. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time-series motif from multi-dimensional data based on MDL principle. Mach. Learn. **58**(2), 269–300 (2005)
15. Tanaka, Y., Uehara, K.: Discover motifs in multi-dimensional time-series using the principal component analysis and the MDL principle. In: Proceedings of the Springer MLDM, pp. 252–265 (2003)
16. Trasarti, R., Olteanu-Raimond, A.M., Nanni, M., Couronné, T., Furletti, B., Giannotti, F., Smoreda, Z., Ziemlicki, C.: Discovering urban and country dynamics from mobile phone data with spatial correlation patterns. Telecommun. Policy **39**(3–4), 347–362 (2015)
17. Zhang, C., Zheng, Y., Ma, X., Han, J.: Assembler: efficient discovery of spatial co-evolving patterns in massive geo-sensory data. In: Proceedings of the ACM SIGKDD, pp. 1415–1424 (2015)