MISCELA: Discovering Correlated Attribute Patterns in Time Series Sensor Data

Kei Harada, Yuya Sasaki, Makoto Onizuka
Graduate School of Information Science and Technology, Osaka University, Japan
{harada.kei, sasaki, onizuka}@ist.osaka-u.ac.jp

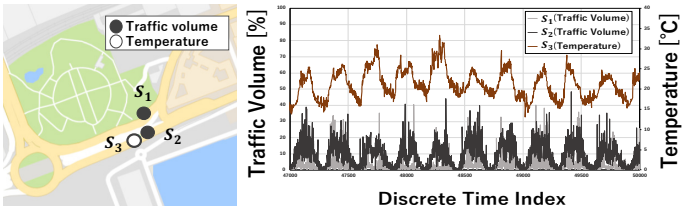
***Abstract*—The urban condition is monitored by a wide variety of sensors with several attributes such as temperature and traffic volume. It is expected to discover the correlated attributes to accurately analyze and understand the urban condition. Several mining techniques for spatio-temporal data have been proposed for discovering the sets of sensors that are spatially close to each other and temporally correlated in their measurements. However, they cannot discover correlated attributes efficiently because their targets are correlated sensors with a single attribute. In this paper, we introduce a problem of discovering correlations among multiple attributes, which we call *correlated attribute pattern (CAP) mining*. Although the existing spatio-temporal data mining methods can be extended to discover CAPs, they are inefficient because they extract unnecessary correlated sensors that do not have CAPs. Therefore, we propose a CAP mining method *MISCELA* to efficiently discover CAPs. In MISCELA, we develop a new tree structure called *CAP search tree*, by which we can effectively prune the unnecessary patterns for the CAP mining. Our experiments using real sensor datasets show that the response time of MISCELA is up to 79 % faster compared to the state-of-the-art.**

***Index Terms*—Spatio-temporal data mining, Smart city, Co-evolving pattern**

I. 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 the traffic volume, and the change of the temperature. Municipalities analyze the urban conditions and make a decision for the urban planning by using such sensor data. For example, Santander, Spain monitors the traffic volumes within the city and informs people of the real-time traffic information [1]. The accumulated traffic data are used to several urban management such as the traffic prediction, the road extension, and the traffic signal control. In these services, it is useful to discover the sets of roads which are spatially close and whose traffic volumes increase or decrease 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 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], [3].

Although the SCP mining discovers meaningful patterns for analyzing the urban environments, it assumes sensors with a single attribute. Many cities typically monitor multiple attributes such as the temperature and the traffic volume to analyze the urban environments from diversified viewpoints. To accurately analyze and understand the urban environment, it is expected to discover *correlated attributes* which are spatially close to each other and temporally co-evolving. However, the SCP mining does not discover correlated attributes because its target is a single attribute.



(a) Sensor location (b) Correlation

Fig. 1: The correlation between traffic volume and temperature in Santander

Example : Figure 1 shows an example of the SCP mining in Santander. There are two traffic sensors s_1 and s_2, and a temperature sensor s_3. These sensors are spatially close to each other, and the measurements of them co-evolve frequently. The SCP mining can discover a set of s_1 and s_2 as correlated sensors. However, it does not discover that traffic volume and temperature are correlated. Municipalities can more accurately understand the traffic behavior by using the data of both the traffic volume and the temperature in the area.

The correlated attributes are computed from multiple attributes and thus provide more diversified knowledge than the patterns from the SCP mining. City managers can obtain more helpful and interesting knowledge in an urban environment from the correlated attributes. Thus, it would contribute to advanced urban management.

In this paper, we introduce a problem of discovering correlations among different attributes, which is called the *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 extend the existing SCP mining methods to discover CAPs, they are inefficient because they extract unnecessary correlated sensors that do not have CAPs. 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. We develop a novel data structure called the *CAP search tree* to facilitate efficient CAP mining. We

can effectively prune the unnecessary patterns for the CAP mining by using the CAP search tree. We conduct experiments with three real sensor datasets in Santander and China. Our experiments show that MISCELA reduces the response time by up to 79% compared to that of the state-of-the-art SCP mining method [2]. In the experiments, the CAP mining discovers meaningful CAPs such as Figure 1.

The main contributions of this paper are as follows.

- We introduce a new problem, CAP mining, that aims to discover correlations among different attributes. It contributes to the advanced urban management.
- 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 can efficiently discover all CAPs by reducing unnecessary candidates generation by the CAP search tree.
- We conduct experiments with three real sensor datasets. The results demonstrate that MISCELA more efficiently discovers CAPs than the state-of-the-art.

The rest of paper is organized as follows. We formulate the CAP mining in Section II and propose the novel CAP mining method MISCELA in Section III. We conduct the CAP mining experiments in Section IV. After that, we summarize the past typical work related to our work in Section V, followed by the conclusion in Section VI.

II. Problem Description

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 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 in $\boldsymbol{S}$. In practice, the correlation among attributes is evaluated as the correlation among sensors which measure different attributes. The spatial correlation among sensors is evaluated based on the spatial closeness of sensors. Hence, we define a *spatially connected set* and a *spatially connected congeneric set* as follows.

Definition 1 (Spatial 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 $\boldsymbol{G} \subseteq \boldsymbol{S}$ and attribute a, if $\forall s \in \boldsymbol{G}$ have the same attribute a, it is called spatially connected congeneric set, which is denoted as $\boldsymbol{G}_a$.

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

Definition 3 (Change Rate): Given sensor s_i and timestamp t_j, the change rate $r_i[t_j]$ of s_i at the timestamp t_j is

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

Definition 4 (Evolving Timestamp): Given evolving rate ε and attribute a, timestamps with the top-ε% (absolute) change rate in the whole sensor data with a are the evolving timestamps for a. Let $\Theta_a = ({\theta_a}^+, {\theta_a}^-)$ be the evolving threshold for a, timestamp t_j is called the positive and negative evolving timestamp if $r_i[t_j] \geq {\theta_a}^+$ and $r_i[t_j] \leq {\theta_a}^-$, respectively.

Next, we define a *co-evolution* of a spatially connected congeneric set and a *co-evolution among μ attributes*.

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

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

If a co-evolution among attributes appears frequently, we call it the *correlated attribute pattern*.

Definition 7 (Correlated Attribute Pattern): Let $C^{*_1,\ldots,*_\mu}_{a_1,\ldots,a_\mu}$ be a 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, and the maximum number of CAP attributes μ, the CAP mining discovers all the correlated attribute patterns which contain two to μ attributes.

III. 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.

A. Outline of MISCELA

According to the Definition 7, the attributes on the sensor sets are a CAP if (1) the sensors are spatially connected, (2) the number of evolving timestamps is larger than ψ, and (3) the sensors contain two to μ attributes. We can discover all CAPs by searching all the spatially connected sets within a given sensor set. However, it is inefficient because there are typically a lot of spatially connected sets which do not have CAPs. Therefore, we only search the spatially connected sets which have CAPs. For efficient search, furthermore, we take an expansion-based approach. We propose a tree structure called the *CAP search tree* to effectively expand a spatially connected set. 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.

B. 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 [4]. 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.

C. 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, which specifies the evolving timestamps. Then, we extract both positive and negative evolving timestamps whose absolute change rate is larger than the evolving thresholds.

D. 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 the concept of *sensor graph*, *connected sub-graph*, and *connected component*.

Definition 8 (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 no longer than the distance η.

Definition 9 (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.

Definition 10 (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 [5]. DBSCAN is one of the clustering algorithm, which groups together points 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 the connected components. DBSCAN simultaneously constructs sensor graphs and find connected components.

Example : Figure 2 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 $\boldsymbol{G}_S$.

E. 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 in-

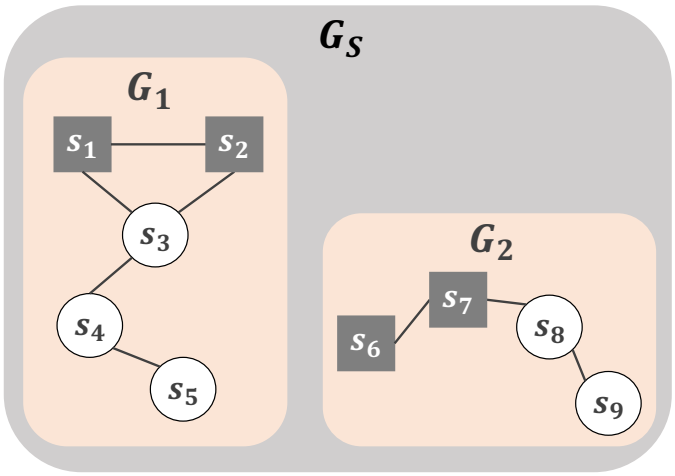


Fig. 2: The sensor graph of the sensor set $\boldsymbol{S}$

creases. 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 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 11 (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, $\boldsymbol{X} = \boldsymbol{Y} \backslash \{s'\}$ 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

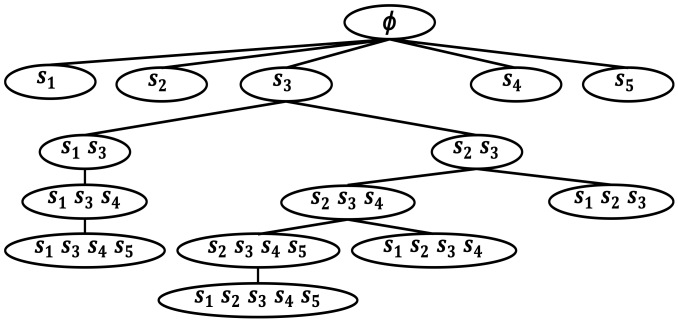


Fig. 3: The CAP search tree of $\boldsymbol{G}_1$ in Figure 2

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.

Example : Consider the connected component $\boldsymbol{G}_1$ in Figure 2. 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 3 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. □

F. 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 and 2. 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

Algorithm 1 MISCELA

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

1: **ForEach** $s_i \in \boldsymbol{S}$ **do**
2: Bottom-up Linear Segmentation(s_i)
3: **end for**
4: **ForEach** attribute a that $\boldsymbol{S}$ contains **do**
5: $\Theta_a \leftarrow$Evolving Threshold Estimation($\boldsymbol{G}_a$,ε)
6: **ForEach** $s_i \in \boldsymbol{G}_a$ **do**
7: **for** $j = 0 \rightarrow$ length of $\boldsymbol{T}$ **do**
8: **if** $r_i[t_j] \geq \theta_a^+$ **then**
9: t_j is a positive evolving timestamp
10: **end if**
11: **if** $r_i[t_j] \leq \theta_a^-$ **then**
12: t_j is a negative evolving timestamp
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: $\mathbb{G}_S \leftarrow$ DBSCAN($\boldsymbol{S}$,η,$MinPts = 2$)
18: **ForEach** $\boldsymbol{G} \in \mathbb{G}_S$ **do**
19: **ForEach** $s \in \boldsymbol{G}$ **do**
20: CAP Search($\boldsymbol{G}$,$\{s\}$,ψ)
21: **end for**
22: **end for**

Algorithm 2 CAP search

Input: connected component $\boldsymbol{G}$, connected sub-graph $\boldsymbol{X} \subseteq \boldsymbol{G}$, minimum support ψ
Output: CAPs on $\boldsymbol{X}$

1: $\mathbb{Y} \leftarrow$ the sets of sensors whose $parent$ is $\boldsymbol{X}$
2: **ForEach** $\boldsymbol{Y} \in \mathbb{Y}$ **do**
3: $C_y \leftarrow$ CAPs on $\boldsymbol{Y}$ with CAP Computation($\boldsymbol{Y}$,ψ)
4: **if** C_y is not empty **then**
5: output C_y
6: CAP search($\boldsymbol{G}$,$\boldsymbol{Y}$,ψ)
7: **end if**
8: **end for**

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 $\boldsymbol{Y}$.

G. Time Complexity

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 $O(nT + n \log n + \max(|\mathbb{G}_S| hd2^{\lambda}, |\mathbb{G}_S| hd2^{\mu}))$.

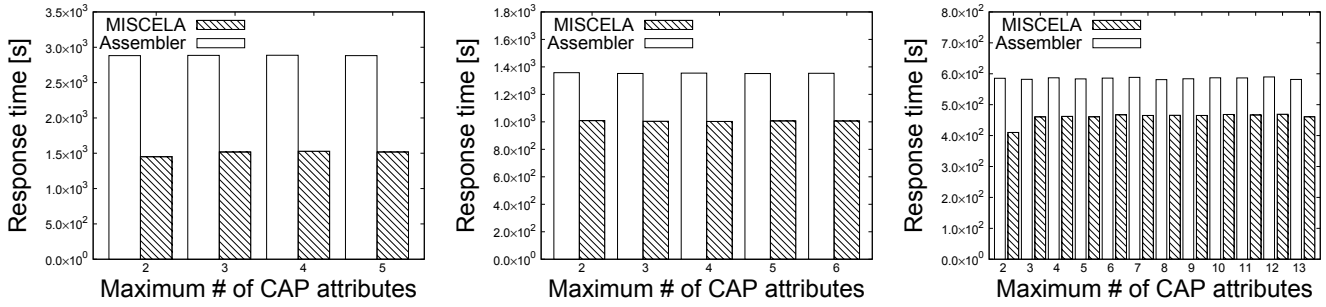
Proof : Since MISCELA contains four steps, we describe the time complexity of each step. Then, We describe the total time complexity of MISCELA.

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 [4]. 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 co-evaluations among μ attributes (Definition 6) 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 $O(\max(|\mathbb{G}_S| hd2^{\lambda}, |\mathbb{G}_S| hd2^{\mu}))$.

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

IV. EXPERIMENTS

In this section, we evaluate the efficiency and usefulness of MISCELA. 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* [2] which is the state-of-the-art algorithm for the SCP mining. Assembler discovers SCPs which contains both necessary and unnecessary sensor sets for the CAP mining. Then, we use it with a filter



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

Fig. 4: The maximum number of CAP attributes μ

which outputs only necessary sensor sets (i.e. CAPs) over 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 to that of Assembler with varying the following parameters;

- The maximum number of CAP attributes μ
- The evolving rate ε
- The minimum support ψ

The algorithms of MISCELA and Assembler are implemented in Python. The experiments are conducted on a computer with Intel Xeron E5-2620 2.4GHz CPU and 32GB memory.

A. Experimental Setup

Our experiments use three real sensor datasets; (1) five attributes daily sensor data collected in Santander, Spain from 2016/3/1 to 2016/9/30, (2) six attributes daily sensor data collected in China from 2016/9/1 to 2018/8/31, and (3) 13 attributes daily sensor data collected in China from 2016/9/1 to 2018/8/31. We obtained the Santander dataset from *FESTIVAL*[1] and the two China datasets from *envicloud.cn*[2]. Table I shows the detail of them. We use $\eta = 800$m for the Santander dataset and $\eta = 200$km for the China datasets. We set these distance thresholds to divide sensors in each dataset into around 20 connected components. For all the datasets, we use $\varepsilon = 50\%$, $\mu = 2$, and $\psi = 500$ as default parameters.

B. Efficiency Test

In this section, we describe the results of efficiency experiments. We show the response time of MISCELA and Assembler varying three parameters μ, ε, and ψ, respectively. Throughout all the experiments, MISCELA is always faster than Assembler. Our experiments show that MISCELA reduces the response time by up to 79% compared to that of Assembler.

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

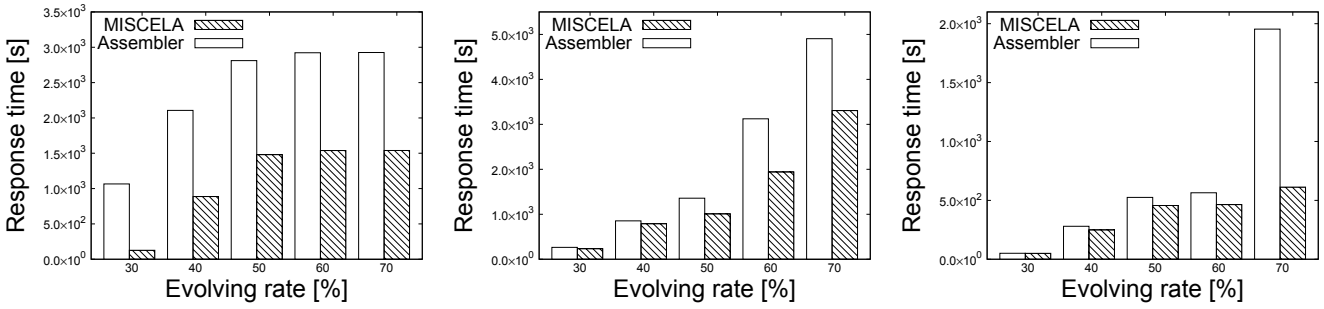
[2]http://www.envicloud.cn/

TABLE I: 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
		SO2	1573
		NO2	1573
		CO	1573
		O3	1573
China$_{13}$	730	PM2.5	370
		PM10	370
		SO2	370
		NO2	370
		CO	370
		O3	370
		Sunny-percent	370
		Rainy-percent	370
		Rain	370
		Temperature	370
		Air-pressure	370
		Humidity	370
		Wind speed	370

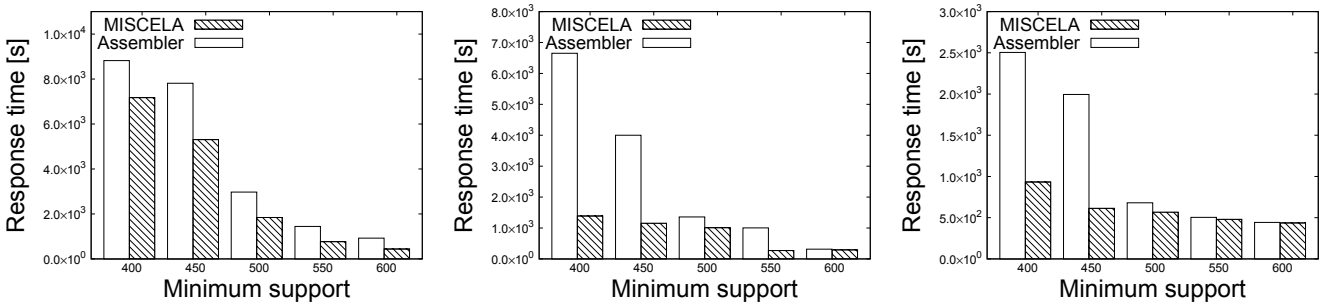
1) The Maximum Number of CAP Attributes μ: First, we describe the experimental result varying the maximum number of CAP attributes μ. We vary μ from 2 to the number of attributes in each dataset (e.g. from 2 to 5 for the Santander dataset). Figure 4 shows the result of the experiment. We can see that MISCELA is faster than Assembler in all the dataset with any μ. Moreover, the result of China$_{13}$ dataset indicates that MISCELA is more efficient with a small μ. This is because MISCELA ignores more connected sub-graphs which have no CAPs when we set a small μ, while Assembler exploits the sub-graphs.

2) Evolving Rate ε: Next, we describe the experimental result varying the evolving rate ε. We measure the response time of both methods with $\varepsilon = 30\%, 40\%, 50\%, 60\%$, and 70%, respectively. Figure 5 shows that MISCELA is faster than Assembler in all the datasets with any ε. The response time increases as the evolving rate ε increases. The setting



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

Fig. 5: Evolving rate ε



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

Fig. 6: Minimum support ψ

TABLE II: # of CAP computations varying μ

μ	2	3	4	5
MISCELA	424,510	425,062	425,062	425,062
Assembler	814,376	814,376	814,376	814,376

TABLE III: # of CAP computation varying ε

ε	30	40	50	60	70
MISCELA	32,743	231,502	424,510	424,541	424,541
Assembler	230,405	577,430	814,376	814,473	814,473

of the large ε causes that the number of extracted evolving timestamps is large.

3) Minimum Support ψ: Finally, we describe the experimental result with varying the minimum support ψ. We measure the response time of both methods with $\psi = 400, 450, 500, 550$, and 600, respectively. We show the result of the experiment in Figure 6. It shows that MISCELA is faster than Assembler in all the cases. In particular, MISCELA reduces the response time 79% compared to 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 of the minimum support. This is because the smaller minimum support detects the more CAPs.

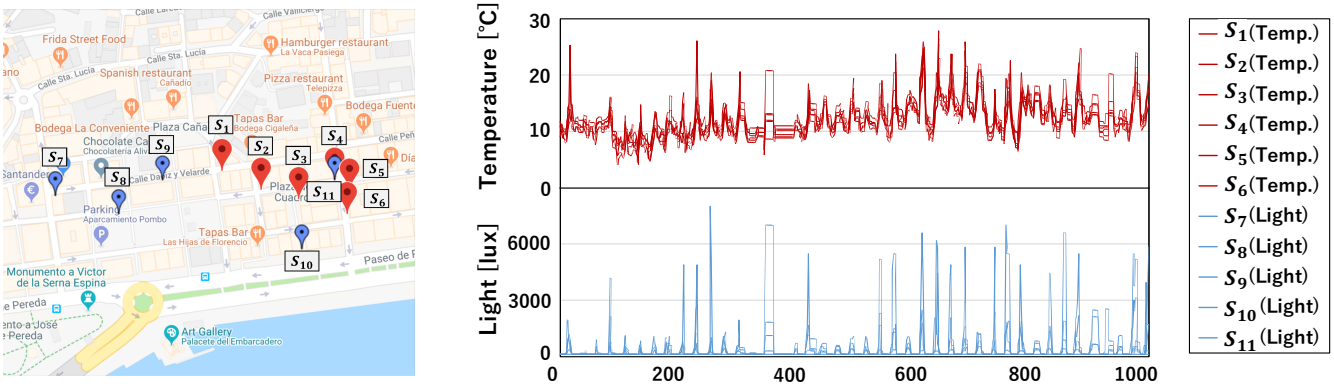
TABLE IV: # of CAP computation varying ψ

ψ	400	450	500	550	600
MISCELA	6,546,796	1,413,067	424,510	163,374	84,768
Assembler	8,091,031	2,306,391	814,376	334,775	166,001

C. Theoretical Analysis

We discuss the results from the viewpoint of the time complexity of the CAP search. The time complexity of the CAP search is $O(\max(|\mathbb{G}_S|hd2^{\lambda}, |\mathbb{G}_S|hd2^{\mu}))$. The expression indicates that the computation cost of CAP search increases as μ increases. Moreover, ε and ψ implicitly affect the computation cost of the CAP search. When ε is large and/or ψ is small, the height of CAP search tree h is large This is because more CAPs are discovered with a larger number of evolving timestamps and/or a smaller minimum support. Since the computation cost of the CAP search becomes large with a large h, a large ε or a small ψ cause a large response time of CAP search.

We can see the correctness of the discussion in Tables II, III, and IV. Tables II, III, and IV show that the number of CAP computations for Santander dataset with varying μ, ε, and ψ, respectively. The large number of CAP computation indicates a large computation cost of CAP search. From these results, we can directly understand the reason that MISCELA is faster than Assembler. The response time of the CAP search



(a) Sensor location

(b) Correlation

Fig. 7: The CAP in Santander

increases in proportion to the number of CAP computations. We here omit the results of the China datasets because they have a similar tendency to those of the Santander dataset.

D. Examples of CAPs

We show a meaningful CAP in Santander. Figure 7 shows locations of sensors and 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. In the real world, the sensors are in a downtown area within the city, and this pattern indicates that the downtown gets a lot of sunshine during the day time and they are not in the shade.

V. 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. Lin et al. [6] 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. [7] developed an algorithm that discovers approximate motifs in linear time. Mueen et al. [8] 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. [9], [10] proposed an algorithm that transforms multi-dimensional time series data into a sequence of symbols using principal component analysis. Minnen et al. [11] 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. [12] 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. [13] 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. [2] 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. [3] 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. [14] 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 Assember which is the state-of-the-art for the SCP mining.

VI. CONCLUSION

In this paper, we introduced the problem called correlated attribute pattern (CAP) mining, which discovers the correlation among attributes in multi-attributes sensor set. We proposed the efficient method, MISCELA, for the CAP mining. MISCELA effectively prunes the unnecessary computations for the CAP mining. We conducted the experiments using three real sensor datasets. MISCELA can efficiently discover CAPs in multi-attribute sensor sets compared to the SCP mining method. The results of our experiments showed that the CAP mining obtains several meaningful patterns.

In our future work, we attempt to discover regional features among the attributes in a certain area. Additionally, it is interesting to discover outliers in areas characterized by CAPs.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Numbers JP15K21069 and JP16H01722.

REFERENCES

[1] Luis Sanchez, Luis Muñoz, Jose Antonio Galache, Pablo Sotres, Juan R Santana, Veronica Gutierrez, Rajiv Ramdhany, Alex Gluhak, Srdjan Krco, and Evangelos Theodoridis. Smartsantander: IoT experimentation over a smart city testbed. *ELSEVIER Computer Networks*, 61:217–238, 2014.

[2] Chao Zhang, Yu Zheng, Xiuli Ma, and Jiawei Han. Assembler: efficient discovery of spatial co-evolving patterns in massive geo-sensory data. In *Proceedings of the ACM SIGKDD*, pages 1415–1424, 2015.

[3] Yun Cheng, Xiucheng Li, and Yan Li. Finding dynamic co-evolving zones in spatial-temporal time series data. In *Proceedings of the ECML PKDD*, pages 129–144, 2016.

[4] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings of the IEEE ICDM*, pages 289–296, 2001.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the ACM SIGKDD*, pages 226–231, 1996.

[6] JLEKS Lonardi and Pranav Patel. Finding motifs in time series. In *Proceedings of the ACM SIGKDD*, pages 53–68, 2002.

[7] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ACM SIGKDD*, pages 493–498, 2003.

[8] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact discovery of time series motifs. In *Proceedings of the SIAM international conference on data mining*, pages 473–484, 2009.

[9] Yoshiki Tanaka and Kuniaki Uehara. Discover motifs in multi-dimensional time-series using the principal component analysis and the MDL principle. In *Proceedings of the Springer MLDM*, pages 252–265, 2003.

[10] Yoshiki Tanaka, Kazuhisa Iwamoto, and Kuniaki Uehara. Discovery of time-series motif from multi-dimensional data based on MDL principle. *Springer Machine Learning*, 58(2):269–300, 2005.

[11] David Minnen, Charles Isbell, Irfan Essa, and Thad Starner. Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery. In *Proceedings of the IEEE ICDM*, pages 601–606, 2007.

[12] Roberto Trasarti, Ana-Maria Olteanu-Raimond, Mirco Nanni, Thomas Couronné, Barbara Furletti, Fosca Giannotti, Zbigniew Smoreda, and Cezary Ziemlicki. Discovering urban and country dynamics from mobile phone data with spatial correlation patterns. *Elsevier Telecommunications Policy*, 39(3-4):347–362, 2015.

[13] Yasuko Matsubara, Yasushi Sakurai, Willem G Van Panhuis, and Christos Faloutsos. FUNNEL: automatic mining of spatially coevolving epidemics. In *Proceedings of the ACM SIGKDD*, pages 105–114, 2014.

[14] Marwan Hassani, Emmanuel Müller, Pascal Spaus, Adriola Faqolli, Themis Palpanas, and Thomas Seidl. Self-organizing energy aware clustering of nodes in sensor networks using relevant attributes. 2010.