Online Temporal-Spatial Analysis for Detection of Critical Events in Cyber-Physical Systems

Zhang Fu, Magnus Almgren, Olaf Landsiedel, Marina Papatriantafilou
Chalmers University of Technology. Email:{zhafu,magnus.almgren,olafl,ptrianta}@chalmers.se

***Abstract*—Cyber-Physical Systems (CPS) employ sensors to observe physical environments and to detect events of interest. Equipped with sensing, computing, and communication capabilities, Cyber-Physical Systems aim to make physical-systems smart(er). For example, smart electricity meters nowadays measure and report power consumption as well as critical events such as power outages. However, each day, such sensors report a variety of warnings and errors: many merely indicate transient faults or short instabilities of the physical system (environment). Thus, given the big volumes of data, the time-efficient processing of these events, especially in large-scale scenarios with hundreds of thousands of sensors, is a key challenge in CPSs.**

Motivated by the fact that critical events of CPSs often have *temporal-spatial* properties, we focus on identifying critical events by an online *temporal-spatial* analysis on the data stream of messages. We explicitly model the online detection problem as a *single-linkage clustering* on a data stream over a *sliding-window*, where the inherent computational complexity of the detection problem is derived. Based on this model, we propose a *grid-based* single-linkage clustering algorithm over a sliding-window, which is an online time-space efficient method satisfying the quick processing demand of big data streams. We analyze the performance of the proposed approach by both a series of propositions and a large, real-world data-set of deployed CPS, composing 300,000 sensors, over one year. We show that the proposed method identifies above 95% of the critical events in the data-set and save the time-space requirement by 4 orders of magnitude compared with the conventional clustering method.

I. Introduction

In Cyber-Physical systems, sensors report a variety of warnings and errors. However, many indicate transient faults or short instabilities of the physical systems (environments). This prohibits operators from reacting to each individual event. Similarly, manually filtering these is a cumbersome process. Thus, given the big volumes of data, the time-efficient processing of these events, especially in large-scale scenarios with hundreds of thousands of sensors, is a key challenge.

There is a need for automatic methods to timely detect and distinguish transient faults and short-term instabilities from critical events, such as power outages or dangerous voltage fluctuations in the electrical grid. In such events, alarms from the affected sensors share key properties in terms of spatial and temporal relation. For example, when a power outage occurs and affects a neighborhood, meters residing within that area detect it at a similar time (if not simultaneously).

Motivated by the above example, we focus on online analysis of data in CPSs: We correlate individual messages from sensors according to their *temporal-spatial* properties and show that our method efficiently identifies and locates critical events in the raw message stream. Aiming at effective online processing both in terms of high accuracy and low processing complexity, we face the following two key challenges:

Implicit temporal-spatial relation: Usually, the temporal-spatial relation of the messages is implicitly defined, since it reflects the nature of the physical system. Although many *clustering algorithms* exist for correlating similar data points [8], there is still a need for a model to map the problem of temporal-spatial analysis of data in CPSs to a corresponding clustering problem at the algorithmic level.

Big data volume and time-critical processing: Online processing of messages which are continuously generated from a large sensor population inevitably encounters the challenge of heavy computational complexity, especially when the data analysis is time-critical. Many approaches provide efficient methods for clustering data streams [13]. However, we find that none of them are suitable for capturing the temporal-spatial properties identified by this paper based on the data from a real deployed CPS (see the related work in Section V). It is still challenging to develop an appropriate time-space efficient clustering method for an explicit temporal-spatial analysis on data streams.

Addressing the above challenges, we make two contributions in this paper:

1) Based on traces from an AMI of 300,000 electrical meters, we analyze the temporal-spatial patterns of critical events, such as power outages. We model the temporal-spatial analysis for detecting critical events through the problem of *single-linkage clustering* of a data stream over a *sliding-window*.
2) We provide $G\text{-}SLC$, a *grid-based single linkage clustering algorithm*, for efficient clustering data points from a real-time large data stream. We analyze the performance of $G\text{-}SLC$ and evaluate it with a real-world data-set. We show that $G\text{-}SLC$ reduces the time-space complexity by almost 4 orders of magnitude compared to the conventional clustering method. $G\text{-}SLC$ can help identifying critical events at high accuracy (around 95%).

The remainder of this paper is organized as follows: Section II gives the problem statement and models the problem as single-linkage clustering a data stream over a sliding-window. In Section III, we propose $G\text{-}SLC$ with detailed explanations; we also analyze its performance by showing a series of propositions. In Section IV, we evaluate it through experiments using a real-world data-set. Section V discusses related work and we conclude in Section VI.

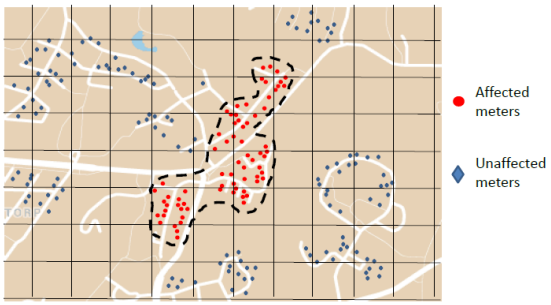

Fig. 1: Example of locations of meters affected in the same power outage event. All the red circles depict affected meters which are connected to the same power substation through power lines. The unaffected meters in the surrounding area are depicted as blue dots. We can see that the affected area (shaped out by the dash line) almost follows along the roads and its shape can be arbitrary.

II. Problem Statement and Analysis

A. Problem Statement

Messages indicating the events of interest are reported by the sensors over time, forming a data steam $\mathcal{S} = s_1, s_2, \cdots$, where s_i is a data record, i.e., a message. We use t_i to denote the time-stamp of s_i, which is the message-generation time. For a pair of data records s_i and s_j, we use $d(s_i, s_j)$ to denote the geographical distance between the corresponding sensors.

Take the electricity grid into consideration. Basically the power-distribution network follows the roads in the residential area. If a part of the electricity network goes down, the affected area commonly stretches out along the roads. All affected meters suffer the power failures almost at the same time and the alarm messages are sent within a short time interval (e.g. a couple of seconds). Figure 1 shows an example derived from a real power outage. The figure depicts the spatial relations among the affected meters. To detect such a power outage, we need to identify a *temporal-spatial related set* of records from a continuous data stream of alarm messages.

Definition 1. *(**Temporal-Spatial Related Set**) Let $\mathcal{S}_*$ be a set of records from the data stream, where $|\mathcal{S}_*| \geq 2$. We say $\mathcal{S}_*$ is a temporal-spatial related set if it satisfies two conditions: (1) $\forall s_i, s_j \in \mathcal{S}_* : |t_i - t_j| \leq \epsilon_t$; (2) For each pair of records (s, s') in $\mathcal{S}_*$, there exists a set of records in $\mathcal{S}_*$, which can form a sequence $\{s_1, \cdots, s_k\}$, $k \geq 2$, where $s_1 = s$ and $s_k = s'$, such that $\forall i < k : d(s_i, s_{i+1}) \leq \epsilon_d$; ϵ_t and ϵ_d are called the time and distance thresholds, respectively.*

B. Clustering Over Sliding Window

To detect temporal-spatial related sets, we – according to definition 1 – perform a *spatial clustering on a data stream over time-based sliding-windows*. The sliding-windows filter time irrelevant records, whereas spatial clustering identifies spatial relevant records.

We first give a formal model of *data clustering over time-based sliding-windows*: We use W_τ to denote the window that starts at time τ. The size of the window is denoted by ω. Thus all data records arriving during time interval $[\tau, \tau + \omega)$ belong to window W_τ. We perform data clustering on the records of each time window. For example, when at time $\tau + \omega$ the clustering is conducted based on data records in W_τ. Then, the sliding-window will move forward by ω_a time units (which is called *the window advance*) and the new window is $W_{\tau+\omega_a}$. All old data records outside the new window will be deleted and not be considered for future computation.

By choosing an appropriate window size, we can show that all the data records that belong to the same temporal-spatial sets can be processed in at least one time-window. In particular, we give the following proposition[1]:

Proposition 1. *Let s_i and s_j be two records, $|t_i - t_j| \leq \epsilon_t$. By choosing the size of the sliding-window $\omega \geq \epsilon_t + \omega_a$, we can process s_i and s_j in at least one time-window.*

Considering the data records in a window, the problem of partitioning them into disjoint sets according to condition 2 in Definition 1 is equivalent to *single-linkage clustering with stop condition of ϵ_d* [10], we use SLC to denote it.

Proposition 2. *Let s_i and s_j be two records in the same window. They belong to the same spatial related set iff they belong to the same cluster in SLC.*

Given Propositions 1 and 2, finding temporal-spatial related sets reduces to SLC on a data stream $\mathcal{S}$ over a sliding-window. A straightforward method for conducting SLC over a sliding-window is to keep all records for the current window. When the window shifts, we compute the clusters based on the current records in the window; then all the records that are outside the new window will be deleted. Suppose there are n records in the current window, SLC can be achieved as follows: First compute the proximity matrix for all pairs of records; the values in the matrix are the distances of all pairs of clusters (initially each record forms a cluster). In each step, the pair of clusters which has the minimum value is merged into a new cluster and the distances bewteen the latter and the remaining clusters are updated. This repeats until all the values in the proximity matrix are greater than ϵ_d or all the records are in one cluster. The method is derived from the *SLINK* algorithm [12]. It is shown that both the time and space complexity of the algorithm are $O(n^2)$, which is known to be the best achievable complexity for single-linkage clustering [5]. This complexity is acceptable for small numbers of records. However, CPSs commonly continuously generate a large volume of records. Often for these the computing time is not acceptable, especially for critical events that need quick reactions.

III. Time-Space Efficient Method

In the classic SLC method, we observe the following: when a set of records is close to each other, then these can be represented as a group and the clustering can be done at the group level. Based on that, we propose our *grid-based* method, denoted as $G\text{-}SLC$. The monitored geographical area is modeled as a 2-D space and is partitioned into a grid.

[1]Due to the space constraint, proofs of all the propositions are omitted.

We use $c_{i,j}$ to denote the cell of the i^{th} interval and j^{th} interval of x and y dimension (which are the longitude and latitude), respectively. Each cell is a square with edge length ϵ_d, corresponding to the SLC method. Since each data record has a location attribute, it can quickly be mapped into the cell which covers its location. The grid in Figure 1 is an example. Before detailing $G\text{-}SLC$, we give two definitions:

Definition 2. *(***Neighbor cells***) Let $c_{i,j}$ and $c_{l,k}$ be two cells in the grid. They are neighbor cells if $|i-l| \leq 1$ and $|j-k| \leq 1$. We use $\mathcal{N}^c$ to denote the set of neighbor cells of cell c.*

Definition 3. *(***Active cell***) Cell $c_{i,j}$ is called an active cell, if there is at least one record mapped to it in the current window.*

A. Design details

Next, we give a detailed description of $G\text{-}SLC$, our proposed grid-based method for efficient single-linkage clustering of records in a data stream over a sliding-window. We first explain how to maintain the state of the cells. Then we discuss our procedure of adding each newly received record. After that we explain how the clustering is conducted when the time window shifts and how the cell states are updated.

1) Maintain cell weights for sliding-window: The state of each active cell maintained in the memory is its *weight* which is defined as the number of records mapped to the cell in the current window. Since only active cells need to be considered in the clustering procedure, only their weights are maintained.

According to the model of sliding-window, when the window shifts some old records may reside outside the new window, then the weights of the corresponding cells have to be updated. Suppose the window shifts from W_τ to $W_{\tau+\omega_a}$, then the records whose timestamps are within the time interval $[\tau, \tau+\omega_a)$ have to be deleted. Given this fact, we maintain the weight of each cell with a number of *buckets* (each of them is just an integer variable). Here we assume that ω is divisible by ω_a, thus the number of buckets for each cell is $\frac{\omega}{\omega_a}$ and they are kept in an array. In window W_τ, for an active cell $c_{i,j}$, bucket $c_{i,j}.b[k]$ maintains the number of records which are mapped to $c_{i,j}$ and have timestamps within $[\tau + k\omega_a, \tau + (k+1)\,\omega_a)$, where $k \in \left\{0, 1, \cdots, \frac{\omega}{\omega_a} - 1\right\}$. So we have $c_{i,j}.weight = \sum_k c_{i,j}.b[k]$.

2) Upon receiving a new record: When a new record, say s, arrives during time window W_τ, the index of the cell which covers the location of s is calculated. If the cell is not active (active cells are maintained in $active\_cell\_list$ which can be implemented using a hash table), then the state of the cell is initiated and added to $active\_cell\_list$. The value of the correct bucket of the cell will be increased by one. Algorithm 1 shows the pseudo-code.

3) Clustering active cells: For each time window, $G\text{-}SLC$ conducts the clustering on all active cells. $G\text{-}SLC$ uses an auxiliary data structure $neighbor\_list$ for doing that. The procedure of the clustering is the following:
(i) A cluster is created for the first unprocessed active cell in $active\_cell\_list$, this cell is marked as *ready* and is put into $neighbor\_list$. (ii) For each cell in $neighbor\_list$, add it to the current cluster, mark it as processed and remove it from $neighbor\_list$. Then insert all its active neighbor cells that are unprocessed and unready in $neighbor\_list$, then mark them as ready. This step is repeated until $neighbor\_list$ is empty. (iii) If there exist unprocessed cells in $active\_cell\_list$, then step 1 and 2 are repeated. The procedure terminates when there is no unprocessed cells in $active\_cell\_list$. The pseudo-code is shown in Algorithm 2.

Algorithm 1: Upon receiving new record s

```
// Suppose current window is W_τ
if s.timestamp ∈ W_τ then
    (i, j) ← find cell index for s;
    k ← ⌊(s.timestamp − τ)/ω_a⌋;
    if c_{i,j} ∉ active_cell_list then
        initiate c_{i,j} and put c_{i,j} into active_cell_list;
    c_{i,j}.b[k] + +;
else
    drop s;
```

Algorithm 2: Clustering active cells in the current window

```
while not reach the end of active_cell_list do
    c ← next cell in active_cell_list;
    if c is unprocessed then
        Create a new cluster Cluster_*;
        Put c into neighbor_list and mark it as ready;
        while neighbor_list != ∅ do
            c ← next cell in neighbor_list;
            Add c into Cluster_* and mark c as processed;
            Remove c from neighbor_list;
            forall the c_* ∈ N^c do
                if c_* is active & c_* is unprocessed & c_* is
                unready then
                    Put c_* into neighbor_list and mark it as
                    ready;
```

4) Upon sliding-window shifting: When the sliding-window shifts, the clusters have to be generated. Thus Algorithm 2 is conducted over the cells in $active\_cell\_list$. After that the weights of all active cells will be updated as well as their buckets. If the weight of a cell becomes zero, then it becomes inactive and is deleted from the memory. The pseudo-code is shown in Algorithm 3.

Algorithm 3: Upon time window shifting

```
B ← ω/ω_a;
if active_cell_list != ∅ then
    run Algorithm 2 on active_cell_list;
    Output clustering results;
    forall the c ∈ active_cell_list do
        if c.weight − c.b[0] > 0 then
            for k ← 1 to B − 1 do
                c.b[k − 1] ← c.b[k];
            c.b[B] ← 0;
        else
            delete c from active_cell_list;
```

B. Performance analysis

First we give two propositions showing the computational complexity of G-SLC.

Proposition 3. *The time complexity of processing a newly arrived record is* $O(1)$.

Proposition 4. *The time complexity of Algorithm 2 is* $O(N)$*, where* N *is the number of active cells.*

Propositions 3 and 4 show that the computational complexity of G-SLC is much smaller than the classical single-linkage clustering method whose complexity is $O(n^2)$. The space complexity of G-SLC is also much smaller than $O(n^2)$, since only the active cells are kept in the memory. The space requirement for $active\_cell\_list$ and $neighbor\_list$ is $O(N)$. Hence we have the following proposition:

Proposition 5. *The space complexity of* G-SLC *is* $O(N)$*, where* N *is the number of active cells.*

To analyze the clustering performance, we use the classical SLC as the baseline. We first show that G-SLC will not miss clusters compared with SLC with the following proposition:

Proposition 6. *Let* s_i *and* s_j *be two records in the same time window. If* s_i *and* s_j *belong to the same cluster in* SLC*, then they belong to the same cluster in* G-SLC.

Proposition 6 implies that G-SLC will not generate more clusters than SLC for the same set of records. However, G-SLC may be more likely to include noise records into the existing clusters than SLC, due to cell-level aggregation. It is complicated to measure accurately the probability that a noise record is assigned into an existing cluster in G-SLC, however, we try to give a non-trivial upper bound on that probability, which is shown in the following proposition:

Proposition 7. *Let* $\mathcal{C}$ *be a cluster in* G-SLC *which contains* $N \geq 2$ *active cells. Suppose a noise record* s *has equal probability to be assigned to any cell, that is* $\frac{1}{\mathcal{G}}$*, where* $\mathcal{G}$ *is the total number of cells in the grid. The probability that* s *joins* $\mathcal{C}$ *is at most* $\frac{5N+4}{\mathcal{G}}$.

Let us take an example based on the studied AMI in this paper which covers a city with roughly about $450km^2$ in size. Suppose we choose $\epsilon_d = 200m$, then the total number of cells for covering $450km^2$ is around 11,000. From our study of the power outages, usually the size of an affected area is less than $1km^2$ covering around 25 cells. According to proposition 7, the probability that a noise record belongs to the cluster of that power outage is less than 0.012.

IV. Experimental Evaluation

In this section we evaluate the performance of G-SLC. We begin by briefly introducing our real-world data-set. Next, we show the computational performance and clustering performance in Section IV-B and connect the results to our analytical results in Section III-B. Section IV-C discusses the performance of G-SLC in detecting critical events.

A. Data-Set Description

The evaluation utilizes a data-set from an Advanced Metering Infrastructure (AMI) of an European city with roughly 600,000 inhabitants and about 450 km^2 in size. The AMI contains around 300,000 smart meters which wirelessly report faults and errors in power lines. In this paper, we focus on power outages and our data-set includes all *power failure* messages received in the year 2012. A record contains a meter ID, GPS coordinates of the meter, and a timestamp indicating when the power failure was detected by the meter.

In addition, we utilize power outage reports from customers to the *call center* of the electricity company. The location and the affected meters are known for each such reference power outage. We use these to calibrate the parameters of our G-SLC method and to evaluate its detection performance.

B. Performance Evaluation

We implemented G-SLC in JDK 1.7 on an Intel Xeon E5645 2.4GHz machine with 48GB memory. The operating system is Ubuntu 12.04 with Linux kernel version 3.2.0-53. We also implemented the SLC method whose performance acts as the baseline in the evaluation. Table I shows the values of all the parameters of G-SLC used in the current evaluation.

Space requirement The performance of G-SLC heavily depends on the number of active cells in the sliding-window. Therefore, we use G-SLC to process the whole data-set to compare the number of active cells with the number of records in each time window. To show that, we measure the ratio between the number of records and the active cells in the same window. We call it *aggregation ratio*. Figure 2 shows the box plot of aggregation ratios for different cell sizes. We can see that the aggregation ratio increases when the cell size is enlarged. For some windows, the aggregation ratio are above 100, some of them are even more than 200 when ϵ_d is greater than $400m$. This means that the space requirement of G-SLC can be less than the one of SLC by 2 orders of magnitude when used in the real setting.

Computation time To study the computational time, we use SLC and G-SLC to process the whole data-set, respectively; the computation times are shown in Figure 3. We can see that G-SLC is approximately 2500 times faster than SLC. Since the time complexity of SLC is $O(n^2)$ and the time complexity of G-SLC is $O(N)$ (where n is the number of records and N is the number of active cells), if n is greater than N by 2 orders of magnitude, then the computational time of SLC can be greater than G-SLC by 4 orders of magnitude.

Clustering performance To evaluate the clustering performance of G-SLC, we use SLC as baseline. For each time window, we measure the *cluster approximation ratio* which is defined as the ratio between the number of clusters generated by G-SLC with that generated by SLC. For different values of ϵ_d, we make SLC and G-SLC process the whole data-set and measure the average cluster approximation-ratio over all time windows. Figure 4 shows the results. From the figure we can see that the average ratio is very close to 1, meaning that

Parameter	Notation	Value
window size	ω	5 minutes
window advance	ω_a	1 minute
cell size	ϵ_d	200m, 400m, 800m, 1600m

TABLE I: Parameter settings of $G\text{-}SLC$ used in the evaluation.

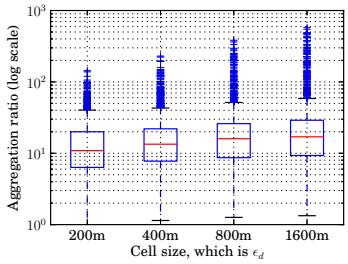

Fig. 2: Box plot of aggregation ratios for different cell sizes. The length of the whiskers is 1.5 IQR.

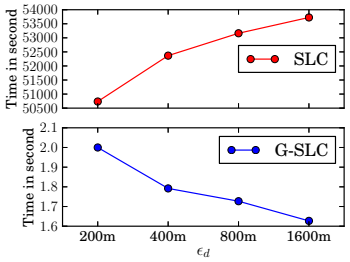

Fig. 3: Computational time of SLC and $G\text{-}SLC$ for processing the whole data-set.

most of the time the clustering results of $G\text{-}SLC$ and SLC are very similar.

C. Detection Evaluation

Detection ratio We design $G\text{-}SLC$ to help detecting critical events from the raw message stream. To evaluate its detection accuracy, we compare the detected power outages with the reference power outages given by the information from the call center. Note, that the reference events only contain parts of the power outages in 2012: Not all power outages resulted in customers calling to notify the electricity provider. For example, short outages at night can go unnoticed. Thus, we cannot utilize it to measure the false positive/negative of $G\text{-}SLC$. Instead, we use the *detection ratio* to measure the proportion of reference events that are detected by $G\text{-}SLC$ when processing the records in the data-set. Here we say a reference event is successfully detected by $G\text{-}SLC$ if: (1) the difference between the time of the reference event and its time detected by $G\text{-}SLC$ is not greater than ω time units; (2) the location of the reference event is covered by the cells of the most significant cluster in $G\text{-}SLC$. We define the significance of a cluster as:

$$\text{significance}(Cluster_i) = \frac{\#\text{ of records in } Cluster_i}{\#\text{ of records in the window}}$$

To facilitate detecting critical events, the clustering method only reports the clusters with significance greater than θ_s (which is called the *significance threshold*) Figure 5 shows the detection ratio of $G\text{-}SLC$ with different settings of the significance threshold and the cell size. We can see that the detection ratio decreases when θ_s increases. This is because higher significance thresholds make the detection method more sensitive to the noise records (e.g., some meters have some transit power failures which accidentally happen in the same time window of a critical power outage). If the scale of the power outage is small (e.g. less than 50), then a small number of noise records (e.g. 5) can make the significance of the largest cluster less than θ_s (e.g 0.9), which causes the event to be undetected.

Another observation is that increasing cell size leads to better detection ratio. Since larger cells may aggregate more records, the largest cluster may contain more records and have higher significance. However, as we will see shortly, increasing cell size harms the *detection precision*.

Detection precision The locations of the detected events by $G\text{-}SLC$ are represented by the cells of the corresponding clusters. Thus, the cell sizes influence the precision of localizing the critical events. To this end, we define the *detection precision* of a critical event as the ratio between the *affected diameter* and the cell size, where the *affected diameter* is the maximum distance between any pair of the records belonging to the event. The higher the value of the detection precision is, the better the critical event can be localized. To evaluate the detection precision of $G\text{-}SLC$, we compute the detection precision values of all the reference power outages over different cell sizes. The results are shown in Figure 6.

From Figure 6, we can see that there are events having high detection precision (i.e. greater than 1) for all the settings of the cell size. These events are the power outages that affect large residential areas. However, on average the detection precision decreases when the cell size becomes larger. In particular, we can see that the median value of the detection precision is around 0.1 when the cell size is $1600m$. This implies that the affected area can be 100 times smaller than a cell. Thus, in such situations, the location of the critical events given by $G\text{-}SLC$ may be very coarse, whereas smaller cell sizes can offer fine-grained localization performance. We can see that when the cell sizes are smaller than $400m$, the median value of the detection precision is above 1.

V. Related Work

We first focus on the related work of processing data of CPSs. Welbourne et al. propose Cascadia [17] which provides a system for processing RFID readings and detecting user-defined (using a declarative query language) events. Gyllstrom et al. propose SASE [7] which is also a system for processing real-time streams of RFID data. It focuses on a complex event language and new query processing methods to implement the language. However, how to conduct temporal-spatial clustering using the queries implemented in Cascadia and SASE is not addressed. Tang et al. propose a clustering method for processing Cyber-Physical data. Their method is similar to the single-linkage clustering. However, unlike our paper which provides a time-space efficient method for processing a real-time data stream, the authors in [14] mainly focused on processing data in a database and the computational complexity of their method is $O(n^2)$. We refer the reader to a survey [9] for further work on processing data in CPSs.

Regarding the algorithmic aspect, our work is related to the field of clustering data streams. There are many algorithms for clustering data streams based-on k-means, such

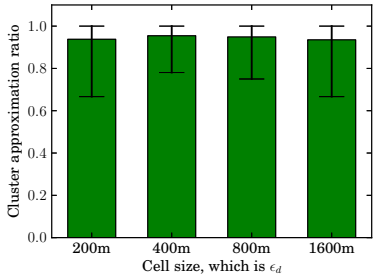

Fig. 4: Average cluster approximation ratio of G-SLC with different cell sizes. The error bars represent the 10^{th} and 90^{th} percentiles.

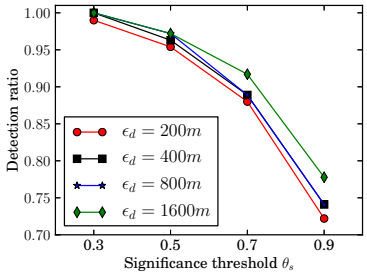

Fig. 5: Detection ratio of G-SLC with different settings of cell size and significance threshold. Increasing θ_s influences the detection ratio, especially for small scale of outages.

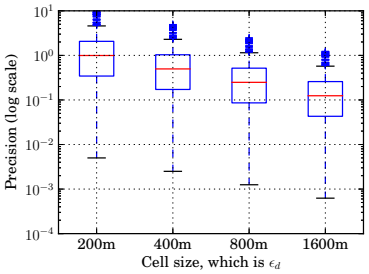

Fig. 6: Box plot of detection precision of all the detected reference events by G-SLC with different settings of the cell size. The length of the whiskers is 1.5 IQR.

as BIRCH [18], CluStream [3] and StreamKM++ [2]. However, k-means clustering can not identify clusters with arbitrary shapes, so it is not suitable for the problem setting of this paper. Due to the limitation of k-means clustering, researchers proposed efficient density-based clustering algorithms to processing data streams and identify clusters of arbitrary shapes [16], [4], [1]. However, all of these works use a *damped window* model which decays the weights of records over time rather than deleting old records. The damped window model is not quite suitable for capturing the temporal features of the temporal-spatial related set. Furthermore, to use density-based clustering methods, such as DBSCAN [6], one needs to define the density threshold in order to filter out noisy data points. However, the deployment density of sensors in a CPS can vary significantly, which makes it difficult to choose such a density threshold. We refer the reader to a survey [13] for further work on data stream clustering.

There are quite few works dealing with grouping objects from a stream of trajectories, such as [15], [19], [11], where the input of the problem is a series of snapshots of moving objects. In these works, tailored clustering methods are used to find companions in each snapshot. However, the clustering methods are not suitable for sliding window and the computational complexity of them are not linear as in this paper.

VI. Conclusions

In this paper, we model the online temporal-spatial data analysis for event detection as the problem of *single-linkage clustering* on a data stream over a *sliding-window*. Based on the model, we show the inherent computational complexity of the problem of identifying critical events. To meet demands for time-space efficient processing, we propose G-SLC, a *grid-based* single-linkage clustering algorithm over a sliding window, which is suitable for quick processing data streams. We analyze the performance of G-SLC through a series of propositions and evaluate it using a data-set of messages from a large-scale deployed CPS. Our results show that G-SLC can conduct the temporal-spatial processing on the data-set 2500 times faster than the classical clustering method and with lower space complexity. Furthermore, G-SLC can help identifying critical events in the data-set with high accuracy.

Acknowledgment

This work has been partially supported by the European Commission Seventh Framework Programme (FP7/2007-2013) through the SysSec Project, under grant agreement 257007, through the FP7-SEC-285477-CRISALIS, through the collaboration framework of Chalmers Energy Area of Advance, and with support from the Swedish Energy Agency under the program Energy, IT and Design.

References

[1] E. Achtert, C. Bohm, H.-P. Kriegel, and P. Kroger. Online hierarchical clustering in a data warehouse environment. In *Data Mining, Fifth IEEE International Conference on*, pages 10–17, Nov 2005.

[2] Marcel R. Ackermann, Marcus Märtens, and Christoph Raupach et al. Streamkm++: A clustering algorithm for data streams. *J. Exp. Algorithmics*, 17:2.4:2.1–2.4:2.30, May 2012.

[3] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *VLDB'03 Volume 29*, pages 81–92, 2003.

[4] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of KDD'07*, pages 133–142, 2007.

[5] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Hierarchical clustering. *Introduction to Information Retrieval*, Chapter 16, 2008.

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[7] Daniel Gyllstrom, Eugene Wu, and Hee-Jin Chae et al. Sase: Complex event processing over streams (demo). In *CIDR*, pages 407–411, 2007.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[9] Fangfang Li, Jia Xu, and Ge Yu. A survey on event processing for cps. In Ruchuan Wang and Fu Xiao, editors, *Advances in Wireless Sensor Networks*, volume 334 of *Communications in Computer and Information Science*, pages 157–166. 2013.

[10] Dana Avram Lupsa. Unsupervised single link hierarchical clustering. *Studia Univ. Babes-Bolyai, Informatica*, 50(2):11–22, 2005.

[11] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

[12] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[13] Jonathan A. Silva and Elaine R. et al Faria. Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13:1–13:31, July 2013.

[14] Lu-An Tang, Xiao Yu, and Sangkyum Kim et al. Multidimensional analysis of atypical events in cyber-physical data. In *Proceedings of ICDE '12*, pages 1025–1036, 2012.

[15] Lu-An Tang, Yu Zheng, and Jing Yuan et al. On discovery of traveling companions from streaming trajectories. In *ICDE*, pages 186–197, 2012.

[16] Li Wan, Wee Keong Ng, Xuan Hong Dang, Philip S. Yu, and Kuan Zhang. Density-based clustering of data streams at multiple resolutions. *ACM Trans. Knowl. Discov. Data*, 3(3):14:1–14:28, July 2009.

[17] Evan Welbourne, Nodira Khoussainova, and Julie Letchner et al. Cascadia: A system for specifying, detecting, and managing rfid events. In *Proceedings of MobiSys '08*, pages 281–294, 2008.

[18] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of SIGMOD '96*, pages 103–114, 1996.

[19] Kai Zheng and Yu Zheng et al. On discovery of gathering patterns from trajectories. In *ICDE*, pages 242–253, 2013.