# Mining Rare Recurring Events in Network Traffic using Second Order Contrast Patterns


Elaheh Alipourchavary
*School of Computing and Information Systems*
*The University of Melbourne*
Melbourne, Australia
ealipourchav@student.unimelb.edu.au

Sarah M. Erfani
*School of Computing and Information Systems*
*The University of Melbourne*
Melbourne, Australia
sarah.erfani@unimelb.edu.au

Christopher Leckie
*School of Computing and Information Systems*
*The University of Melbourne*
Melbourne, Australia
caleckie@unimelb.edu.au



***Abstract*—Data mining techniques such as contrast pattern mining provide a promising approach to detecting and characterizing changes in network traffic. However, a major challenge for network managers is how to prioritize their analysis of these changes, without being overwhelmed by uninformative patterns. In particular, some changes in traffic occur on a regular basis, such as system backups, and it is important to filter out these rare recurring events, so that network managers can focus on new events. In this paper we address the problem of identifying rare recurring events in network traffic, and we propose a novel solution to detecting new events based on the approach of mining second order contrast patterns. Based on an empirical evaluation using a variety of real traffic sources, we show that our method can achieve high accuracy and F1-Score in detecting new events. Our work demonstrates the importance of higher order contrast pattern mining in practice, and provides an effective method for finding such higher order patterns in large datasets.**



***Index Terms*—contrast pattern mining, emerging pattern mining, higher order contrast patterns**


## I. Introduction

Identifying and characterizing significant changes in the traffic of a network is a challenging task in network and security management. For example, changes in the types of traffic flows in a network may indicate malicious activity, a network fault, or a change in legitimate users' behavior. Contrast pattern mining (CPM) is a promising approach to extract emerging trends and changes, and it has been used successfully in many applications such as network traffic analysis [1], [2], medical diagnosis [3], [4], and customer behavior analysis [5].

CPM (aka emerging pattern mining [6]) finds contrast patterns (CPs) that occur frequently in one target dataset and infrequently in another background dataset (e.g., between two different days), where a CP is a significant change between two datasets. While there has been a substantial body of work on mining CPs [1], [2], [4], [6]–[10], the focus of these approaches is mainly on extracting all or a substantial subset of the patterns, where the number of extracted CPs may be combinatorially large in big, high-dimensional datasets. Thus, a major challenge for network managers and analysts is how to prioritize their analysis of these patterns to quickly recognize what is happening in their network.

In particular, some changes in traffic occur on a regular basis and exhibit periodic behavior over time, which we call *rare recurring patterns* (rare patterns/events for short). These rare events, which may overload the network and cause congestion, could be a regular radio competition, weekly scheduled system backups, or webinars. For example, a webinar that is held once a week may appear as unusual traffic to security analysts, and may be misinterpreted as a malicious attack. Given the resource limitations and the need for a rapid response, security analysts need to filter out these rare events and prioritize their focus on the *new patterns/events*, which are more likely to be a fault or malicious behavior. While rare events are still unusual, they have been previously identified by network managers and a longer-term response plan may already be in place. By separating out the rare recurring problems from the new problems that have not been seen before, network managers can better prioritize their effort. In our approach, we adapt a model of analysis based on time-windows, to detect *normal* and *change* windows. Change windows can be one of two different types: rare recurring change windows and new change windows (rare windows and new windows for short). Our aim is to detect new change windows, which contain new events of potential interest to security analysts, while filtering out rare change windows that contain rare, recurring events.

*Example 1:* In Fig. 1, the target dataset $D_t$ has four windows $\{w_1, \ldots, w_4\}$. $w_1$ has three records similar to the reference dataset $D_b$, called a normal window. However $w_2$ to $w_4$, each have five records that are very different to the reference dataset, and we call them change windows, e.g., $\{bcg\}$ and $\{dg\}$ are two CPs between $w_2$ and $D_b$, which are also repeated in $w_4$. Thus, we call $w_2$ and $w_4$ rare recurring windows. In contrast, $\{ge\}$ and $\{efh\}$ are two new CPs between $w_3$ and $D_b$, that do not repeat in other windows. Thus, we call $w_3$ a new window.

In this paper, we focus on *(i)* how to identify all rare and new windows in a sequence of records. In particular, we consider *(ii)* how to filter out the rare windows from the new windows. Our approach is based on using a special type of CP, called *jumping emerging patterns* (JEPs). JEPs are patterns that appear in a target dataset while being absent in a background dataset [11]. For example, in Fig. 1, pattern $\{bcg\}$

| | $w_1$ | | | $w_2$ | | | | | $w_3$ | | | | | $w_4$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **TID** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** |
| $\mathbf{D_t}$ | abc | a | ab | d | ac | bcg | bcg | dg | ge | efh | efh | ab | fh | a | ac | bcg | bcg | dg |

| **TID** | **1** | **2** | **3** |
|---|---|---|---|
| $\mathbf{D_b}$ | ab | abc | b |

Fig. 1. Example datasets with four windows.

is a JEP that occurs frequently (40%) in the target window $w_2$, but does not occur in the background dataset $D_b$. Similarly, the pattern $\{ge\}$ and $\{fh\}$ are two JEPs in $w_3$.

We propose a novel solution to identify rare events based on the approach of mining *second order contrast patterns* (SOCPs). While contrast patterns (referred to as first order contrast pattens (FOCPs) in this work) discover significant changes between two datasets, SOCPs discover significant changes between two sets of FOCPs. We propose a new algorithm, called *RCEP* (Rare recurring Change detection by Emerging Patterns), that extracts SOCPs over a sequence of records in two steps. In the first step, *RCEP* generates FOCPs to identify both types of change windows, but in this step we do not know which are rare windows and which are new windows. So, in the second step, using SOCP mining, *RCEP* filter outs rare windows and identifies any new windows. An interesting feature of *RCEP* is that it does not use the CPs themselves. Instead, it uses statistical measures of the aggregated CPs (length, frequency and size of the patterns) to discriminate new and rare windows. To the best of our knowledge, *RCEP* is the first algorithm that uses higher order contrast pattern mining to detect rare recurring events.

Our experimental results show that our proposed *RCEP* algorithm achieves considerably higher performance in terms of accuracy and F1-Score over an existing approach [2]. Our work also demonstrates the importance of SOCP mining in practice, and provides an effective method for finding such higher order patterns in large datasets.

**Contributions of this paper:** Our main contributions are: *(1)* We introduce a new concept of *second order contrast patterns* to discriminate between rare events and new events. *(2)* We propose a novel algorithm, called *RCEP*, to extract FOCPs and SOCPs over a sequence of records. By utilizing the SOCPs, *RCEP* is able to detect new events effectively. *(3)* We introduce several statistical features of CPs, which are used by *RCEP* to discriminate new events from rare events. *(4)* We show the practicality of our algorithm on three network traffic datasets. We evaluate our algorithm in terms of accuracy and F1-Score, and show that SOCPs are a powerful method for detecting the new events.

## II. Related Work

Many approaches have been proposed for CPM, such as border-based algorithms [2], [6], tree-based approaches [1], [9], and Zero-Suppressed Binary Decision Diagrams [10]. Despite the numerous applications presented in the literature for CPM [1], [3]–[5], [12], we are not aware of any study for distinguishing between rare events and new events. While there have been many works in the signal processing literature for periodic or recurring signal detection [13], [14], those methods focus on recurring patterns in the amplitude of multi-dimensional time series, where each dimension of the time series is usually a continuous, real-valued signal. In contrast, in our approach we analyse a sequence of records, where each record comprises a combination of discrete *items*, and we aim to find an *itemset* that identifies the combination of *items* that form a pattern. Moreover, while there have been many works in periodic pattern mining such as [15], those methods search for repeated patterns under normal conditions and they do not detect CPs. In contrast, our work aims to discover what is changing from normal, and then identifies rare patterns to distinguish them from any new patterns.

In [16], the authors proposed an algorithm to detect periodic CPs based on a time-window model. However, they simply check whether a single CP is periodic or not, while our approach is based on statistical measures for the aggregation of patterns, and by using SOCPs, we filter out rare windows and classify different windows as normal or new windows. In [17], the authors proposed the *OCLEP* algorithm to detect masquerader attacks using JEPs. They used the *average length* of patterns to classify new test instances. In [2], they proposed a new offline version for anomaly detection, named *OCLEP+*, which is very similar to the *OCLEP*, except they used the *minimum length* statistic as the cutoff threshold. However, our focus is not anomaly detection. In fact, our problem is a form of summarization of a sequential dataset. By using SOCPs on a sequence of records, we filter out the rare windows in order to summarize the new windows that contain a significant number of contrast patterns that we have not seen before. In addition, *OCLEP* and *OCLEP+* use the *BorderDiff* algorithm [6] to extract JEPs. *BorderDiff* only extracts the borders of JEPs and does not calculate the *support count* of each pattern, while *RCEP* uses the *EPClose* algorithm [7], which is a *tree-based* approach that also extracts the *support counts* of each pattern. Since *OCLEP+* detects changes by using contrast patterns, we use this method as a benchmark, and in Section V we explain how to adapt the offline *OCLEP+* method to our sequential scenario.

## III. Definitions and Problem Statement

Let $I = \{i_1, i_2, \ldots, i_M\}$ be the set of distinct items in a *transaction dataset D* where a *transaction T* is a non-empty set of items $\{i_1, i_2, \ldots, i_m\}$ and $i_j \in I$. A transaction may occur several times in *D*. An *itemset* or *pattern X* is any subset of $I$. We use the terms itemset and pattern interchangeably throughout this work. An itemset *X* is contained in a transaction *T* if $X \subseteq T$. We define a target dataset $D_t$ of $n_t$ as a sequence of $K$ non-overlapping time-windows $(w_1, \ldots, w_K)$, where each window is a batch of transactions. We denote the set of all transactions in a window $w_i$ as $\mathcal{TS}(w_i)$. The size of each window is the number of transactions it contains, denoted as $n_{w_i}$. The number of transactions in a window $w_i$ containing pattern *X* is called the *support count* of *X*, denoted as $SC(X, w_i)$. The *support* of itemset *X* is the fraction of transactions in a window $w_i$ that contain *X*, and is given by $supp(X, w_i) = \frac{SC(X,w_i)}{n_{w_i}}$.

*Definition 1:* The growth rate of a pattern *X* for a window $w_i$ compared to a background dataset $D_b$, denoted $gr(X, w_i)$, is defined as:

$$gr(X, w_i) = \begin{cases} 0 & supp(X, w_i) = 0 \ \&\ supp(X, D_b) = 0 \\ \infty & supp(X, w_i) \neq 0 \ \&\ supp(X, D_b) = 0 \\ \frac{supp(X,w_i)}{supp(X,D_b)} & otherwise. \end{cases}$$

*Definition 2:* A contrast pattern *X* is a pattern whose *support* is significantly different from a target window $w_i$ to a background dataset $D_b$. Given a growth rate threshold $\rho > 1$, *X* is a contrast pattern for a window $w_i$ if $gr(X, w_i) \geq \rho$.

*Definition 3:* A jumping emerging pattern (JEP) for a window $w_i$ compared to a background dataset $D_b$ is a subset of contrast patterns such that $supp(X, D_b) = 0$ and $supp(X, w_i) > 0$, i.e., $gr(X, w_i) = \infty$.

*Example 2:* In Fig. 1, the background dataset $D_b$ has 3 transactions with *transaction IDs* (TID) from 1 to 3. The target dataset $D_t$ has 18 transactions, i.e., $n_t = 18$, which is divided into 4 windows $w_1, \ldots, w_4$. All windows have the same time period, but the number of transactions in them are not the same. The size of all windows is 5, except $w_1$ whose size is 3, i.e., $n_{w_1} = 3$. Each transaction is a subset of the itemset $I = \{a, b, c, d, e, f, g, h\}$. Suppose $\rho = 1.5$. In $w_2$, the pattern $\{bc\}(2:1)$[1], is a CP, since its growth rate is $gr = 2/1$, which is bigger than the threshold $\rho = 1.5$; $\{c\}(3:1)$ is another CP with $gr = 3$. The patterns $\{bcg\}(2:0)$ and $\{g\}(3:0)$ are two JEPs of $w_2$, because these patterns did not occur in $D_b$.

We use the *EPClose* algorithm [7] to extract contrast patterns. *EPClose* generates all CPs from *closed patterns* [18], which are those patterns that have no proper supersets with the same support. Following the approach in [7], we use *closed jumping emerging patterns* (CJEPs), which are the most specific JEPs, i.e., patterns that are both closed and JEPs. We focus on JEPs because their discriminative power is much stronger than that of ordinary CPs. We use closed patterns as they can eliminate the most general patterns, and by reducing the redundant patterns they improve the scalability of our algorithm. We use $EPClose(D_t, D_b)$ to denote the returned set of CJEPs. For example in Fig. 1 for $D_t = w_2$, $EPClose(w_2, D_b) = \{bcg\}(2:0), \{dg\}(1:0), \{g\}(3:0), \{d\}(2:0)$. Here, we introduce a new type of CP, called *second order contrast patterns*. In Definition 4, the name SOCP could more precisely be *second order closed jumping emerging pattern*, but since we can apply SOCP to any type of CPs, and for simplicity, we call it a *second order contrast pattern*.

*Definition 4:* Given two sets of CJEPs, denoted as a background set $\Delta_b$ and a target set $\Delta_t$, a *second order contrast pattern* $X'$ is a closed jumping emerging pattern from $\Delta_b$ to $\Delta_t$ such that $gr(X', \Delta_t) = \infty$.

***Problem statement***: Given a background dataset $D_b$ as a normal reference dataset, and a target dataset $D_t = \{w_1, w_2, \ldots, w_k\}$, comprising a mixture of normal windows, rare windows $w_{rc}$, and new windows $w_{nc}$, we investigate *(i)* how to detect all $w_{rc}$ and $w_{nc}$ using FOCPs, denoted as $\Delta_{w_i} = EPClose(w_i, D_b)$ where $i = \{1, \ldots, k\}$. In particular, we consider *(ii)* how to discriminate between $w_{rc}$ and $w_{nc}$ in order to identify $w_{nc}$, using SOCPs, denoted as $\Delta'_{w_j} = EPClose(\Delta_{w_j}, \Delta_{w_i})$ where $\Delta_{w_j}$ and $\Delta_{w_i}$ are FOCPs and $j > i$.

## IV. Our Approach: RCEP

In this section we introduce our *RCEP* algorithm for detecting rare and new patterns. It consists of two main phases of training and testing. In the training phase, *RCEP* calculates some statistical measures of the aggregated CJEPs, and then a cut-off threshold is derived from these measures. In the test phase, the same statistics are computed for transactions of each window and labeled as *normal* or *new* according to the training phase cut-off threshold. One interesting feature of *RCEP* is that instead of saving all generated CJEPs on each window in memory, it only needs to save some statistics of the patterns, which reduces the memory requirements for *RCEP*. Another feature of *RCEP* is that by using SOCPs, it applies a second stage that filters out rare windows, and thus prioritises new windows. In this section, we first describe what kind of statistical measures we can derive from CJEPs. Then, we explain how to mine FOCPs and SOCPs using the *RCEP* algorithm.

### A. Observations on Statistical Measures of CJEPs

For the detection of rare and new patterns, utilizing all generated CJEPs can be prohibitive in terms of memory requirements. Instead, we propose to use several statistical measures of aggregated patterns. The measure used for FOCPs is the *maximum support count* of patterns in each window, and the measures stored for each collection of SOCPs are: the *number of generated CJEPs*, the *minimum length* of patterns, and the *variance of the length* of patterns in each window.

[1] Given $k \geq 1$ $\{a_1a_2...a_k\}(n:m)$ shows that the pattern $\{a_1a_2...a_k\}$ repeats $n$ times in $w_i$ and $m$ times in $D_b$

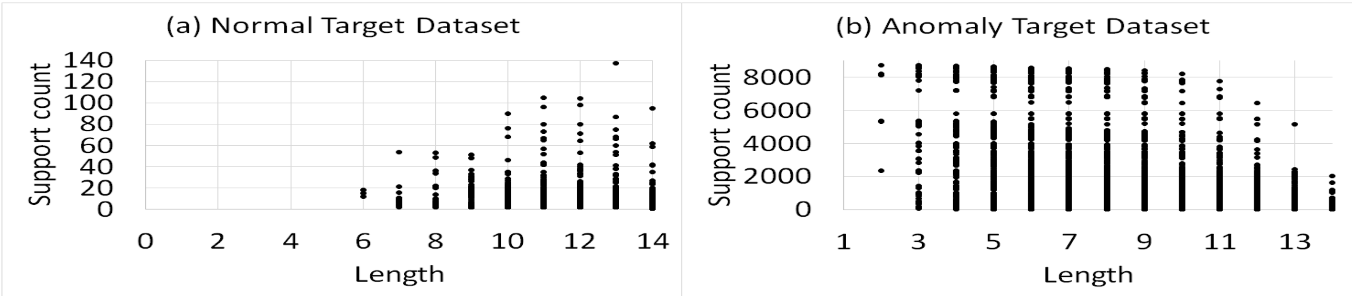


Fig. 2. Behavioral observation of CPs in Kyoto dataset.

These measures are defined below, and the reason for their selection is based on the following experimental observations.

Our premise is that CPs extract knowledge between *different classes of data* more strongly than *the same classes*. To verify this, consider an example based on the Kyoto network traffic dataset [19]. The Kyoto dataset consists of two classes of traffic: normal and anomalous. We run two separate experiments: in the first case we use *normal data* for both the background and target datasets, and in the second case we use normal traffic for the background dataset and anomalous traffic for the target dataset. The background dataset contains 10,000 normal transactions, selected randomly from 15 July 2007 of the Kyoto dataset, and the target dataset consists of 20,000 transactions in each experiment, selected randomly from the 16 July 2007 (other settings are the same as in the Experimental section). Fig. 2 shows the *length* (the number of *items* in each pattern) and the *support count* of each pattern in the two experiments. It is evident that these two measures have significant differences in the two experiments. When both datasets come from the normal class (Fig. 2(a)), the *minimum length* is 6, while in the experiment with two different classes (Fig. 2(b)), the *minimum length* is 2. In terms of *support count*, when both datasets have the same class the *maximum support count* is 137, but for the different classes it is 8700. The other interesting difference between the two experiments is the *number of patterns* and the *variance of the length*. When the target dataset is normal, the number of the generated patterns is 1650, which is far fewer than when the target dataset is anomalous traffic with 7957 patterns. In addition the *variance of the length* is 2.9 and 6.8 for the two experiments, respectively. These experiments lead to the following key observation:

*Property 1:* When $D_t$ and $D_b$ contain two different classes, $EPClose(D_t, D_b)$ tends to generate a higher number of short patterns with high support count and high variance in length. In contrast, if both datasets contain the same class, $EPClose(D_t, D_b)$ extracts a smaller number of long patterns with lower support count and low variance in length.

### B. Defining the Statistical Measures of CJEPs

In this section we study several statistical measures in more detail. For FOCP, we use the *maximum support count* of the aggregated patterns as a cut-off threshold.

*Definition 5:* Given a non-empty set $S = \{(X_1 : SC(X_1)), (X_2 : SC(X_2)), \ldots, (X_k : SC(X_k))\}$ of patterns, where $\{X_1, X_2 \ldots, X_k\}$ are CJEPs and $\{SC(X_1), \ldots, SC(X_k)\}$ are *support counts* of the patterns, the maximum support count metric is defined as follows:

$$maxSupCount(S) = max(SC(X_i) \mid i = 1, \ldots, k)$$

For SOCP, we propose a heuristic measure, named *novelty*, that is presented in Definition 6, to find the best threshold to identify new windows. The intuition behind this measure is that we aim to maximize the gap between the boundaries of normal patterns and new patterns. This novelty measure is based on the observations from the previous section that the *number of patterns* and the *variance of length* tends to be higher for anomalous patterns, while the *minimum length* of an anomalous pattern tends to be lower.

*Definition 6:* Given a non-empty set $S = \{X_1, X_2, \ldots, X_k\}$ of patterns, where $\{X_1, X_2 \ldots, X_k\}$ are CJEPs, the $novelty$ measure is defined as follows:

$$minLen(S) = min(|X_i| \mid X_i \in S \ , \ i = 1, \ldots, k)$$

$$varLen(S) = \sum_{i=1}^{k}(|X_i| - \mu)^2/k - 1 \ , \quad where \ \mu = \sum_{i=1}^{k}|X_i|/k$$

$$novelty(S) = \frac{k * varLen}{minLen}$$

### C. Mining First and Second Order Contrast Patterns

Fig. 3 gives an overview of the operation of *RCEP*, while the complete process of training and testing in *RCEP* is presented in Algorithms 1 and 2, respectively. The *RCEP* algorithm performs a two-stage training phase. In training, we only use the normal traffic. Let $D_b$ and $D_t$ be the background and the target datasets, respectively, both consisting of normal traffic. We divide $D_t$ into $K$ windows, according to Fig. 3. In the first stage of training, *RCEP* computes a set of FOCPs on each window, i.e., $\Delta_{w_i} = EPClose(\mathcal{TS}(w_i), D_b)$, in line 2 of Algorithm 1, and for each $\Delta_{w_i}$ the $maxSupCount$ measure

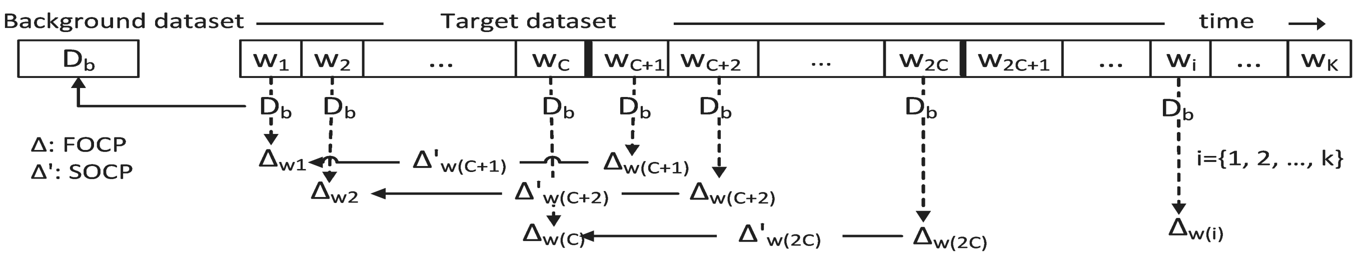


Fig. 3. *RCEP* overview.

**Algorithm 1:** Training phase

**Input:** $D_b$: reference dataset, $D_t = \{w_1, \ldots, w_K\}$: target dataset of normal instances with K windows, $r$: percentile for SOCP cutoff value, $C$: initial windows

**Output:** FOCPCutoff, SOCPCutoff

1 **for** $w_i \in \{w_1, \ldots, w_K\}$ **do**
2 $\quad \Delta_{w_i} = EPClose(\mathcal{TS}(w_i), D_b)$; `// FOCP`
3 $\quad$ Calculate maxSupCount($\Delta_{w_i}$) and add it to MaxSCList;
4 $\quad$ **if** $i > C$ **then**
5 $\quad\quad \Delta'_{w_i} = EPClose(\Delta_{w_i}, \Delta_{w_{(i-C)}})$; `// SOCP`
6 $\quad\quad$ Calculate $minLen(\Delta'_{w_i})$, $varLen(\Delta'_{w_i})$, $|\Delta'_{w_i}|$, and add them to MinLenList, VarLenList, PatNumList, respectively;
7 $\quad$ **end**
8 **end**
9 FOCPCutoff=$Max(SC \mid SC \in MaxSCList)$;
10 Sort the MinLenList in descending order and the VarLenList and PatNumList in ascending order, and get **r**-percentile of them as $mlen$, $maxVarLen$ and $maxPatNum$, respectively;
11 SOCPCutoff=$\dfrac{maxPatNum * maxVarLen}{mlen}$;

**Algorithm 2:** Testing phase

**Input:** $D_b$: reference dataset, $D_t = \{w_1, \ldots, w_K\}$: target dataset with K windows, $C$: initial windows, FOCPCutoff, SOCPCutoff

**Output:** Is $w_i$ normal or new window?

1 **for** $w_i \in \{w_1, \ldots, w_K\}$ **do**
2 $\quad \Delta_{w_i} = EPClose(\mathcal{TS}(w_i), D_b)$; `// FOCP`
3 $\quad FOCPMeasure(\Delta_{w_i}) = maxSupCount(\Delta_{w_i})$; `// Definition 5`
4 $\quad$ **if** $FOCPMeasure(\Delta_{w_i}) > FOCPCutoff$ **then**
5 $\quad\quad$ classify $w_i$ as change window; otherwise as normal window;
6 $\quad$ **end**
7 $\quad$ **if** $i > C$ **then**
8 $\quad\quad \Delta'_{w_i} = EPClose(\Delta_{w_i}, \Delta_{w_{(i-C)}})$; `// SOCP`
9 $\quad\quad SOCPMeasure(\Delta'_{w_i})$=$novelty(\Delta'_{w_i})$; `// Definition 6`
10 $\quad\quad$ **if** $SOCPMeasure(\Delta'_{w_i}) > SOCPCutoff$ **then**
11 $\quad\quad\quad$ classify $w_i$ as new window; otherwise as normal window;
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 **end**

is calculated according to Definition 5. After $C$ number of windows, called the *initial windows*, *RCEP* starts the second stage of training to mine the SOCPs using the generated FOCPs, i.e., $\Delta'_{w_i} = EPClose(\Delta_{w_i}, \Delta_{w_{(i-C)}})$, in line 5 of Algorithm 1. In line 6, *RCEP* computes the *number of CJEPs*, the *minimum length* and the *variance of the length* of the patterns for each set of SOCP, and saves them in three separate lists. After extracting FOCPs and SOCPs for all windows, in the last step, *RCEP* calculates the cut-off thresholds from the saved lists.

For the FOCP cut-off threshold, we simply return the maximum of the $maxSupCount$ list as the *FOCPCutoff* threshold. For SOCP, further steps are needed to select an appropriate cut-off threshold (lines 10-11). Specifically, we first sort the variance length list in ascending order. Let $m$ and $n$ be the minimum and maximum of the *variance length*s. Then we can select any value of $r$ between $m$ and $n$, $m \leq r \leq n$ as a cut-off value. The same mechanism is applied to the *pattern number* list. For the *minimum length* list, it is sorted in descending order. Finally, in line 11, the *novelty* measure is calculated according to Definition 6, and is returned as *SOCPCutoff* threshold.

Some applications demand low false negative (FN) rates (the number of positive samples that are classified as negative), or low false positive (FP) rates (the number of negative samples that are identified as positive). A higher cut-off threshold $r$ (i.e., closer to the maximum) leads to a higher FN rate, and many windows may be detected as normal windows, even though they are actually anomalous. A lower cut-off threshold (i.e., closer to the minimum) causes an increase in the FP rate, resulting in normal windows being identified as anomalous. For recurring problems, since it is highly desired to reduce the number of normal windows that are identified as change windows (a low FP), we select a higher cut-off value for $r$.

The testing phase (Algorithm 2), similar to the training phase, also consists of two stages of FOCP and SOCP mining. The background dataset $D_b$ consists of the normal traffic, and the target dataset $D_t$, as our test dataset, consists of a mixture of both normal and anomalous traffic. Similar to the training phase, $D_t$ is a sequence of $K$ windows. In the first stage, *RCEP* applies $EPClose$ to each window and derives a set of FOCPs in line 2. In the second stage, after $C$ number of *initial windows*, *RCEP* mines the SOCPs, in line 8. By filtering out the rare windows using SOCPs, windows are classified as normal or new according to the statistical measures in Definition 6.

*RCEP* requires a small amount of space only for saving $C$ sets of CPs and the list of statistical measures (besides that used by the training data). The parameter $C > 1$ depends on the application and domain knowledge, and it needs to reflect the typical variation that we see on a regular basis. In our setting, we would like to see a regular cycle of normal behaviour over a 24 hour period to analyse a day of network traffic, hence we choose $C = 24$ corresponding to the 24 hours of a day. In Algorithm 1 the measures are sorted and the cut-off is at the $r^{th}$ percentile, where $r = 95\%$ in our experiments.

## V. Experimental Results

To evaluate the effectiveness of the proposed *RCEP* algorithm, we compare it with *OCLEP+* [2] (we used the source code from the author, and both algorithms are in Java). For empirical evaluation, three benchmark network traffic datasets are used, namely Kyoto 2006+ [19], KDD'99 and BGU from the UCI data repository. In the Kyoto dataset, we used the 14 conventional features [19] of 4 days of 15-18 July 2007. For KDD'99, we randomly selected the 10 features according to [20]. In BGU, we used all 23 main features in time windows of 1 minute, and the traffic of three devices ("Ennio doorbell", "Ecobee thermostat", and "SimpleHomeXCS7-1003-WHT security camera") are used for evaluation. We combined and shuffled the benign traffic of these devices. For attack traffic, different attack types of these devices from the *gafgyt* botnet were used. We combined the *combo* attack of doorbell, *scan* attack of thermostat, and *junk* attack of security camera. After combining and shuffling the benign and attack traffic we used it as BGU dataset. The continuous features of the network traffic datasets were discretized using the *equal-frequency unsupervised discretization* method, and the number of bins for discretization are 4, 5 and 2 in the Kyoto, KDD'99 and BGU, respectively. Also, the number of generated items after discritization are 108, 38 and 46 in the Kyoto, KDD'99 and BGU, respectively. Each dataset consist of two classes of normal and anomalous traffic. All experiments were run on a 2.6GHz CPU with 16GB of memory running Windows 7.

**Evaluation scenario:** At present, there is no existing traffic dataset with labelled instances of recurring events that is available for use as a benchmark for our algorithm. To overcome this, we have made use of several widely used benchmark traffic datasets as the background traffic for our evaluation, and introduced some rare and recurring events. This allows us to evaluate our method under controlled conditions to systematically assess its accuracy. We used the transactions of the above mentioned datasets to create a data stream for each dataset. For example, the Kyoto dataset has nearly 500,000 transactions and we used those transactions to create our designed data stream for different days as follows. In the training phase, 200 and 54,000 normal transactions were selected as the background and target datasets, respectively. The target dataset is divided into 270 windows, each containing 200 transactions. The *initial window* was set to ($C = 24$). To select the *r-percentile* parameter we conducted a grid-search over the range $r = 80\%$ to $r = 100\%$, and found that the best results were obtained over all three datasets for $r = 95\%$. In the testing phase, the same normal background dataset was used, but for the target dataset we used 192 windows that consist of a mixture of normal and anomalous traffic, equivalent to 8 days worth of data. It is worth noting that the transactions of the target dataset in the training phase do not overlap with the normal traffic of the target dataset in the test phase. We also set $C = 24$. The total size of the target test dataset is 66,400 transactions as follows: We injected a mixture of normal and anomalous traffic into hours (windows) 2 and 4 of each day as rare recurring traffic. So, in total, we have 16 rare windows. We also injected a mixture of normal and anomalous traffic in 12 random windows of 40, 42, 55, 57, 92, 94, 103, 106, 156, 158, 160, 174 as *new* windows. Both the rare and new windows have the same size with 1200 transactions, and both contain 80% of anomalous traffic. For rare traffic we used a *recurring percentage* of 80%. The *recurring percentage* shows what proportion of the rare traffic would be repeated in each rare window. The other remaining windows, i.e., 164 windows, each consist of 200 normal transactions (normal windows).

**Adapting *OCLEP+* to our scenario**: The *OCLEP+* algorithm is an anomaly detection method. It extracts only FOCPs, and then classifies each test instance according to its *minimum length* as normal or anomalous. However, our approach is based on SOCPs to classify each window as normal or new according to aggregated patterns. So, to adapt *OCLEP+* to our scenario, in each window, we find the minimum length for each test instance. This produces a set of minimum lengths. We sort them in decreasing order, and select the minimum length in the $p^{th}$ percentile of this set as the decision threshold. This means that if $p\%$ of test instances in a window are normal, then we classify that window as normal. We implemented *OCLEP+* for SOCP in a similar way. To be as fair as possible to *OCLEP+*, we set different decision thresholds $p$ not only for FOCPs and SOCPs separately, but also for each dataset separately to get the best possible results. The value of $p$ for FOCPs and SOCPs is 20% and 40% for Kyoto, 30% and 10% for KDD'99, and 30% and 45% for BGU, respectively.

We measure performance in terms of *accuracy* and *F1-Score*. *Accuracy* is the ratio of correctly predicted cases to the total cases, and reflects how many windows are classified correctly. *F1-Score* is a harmonic mean of *precision* and *recall*, and gives a better measure of the incorrectly classified cases than *accuracy*. *Precision* is a measure of the correctly

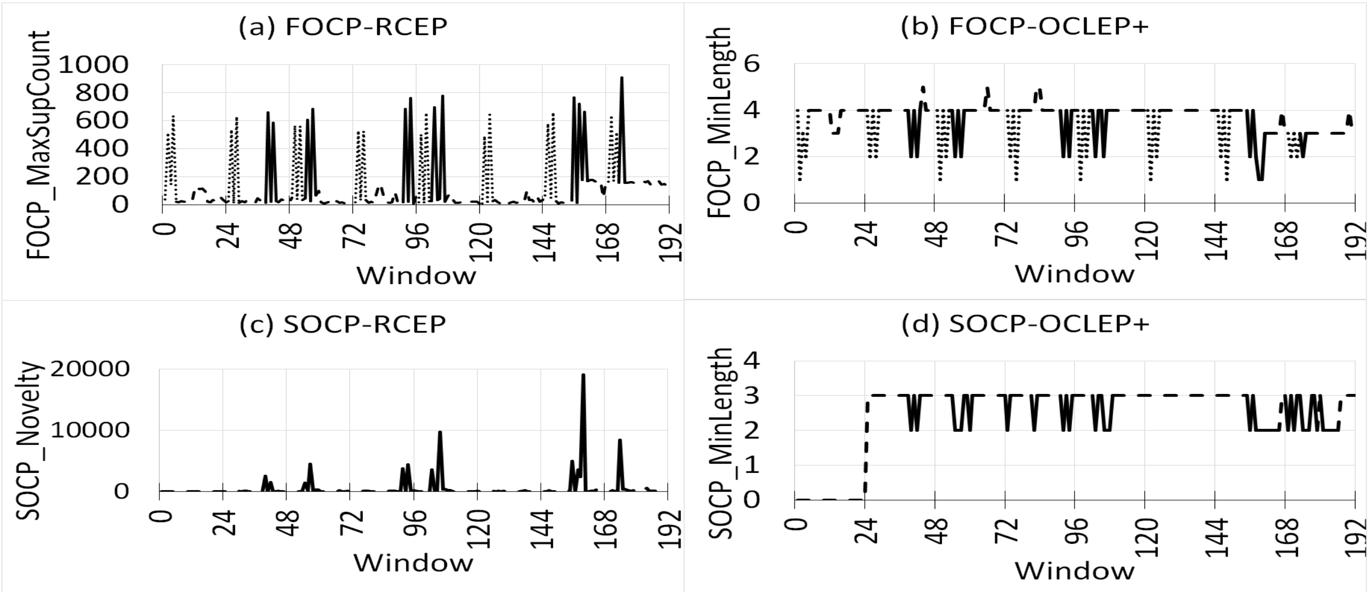


Fig. 4. Results for FOCPs and SOCPs (solid lines show *new* windows).

TABLE I
PERFORMANCE COMPARISON FOR SOCP (ACC=ACCURACY)

| Method | RCEP | | | | OCLEP+ | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Precision | Recall | F1-Score | Acc | Precision | Recall | F1-Score | Acc |
| **Kyoto** | 86 | 100 | 92 | 99 | 34 | 92 | 50 | 87 |
| **KDD'99** | 92 | 100 | 96 | 99 | 19 | 83 | 31 | 73 |
| **BGU** | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

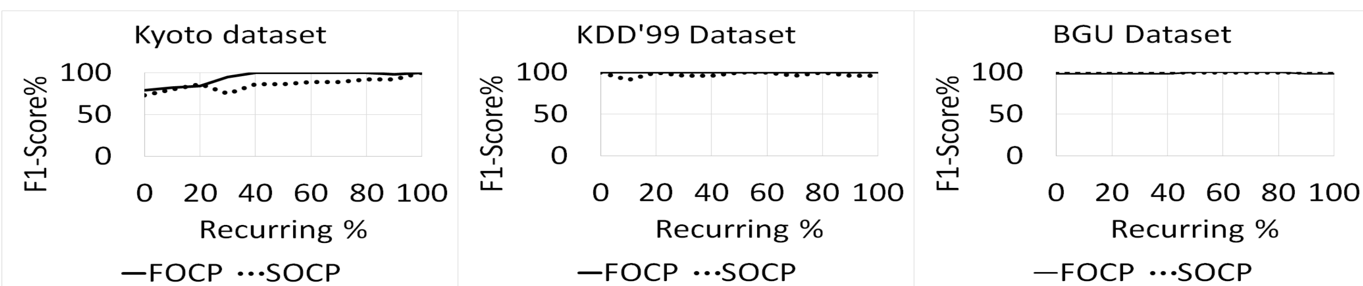


Fig. 5. F1-Score comparison for FOCP and SOCP in *RCEP*.

identified positive cases to all the predicted positive cases, *recall* is the correctly identified positive cases to all the actual positive cases, and *F1-Score*= $\frac{2*precision*recall}{precision+recall}$.

The statistical measures of *RCEP* and *OCLEP+* on a sequence of windows are shown in Fig. 4 for the Kyoto dataset. Peaks with dotted lines correspond to rare windows, peaks with solid lines correspond to *new* windows, and the dashed lines are normal windows. In *RCEP*, the cutoff thresholds for FOCP and SOCP in Kyoto are 184 and 500.18, respectively. All windows above these thresholds are identified as changes, and those below the thresholds are labeled as normal. It is clear that FOCPs are able to detect both rare and new windows in the target dataset, but are unable to discriminate between these two types of windows. In contrast, Fig. 4(c), shows that SOCPs are able to accurately filter out the rare windows and detect the new windows. In *OCLEP+*, the cutoff thresholds of *minimum length* for both FOCP and SOCP are 2.05 in the Kyoto dataset. So, all windows with a value less than these thresholds are identified as changes. Fig. 4(d) shows that *OCLEP+* is not able to discriminate between these two types of changes. An interesting point is that the decision gap between normal and change windows in *RCEP* algorithm is much larger than *OCLEP+*, which implies that there is a wider margin for classifying decisions in *RCEP*.

The same results were observed for the two other datasets. Due to space limitations we briefly summarize their SOCP results in Table I. In terms of the FOCPs of these datasets, both algorithms detect all change windows, i.e., a *recall* of 100%. However, in terms of precision, FOCP detects many false positive and is not able to discriminate between rare and new windows. In terms of SOCPs, Table I shows that *RCEP* considerably outperforms *OCLEP+* in the Kyoto and KDD'99 datasets. The main reason for *RCEP*'s performance is its use of the *novelty* measure, which combines several important properties of CPs as identified in Property 1.

In the above results, we used 80% of repeated traffic in the rare windows. We tested the effect of varying the *recurring percentage* from 0% (all traffic is new in each rare window) to 100% (the same traffic is used in all rare windows) as shown in Fig. 5. We observe that the F1-Score of *RCEP* is relatively consistent, ranging from 79% to 100% for FOCPs and from 73% to 100% for SOCPs in all three datasets (while the lowest accuracy is 92% and 95% for FOCPs and SOCPs, respectively). This demonstrates that the performance of *RCEP* is relatively insensitive to the choice of this parameter.

## VI. Conclusion and Future Work

In this paper, we presented a novel approach to discriminate between rare recurring windows and new windows in network traffic based on the approach of mining second order contrast patterns. We proposed a new algorithm, called *RCEP*, that uses several statistical measures of contrast patterns to filter out rare windows on a sequence of transactions. We demonstrated that the *RCEP* algorithm can achieve high accuracy and F1-Score in comparison with an existing approach. As future work, we will evaluate the use of *RCEP* for real-life data streams. Also we will investigate the use of the *RCEP* approach and second order contrast patterns for outlier detection.

## References


[1] E. A. Chavary, S. M. Erfani, and C. Leckie, "Summarizing significant changes in network traffic using contrast pattern mining," in *CIKM*, pp. 2015–2018, 2017.

[2] G. Dong and S. K. Pentukar, "OCLEP+: One-class anomaly and intrusion detection using minimal length of emerging patterns," *arXiv:1811.09842*, 2018.

[3] C. Gao, L. Duan, G. Dong, H. Zhang, H. Yang, and C. Tang, "Mining top-k distinguishing sequential patterns with flexible gap constraints," in *WAIM*, pp. 82–94, 2016.

[4] S. Ghosh, J. Li, L. Cao, and K. Ramamohanarao, "Septic shock prediction for ICU patients via coupled HMM walking on sequential contrast patterns," *J. Biomed. Inform*, 2017.

[5] L. Li, *Contrast Data Mining of Multi-source Heterogeneous Trajectory Data*. PhD thesis, 2020.

[6] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *SIGKDD*, pp. 43–52, 1999.

[7] E. AlipourChavary, S. M. Erfani, and C. Leckie, "Improving scalability of contrast pattern mining for network traffic using closed patterns," in *arXiv:2011.14830*, 2020.

[8] H. S. Pham, G. Virlet, D. Lavenier, and A. Termier, "Statistically significant discriminative patterns searching," in *DaWaK*, pp. 105–115, 2019.

[9] Y. Kameya, "Towards efficient discriminative pattern mining in hybrid domains," *arXiv:1908.06801*, 2019.

[10] E. Loekito and J. Bailey, "Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams," in *SIGKDD*, pp. 307–316, 2006.

[11] J. Li, K. Ramamohanarao, and G. Dong, "The space of jumping emerging patterns and its incremental maintenance algorithms," in *ICML*, pp. 551–558, 2000.

[12] K. Maeda and Y. Kameya, "Associative classification using common instances among conflicting discriminative patterns," in *TAAI*, pp. 1–6, 2019.

[13] N. A. Huynh, W. K. Ng, A. Ulmer, and J. Kohlhammer, "Uncovering periodic network signals of cyber attacks," in *VizSec*, pp. 1–8, 2016.

[14] F. S. Passino and N. A. Heard, "Classification of periodic arrivals in event time data for filtering computer network traffic," *SC*, pp. 1241–1254, 2020.

[15] P. Fournier-Viger, P. Yang, R. U. Kiran, S. Ventura, and J. M. Luna, "Mining local periodic patterns in a discrete sequence," *Inf. Sci.*, pp. 519–548, 2020.

[16] C. Loglisci and D. Malerba, "Mining periodic changes in complex dynamic data through relational pattern discovery," in *NFMCP*, pp. 76–90, 2015.

[17] L. Chen and G. Dong, "Masquerader detection using OCLEP: One-class classification using length statistics of emerging patterns," in *WAIM*, pp. 5–5, 2006.

[18] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient mining of association rules using closed itemset lattices," *IS*, pp. 25–46, 1999.

[19] J. Song, H. Takakura, and Y. Okabe, "Description of kyoto university benchmark data," *www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf, 2006.*

[20] A. A. Olusola, A. S. Oladele, and D. O. Abosede, "Analysis of KDD99 intrusion detection dataset for selection of relevance features," in *WCECS*, pp. 20–22, 2010.

[21] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiotnetwork-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, pp. 12–22, 2018.