Condensed Representations of Changes in Dynamic Graphs through Emerging Subgraph Mining

Angelo Impedovo[a,*], Corrado Loglisci[a,b], Michelangelo Ceci[a,b], Donato Malerba[a,b]

[a] *Department of Computer Science, Universita' degli Studi di Bari Aldo Moro, Bari, Italy*
[b] *CINI, Consorzio Interuniversitario Nazionale per l'Informatica, Bari, Italy*

Abstract

Change mining is one of the main subjects of analysis on time-evolving data. Regardless of the distribution of the changes over the data, often the algorithms return very large sets of results. In fact, one class of algorithms designed for change mining is based on pattern mining, which notoriously suffers from the problem of a huge number of returned patterns. Moreover, the complexity of some types of data, like dynamic graphs, could make the size of the final changes even larger, which makes interpretation difficult or even impossible. This paper represents the first attempt, to our knowledge, to build condensed representations of changes from dynamic graphs. We study changes captured with the pattern (subgraph) mining framework and focus on the discovery of subgraphs able to *i)* represent evident changes and *ii)* convey graph-based information that is not already expressed by other subgraphs. To do this, we revise an existing approach by introducing the notion of *emerging* subgraphs, used to remove uninteresting changes and the notions of closed and maximal subgraphs, used to remove redundant changes. Experiments performed on real-world dynamic graphs show that the condensed representations maintain the accuracy levels of the original approach and often offer a loss-less representation of the detected changes.

Keywords: Emerging Subgraph Mining, Closed Subgraph Mining, Maximal Subgraph Mining, Dynamic Graphs

1. Introduction

Recent advances in data modeling and processing have consolidated the data representation in form of graphs (or networks) as the way to straightforwardly handle complex data, characterized by structured objects and interconnected entities. With the adoption of technologies, able to collect uninterruptedly time-ordered measurements in many real-world application domains (such as communication, mobility and the Internet), the graph-based formalism has been extended to the temporal dimension through so-called *dynamic graphs* [1]. Their main feature is modeling domains as they are observed over time and depicting their time-variability [2]. Dynamic graphs can represent an instantaneous status of a domain as a new graph snapshot, in which nodes and edges can appear, disappear or exhibit differences, compared with the previous snapshot.

For instance, in the scenario of telecommunication, dynamic graphs can represent the flow of call data records produced over time in a geographic area (Figure 1). The nodes denote users, while the edges denote the way in which two users interact (e.g. the SMS and telephone conversation). There, we could register the disappearance of the edges between two users due to malfunctions of the communication infrastructure or insertion of new edges denoting the new ways of communication.

When mining changes on graph data, we should take additional issues into account, with respect to time-series or transactional data [3]. First, changes may regard the whole structure of the graph or only some nodes and edges, because they are likely to be associated to variations of the majority of the domain or just small portions [4]. Second, changes do not only affect the structural aspects, but may have impact also on the descriptive ones. Indeed, the properties of nodes and edges may change, while the structure remains unaltered. Third, regardless of where the changes occur, not all of them are worthy of interest. Indeed, episodic or sporadic evolutions may be of little relevance for the problem at hand, even if they have impact on the whole graph. On the contrary, changes occurring on a portion of the graph, but regularly repeated over time, can be of greater interest [2]. For instance, in Figure 1, the portion of the graph denoted as "subgraph 1" regularly occurs in the time-interval [7:00,7:30], but it is replaced by the portion "subgraph 3" in the time-interval [7:45,8:15]. This means it is always present in the first part, but completely absent in the second part.

An approach taken in the literature is the detection of changes which occur in the frequent subgraphs, as a mirror of what happens in the underlying graph data [5, 6, 3, 7]. However, the subgraphs returned by this kind of solution generally have the brittleness to convey redundant infor-

*Corresponding author
Email address: angelo.impedovo@uniba.it (Angelo Impedovo)

mation, in the sense that many changes can be pleonastic. They might not add information to that already provided by other subgraphs. For instance, in Figure 1, we note that the change of the subgraph 3 (it is absent in [7:00,7:30], but always present in [7:45,8:15]) can also be observed for subgraph 4, which, however, adds information (short messaging between A and B). In fact, subgraph 3 is isomorphic to an induced subgraph of subgraph 4. So, we would prefer to condense subgraph 3 and subgraph 4 to the sole change represented by subgraph 4. Moreover, we observe that subgraph 2 appears only once in [7:00,7:30], which basically seems episodic and therefore of little relevance.

It is evident that not all the changes are equally important and some are just superfluous with respect to the others. Therefore, it becomes necessary to find a condensed representation of change-related subgraphs.

We propose to do so by *i)* detecting *evident* (or stronger) changes, like those represented by subgraph 1, while discarding those represented by subgraph 2, and *ii)* pruning *superfluous* changes, that is, changes which are already represented by others, like the ones in subgraph 3. Thus, we prefer a 'generative' approach, which builds larger and larger subgraphs by combining shorter ones, with respect to a mere partitioning of the whole graph into smaller substructures, which would turn out to be NP-hard. To detect evident changes, the approach first discovers subgraph patterns and then, from these, searches for the *emerging* ones. On the other hand, to prune superfluous changes, the approach removes redundancies present in the structure of the sub-graphs and in their statistical properties. This is done through two alternative strategies, which implement the notions of *closed subgraph* and *maximal subgraph*. The first strategy works before the emerging subgraphs have been mined. More precisely, it mines frequent and infrequent subgraphs and, from these, it selects the closed and maximal ones. Finally, it returns the emerging subgraphs. The second strategy works after the emerging subgraphs have been mined and acts directly on the emerging graphs by identifying the closed emerging subgraphs and maximal emerging subgraphs. The two strategies return different condensed representations of the changes because they work on different spaces of the subgraphs. In the first case, we derive a concise set of the frequent subgraphs and infrequent subgraphs, which could omit candidate changes. On the contrary, in the second case, the redundancies are removed only after the emerging subgraphs have been generated. Thus, we expect that the first strategy generates a set of change-related subgraphs, which is more condensed than the one extracted by the second solution.

We sum up the specific contributions of the paper in the following list:

- we investigate the two-fold problem of detecting evident changes in dynamic graphs and building a condensed representation;
- we deal with the problem by exploiting the framework of subgraph pattern mining, which yields an abstract form of underlying graph data;
- we propose a general computation solution, which considers emerging sub-graphs as evident changes, and embeds the notions of closed subgraphs and maximal subgraphs to prune redundancies. The computational solution is boiled down into two alternative strategies, which differ from each other in the order with which the operation of evident change detection and redundancy removal are performed;
- we study the viability of these solutions on synthetic and real-world dynamic graphs, by evaluating the conciseness of the resulting change-related subgraphs.

The rest of the paper is structured as follows. Section 2 introduces basic notions for the scientific problem studied. In Section 3 we present the two afore-mentioned strategies and report the respective algorithmic details. Section 4 illustrates the experiments performed on synthetic and real-world data and the results in terms of conciseness and efficiency of the changes. The literature related to the present work is discussed in Section 6. Finally, some conclusions are drawn in Section 7.

2. Background and basics

In this section we provide the notations and concepts necessary to formulate the problem at hand.

2.1. Basics

Let $D = \langle G_1, G_2, \ldots, G_n, \ldots \rangle$ be the sequence of graph snapshots [8]. Each G_i corresponds to the snapshot observed at the time-point τ_i and can be formally defined as a labeled graph with labeled edges $G_i \subseteq N \times N \times L$, where N is the set of nodes and L is the set of edge labels.

A *landmark window* $W' = [\tau_i, \tau_m]$ is the sequence of consecutive time-points $\{\tau_i, \ldots, \tau_m\}$ and is built by enqueuing the sequence of time-points $\{\tau_{n+1}, \tau_m\}$ to the window $W = [\tau_i, \tau_n]$, $\tau_i < \tau_n < \tau_m$ [9]. For simplicity, we use the notion of landmark window to refer also to the sequence of snapshots $W' = [G_i, \ldots, G_m]$ observed at the time-points $\{\tau_i, \ldots, \tau_m\}$, respectively, so, width $|W'|$ equals $m - i + 1$, that is, the number of snapshots (and time-points) collected in W'.

Let $P = \{(u, v, e) \mid u, v \in V, e \in E\}$ be a subgraph pattern (or simply a subgraph), we denote as $cov(P, W) = \{G \in W \mid P \subseteq G\}$ the set of graph snapshots in which P is present, that is, an isomorphism exists from P to an inducted subgraph of G_i.

The count of the snapshots (out of all the snapshots) in which P occurs quantifies the relative frequency of P in a window W, which we call *support* of a subgraph P in W and denote as $sup(P, W)$. Whenever the value of $sup(P, W)$ in W exceeds a user-defined threshold $minSUP$ ($sup(P, W) \geq minSUP$), P is *frequent* in W, otherwise P is infrequent ($minSUP \in [0, 1]$, $sup(P, W) \in [0, 1]$). We

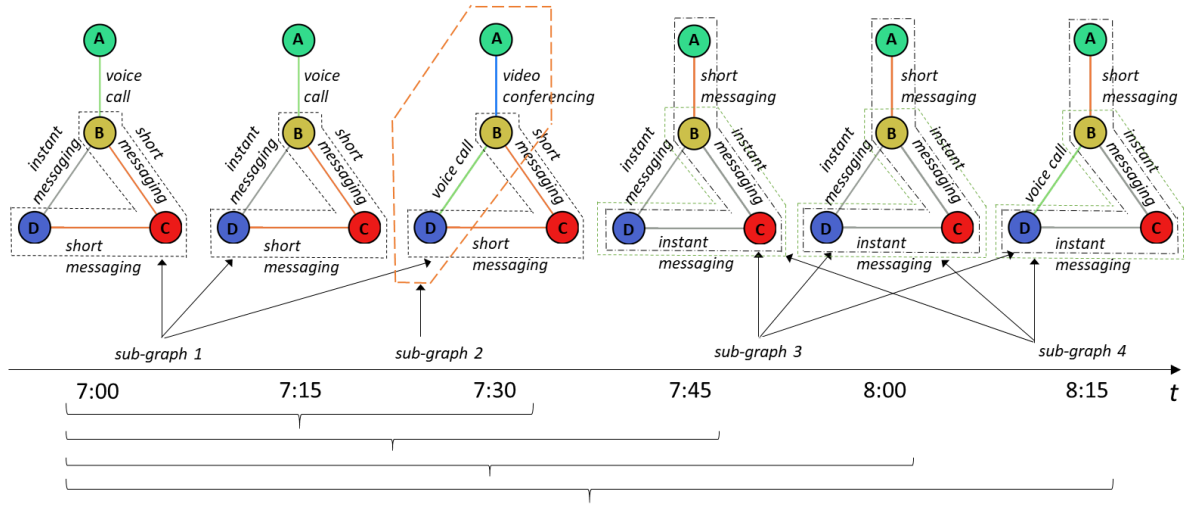


Figure 1: Representation of the scenario of the telecommunication in the form of evolving network.

denote the set of all the frequent subgraphs in the window W as F_W.

The construction of the subgraphs is a process which is basically identical to the discovery of item-sets [10]. It relies on a lattice, a structure which arranges subgraphs in a partial order relation, that is, a generality order $\geqslant$ similar to the subset-containment relation between two sets. The lattice is arranged over generality levels: given P and Q two subgraphs, we say that P is more general than Q ($P \geqslant Q$), if $P \subseteq Q$. At the same level there are subgraphs having the same number of edges (u, v, e), while, at the $(k{+}1)$-th level there are subgraphs having (k+1) edges, that is, one edge more than the subgraphs at the k-th level. So, the generality order $\geqslant$ does not hold on the subgraphs of the same level, but holds on those of different levels. Finally, the generality order $\geqslant$ is characterized by the anti-monotonicity property with respect to the support. In the sense that, if the set P is infrequent in window W, the subgraphs that are more specific will be infrequent in the W too. It should be noted that the generality order between the subgraph P and subgraph Q ($P \subseteq Q$) corresponds to having P isomorphic to an induced subgraph of Q.

Three categories of subgraphs are fundamental to detect non-redundant changes in this work, *emerging subgraphs*, *closed subgraphs* and *maximal subgraphs*, which we formalize in the following:

Definition 1 (Emerging subgraph (ES))**.** *Let P be a subgraph and W and W' be two landmark windows. P is emerging if the associated growth-rate $GR(P, W, W')$ exceeds a user-defined threshold minGR, where*

$$GR(P, W, W') = \frac{max(sup(P, W), sup(P, W'))}{min(sup(P, W), sup(P, W'))} \in [1, \infty) \quad (1)$$

The set of the emerging subgraphs between windows W and W' according to minGR is denoted as $ES(W, W')$.

The notion of the emerging subgraph is inspired by the one of emerging patterns [11], which was originally formulated to work on stationary data and address a predictive task by means of a discriminative approach. In the current manuscript, ES operates in another setting (that is, time-changing data) and for a descriptive task, that is, quantifying the variations, relative to one subgraph, between two landmark windows of graph snapshots.

The *growth-rate* $GR(P, W, W')$ of a subgraph P symmetrically estimates the increase and decrease of the support of P from W to W', as suggested by the following properties:

Lemma 2.1. *Let P be a subgraph and W and W' two landmark windows. If the support of P increases from W to W', then $GR(P, W, W') > 1$.*

Proof. The support increase from W to W' means that $sup(P, W') > sup(P, W)$. Then $max(sup(P, W), sup(P, W')) = sup(P, W')$ and $min(sup(P, W), sup(P, W')) = sup(P, W)$. It follows that $GR(P, W, W') = \frac{sup(P,W')}{sup(P,W)}$, which is always greater than 1, QED. □

Lemma 2.2. *Let P be a subgraph and W and W' two landmark windows. If the support of P decreases from W to W', then $GR(P, W, W') > 1$.*

Proof. The support decrease from W to W' means that $sup(P, W) > sup(P, W')$. Then, $max(sup(P, W), sup(P, W')) = sup(P, W)$ and $min(sup(P, W), sup(P, W')) = sup(P, W')$. It follows that $GR(P, W, W') = \frac{sup(P,W)}{sup(P,W')}$, which is always greater than 1, QED. □

However, the notion reported above introduces a bias that tends to identify changes especially due to the statistical properties, with the risk of neglecting the structural

aspect. Indeed, we can discover emerging subgraphs that are frequent in both W and W', but that would provide no structural change of the whole graph (having many occurrences in both windows). For this reason, we further refine the set of emerging subgraphs to only those which are frequent in only one window among W and W'. In particular, subgraphs which are frequent in W and have become infrequent in W' and, vice-versa, those which have become frequent in W' and are infrequent in W. This decision, however, allows us to preserve as much topological information as possible, and guarantees the removal of any redundancies. This is done through the following two definitions:

Definition 2 (Closed subgraph (CS)). *Let W be a window, S be a sub-space of the lattice built on W, $P \in S$ be a subgraph: P is closed iff no subgraph $Q \in S$ exists, and Q is more specific than P ($P \geqslant Q$), such that $cov(Q, W) = cov(P, W)$.*

Intuitively, P is closed if there is no subgraph Q that is present in the same snapshots in which P is present (with the same support) and that are more specific than P. Clearly, P and Q will occur in the same sub-set of snapshots.

It should be noted that Definition 2 differs from the concept of a closed graph pattern [12, 13], which is instead oriented to a quantitative notion of redundancy and is used to prune a subgraph, if it occurs in a number of snapshots identical to the number of its super-graphs. In practice, in [12, 13] it is sufficient that the support of P is identical to that of Q to consider Q as closed, even if they occur in different snapshots. This is clearly not desirable in time-changing scenarios, where the snapshots are temporally ordered. On the contrary, in the current manuscript we consider a notion of redundancy which encompasses the topological aspect and prunes a subgraph, if it expresses the structural information already conveyed by more specific subgraphs, and it is present in the same snapshots.

Definition 3 (Maximal subgraph (MS)). *Let W be a window, S be a sub-space of the lattice built on W, $P \in S$ be a subgraph: P is maximal iff no subgraph $Q \in S$ exists, and Q is more general than P ($P \geqslant Q$).*

Intuitively, the notion formulated above requires less and accounts for only the topological aspect. Thus, we consider the most general subgraph of a sub-space of the lattice to be a condensed representation, by consequently pruning the most general subgraphs, without particular conditions in the supporting snapshots.

Therefore, it is evident that the notion of CS allows us to build a loss-less condensed representation of the subgraphs, since it preserves structural and "frequentistic" information of the redundant subgraphs we discarded. This is something that the notion of MS cannot guarantee because it neglects the information related to the supporting snapshots. Consequently, the set of the changes built by Definition 3 is generally more concise than the one derived with Definition 2.

3. Condensed representations of change-related subgraphs

The combinatorial search used in many sub-graph mining algorithms leads to generating such a huge number of patterns that seeking emerging subgraphs might be time-consuming for the machines and virtually impossible for humans. This justifies an initiative focused on returning only a concise set.

The computational solution we propose relies on two algorithmic decisions used also in [3], that is, *i)* a lattice of subgraphs to represent the whole graph in a computational form, and *ii)* two time-windows (W and W') to collect the snapshots and keep the subgraphs updated with the new occurrences. The lattice is built only once (that is, at the beginning of the analysis) and initialized with the snapshots included in window W. Then, we determine the supporting snapshots $cov(Q, W)$ of each subgraph P in W, so that we can distinguish the frequent ones from the infrequent ones, with respect to the threshold $minSUP$. Next, we use a window W' to add a block Π of new snapshots to those of W and determine the supporting snapshots on a larger window (W'). Finally, we revise the support values for all the subgraphs.

This update could result in increasing or decreasing the support of the subgraphs and lead frequent subgraphs in W (F_W) to becoming infrequent in W' and infrequent subgraphs in W to becoming frequent in W' ($F_{W'}$). These feed two alternative strategies, whose common purpose is removing redundancies present in the subgraphs that have been previously updated. They differ from each other in the order in which they perform the two operations, that is, the identification of emerging subgraphs and the generation of their condensed representation. To identify the emerging subgraphs, both strategies apply Definition 1 to the subgraphs previously updated, without necessarily visiting the whole lattice via the property of anti-monotonicity. To build a condensed representation, the strategies adopt Definition 2 and Definition 3. In the following we report further algorithmic details.

3.1. Discovery of emerging subgraphs from closed/maximal subgraphs

The first strategy prunes redundancies from the updated subgraphs and then discovers evident changes. To do this it identifies the CS (or MS) from the sub-space of the lattice containing the subgraphs which were frequent (infrequent) in W and become infrequent (frequent) in W'. More precisely, it first visits the subgraphs which have the smallest number of edges (the first level of the lattice). Then it moves downwards by exploring (the sub-spaces of) subgraphs which are more specific (according to the

generality order $\geqslant$) than the subgraphs in $(F_W - F_{W'}) \cup (F_{W'} - F_W)$, that is, those that have become frequent, but which were infrequent, and those that have become infrequent, but which were frequent.

To seek CS, we exploit the optimization provided by the anti-monotonicity property of the generality order, with respect to the support. This allows us to interrupt the exploration as we find out a subgraph Q, whose supporting snapshots $cov(Q, W)$ differ from those of its more general subgraph P, that is, $(cov(P, W) \neq cov(Q, W))$. When this happens P is conserved as non-redundant and all the subgraphs that are more general than P, with the same supporting snapshots (e.g.,Q), are removed. Finally, the search for another CS re-starts from Q.

To seek MS, we explore the subspace of the lattice with new frequent subgraphs and the subspace with new infrequent subgraphs from the last level. In particular, we visit the lattice upwards and, by exploiting the generality order $\geqslant$, take the subgraphs P which have the largest number of edges and which are collocated at the last level. Contemporaneously, we discard all the subgraphs which have less edges and which are more general than P. Thus, the majority of the redundancies, especially the subgraphs collocated at the higher levels, are already removed at the beginning of the operation.

Once the redundancies have been discarded and non-redundant subgraphs have been identified (either by Definition 2 or by Definition 3), we check whether they are also emerging, by evaluating the ratio of the support values (already computed in W and W'), as indicated in Definition 1.

For the sake of brevity, we report the algorithmic procedure only for the CS. In Algorithm 1, by implementing Definition 2, the procedure checks whether the current subgraph P is redundant for the sub-space S.

Algorithm 1: Discovery of emerging subgraphs from closed subgraphs.

```
   inputs : F_W, F_W', minGR
   output: ESs the emerging subgraphs between W
           and W'
1  ESs ← ∅
2  S ← (F_W − F_W') ∪ (F_W' − F_W)
3  foreach P ∈ S do
4  |  if isNonRedundant(P, S) then
5  |  |  if GR(P, W, W') > minGR then
6  |  |  |  ESs ← ESs ∪ {P}
7  |  |  end
8  |  end
9  end
```

3.2. Discovery of closed/maximal subgraphs from emerging subgraphs

On the other hand, the second strategy first identifies the most evident changes, that is, emerging subgraphs, and then removes redundancies from them. To this end, it searches for emerging subgraphs from the sub-space of frequent subgraphs (at the time of W') and from the sub-space of infrequent subgraphs (at the time of W'), which, according to what has been said in Section 2, contain subgraphs which express changes. For this operation, however, we cannot avoid exhaustively scanning the branches of the two sub-spaces, because the property of anti-monotonicity does not hold for the growth-rate [14].

The discovery of ES returns two sub-spaces, which are likely to be smaller than the sub-space containing the infrequent subgraphs $(F_W - F_{W'})$ and sub-space containing frequent subgraphs $(F_{W'} - F_W)$ respectively. However, these sub-spaces can have "holes", that is, parts of the lattice that cannot be reached. This means we might have subgraphs that are not emerging but are more specific or more general than other subgraphs that are emerging. This does not prevent us from using the generality order $\geqslant$ to remove redundancies from those (reduced) sub-spaces. Thus, when seeking CS, we explore the lattice downwards by starting from the first level, skip the subgraphs P, which are not emerging, but are more specific than those emerging, and finally evaluate the subgraphs Q which are emerging and are more specific than P. When seeking MS instead, however, we explore the lattice of only ES from the last level. In particular, we explore the lattice upwards and, by exploiting the generality order $\geqslant$, take the subgraphs P which have the largest number of edges and are collocated at the last level. At the same time, we remove all the emerging subgraphs which have less edges and are more general than P. The advantage of this operation is to remove immediately the majority of the redundancies, that is, the emerging subgraphs collocated at the higher levels.

For the sake of brevity, we report the algorithmic procedure only for the CS. In Algorithm 2 the procedure implements Definition 2 and works on two sub-spaces of emerging subgraphs (contrary to Algorithm 1). Thus, it checks whether the current subgraph P is redundant for sub-space S.

4. Experiments

The empirical evaluation considers the following research questions:

- How much redundant information are the condensed representations able to remove (*space savings*)?
- What is the ability of the condensed representations to capture the changes expressed by the original sets of ES (*accuracy*)?
- Are the proposed approaches scalable?

To answer these questions, we performed a comparative and quantitative evaluation between the different procedures for mining the condensed representations of ES. In

Algorithm 2: Discovery of closed subgraphs from emerging subgraphs.

inputs : $F_W, F_{W'}, minGR$
output: ESs non-redundant subgraphs from emerging subgraphs between W and W'

$ESs \leftarrow \emptyset$
$S \leftarrow \{P \in (F_W - F_{W'}) \mid GR(W, W', P) > minGR\}$
$S' \leftarrow \{P \in (F_{W'} - F_W) \mid GR(W, W', P) > minGR\}$
foreach $P \in S$ **do**
  if $isNonRedundant(P, S)$ **then**
    $ESs \leftarrow ESs \cup \{P\}$
  end
end
foreach $P \in S'$ **do**
  if $isNonRedundant(P, S')$ **then**
    $ESs \leftarrow ESs \cup \{P\}$
  end
end

particular, we compared the performance of the four variants of the general computational approach which have been designed by implementing the two notions of non-redundant subgraphs (CS and MS) in the two strategies reported in Section 3.1 and Section 3.2 respectively. We will refer to the variant for the *i)* discovery of emerging subgraphs ES from closed subgraphs CS named as *CloPre*, *ii)* discovery of emerging subgraphs ES from maximal subgraphs MS as *MaxPre*, *iii)* discovery of closed subgraphs CS from emerging subgraphs ES as *CloPost*, and *iv)* discovery of maximal subgraphs MS from emerging subgraphs ES as *MaxPost*.

The experiments have been organized in order to evaluate the four variants in term of quantitative measures described in the following:

- *average space savings* of the condensed representations against the reference changes, that is, the sets of the original ES.
- *average accuracy* of the condensed representations in representing the same structural information as the original ES.
- scalability of the proposed approaches.

By following the practice used in the literature of data stream mining on dynamic graphs, we built the landmark windows by enqueuing blocks of snapshots rather than individual snapshots. In these experiments, the blocks are built with 100 snapshots (Π=100), which has been chosen by trading-off the mining efficiency (because of the update of the subgraphs) and the completeness of the subgraphs denoting changes. In fact, maintaining the updated sets of frequent patterns in a streaming environment can introduce inefficiency when processing single data points, whereas processing blocks of snapshots can alleviate the computational efforts, as also pointed out in [15]. Moreover, keeping the number of blocks fixed allows us to make a fair comparison. Indeed, this puts the four procedures in the same conditions in which the algorithm [3] works, and allows us to fairly compare the condensed representation discovered by CloPre, CloPost, MaxPre and MaxPost with the original ES.

As we will observe, the empirical results indicate that the condensed representations *i)* compress the change-related information and *ii)* maintain the frequentistic and structural information expressed by the original set of change-related subgraphs.

The experiments were run on an Intel i7 64bit @3.4 GHz desktop running Windows. The datasets we used are time-stamped graphs and are stored on the desktop machine as data files. In the following, we first describe the datasets used and then explain the experimental sessions by discussing the results obtained.

4.1. Dataset Description

The experiments were performed on i) 5 real-world dynamic graphs and ii) 8 synthetic dynamic graphs. Table 1 reports the basic statistics and graph-based indices for each dataset. More specifically, we report the total number of snapshots $|D|$, the minimum/average/maximum number of nodes over the snapshots $|N_i|$, the minimum/average /maximum number of edges over the snapshots $|E_i|$, and the minimum/average/maximum snapshot diameter $diam(G_i)$.

The synthetic datasets were generated with a dynamic graph generator we designed[1], which builds dynamic graphs by appending blocks of graph-snapshots built according to different generative criteria (to ensure the presence of changes between two consecutive blocks). The first snapshot of each block is built according to a random scale-free network generator, the graph obtained is then replicated for the remaining snapshots of the block. We used a scale-free network generator to simulate the preferential attachment rule, which is typical in many real world networks [16]. Each remaining snapshot of a block is then perturbed by randomly adding edges and removing nodes with a probability equal to 2%. We used the generator by tuning the maximum number of nodes, which produced the datasets named *synth nodes 01, synth nodes 02, synth nodes 03* and *synth nodes 04*. We also used the generator by tuning the maximum number of graph snapshots, which produced the datasets named *synth graphs 01, synth graphs 02, synth graphs 03* and *synth graphs 04*. In particular, *synth nodes 01, synth nodes 02, synth nodes 03* and *synth nodes 04* contain dynamic graphs of 1000 snapshots, each of which contains an increasing number of nodes (10, 100, 200 and 300, respectively). On the contrary, *synth graphs 01, synth graphs 02, synth graphs 03* and *synth*

[1] https://bitbucket.org/netminerteam/randomdynamicgraphgen/

Table 1: Characteristics of the dynamic graphs considered. Values are aggregated over time-points.

dataset	$\|D\|$	$\|N_i\|$			$\|E_i\|$			$diam(G_i)$		
		min	avg	max	min	avg	max	min	avg	max
keds	11070	2	10	36	1	12	91	1	3	9
nodobo	41344	2	8	26	1	130	2600	1	1	8
noaa	7670	1418	1637	1738	2601	3297	3917	44	51	76
wikitalk	2185	1	1923	18344	1	3586	28017	1	14	45
mawi	51809	2	17	360	1	15	343	1	1	1
synth nodes 01	1000	5	6	9	4	9	14	2	3	5
synth nodes 02	1000	95	98	100	256	288	329	8	10	14
synth nodes 03	1000	199	199	200	894	988	1067	7	8	9
synth nodes 04	1000	300	300	300	1978	2083	2196	6	6	7
synth graphs 01	200	40	46	50	81	95	111	7	9	14
synth graphs 02	2000	40	46	50	81	95	112	7	9	14
synth graphs 03	20000	40	46	50	81	95	111	7	9	14
synth graphs 04	200000	40	46	50	81	95	111	7	9	14

graphs 04 contain dynamic graphs with an increasing number of snapshots (200, 2000, 20000 and 200000, respectively), each of which contains 50 nodes.

The *keds* dataset[2] [17, 18] depicts the socio-political interactions, occurring between nations and world-wide organizations in the gulf region, as depicted in the news reports collected day by day from April 1979 to July 2009. In this dataset 208 different nodes correspond to nations and world-wide organizations operating in the gulf region, while 20 different edge labels are used to denote the type of political relationships occurring between the nodes. Mining change-related subgraphs in keds means identifying sudden changes in the socio-political international context among nations and world-wide organizations.

The *nodobo* dataset[3] [19] concerns the state of the telecommunication network made of transactions (phone calls, SMSs and bluetooth interactions) between 27 students from a Scottish state high-school, from September 2010 to February 2011. In its original form the dataset collected the transactional records of 13035 phone calls, 83542 SMSs and 5292103 bluetooth interactions. The duration of the phone calls and the length of the SMSs were also reported. When building the dynamic graph counterpart,we built a dynamic graph in which the nodes represent the students, while 11 edge labels denote the modalities of communication. In particular, the edge labels were generated by discretizing, with an equal-width in 5 bins, the duration of the phone calls and the length of the text messages between two students, respectively. On the other hand, we used a single label for referring to the presence of the bluetooth connection. Here the change mining supports the identification of changes in the communication modalities between the students.

The *noaa* dataset[4] [20] was developed in the Reanalysis project by the National Center for Environmental Prediction and the National Center for Atmospheric Research. In its original form the dataset gathered the atmospheric measurements of different meteorological quantities (e.g.: air temperature, wind speed and relative humidity) made by geo-localized sensors equally distributed over the space. When building the dynamic graph, we focused on the relative humidity measurements from January 1st 1990 to December 31st 2010, recorded on a daily basis over an area roughly spanning North-Central America. Clearly, nodes from the dynamic graph denote sensors, while edge labels are nominal values denoting the relative humidity between two linked sensors. As for *nodobo*, the edge labels were generated by applying an equal-width discretization with 10 bins to the values of relative humidity. Clearly, the main goal is to identify the climatic changes occurring in the geographic area considered.

The *wikitalk*[5] dataset depicts the network of interactions among the authors of Wikipedia, the free encyclopedia, observed day by day from December 2001 to January 2008. More specifically, 1140141 nodes correspond to the authors, while 1 edge label is used to denote an edit performed by an author towards the wikipedia talk page of another author. The goal is to identify interesting variations in the interactions among the authors over time.

The *mawi*[6] was developed in the MAWI Project by the Measurement and Analysis on the WIDE Internet Working Group. In its original form the dataset collected the network traffic monitored over the network, in the form of IPv6 packets sent. In this work we have used only a portion of the whole network, that is, the traffic monitored by the sampling point D from January 25th 2005 to January 31th 2005. In particular, we built the dynamic graph by considering the IPv6 addresses as nodes, while 2745 edge labels represent both the communication protocol and the communication port used between two devices. For example, the pair (TCP,80) indicates an HTTP packet. A graph snapshot represents a time-interval of five seconds, during which two IPv6 addresses may communicate through different protocols.

4.2. Space savings of the condensed representations

In this section we report the results obtained on the ability of the four procedures to remove redundancies, while

[2] `http://eventdata.parusanalytics.com/data.dir/levant.html`

[3] `http://nodobo.com/release.html`

[4] `https://coastwatch.pfeg.noaa.gov/erddap/griddap/esrlNcepRe.html`

[5] `https://snap.stanford.edu/data/wiki-Talk.html`

[6] `http://mawi.wide.ad.jp/mawi/samplepoint-D/2005/`

keeping the changes expressed by the original ES. To compute the space savings, we adapt the notion reported in [21] to the changes discovered over two successive landmark windows W and W'.

In particular, we compute the space savings as the relative number of ES saved by the condensed representations of change-related subgraphs CS and MS, respectively:

$$S_{CS} = 1 - \frac{|CS(W,W')|}{|ES(W,W')|} \quad S_{MS} = 1 - \frac{|MS(W,W')|}{|ES(W,W')|}$$

Figure 2 reports the averaged space savings when tuning *minGR*. We observe that MaxPre and MaxPost (Figures 2(b) and 2(d)) have greater space savings (that is, they remove greater portions of redundancies) than the procedures CloPre and CloPost (Figures 2(a) and 2(c)). This is expected, since the MS compress the sets of ES better than CS on average. In particular, by considering the algorithms CloPre and CloPost, we observe that the proportion of ES which are redundant (that is, not satisfying Definition 2) ranges between 5% and 20% on average. On the other hand, by considering the algorithms MaxPre and MaxPost, the proportion of ES which are redundant (not satisfying Definition 3) ranges between 10% and 27%. However, no clear tendency emerges when looking at the space savings when tuning *minGR*.

The space savings do not exhibit relevant differences when observing the impact of the chosen strategy (CloPre vs. CloPost and MaxPre vs. MaxPost). In particular, the space savings of CloPost and MaxPost are slightly higher than those of CloPre and MaxPre, respectively. The NODOBO dataset is an exception, in which MaxPost outperforms MaxPre by 10 points on average. Therefore, we conclude that there is not a clear difference, in terms of space saving, between the first strategy (Section 3.1) and the second strategy (Section 3.2).

4.3. Accuracy of the condensed representations

In this section we evaluate whether the condensed representations cover the portions of the network which have determined the change. To do so, we compare, in the sense of the Jaccard similarity, the edges from the closed/maximal emerging subgraphs (the condensed representation) with the edges forming the emerging subgraphs (the reference changes). In this perspective, a value of similarity equal to 1 indicates that each edge belonging to an emerging subgraph is kept in the condensed representation of the change (at least one closed/maximal emerging subgraph exists which contains it).

More precisely, let C be the set of change-related subgraphs (either $ES(W,W')$, $CS(W,W')$, or $MS(W,W')$), we focus on the following sets of change-related edges:

- $\Delta_C = \bigcup P \in C$, such that $sup(P,W') > sup(P,W)$. Intuitively, this is built on the edges of the newly frequent subgraphs, which are emerging (Δ_{ES}), closed emerging (Δ_{CS}), or maximal emerging (Δ_{MS}), depending on C.
- $\nabla_C = \bigcup P \in C$, such that $sup(P,W') < sup(P,W)$. Intuitively, this is built on edges of the newly infrequent subgraphs, which are emerging (Δ_{ES}), closed emerging (Δ_{CS}), or maximal emerging (Δ_{MS}), depending on C.

In particular, we consider the following Jaccard similarities: i) $J(\Delta_{ES}, \Delta_{CS})$, ii) $J(\Delta_{ES}, \Delta_{MS})$, iii) $J(\nabla_{ES}, \nabla_{CS})$ and iv) $J(\nabla_{ES}, \nabla_{MS})$. Thus, high values of similarity indicate the ability of the CS and MS to include in the condensed representations the same portions of the dynamic graph depicted by the original set of ES.

Table 2 and Table 3 report the values of the Jaccard similarity. In Table 2 we observe that the condensed representation built with the CS perfectly depicts the whole structural information (there is nothing missing). This is not expected for two reasons: i) the notion of CS allows us to preserve the occurrences and topological characterization of the redundant subgraphs and ii) this consideration is also confirmed in the results obtained by CloPre and CloPost which enjoy the independence of the accuracy from the search strategy of CS. Therefore, both CloPre and CloPost can be deemed as loss-less compression algorithms of change-related subgraphs. The same consideration can be drawn for MaxPost. This does not happen for the procedure MaxPre (see Table 3). This is expected because, when inspecting the set of the MS, the subgraphs emerging are often isomorphic to those which are not emerging. This results in capturing only the changes associated to smaller portions of the dynamic graph and missing the changes of larger portions. Consequently, the accuracy drops.

4.4. Scalability of the four procedures

In this section, we argue the scalability of CloPre, CloPost, MaxPre and MaxPost in building the condensed representations on synthetic dynamic graphs. In particular, we show how the four procedures scale when they work on dynamic graphs with *i)* an increasing number of nodes and *ii)* an increasing number of network snapshots.

For this analysis we focus on the running times of the four procedures compared with three competitors. The first competitor is the algorithm KARMA [3], which we can consider as a baseline. It was originally designed to detect changes based on pattern mining, but it is unable to remove redundancies. However, to design the other two competitors, we implemented two variants of KARMA by exploiting two existing alternative techniques, *LCM* and *LCM-Max*[22], as core procedures for mining closed and maximal subgraphs. Thus, we have two competitors developed to detect changes by removing redundancies from mined patterns. However, originally *LCM* and *LCM-Max* were designed to work on itemsets, so, for the variants we used, the itemsets are built as the set of elements of the form $(node, node, edge)$, taken from the datasets used. The way KARMA handles incoming blocks of snapshots and time windows remains the same; what changes is the set of subgraphs on which it works. More precisely, the

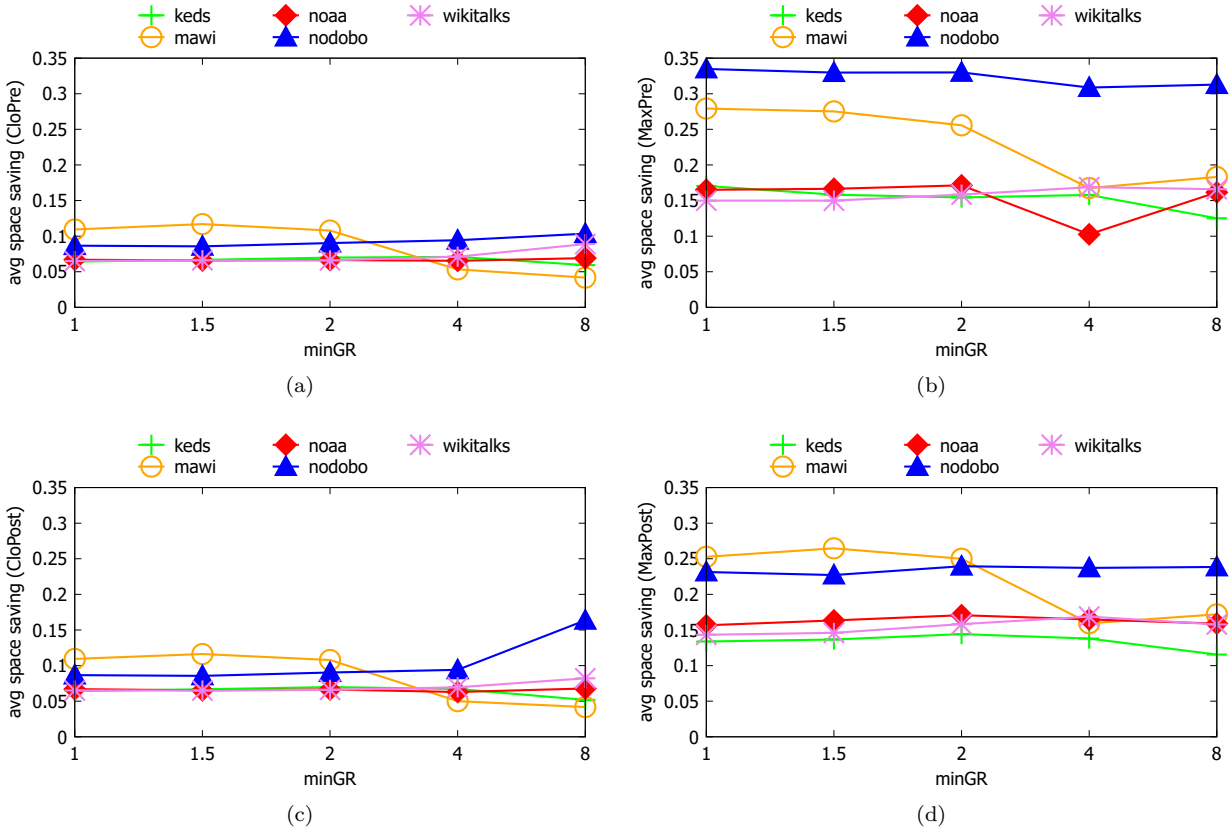


Figure 2: Average space savings of the condensed representations produced by CloPre 2(a), MaxPre 2(b), CloPost 2(c) and MaxPost 2(d) when tuning *minGR*.

algorithm LCM and LCM-Max work on *frequent closed subgraphs* and *frequent maximal subgraphs*, respectively, in place of the frequent subgraphs and infrequent subgraphs.

From a computational perspective, both variants will search for condensed representations of emerging subgraphs, by following the blueprint of KARMA. However, the maximal and closed subgraphs are discovered without the intermediate step of frequent subgraph mining.

In order to stress the algorithms, the synthetic dynamic graphs we used have been configured to process blocks of 10 snapshots (Π=10), Whereas, the condensed representations have been searched by setting *minGR* to 1.2. Figure 3 shows the running times of the four procedures proposed and three competitors, when run on synthetic datasets having different magnitude orders of snapshots (*synth graphs 01, synth graphs 02, synth graphs 03* and *synth graphs 04*). We see that all the algorithms scale linearly with the number of the snapshots (the x-axis is in logarithmic scale and the y-axis in linear scale). This behaviour is not expected and is coherent with the theoretical complexity models discussed in Appendix A. In fact, since KARMA processes dynamic graphs in a block-wise fashion, any variant built on KARMA behaves similarly, that is, it analyzes $\frac{|D|}{|\Pi|}$ blocks of graph snapshots, each consisting of $|\Pi|$ snapshots.

By analyzing separately the times of the procedures CloPre and CloPost (Figure 3(a)) and those of the procedures MaxPre and MaxPost (Figure 3(b)), we grasp the empirical evidence, in terms of time-consumption, of the difference between the two modalities of removing redundancy. The generation of condensed representations, based on the maximal subgraphs, requires a greater time-consumption because the check of definition 3 involves a larger set of subgraphs, that is, those which meet the generality order. On the other hand definition 2 involves the subgraphs which have to also meet the equality of the supporting snapshots sets.

Moreover, Figure 3 reveals that the four procedures proposed are faster than the competitors LCM and LCM-Max. In particular, LCM-Max is slower by one magnitude order. This result lies in different algorithmic decisions, among which is the use of projected databases, which, on the one hand, allows the algorithm to operate only on the subgraphs really appearing in the database, and, on the other hand, it has the disadvantage of repeatedly scanning the databases previously created.

Figure 4 shows the running times of the four procedures proposed and three competitors when run on synthetic datasets having different magnitude orders of nodes (they are *synth nodes 01, synth nodes 02, synth nodes 03* and *synth nodes 04*). These confirm two considerations we

dataset	avg $J(\Delta_{ES}, \Delta_{CS})$ @ minGr									
	CloPre					CloPost				
	1.0	1.5	2.0	4.0	8.0	1.0	1.5	2.0	4.0	8.0
keds	1	1	1	1	1	1	1	1	1	1
mawi	1	1	1	1	1	1	1	1	1	1
noaa	1	1	1	1	1	1	1	1	1	1
nodobo	1	1	1	1	1	1	1	1	1	1
wikitalks	1	1	1	1	1	1	1	1	1	1

dataset	avg $J(\nabla_{ES}, \nabla_{CS})$ @ minGr									
	CloPre					CloPost				
	1.0	1.5	2.0	4.0	8.0	1.0	1.5	2.0	4.0	8.0
keds	1	1	1	1	1	1	1	1	1	1
mawi	1	1	1	1	1	1	1	1	1	1
noaa	1	1	1	1	1	1	1	1	1	1
nodobo	1	1	1	1	1	1	1	1	1	1
wikitalks	1	1	1	1	1	1	1	1	1	1

Table 2: Averaged accuracy scores when approximating i) Δ_{ES} with Δ_{CS} and built from closed subgraphs, discovered with CloPre and CloPost, and ii) ∇_{ES} with ∇_{CS} built from closed subgraphs discovered with CloPre and CloPost.

dataset	avg $J(\Delta_{ES}, \Delta_{MS})$ @ minGr									
	MaxPre					MaxPost				
	1.0	1.5	2.0	4.0	8.0	1.0	1.5	2.0	4.0	8.0
keds	**0.98**	**0.99**	1	1	1	1	1	1	1	1
mawi	**0.97**	**0.98**	**0.98**	1	1	1	1	1	1	1
noaa	**0.99**	**0.99**	**0.99**	1	1	1	1	1	1	1
nodobo	**0.97**	**0.99**	**0.97**	**0.99**	**0.99**	1	1	1	1	1
wikitalks	**0.99**	**0.98**	1	1	1	1	1	1	1	1

dataset	avg $J(\nabla_{ES}, \nabla_{MS})$ @ minGr									
	MaxPre					MaxPost				
	1.0	1.5	2.0	4.0	8.0	1.0	1.5	2.0	4.0	8.0
keds	1	1	1	**0.99**	**0.99**	1	1	1	1	1
mawi	**0.99**	**0.99**	1	**0.99**	1	1	1	1	1	1
noaa	**0.99**	**0.99**	1	**0.99**	**0.99**	1	1	1	1	1
nodobo	**0.99**	**0.99**	1	**0.99**	**0.99**	1	1	1	1	1
wikitalks	**0.99**	**0.99**	1	1	**0.99**	1	1	1	1	1

Table 3: Averaged accuracy scores when approximating i) Δ_{ES} with Δ_{MS} and built from maximal subgraphs, discovered with MaxPre and MaxPost, and ii) ∇_{ES} with ∇_{MS} built from maximal subgraphs discovered with MaxPre and MaxPost.

made for Figure 3, that is, i) the difference between the removal of redundancies with closed subgraphs and with maximal subgraphs and ii) the difference between the variant LCM and LCM-Max and the four procedures.

These two scalability studies also reveal that the second strategy (CloPost and MaxPost) is faster than the first strategy (CloPre and MaxPre), which is a result we attribute to the difference of the input sub-graph sets. The first strategy works on the frequent and infrequent subgraphs, while the second strategy operates on emerging subgraphs, which form a sub-space derived from the frequent and infrequent subgraphs.

To sum up these results, we have that i) the running times of KARMA are a lower bound for every other algorithm considered; ii) the running time of LCM is an upper bound for CloPre and CloPost and iii) the running time of LCM-Max is an upper bound for MaxPre and MaxPost.

4.5. Impact of the minimum growth-rate on the accuracy

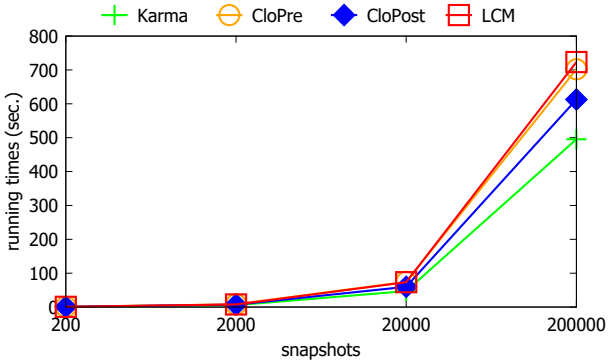
In this section we evaluate the impact of the threshold *minGR* on the accuracy, and quantify the distance (computed with the Jaccard similarity in Section 4.3) of the condensed representations from the set of all the changes, that is, the set of subgraphs which were frequent (infrequent) in W and became infrequent (frequent) in W'. To obtain that set, we performed KARMA at *minGR*=1.0. For these experiments we considered only CloPost and MaxPost because they showed the best accuracy in representing the change conveyed by the ESs discovered by KARMA when tuning *minGR* (Tables 2 and 3).

Figures 5 and 6 plot the averaged accuracy scores relative to the distances i) between Δ_{ES} and Δ_{CS} (Δ_{MS}) and ii) between ∇_{ES} and ∇_{CS} (∇_{MS}).

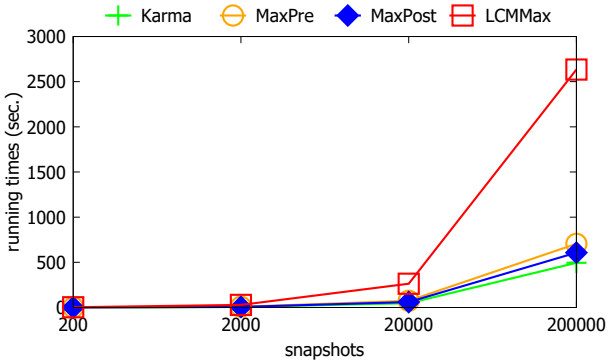
We observe a general drop of the similarity, and, consequently, of the accuracy, when *minGR* increases. This means that the value of the growth-rate is a strong contributor to the removal of the redundancy, regardless of the specific procedure. This is an expected results, in fact higher values of *minGR* let the algorithms discard an increasing number of candidate emerging subgraphs, regardless of the search space explored, as the minGR considerably prunes the search space of frequent, closed and maximal subgraphs.

Another consideration can be made on the different tendency of the plots concerning Δ_{ES} compared to those concerning ∇_{ES}. The similarity computed on Δ_{ES} suddenly decreases with respect to the similarity on ∇_{ES}. This is explained by the different number of ESs: the set of newly frequent subgraphs (Δ_{ES}) is generally smaller than the set of newly infrequent ones (∇_{ES}).

We can conclude that the appropriate choice of the value of *minGR* is pivotal for the accuracy of the condensed representations. In fact, it is evident that for low values of *minGR* the representation is more accurate, al-

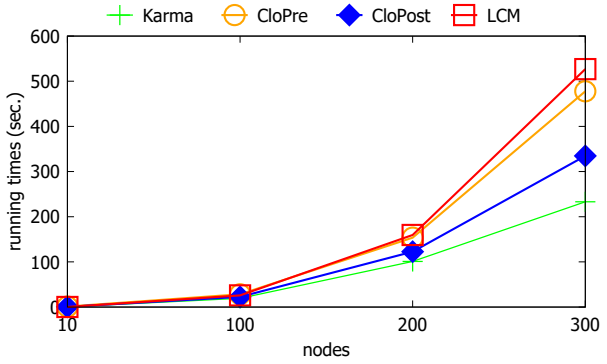


(a)

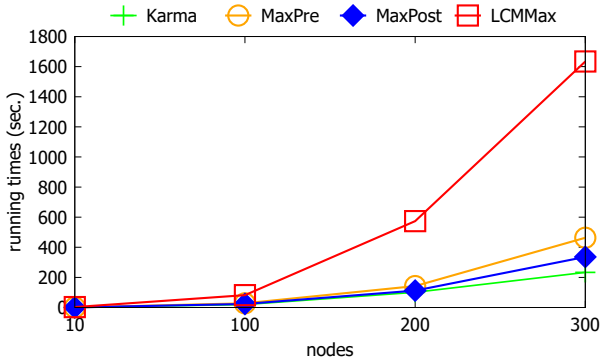


(b)

Figure 3: Running times of i) Karma, CloPre, CloPost and LCM, and ii) Karma, MaxPre, MaxPost and LCM-Max, while tuning the number of graph snapshots from the dynamic graphs ($|\Pi| = 10$, $minGR$=1.20).



(a)



(b)

Figure 4: Running times of i) Karma, CloPre, CloPost and LCM, and ii) Karma, MaxPre, MaxPost and LCM-Max, while tuning the number of nodes of each snapshot from the dynamic graph ($|\Pi| = 10$, $minGR$=1.20).

though it consists of a large number of emerging subgraphs. On the contrary, high values of $minGR$ make the representations become inaccurate, although they consists of a highly reduced number of emerging subgraphs.

5. A practical example of condensed representation

Up to this point, we have discussed the major aspects of mining closed subgraphs and maximal subgraphs from the emerging ones. In this section we introduce a clarifying case-study of how condensed representations, consisting of closed and maximal subgraphs respectively, reduces the number of subgraphs which will be presented to expert users for further inspection.

Both the examples reported have been taken by inspecting the emerging subgraphs discovered by CloPost and MaxPost at the same change point discovered by the KARMA algorithm on the *keds* dataset. In particular, we firstly report the distribution of the emerging subgraphs (ESs) discovered by the three algorithms over time, and then we comment on some differences between some of the sub-networks which have been discovered.

In the socio-political domain of the *keds* dataset every emerging subgraph may refer to an event that really occurred among the nations and the world-wide organizations involved in the subgraph. And it may provides arguments about the temporal collocation of the change. In this scenario, the condensed representations of changes, provided by CloPost and MaxPost, may help the expert users in framing what is happening in complex environments in a more concise manner.

For example, by observing the distribution of emerging subgraphs over time in Figure 7, three peaks of more than than 50 emerging subgraphs clearly emerge. Every single peak refers to a major historical event. In particular, the first one (the beginning of 1991) can be associated to the operations Desert Shield and Desert Storm from the First Gulf War. The second one (September 2000) is associated to the Second Intifada, while the third peak (from late 2001 to early 2002) refers to the invasion of Afghanistan in response of the September 11th attacks. However, many other peaks are present, for example, the one in 2006 which corresponds to the Lebanon War.

Grasping the dynamics involved in such historical events is essential. The domain expert should match each emerging subgraph with time-lines from news broadcasting websites [23], which can be a time-consuming task when the number of emerging subgraphs grows. Closed emerging

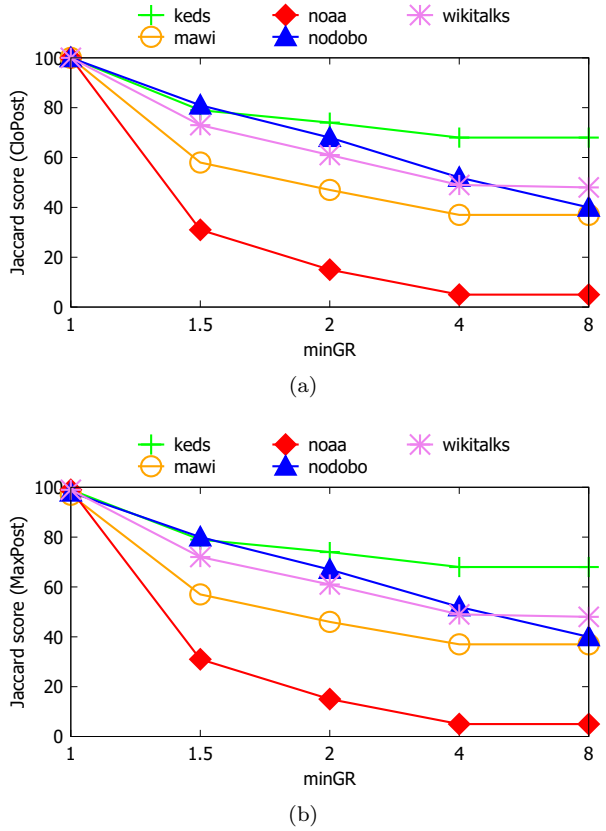


Figure 5: Jaccard scores between Δ_{ES} (from ESs discovered by Karma with $minGR = 1.00$) and i) Δ_{CS}, built with ESs discovered by CloPost, and ii) Δ_{MS}, built with ESs discovered by MaxPost when tuning minGR.

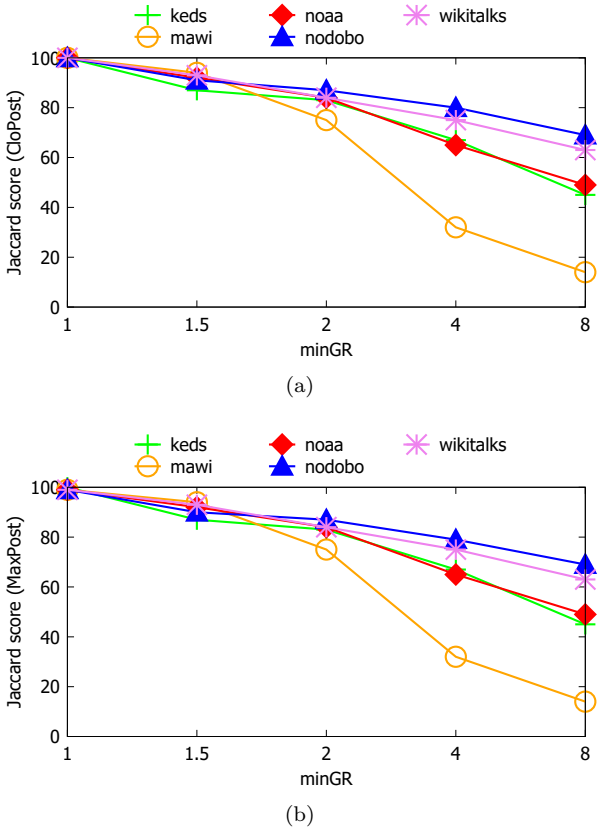


Figure 6: Jaccard scores between ∇_{ES} (from ESs discovered by Karma with $minGR = 1.00$) and i) ∇_{CS}, built with ESs discovered by CloPost, and ii) ∇_{MS}, built with ESs discovered by MaxPost when tuning minGR.

subgraphs (CESs) and maximal emerging subgraphs (MESs) help the user by providing sets of subgraphs which i) are of a moderately reduced size, and ii) eliminate the redundancies contained in the emerging subgraphs (ESs) discovered by KARMA. We report a simple example by inspecting the emerging subnetworks involved in the fluctuation shortly after the "March 9th 1991" peak, that is, at the "September 5th 1991" peak. The algorithms have discovered the emerging subgraphs reported in Table 4.

We clearly observe a reduced number of both the CESs and MESs with respect to the ESs. In particular, we note that only P_2, P_3 and P_4 are retained in the CESs. In fact, P_1 is not closed, and therefore discarded, as a more specific emerging sub-graph exists, namely P_4, covered by the same graph snapshots in W'. In this perspective, the condensed representation of the change has discarded the emerging subgraphs also associated to statistical redundancies, that are general emerging subgraphs with the same support of more specific emerging subgraphs. While looking at the MESs, we note that only P_2 and P_4 are retained, while both P_1 and P_3 are discarded. In fact, the set of CESs is still carrying on the structural renduncancy denoted by P_3, as the edge "(usa, consult, israel)" is still entirely contained in P_4. By also discarding P_3, we ensure that the MESs is the minimal set of emerging subgraphs on which all the edges involved in the change are contained. By eliminating statistical redundancies (in the case of CESs) and structural redundancy (in the case of MESs), the algorithms are able to reduce the number of emerging subgraphs which are necessary to give a comprehensive interpretation of historical events, in the case of the keds dataset. What differs between i) CloPre and CloPost and ii) MaxPre and MaxPost is the closure and maximality evaluation. Both CloPre and MaxPre evaluate the emergency on closed frequent subgraphs and maximal frequent subgraphs, respectively. Therefore, it may happen that i) a closed emerging subgraph (as discovered by CloPost) is not frequent closed, and ii) a maximal emerging subgraph (as discovered by MaxPost) is not frequent maximal. For the afore-mentioned reasons, the subgraphs would be discarded, thus denoting a loss of the structural information associated to its edges.

6. Related works

The study reported in this manuscript falls at the intersection of different fields, such as Pattern discovery, Graph mining and Change detection, but basically it focuses on two main problems, that is, condensed representation of

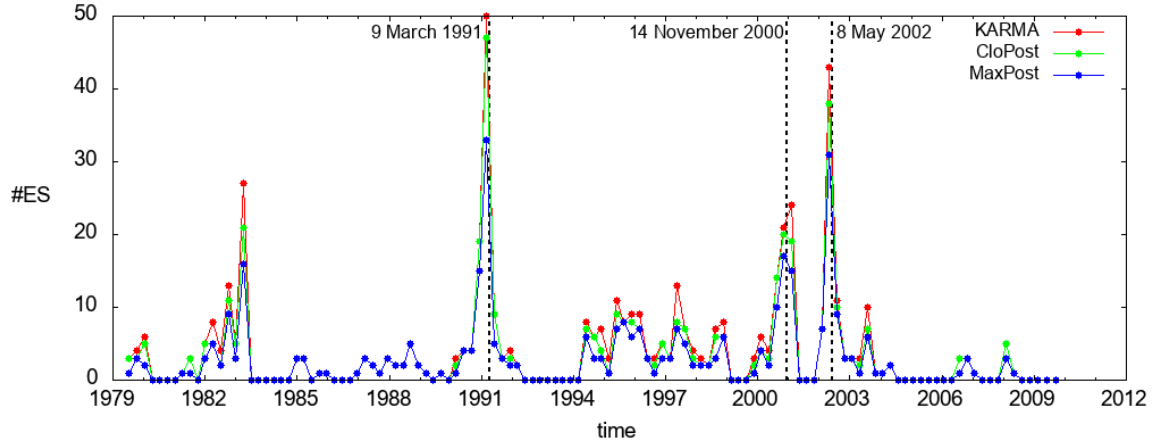


Figure 7: Distribution of the emerging subgraphs *#ES* discovered by i) KARMA, ii) CloPost and iii) MaxPost, from the keds dataset over the time ($minGR = 1.20$, $minSUP$=0.1).

type	P	$sup(P, W)$	$sup(P, W')$	GR
ESs	P_1 : (israel, consult, usa)	**0.12**	0.08	1.47
	P_2 : (israel, fight, palestine)	0.08	**0.12**	1.375
	P_3 : (usa, consult, israel)	**0.15**	0.01	1.56
	P_4 : (israel, consult, usa),(usa, consult, israel)	**0.12**	0.08	1.47
CESs	P_2 : (israel, fight, palestine)	0.08	**0.12**	1.375
	P_3 : (usa, consult, israel)	**0.15**	0.01	1.56
	P_4 : (israel, consult, usa),(usa, consult, israel)	**0.12**	0.08	1.47
MESs	P_2 : (israel, fight, palestine)	0.08	**0.12**	1.375
	P_4 : (israel, consult, usa),(usa, consult, israel)	**0.12**	0.08	1.47

Table 4: List of emerging subgraphs discovered between W=[1991 Sep 5, 1991 Dec 3] and W'=[1991 Sep 5, 1992 Mar 2] by KARMA (ESs), CloPost (CESs), and MaxPost (MESs), respectively. We report in bold the greatest support per each subgraphs.

patterns and change mining in dynamic graphs. The seminal contributions on condensed representations can be attributed to the closed patterns described in [24], which introduced the use of Formal concept analysis and, more precisely, closure operators, to explore the lattice of concepts. Different research threads have been developed since then, for instance, investigating the extension of closed patterns to multi-level itemsets [25, 26].

Another substantial research thread focuses on temporal/evolving data. For the temporal sequences, three main alternatives to compress sequential patterns are extensions of what we have in frequent itemset mining, and they are frequent maximal sequences, frequent closed sequences and frequent generator sequences. Indeed, they work on redundancy removal by considering the containment relationship between sequences and the equality of the support values. A frequent maximal sequence is a frequent sequence that is not strictly contained in another frequent sequence [27]. A frequent sequence is closed if no nother super-sequence exists [28]. A frequent sequence is a generator if no proper sub-sequence exists having the same support [29]. [30] proposes a criterion based on the equivalence of projected databases to terminate early the generation of candidates which are sub-sequences of high utility patterns having the same support only, without requiring particular conditions in the utility values. In [31] the removal of the redundancy of periodic patterns with respect to the support is combined with the removal of the redundancy on the gaps (inter-distance of the repetitions). The method relies on two pruning techniques (addressing the two redundancy notions respectively) to output closed periodic patterns with the least-general gaps.

Another research thread, in which this work can be collocated, is developing around the inherent structure of the data investigated. The concept of maximal co-location patterns has been introduced to create a condensed representation of co-location patterns, which is a form of regularities on spatial data with two-fold features. The compressed set is able to infer the original patterns, but not their participation index values, which is the typical parameter of the spatial co-locations [32]. This has been addressed in [33] through the notion of closed co-location patterns, which enables the generation of the so-called prevalent co-locations. A co-location pattern is closed if no proper co-location exists with the identical value of the participation index.

Data with a more articulated structure are represented by the graphs. The search for condensed representations often has a common denominator, which is the selection of patterns on the basis of statistical or entropy-based parameters. The methodology described in [34] selects only significant patterns in a graph database. The algorithm is able to search for the most dissimilar graph patterns by employing a branch-and-bound mining strategy. In contrast to closure-based methods, the work proposed in [35] selects the maximal patterns in a graph database. The

natural evolution of the graph data is represented by those which change as time goes by. One of the first studies on the concise representations of patterns is reported in [36]. The authors introduce the notion of minimal contrast patterns, in order to capture the smallest edge sets appearing in one class of graphs, but never in another class of graphs. This method differs from ours because it focuses on the smallest subgraphs (against the largest), which could also be disconnected (against the connected ones). A similar problem has been explored in [37], which searches for the so-called emerging pruned graph patterns by means of a constraint-based approach. Pruned patterns are minimal representatives of the emerging subgraphs, which are in their turn extracted during the process of generation of frequent subgraphs, under the satisfaction of input structural constraints. However, the use of user-defined constraints would not guarantee coverage of the complete set of the emerging sub-structures. Moreover, compressed pattern-based representations of changes have been achieved by selecting the patterns that best compress the data. For example, [38] investigates the sequential change detection using the minimum description length (MDL) principle. Firstly, the authors compute MDL-change statistics, as the difference in terms of minimum encoding length required for the case where the change does not occur and the case where it occurs. Then the statistics are aggregated over sequences of time-windows. The same authors extend that research by focusing on 'metachanges', that is, variations on when and how changes occur [39]. On the other hand, the StreamKRIMP [40] change detection algorithm is built on top of patterns discovered by the KRIMP algorithm proposed in [41], which is designed around the MDL principle. StreamKRIMP incrementally keeps track of an MDL code table encoding the data-generating distribution. An MDL-optimal codeword is assigned to each frequently occurring itemset in the stream. New blocks of items are tested as to whether they fit well with the distribution encoded in the current code table. When this does not happen a data distribution change is alerted.

In [42] the authors work on dynamic graphs, and more precisely on streaming data. However, the problem investigated does not concern concise descriptions of the changes, but the discovery of closed patterns, by accounting also for the typical issues of the data stream concept drift. Closed subgraphs are mined by verifying the conditions of closure (support and containment) on weighted input transactions, which probably mirrors the variability of the streaming environments. They propose several techniques which differ in assumptions about the data stream distribution. To our knowledge, there have been very few attempts to perform change detection. [43] adopts closed contrast patterns to summarize changes in network traffic and distinguish between attacks and normal connections. Closed patterns are discovered separately on attack connections and normal connections and the contrast ones are identified as the closed ones that statistically discriminate one class of connection from another.

7. Conclusions

In this manuscript we have faced the problem of mining condensed representations of change-related subgraphs. More specifically, we have investigated the extraction of closed and maximal subgraphs from emerging subgraphs and, viceversa, the extraction of emerging subgraphs starting from closed or maximal subgraphs, by quantitatively evaluating four distinct procedures (namely CloPre, CloPost, MaxPre and MaxPost). The joint notions of closed subgraphs, maximal subgraphs and emerging subgraphs allowed us to take advantage of previous techniques in graph mining and pattern mining when characterizing observed changes in dynamic graphs.

Empirical results have shown that the condensed representations i) store the complete structural knowledge of how the dynamic graph has changed over time, and ii) may compress the number of subgraphs involved in the characterization of the changes. As future work, we plan to explore two different directions, namely the adoption of alternative pattern formalisms, to represent change occurrences, and the use of alternative condensed representations, to compress the set of change-related patterns. As to the first point, we intend to evaluate the use of the patterns in the form of frequent subtrees. However, these, on one hand, may lead to significantly restrict the search space of emerging patterns [44] and improve the scalability, on the other hand, they could give a marginal contribution to the compression of changes, thus resulting in a trade-off problem against scalability and compression performance. As to the second point, we deem be worthwhile exploring the influence of alternative condensed representation of patterns, such as, strongly closed itemsets [45] both on the compression rates and on the performance of the solutions.

Acknowledgments

We acknowledge the support of the MIUR - Ministero dell'Istruzione dell'Università e della Ricerca through the project "TALIsMan - Tecnologie di Assistenza personALizzata per il Miglioramento della quAlità della vitA" (Grant ID: ARS01_01116), funding scheme PON RI 2014-2020. We would also like to thank Lynn Rudd for her help in reading the manuscript.

References

[1] P. Fournier-Viger, G. He, C. Cheng, J. Li, M. Zhou, J. C.-W. Lin, U. Yun, A survey of pattern mining in dynamic graphs, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (2020) e1372.

[2] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, N. F. Samatova, Anomaly detection in dynamic networks: a survey, Wiley Interdisciplinary Reviews: Computational Statistics 7 (2015) 223–247.

[3] C. Loglisci, M. Ceci, A. Impedovo, D. Malerba, Mining microscopic and macroscopic changes in network data streams, Knowl.-Based Syst. 161 (2018) 294–312.

[4] J. Leskovec, L. Backstrom, R. Kumar, A. Tomkins, Microscopic evolution of social networks, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008, pp. 462–470.

[5] A. Chaturvedi, A. Tiwari, N. Spyratos, minstab: Stable network evolution rule mining for system changeability analysis, IEEE Transactions on Emerging Topics in Computational Intelligence (2019) 1–10.

[6] E. Scharwächter, E. Müller, J. F. Donges, M. Hassani, T. Seidl, Detecting change processes in dynamic networks by frequent graph evolution rule mining, in: IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain, pp. 1191–1196.

[7] R. Ahmed, G. Karypis, Algorithms for mining the evolution of conserved relational states in dynamic networks, Knowl. Inf. Syst. 33 (2012) 603–630.

[8] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, N. F. Samatova, Anomaly detection in dynamic networks: a survey, WIREs Computational Statistics 7 (2015) 223–247.

[9] J. Gama, M. M. Gaber, Learning from data streams: processing techniques in sensor networks, Springer, 2007.

[10] P. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, 2005.

[11] G. Dong, J. Li, Efficient mining of emerging patterns: Discovering trends and differences, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999, pp. 43–52.

[12] X. Yan, J. Han, Closegraph: mining closed frequent graph patterns, in: L. Getoor, T. E. Senator, P. M. Domingos, C. Faloutsos (Eds.), Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003, ACM, 2003, pp. 286–295.

[13] C. Borgelt, T. Meinl, M. R. Berthold, Advanced pruning strategies to speed up mining closed molecular fragments, in: Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: The Hague, Netherlands, 10-13 October 2004, IEEE, 2004, pp. 4565–4570.

[14] A. Soulet, B. Crémilleux, F. Rioult, Condensed representation of emerging patterns, in: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings, volume 3056 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 127–132.

[15] M. Deypir, M. H. Sadreddini, S. Hashemi, Towards a variable size sliding window model for frequent itemset mining over data streams, Computers & Industrial Engineering 63 (2012) 161–172.

[16] D. Chakrabarti, C. Faloutsos, Graph Mining: Laws, Tools, and Case Studies, Synthesis Lectures on Data Mining and Knowledge Discovery, Morgan & Claypool Publishers, 2012.

[17] P. A. Schrodt, S. G. Davis, J. L. Weddle, Political Science: KEDS–A Program for the Machine Coding of Event Data, Social Science Computer Review 12 (1994) 561.

[18] U. Brandes, J. Lerner, Visualization of conflict networks, Nato Security Through Science Series - E: Human and Societal Dynamics 36 (2008) 169.

[19] S. Bell, A. McDiarmid, J. Irvine, Nodobo: Mobile phone as a software sensor for social network research, in: Proceedings of the 73rd IEEE Vehicular Technology Conference, VTC Spring 2011, 15-18 May 2011, Budapest, Hungary, pp. 1–5.

[20] E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, A. Leetmaa, B. Reynolds, M. Chelliah, W. Ebisuzaki, W. Higgins, J. Janowiak, K. C. Mo, C. Ropelewski, J. Wang, R. Jenne, D. Joseph, The NCEP/NCAR 40-Year Reanalysis Project., Bulletin of the American Meteorological Society 77 (1996) 437–472.

[21] J. Uthayakumar, T. Vengattaraman, P. Dhavachelvan, A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications, Journal of King Saud University - Computer and Information Sciences (2018).

[22] T. Uno, M. Kiyomi, H. Arimura, LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets, in: FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004.

[23] G. B. Tran, M. Alrifai, D. Q. Nguyen, Predicting relevant news events for timeline summaries, in: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume, pp. 91–92.

[24] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Efficient mining of association rules using closed itemset lattices, Inf. Syst. 24 (1999) 25–46.

[25] C. Loglisci, D. Malerba, Mining multiple level non-redundant association rules through two-fold pruning of redundancies, in: P. Perner (Ed.), Machine Learning and Data Mining in Pattern Recognition, 6th International Conference, MLDM 2009, Leipzig, Germany, July 23-25, 2009. Proceedings, volume 5632 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 251–265.

[26] T. Hashem, C. F. Ahmed, M. Samiullah, S. Akther, B.-S. Jeong, S. Jeon, An efficient approach for mining cross-level closed itemsets and minimal association rules using closed itemset lattices, Expert Systems with Applications 41 (2014) 2914–2938.

[27] P. Fournier-Viger, C. Wu, A. Gomariz, V. S. Tseng, VMSP: efficient vertical mining of maximal sequential patterns, in: M. Sokolova, P. van Beek (Eds.), Advances in Artificial Intelligence - 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings, volume 8436 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 83–94.

[28] B. Le, H. Duong, T. Truong, P. Fournier-Viger, Fclosm, fgensm: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy, Knowledge and Information Systems 53 (2017) 71–107.

[29] P. Fournier-Viger, A. Gomariz, M. Sebek, M. Hlosta, VGEN: fast vertical mining of sequential generator patterns, in: L. Bellatreche, M. K. Mohania (Eds.), Data Warehousing and Knowledge Discovery - 16th International Conference, DaWaK 2014, Munich, Germany, September 2-4, 2014. Proceedings, volume 8646 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 476–488.

[30] T. Truong, H. Duong, B. Le, P. Fournier-Viger, Fmaxclohusm: An efficient algorithm for mining frequent closed and maximal high utility sequences, Engineering Applications of Artificial Intelligence 85 (2019) 1 – 20.

[31] S. Akther, M. R. Karim, M. Samiullah, C. F. Ahmed, Mining non-redundant closed flexible periodic patterns, Engineering Applications of Artificial Intelligence 69 (2018) 1 – 23.

[32] J. S. Yoo, M. Bow, Mining maximal co-located event sets, in: J. Z. Huang, L. Cao, J. Srivastava (Eds.), Advances in Knowledge Discovery and Data Mining - 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24-27, 2011, Proceedings, Part I, volume 6634 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 351–362.

[33] L. Wang, X. Bao, H. Chen, L. Cao, Effective lossless condensed representation and discovery of spatial co-location patterns, Inf. Sci. 436-437 (2018) 197–213.

[34] X. Yan, H. Cheng, J. Han, P. S. Yu, Mining significant graph patterns by leap search, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008, pp. 433–444.

[35] J. Huan, W. Wang, J. F. Prins, J. Yang, SPIN: mining maximal frequent subgraphs from graph databases, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004, pp. 581–586.

[36] R. M. H. Ting, J. Bailey, Mining minimal contrast subgraph patterns, in: J. Ghosh, D. Lambert, D. B. Skillicorn, J. Sri-

vastava (Eds.), Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA, SIAM, 2006, pp. 639–643.
[37] G. Poezevara, B. Cuissart, B. Crémilleux, Extracting and summarizing the frequent emerging graph patterns from a dataset of graphs, J. Intell. Inf. Syst. 37 (2011) 333–353.
[38] K. Yamanishi, K. Miyaguchi, Detecting gradual changes from data stream using mdl-change statistics, in: J. Joshi, G. Karypis, L. Liu, X. Hu, R. Ak, Y. Xia, W. Xu, A. Sato, S. Rachuri, L. H. Ungar, P. S. Yu, R. Govindaraju, T. Suzumura (Eds.), 2016 IEEE International Conference on Big Data, Big-Data 2016, Washington DC, USA, December 5-8, 2016, IEEE Computer Society, 2016, pp. 156–163.
[39] S. Fukushima, K. Yamanishi, Detecting metachanges in data streams from the viewpoint of the MDL principle, Entropy 21 (2019) 1134.
[40] M. van Leeuwen, A. Siebes, Streamkrimp: Detecting change in data streams, in: Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I, pp. 672–687.
[41] J. Vreeken, M. van Leeuwen, A. Siebes, Krimp: mining itemsets that compress, Data Min. Knowl. Discov. 23 (2011) 169–214.
[42] A. Bifet, G. Holmes, B. Pfahringer, R. Gavaldà, Mining frequent closed graphs on evolving data streams, in: C. Apté, J. Ghosh, P. Smyth (Eds.), Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011, ACM, 2011, pp. 591–599.
[43] E. A. Chavary, S. M. Erfani, C. Leckie, Summarizing significant changes in network traffic using contrast pattern mining, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, ACM, New York, NY, USA, 2017, pp. 2015–2018.
[44] A. Impedovo, M. Ceci, T. Calders, Efficient and accurate non-exhaustive pattern-based change detection in dynamic networks, in: Discovery Science - 22nd International Conference, DS 2019, Split, Croatia, October 28-30, 2019, Proceedings, pp. 396–411.
[45] D. Trabold, T. Horváth, Mining strongly closed itemsets from data streams, in: Discovery Science - 20th International Conference, DS 2017, Kyoto, Japan, October 15-17, 2017, Proceedings, pp. 251–266.

Appendix A. Computational complexity

We studied the time complexity of the proposed algorithms by analyzing the worst-case. Let W be the landmark window, Π be the data block of graph snapshots, such that $W' = W \cup \Pi$, $n = |N|$ is the number of nodes, and $m = |E|$ is the number of edge labels.

The computational complexity is related to the following factors: i) the cost of building and revising the sets of at most $O(2^{nkm})$ frequent subgraphs in the lattice, and ii) the cost $O(C)$ for the selection of emerging subgraphs, whose upper bound is $C = |F_W - F_{W'}| + |F_{W'} - F_W|$. Then, under the assumption that each of the n nodes is connected to at most other k nodes [3], we showed that the time complexity in the worst-case is $O(2^{nkm} \cdot |\Pi| + C)$.

The procedures described so far are: i) considering two sets F_W and $F_{W'}$ of frequent subgraphs in time proportional to $O(2^{nkm})$ and ii) extracting a set of emerging subgraphs in $O(C)$. All four algorithms search the condensed representation of changes by checking, although in a different order, the growth-rate, the closure and the maximality on at most $O(C)$ subgraphs. Assuming that each subgraph possesses on average b super-graphs in the lattice, then the evaluation of the maximality condition can be performed in $O(1)$ by keeping counters for each subgraph. On the contrary, the evaluation of the closure requires exactly b coverage comparisons having at most $|W'|$ elements. For this reason, the cost of evaluating the closure is proportional to $O(b \cdot |W'|)$.

This means that the final complexity in the worst-case is i) $O(2^{nkm} \cdot |\Pi| + C)$, when mining the maximal subgraphs, and ii) $O(2^{nkm} \cdot |\Pi| + C \cdot b \cdot |W'|)$, when mining the closed subgraphs. Therefore, the computational complexities remain exponential in the number of triples in the worst-case scenario, with the lower bound discussed in [3]. This study takes into account the nature of the data considered, while at the same time it does not rely on a specific frequent pattern mining algorithm.

In conclusion, we ascertain that the four procedures do not introduce time complexity degrees to the frequent pattern mining problem (under the $O(\cdot)$ operator), while addressing a *wider* problem, that is, the detection of changes on dynamic graphs. As explained, the computational solution does not only rely on the frequent pattern mining task, but additionally discovers emerging subgraphs and identifies closed/maximal subgraphs.