# Mining Frequent Sequential Subgraph Evolutions in Dynamic Attributed Graphs

Zhi Cheng[1], Landy Andriamampianina[1,2], Franck Ravat[2], Jiefu Song[2], Nathalie Vallès-Parlangeau[3], Philippe Fournier-Viger[4], and Nazha Selmaoui-Folcher[5]

[1] Activus Group, 1 Chemin du Pigeonnier de la Cépière, 31100 Toulouse, France
`{zhi.cheng, landy.andriamampianina}@activus-group.fr`
[2] IRIT-CNRS (UMR 5505) - Université Toulouse 1 Capitole (UT1), 2 Rue du Doyen Gabriel Marty F-31042 Toulouse Cedex 09, France
`{franck.ravat, jiefu.song}@irit.fr`
[3] Université de Pau et des Pays de l'Adour, France
`nathalie.valles-parlangeau@univ-pau.fr`
[4] Shenzhen University, Shenzhen, China
`philfv@szu.edu.cn`
[5] University of New Caledonia, PPME, BP R4, F-98851 Nouméa, New Caledonia
`nazha.selmaoui@unc.nc`

**Abstract.** Mining patterns in a dynamic attributed graph has received more and more attention recently. However, it is a complex task because both graph topology and attributes values of each vertex can change over time. In this work, we focus on the discovery of frequent sequential subgraph evolutions (FSSE) in such a graph. These FSSE patterns represent frequent evolutions of general sets of connected vertices' attribute values. A novel algorithm, named FSSEMiner, is proposed to mine FSSE patterns. This algorithm is based on a new strategy (graph addition) to guarantee mining efficiency. Experiments performed on benchmark and real-world datasets show the interest of our approach and its scalability.

**Keywords:** dynamic attributed graph · frequent sequential pattern · graph mining

## 1 Introduction

Dynamic attributed graphs have recently received a lot of attention [6]. The reason is that this graph type provides a rich representation of real-world phenomena. It has been widely used to describe many complex datasets (e.g. spatio-temporal data, health data, biological data or social data) [2,3,8]. A dynamic attributed graph depicts a time-ordered sequence of graphs to capture the evolution of a real-world phenomena. Specifically, vertices and edges between vertices of each graph of the sequence represent respectively objects and spatial relations or other types of interactions between objects that are valid at the graph timestamp. Attributes are used to complete the semantics of vertices. Objects and their relations may evolve over time. Indeed, changes may happen in two levels: at the topological level, there may be addition and removal of objects and relations; at the object level, changes may also happen in attribute values. Mining patterns in a dynamic attributed graph allows analysing how objects, relations and attribute values of objects evolve over time.

Existing pattern mining approaches in dynamic attributed graphs allow following evolution within an individual vertex [8] or a set of vertices [2,3,5,7,1]. However, none of these algorithms allows finding frequent sequential evolutions for general sets of connected vertices (i.e., frequent subgraphs) in a dynamic attributed graph. For instance, in the case of monitoring the spread of a virus, existing patterns may reveal sequential changes of individuals' health status within a specific group. However, if this specific group is special (for example, they have innate resistance to this virus, or they were vaccinated), existing patterns lose general representativeness and cannot provide meaningful analysis for virus transmission. Indeed, the objective of doctors is to understand how virus spread among general groups instead of specific ones. Therefore, our objective is to find frequent sequential evolutions of general sets of connected vertices. To do so, we propose a novel pattern type denoted as *frequent sequential subgraph evolutions (FSSE)*. After describing a formal representation of FSSE (Section 3), we present the *FSSEMiner* algorithm to find FSSE patterns in a dynamic attributed graph (Section 4). The scalability of the algorithm is studied through some experimental assessments based on both real-world and benchmark datasets (Section 5).

## 2 Related work

Several research works are proposed to analyse dynamic attributed graphs. Desmier et al. [2] defined *cohesive co-evolution patterns*. Such patterns represent a set of vertices with the same attribute variations and a similar neighbourhood during a time interval. The authors extended their work in [3] by integrating constraints on topology and on attribute values to extract maximal dynamic attributed subgraphs. Yet, these patterns do not represent sequential changes.

Kaytoue et al. [8] defined the *triggering pattern* problem, which allows finding temporal relationships between attribute values and topological properties of vertices. The TRIGAT algorithm allows mining triggering patterns by using a projection strategy. However, triggering patterns do consider neighbouring vertices, neither their evolutions since they focus on a single vertex.

Fournier-Viger et al. [5] proposed *significant trend sequences*. Such patterns allow discovering the influence of attribute variations of a single vertex on its neighbours. The authors extended these patterns in [7] by defining *attribute evolution rules* (AER) to discover the influence of multiple vertices on other vertices. The AER-Miner algorithm allows mining AER patterns by using breadth first search (BFS) strategy. Yet, AER patterns represent sequential evolutions of attributes instead of connected vertices.

Cheng et al. [1] proposed *recurrent patterns*, which are frequent sequences of sets of connected vertices. Each sets of connected vertices of a recurrent pattern represent attribute variations of vertices. The RPMiner algorithm, based on successive graph intersections strategy, allows mining recurrent patterns. However, recurrent patterns focus only on evolutions of specific set of vertices, as they depend on their vertices' temporal occurrences instead of considering the spatio-temporal occurrences.

In response to the previous limitations, we propose a novel pattern denoted as *frequent sequential subgraph evolutions (FSSE)*. Compared to existing work, the main advantage of this pattern is to consider evolutions independently of subgraphs in which they occur. In a spatio-temporal context, it means that such patterns would highlight phenomena independently of their locations. However, none of the previous algorithms mines FSSE. Indeed, mining such patterns in a dynamic attributed graph is so complex that all existing strategies cannot guarantee both the mining efficiency and the completeness of patterns. For this purpose, we propose a novel algorithm called *FSSEMiner* to mine FSSE. The algorithm is based on a new strategy called *graph addition* to guarantee mining efficiency. It requires traversing only once each graph, instead of an exponential graph traversing operation as in most existing mining strategies.

## 3 Notations

### 3.1 Dynamic Attributed Graph

A dynamic attributed graph, denoted as $\mathcal{G} = \langle G_{t_1}, G_{t_2}, ..., G_{t_{max}} \rangle$, represents the evolution of a graph over a set of ordered and consecutive timestamps $T = \{t_1, t_2, ..., t_{max}\}$. It is composed of the set of vertices denoted as $\mathcal{V}$. The set of attributes $\mathcal{A}$ is used to describe all the vertices. Each attribute $a \in \mathcal{A}$ is associated with a domain value. A domain value $\mathbb{D}_a$ (numerical or categorical) is associated to each vertex and attribute $a \in \mathcal{A}$. For each time $t \in T$, $G_t = (V_t, E_t, \lambda_t)$ is an attributed undirected graph where: $V_t \subseteq \mathcal{V}$ is the set of vertices, $E_t \subseteq V_t \times V_t$ is the set of edges, and $\lambda_t : V_t \rightarrow 2^{\mathcal{A}\mathbb{D}}$ is a function that associates each vertex of $V_t$ with a set of attribute values $\mathcal{A}\mathbb{D} = \cup_{a \in \mathcal{A}}(a \times \mathbb{D}_a)$.

When attribute values are numerical, a dynamic attributed graph is usually preprocessed to derive trend values from these attributes, since we are not interested in their absolute variations [6]. A trend is an increase $(+)$, decrease $(-)$ or stability $(=)$ which means the value of a vertex's attribute increase, decrease or does not change between two consecutive timestamps.

### 3.2 A New Pattern Domain

***Definition 1 (frequent subgraph)*** Let $(\lambda, Occurrence(\lambda)\, in\, T')$ be an attributed subgraph of $\mathcal{G}$, where $\lambda$ is a set of attribute values (trends or categorical values) representing a pattern and $Occurrence(\lambda)\, in\, T'$ represents the occurrences of $\lambda$ in the set of times $T' \subseteq T$. More precisely, $Occurrence(\lambda)\, in\, T'$ is a set of subgraphs such that $Occurrence(\lambda) \subseteq V_t$ where $t \in T', V_t \subseteq \mathcal{V}$. As shown in Fig. 1, $\langle \{a_1 - a_2+, a_1 - a_2+, a_1 + a_2 =)\}, \{t_1 : (1, 2, 3)|t_1 : (7, 8, 10)|t_1 :$

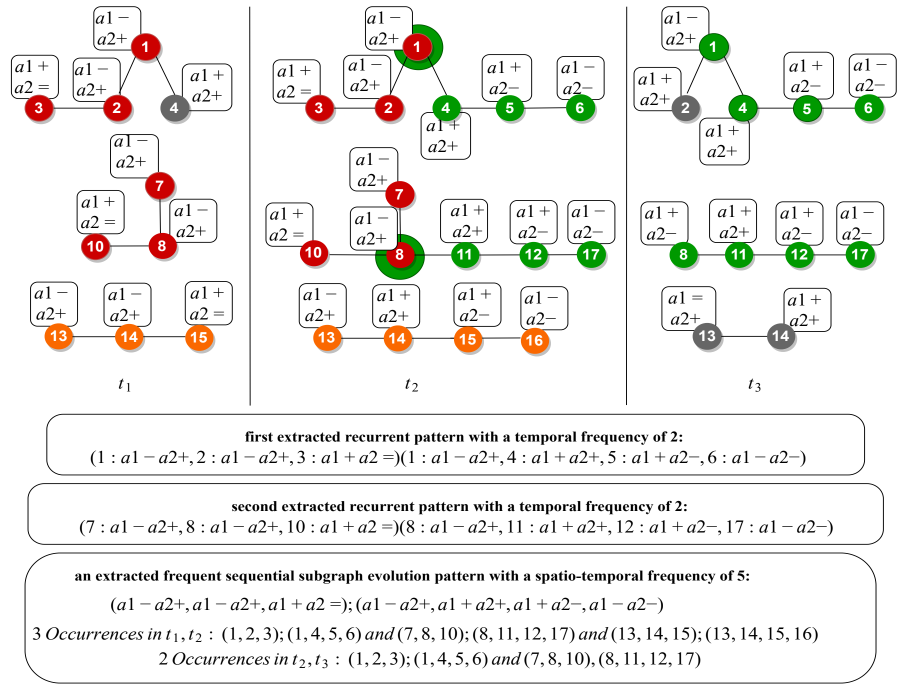

$t_1$
$t_2$
$t_3$
first extracted recurrent pattern with a temporal frequency of 2:
$(1 : a1-a2+, 2 : a1-a2+, 3 : a1+a2 =)(1 : a1-a2+, 4 : a1+a2+, 5 : a1+a2-, 6 : a1-a2-)$
second extracted recurrent pattern with a temporal frequency of 2:
$(7 : a1-a2+, 8 : a1-a2+, 10 : a1+a2 =)(8 : a1-a2+, 11 : a1+a2+, 12 : a1+a2-, 17 : a1-a2-)$
an extracted frequent sequential subgraph evolution pattern with a spatio-temporal frequency of 5:
$(a1-a2+, a1-a2+, a1+a2 =); (a1-a2+, a1+a2+, a1+a2-, a1-a2-)$
3 *Occurrences in* $t_1, t_2$ : $(1,2,3); (1,4,5,6)$ *and* $(7,8,10); (8,11,12,17)$ *and* $(13,14,15); (13,14,15,16)$
2 *Occurrences in* $t_2, t_3$ : $(1,2,3); (1,4,5,6)$ *and* $(7,8,10), (8,11,12,17)$

Fig. 1: Dynamic attributed graph.

$(13,14,15)|t_2 : (1,2,3)|t_2 : (7,8,10)\}\rangle$ is an attributed subgraph in $t_1$ and $t_2$. The set of attribute values $\lambda = (a_1 - a_2+, a_1 - a_2+, a_1 + a_2 =)$ represents a size-1 pattern, i.e., a pattern composed of only one set of attribute values. $Occurrence(\lambda)\, in\, t_1 = \{t_1 : (1,2,3)|t_1 : (7,8,10)|t_1 : (13,14,15)\}$ represents the set of occurrences of $\lambda$ in $t_1$ (in red and orange). $Occurrence(\lambda)\, in\, t_2 = \{t_2 : (1,2,3)|t_2 : (7,8,10)\}$ represents the set of occurrences of $\lambda$ in $t_2$ (in red). So the frequency of the subgraph is 5 because it occurs 5 times.

***Definition 2 (frequent sequential subgraph evolution)*** A frequent sequential subgraph evolution (FSSE) of $\mathcal{G}$ appearing in the time intervals $\{T_1, \ldots, T_k\}$, $T_i \subseteq T$, is a sequence $P = \langle\{\lambda_1; ...; \lambda_n\}, \{T_1 : Occurrence_1(\lambda_1); \ldots; Occurrence_1(\lambda_n), | \ldots |T_k : Occurrence_k(\lambda_1); \ldots; Occurrence_k(\lambda_n)\}\rangle$. The first set represents the frequent sequential subgraphs where each subgraph is separated by semicolons. The second set is composed of all patterns' occurrences, where each occurrence is separated by vertical bars. For each $T_i = \{t_i, ..., t_j\}, 1 \le i \le j \le |T|$, $t_i$ represents the starting time of this occurrence and $t_j$ represents the end time of this occurrence. As shown in Fig. 1, $\langle\{(a1 - a2+, a1 - a2+, a1 + a2 =); (a1 - a2+, a1 + a2+, a1 + a2-, a1 - a2-)\}, \{t_1, t_2 : (1,2,3); (1,4,5,6)|t_1, t_2 : (7,8,10); (8,11,12,17)|t_1, t_2 : (13,14,15); (13,14,15,16)|t_2, t_3 : (1,2,3); (1,4,5,6)|\ t_2, t_3 : (7,8,10); (8,11,12,17)\}\rangle$ is a FSSE starting at $t_1$ and $t_2$. It is a sequence of size 2 composed of a frequent subgraph with the pattern $(a1-a2+, a1-a2+, a1+a2=)$ in $\{t_1, t_2\}$ (in red and orange) and a frequent subgraph with the pattern $(a1-a2+, a1+a2+, a1+a2-, a1-a2-)$ in $\{t_2, t_3\}$ (in green and orange).

The example shown in Fig. 1 can also illustrate the difference between the closest pattern type (i.e, recurrent pattern [1]) and the FSSE. With a frequency of 2, two recurrent patterns are extracted. The first pattern represents a subgraph in red $(1: a1-a2+, 2: a1-a2+, 3: a1+a2=)$ which is followed by another subgraph in green $(1: a1-a2+, 4: a1+a2+, 5: a1+a2-, 6: a1-a2-)$. The frequency of this pattern is 2, as it appears twice over time: the first time from $t_1$ to $t_2$ and the second from $t_2$ to $t_3$. Similarly, another recurrent pattern is a subgraph in red $(7: a1-a2+, 8: a1-a2+, 10: a1+a2=)$ which is followed by another subgraph in green $(8: a1-a2+, 11: a1+a2+, 12: a1+a2-, 17: a1-a2-)$ with a frequency of 2. Moreover, it can be observed that another sequence of subgraphs in orange, $(13: a1-a2+, 14: a1-a2+, 15: a1+a2=)(13: a1-a2+, 14: a1+a2+, 15: a1+a2-, 16: a1-, a2-)$, represents exactly the same evolution as the two patterns extracted above. However, it is not considered as a recurrent pattern as its temporal frequency is 1. In comparison, one extracted frequent sequential subgraph evolution is a subgraph $(a1-a2+, a1-a2+, a1+a2=)$ which is followed by another subgraph $(a1-a2+, a1+a2+, a1+a2-)$ where each subgraph is composed of a general set of vertices. It groups all the frequent and especially all the infrequent recurrent patterns to generate a much more general pattern. Indeed, the frequency of this pattern is considered in one more dimension, the spatial one. So, the spatio-temporal frequency of this pattern is 5.

## 3.3 Interesting Measures and Constraints

Let $P = \langle\{\lambda_1; ...; \lambda_n\}, \{T_1 : Occurrence_1(\lambda_1); \ldots; Occurrence_1(\lambda_n) \,|\, \ldots |T_k : Occurrence_k(\lambda_1); \ldots; Occurrence_k(\lambda_n)\}\rangle$ be a pattern. Several measures and constraints are defined for two purposes: (i) to let the user express his preferences to select patterns via a set of constraints, and (ii) to reduce the search space and improve the efficiency of the algorithm.

***Spatio-temporal frequency*** The frequency constraint, denoted as ***minsup***, is a user-defined threshold to filter patterns which occur more than a minimum number in time and in space. The frequency of $P$ is the number of occurrences of pattern $P$, $sup(P) = k$. Consequently, $P$ is a frequent evolution iff $sup(P) \geq minsup$. For example, in Fig. 1, the frequency of the sequence $(a1-a2+, a1-a2+, a1+a2=); (a1-a2+, a1+a2+, a1+a2-, a1-a2-)$ is 5, as the pattern appears 5 times.

***Connectivity*** During pattern extraction, vertices should be connected by edges to extract potentially correlated evolutions among a set of objects. In Fig. 1, the pattern $(a1-a2+, a1-a2+, a1+a2=); (a1-a2+, a1+a2+, a1+a2-, a1-a2-)$ occurs in $\{t_1, t_2\}$ on a sequence of sets of connected vertices (or subgraphs) such as $(1, 2, 3); (1, 4, 5, 6)$.

***Volume*** The volume measure defines the number of vertices of a subgraph. Let $vol(P) = \min_{\forall i \in [1,n]} |\lambda_i|$ be the volume of a pattern $P$. A pattern $P$ is sufficiently voluminous iff $vol(P) \geq minvol$, where $\boldsymbol{minvol}$ is a minimum number of vertices of a subgraph. The user can also define the maximum number of vertices of a subgraph, denoted as $\boldsymbol{maxvol}$, such as $vol(P) \leq maxvol$. For example, the pattern $(a1-a2+, a1-a2+, a1+a2=);(a1-a2+, a1+a2+, a1+a2-, a1-a2-)$ has a volume of 3.

***Temporal continuity*** An evolution may include different vertices at each timestamp. However, it is difficult for end users to interpret the evolution of vertices without a direct relation between them at each step. Hence, it is desirable to study evolution around a common core of vertices. To do so, a constraint, denoted as $\boldsymbol{mincom}$, is defined to follow a minimum number of common vertices over time. Let denote $Occurrence_j(P)$ is a $j^{th}$ instance of pattern $P$ and $com(Occurrence_j(P)) = |\cap_{\forall i \in 1,...,n} Occurrence_j(\lambda_i)|$ be the common number of vertices occurring in the instance sequence $j$. $P$ is a continuous pattern iff $\forall j \in \{1,..,k\}$ $com(Occurrence_j(P)) \geq mincom$. Consider $P$ the pattern $(a2+, a1-a2+, a1+a2=);(a1-a2+, a1+a2+, a1+a2-, a1-a2-)$ in Fig 1. All instances of the pattern $P$ have at least one common vertex. For instance, the subgraphs of the occurrence $(7, 8, 10); (8, 11, 12, 17)$, at $t_1$ and $t_2$, have one common vertex, which is 8.

## 4 Mining Frequent Sequential Subgraph Evolutions

***Problem Setting*** Given a dynamic attributed graph $\mathcal{G}$, the problem is to extract the complete set of frequent sequential subgraph evolutions in $\mathcal{G}$, denoted as $Sol$, such that $\forall P \in Sol$, (i) $P$ is frequent (i.e., $sup(P) \geq minsup$); (ii) the occurrences of $P$ are connected at each time; (iii) $P$ is sufficiently voluminous (i.e., $minvol \leq vol(P) \leq maxvol$); (iv) $P$ is centered around a core of vertices sufficiently large (i.e., $com(P) \geq mincom$), where $maxvol, minsup$ and $mincom$ are user-defined thresholds.

In this section, we propose an algorithm, called FSSEMiner, to mine FSSE patterns in a dynamic attributed graph. This algorithm can be divided in three main steps: (i) identify subgraphs (Section 4.1); (ii) count the frequency of subgraphs over time (Section 4.2); (iii) construct sequences of subgraphs using frequent subgraphs (Section 4.3). The sequence of the three steps is illustrated via the FSSEMiner algorithm (Algorihtm 1)).

### 4.1 Extraction of Subgraph Candidates

The first step of the algorithm is to construct all possible candidate subgraphs (frequent and infrequent) using a dynamic attributed graph as input (Lines 1-2). More precisely, the algorithm constructs all possible sets of patterns $\lambda_i$ whose volume is between $minvol$ and $maxvol$. For each generated pattern $\lambda_i$, a depth-first search (DFS) strategy is used to compute its occurrences $Occurrence(\lambda_i)$ in each $G_t \in \mathcal{G}$. The anti-monotonicity property is respected to find anti-monotonic subgraphs [4]. The result is the set of subgraphs satisfying the *volume* and *connectivity* constraints and denoted as $\$ = \{\$_i$ set of subgraphs of $G_t, t \in T \mid$

**Algorithm 1:** $FSSEMiner$ : Mining frequent sequential subgraph evolutions

```
Input: $\mathcal{G}$ : a dynamic attributed graph , $minsup$, $minvol$, $mincom$
Output: $Sol$: set of frequent sequential subgraph evolutions satisfying the constraints
/* Step 1: Extraction of subgraph candidates */
1  $\mathbb{S} = \{\mathbb{S}_i$ set of subgraphs of $G_t, t \in T \mid \forall s_i \in \mathbb{S}_i, s_i = (\lambda_i, Occurrence(\lambda_i)\, in\, t), minvol \leq |Occurrence(\lambda_i)| \leq maxvol\}$
2  $Cand_i = \emptyset, \forall i \in \{1, 2, ..., |\mathcal{T}|\}$
/* Step 2: Generation size-1 patterns by graph addition */
3  for $k = 1$ to $|\mathcal{T}|$ do
4      for each $T_1^k \subseteq \mathcal{T}$ such as $t_1 \in T_1^k$ do
5          $\mathbb{P}_{union} = \{\mathbb{S}_{union}$ set of frequent subgraphs in $T_1^k, \mid \forall s_{union} \in \mathbb{S}_{union}, s_{union} = (\lambda, Occurrence_{union}), Occurrence_{union} = Union(Occurrence_t)$ where $(t \in T_1^k)$ and $|Occurrence_{union}| \geq minsup\}$
6          $Cand_1 = Cand_1 \cup \mathbb{P}_{union}$
7      end
8  end
/* Step 3: Extension of patterns */
9  $Sol_i = \emptyset, \forall i \in \{1, 2, ..., |\mathcal{T}|\}$
10 for $i = 2$ to $|\mathcal{T}|$ do
11     for each $T_i^k \subseteq \mathcal{T}$ such as $t_i \in T_i^k$ do
12         for each $P_i \in \mathbb{P}_{union} = \{\mathbb{S}_{union}$ set of frequent subgraphs in $T_i^k \mid \forall s_{union} \in \mathbb{S}_{union}, s_{union} = (\lambda, Occurrence_{union}), Occurrence_{union} = Union(Occurrence_t),$ where $(t \in T_i^k)$ and $|Occurrence_{union}| \geq minsup\}$ do
13             for each $P$ such as $P \in Cand_{i-1}$ do
14                 $P' = ExtendWith(P, P_i)$
15                 if $com(P') \geq mincom$ and $|P'| \geq minsup$ then
16                     $Cand_i = Cand_i \cup \{P'\}$
17                 end
18                 else
19                     $Sol_{i-1} = Sol_{i-1} \cup \{P\}$
20                     $Cand_i = Cand_i \cup \{P_i\}$
21                 end
22             end
23         end
24     end
25 end
26 $Sol = MergeUpdate(\bigcup_{\forall i \in \mathcal{T}} Sol_i)$
```

$\forall s_i \in \mathbb{S}_i, s_i = (\lambda_i, Occurrence(\lambda_i)\, in\, t), minvol \leq |Occurrence(\lambda_i) \leq maxvol\}$. For example, occurrences of $(a1 - a2+, a1 - a2+, a1 + a2 =)$ are extracted at each time. They are represented by the two red connected subgraphs in $t_1$ and $t_2$ in Figure 1.

## 4.2 Generation of Size-1 Patterns by Graph Addition

The second step of the algorithm is to combine the candidate subgraphs generated in the previous step (Section 4.1) to form size-1 patterns (i.e., sequences composed of a single subgraph) (Line 3-8). The construction of size-1 patterns is the fundamental building block for constructing the final patterns. To do so, a new strategy, called graph addition, is proposed. It consists in adding all the occurrences of a candidate subgraph at different times for the same pattern. Then, the algorithm verifies if the spatio-temporal frequency of this candidate subgraph respects the *minsup* constraint.

Let be $n$ times $t_i, ..., t_j \in T$, where $1 \leq n \leq |T|$ and $1 \leq i < j \leq |T|$. The addition of $n$ subgraphs $s_i = (\lambda_i, Occurrence(\lambda_i)\, in\, t_i)$,..., $s_j = (\lambda_j, Occurrence(\lambda_j)\, in\, t_j)$ is denoted as $s_{union} = (\lambda, Occurrence_{union}(\lambda))$ where $\lambda = \lambda_i = ... = \lambda_j$ and $Occurrence_{union}(\lambda) = Occurrence(\lambda_i)\, in\, t_i \cup ... \cup Occurrence(\lambda_j)\, in\, t_j$. $s_{union}$ is a subgraph composed of the union of occurrences of the $n$ initial subgraphs having the same pattern (i.e., same attribute values). If $|Occurrence_{union}(\lambda)| \geq minsup$, the algorithm keeps $s_{union}$ in the mining process. For the special case where $n = 1$ and $i = j$ the result of the addition of a subgraph is itself. However, this case is necessary because a size-1 pattern (one subgraph) could also be spatially frequent in one timestamp.

Graph addition is applied to all sets of time combinations, denoted as $T^k$, $1 \leq k \leq |T|$. The number of linear additions is $|T^k| = 2^{|T|} - 1$ which depends only on the number of timestamps in $\mathcal{G}$. The advantage of the graph addition strategy is to avoid performing a huge amount of subgraph traversals for the generation of patterns of size 1. Let us suppose that $minsup = 4$. In Fig. 1, there is the subgraph $s_1 = \langle\{(a1 - a2+, a1 - a2+, a1 + a2 =)\}, \{t_1 : (1, 2, 3)|t_1 : (7, 8, 10)|t_1 : (13, 14, 15)\}$ in $t_1$ and $s_2 = \langle\{(a1 - a2+, a1 - a2+, a1 + a2 =)\}, \{t_2 : (1, 2, 3)|t_2 : (7, 8, 10)\}$ in $t_2$ having the same pattern. By adding $s_1$ and $s_2$, the pattern $s_{union} = \langle\{(a1-a2+, a1-a2+, a1+a2 =)\}, \{t_1 : (1, 2, 3)|t_1 : (7, 8, 10)|t_1 : (13, 14, 15)|t_2 : (1, 2, 3)|t_2 : (7, 8, 10)\}$ is obtained. It can be observed that $s_1$ and $s_2$ are infrequent. However, $s_{union}$ is frequent after the addition of the graphs.

## 4.3 Extension of Patterns

The final step of the algorithm is to construct the complete sequential patterns by extending each size-1 pattern of each successive set of times generated in the previous step (Section 4.2) (Line 9-26). To do this, size-1 patterns are iteratively extended by checking the *mincom* and *minsup* constraints to connect other consecutive patterns to build sequences of frequent subgraphs. This extension can be achieved by processing the times incrementally. Figure 2 shows an incremental construction of a pattern beginning from $\{t_1, t_2\}$. This figure displays the parallel extensions of a pattern which occurs at $t_1$ and $t_2$. Let $s$, $s'$ and $s^*$ be frequent subgraphs extracted in graph additions. Additions between $S_1$ and $S_2$ result in a set of frequent subgraphs, such that $s = (\lambda, Occurrence(\lambda)\, in\, t_1, t_2) \in S_1 \cup S_2$. Candidate extensions for these subgraphs can only be at $t_2$ and $t_3$ respectively (since gaps are not allowed). Now consider times $t_2, t_3$ and suppose that $s' = (\lambda', Occurrence(\lambda)'\, in\, t_2, t_3)$ is a frequent subgraph of $S_2 \cup S_3$. If $s$ and $s'$ have at least *minsup* occurrences verifying the temporal continuity constraint, then we can extend $s$ with $s'$ to obtain $P = \langle\{\lambda; \lambda'\}, \{t_1, t_2 : Occurrence(\lambda)\, in\, t_1, t_2; t_2, t_3 : Occurrence(\lambda')\, in\, t_2, t_3\}\rangle$. The process continues until no more extensions can be performed. At each iteration, subgraphs can be used to extend patterns from the previous iteration, but they can also be "starting points" for new patterns. For a sequence of 3 subgraphs, the pattern will be constructed and extended seven times (from $\{t_1\}$, from $\{t_2\}$, from $\{t_3\}$, from $\{t_1, t_2\}$, from $\{t_2, t_3\}$, from $\{t_1, t_3\}$, from $\{t_1, t_2, t_3\}$). Although the study of the combination $\{t_1, t_2\}$ does not bring more information

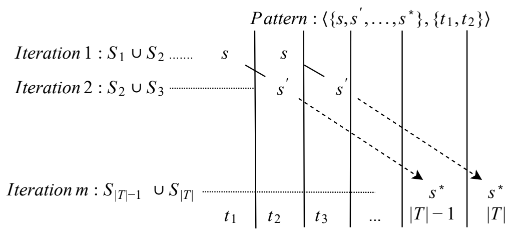

$Pattern : \langle\{s, s', \ldots, s^*\}, \{t_1, t_2\}\rangle$
$Iteration\,1 : S_1 \cup S_2$
$Iteration\,2 : S_2 \cup S_3$
$Iteration\,m : S_{|T|-1} \cup S_{|T|}$
$s$
$s$
$s'$
$s'$
$s^*$
$s^*$
$t_1$
$t_2$
$t_3$
...
$|T|-1$
$|T|$

Fig. 2: Additions and extensions of patterns from $\{t_1, t_2\}$

compared to $\{t_1, t_2, t_3\}$, it allows discovering other patterns to be extended. All these time combinations are therefore necessary. This highlights the importance of the proposed graph addition strategy described above, which requires only $|T|$ times instead of $2^{|T|} - 1$ graph traversals in a dynamic attributed graph.

## 5 Experiments

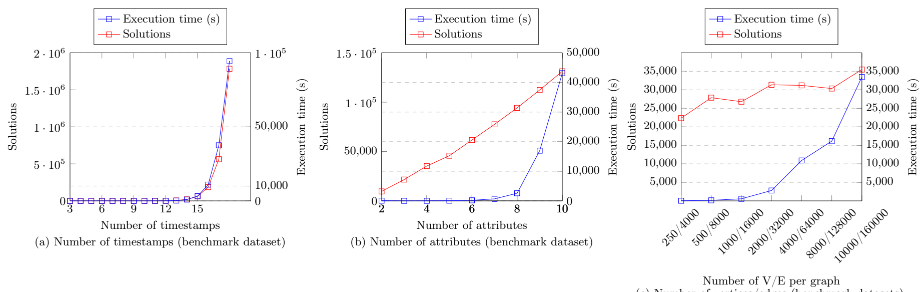
In this section, the performance of the FSSEMiner algorithm has been evaluated. The algorithm was implemented in C ++. Experiments were conducted on a PC (CPU: Intel(R) Core(T:) 3.5GHz) with 16 GB of main memory.

***Datasets*** Benchmark data were generated by varying different parameters: the number of vertices/attributes/edges and the number of graphs of the sequence by setting the other parameters (minsup, minvol, maxvol, mincom), where edges (pairs of vertices) and attribute values follow a uniform distribution. The first real-world dataset is the Domestic US Flights traffic dataset during the Katrina hurricane period (from 01/08/2005 to 25/09/2005) [3]. It is composed of 280 vertices (airports) and 1206 edges in average (flight connections) per timestamp, 8 timestamps (data are aggregated by weeks) and 8 attributes (e.g. number of departures/arrivals/cancelled flights). The second real-world dataset is composed by a travel flows in China dataset[6] and a COVID-19 daily cases dataset[7] during two periods (from 25/01/2020 to 20/03/2020 and from 15/04/2022 to 15/05/2022). It is composed of 232 vertices (cities) and 13260 edges (travel flows between cities) in average, 6 timestamps (data are aggregated every 3 days) and 4 attributes: the size of the city (small, medium-sized, big and megacity according to the population), the total number of new COVID cases, of deaths, of recoveries since 25/01/2020, with 4 values (=:no new cases, +:]0,5],++:]5,15] and +++:]15,]).

***Quantitative results*** We have conducted a quantitative analysis of patterns extracted from benchmark datasets to evaluate the scalability of the proposed algorithm. Fig.3 (a) shows the impact of the number of timestamps on the algorithm's runtimes for 2000 vertices and 8000 edges per graph, 2 attributes and minvol=maxvol=2, mincom=1, minsup=60% of the average number of vertices.

6 https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/FAEZIO

7 https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/MR5IJN



(a) Number of timestamps (benchmark dataset) (b) Number of attributes (benchmark dataset) (c) Number of vertices/edges (benchmark datasets).

Fig. 3: Impacts of parameters on execution times of FSSEMiner

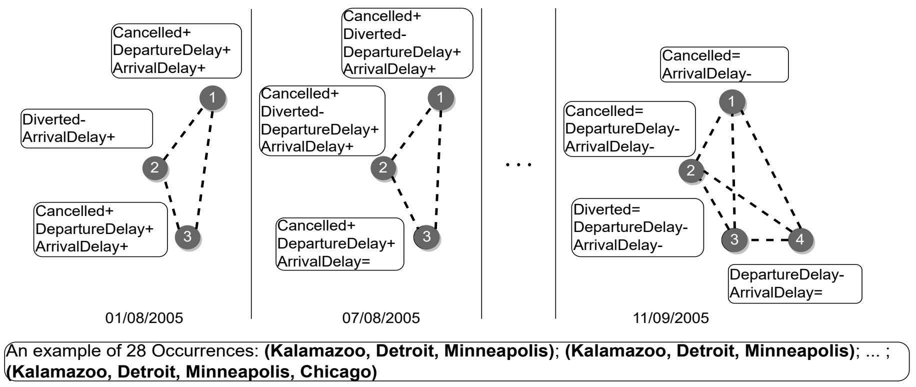


Fig. 4: A pattern in US Flights dataset

This impact is high, but performance of the algorithm is comparable with the one in [1] since it generates more general and more complex patterns. Fig.3 (b) shows the impact of the number of attributes on the algorithm's runtimes for 8 timestamps and the other parameters are fixed as before. Execution times remain low for less than 8 attributes, so for most of the real-world datasets. Fig.3 (c) shows the impact of the number of vertices and edges at each timestamp on the algorithm's runtimes (the other parameters are fixed as before). It can be noticed that the algorithm remains efficient for large dense graphs (10,000 vertices and 160,000 edges at each timestamp).

***Qualitative results*** We have conducted a qualitative analysis of patterns extracted on the real-world datasets. Fig. 4 shows an example of a pattern extracted from the US Flights traffic dataset (minvol =2, maxvol=4, minsup=25, mincom=1). This pattern appears 28 times in the dataset. It shows the impact of hurricanes on US airport traffic for 6 weeks. First, it is observed that delays and cancellations increased at destination and arrival airports, while diverted flights always remained the same when the hurricane came. It shows that hurricanes have strong impact on cancellations but have hardly any impact on flight diversion. Second, it is noticed that cancellations and diverted flights became stable

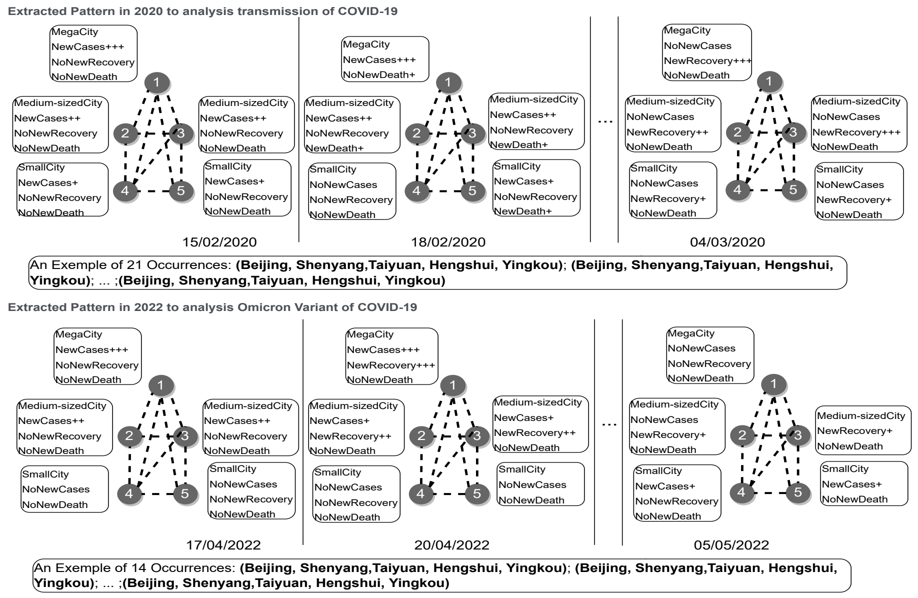

Extracted Pattern in 2020 to analysis transmission of COVID-19
MegaCity NewCases+++ NoNewRecovery NoNewDeath
Medium-sizedCity NewCases++ NoNewRecovery NoNewDeath
Medium-sizedCity NewCases++ NoNewRecovery NoNewDeath
SmallCity NewCases+ NoNewRecovery NoNewDeath
SmallCity NewCases+ NoNewRecovery NoNewDeath
15/02/2020
MegaCity NewCases+++ NoNewDeath+
Medium-sizedCity NewCases++ NoNewRecovery NewDeath+
Medium-sizedCity NewCases++ NoNewRecovery NewDeath+
SmallCity NoNewCases NoNewRecovery NoNewDeath
SmallCity NewCases+ NoNewRecovery NewDeath+
18/02/2020
...
MegaCity NoNewCases NewRecovery+++ NoNewDeath
Medium-sizedCity NoNewCases NewRecovery++ NoNewDeath
Medium-sizedCity NoNewCases NewRecovery+++ NoNewDeath
SmallCity NoNewCases NewRecovery+ NoNewDeath
SmallCity NoNewCases NewRecovery+ NoNewDeath
04/03/2020
1 2 3 4 5
An Exemple of 21 Occurrences: (Beijing, Shenyang,Taiyuan, Hengshui, Yingkou); (Beijing, Shenyang,Taiyuan, Hengshui, Yingkou); ... ;(Beijing, Shenyang,Taiyuan, Hengshui, Yingkou)
Extracted Pattern in 2022 to analysis Omicron Variant of COVID-19
MegaCity NewCases+++ NoNewRecovery NoNewDeath
Medium-sizedCity NewCases++ NoNewRecovery NoNewDeath
Medium-sizedCity NewCases++ NoNewRecovery NoNewDeath
SmallCity NoNewCases NoNewDeath
SmallCity NoNewCases NoNewRecovery NoNewDeath
17/04/2022
MegaCity NewCases+++ NewRecovery+++ NoNewDeath
Medium-sizedCity NewCases+ NewRecovery++ NoNewDeath
Medium-sizedCity NewCases+ NewRecovery++ NoNewDeath
SmallCity NoNewCases NoNewRecovery NoNewDeath
SmallCity NoNewCases NoNewDeath
20/04/2022
...
MegaCity NoNewCases NoNewRecovery NoNewDeath
Medium-sizedCity NoNewCases NewRecovery+ NoNewDeath
Medium-sizedCity NewRecovery+ NoNewDeath
SmallCity NewCases+ NoNewRecovery NoNewDeath
SmallCity NewCases+ NoNewDeath
05/05/2022
An Exemple of 14 Occurrences: (Beijing, Shenyang,Taiyuan, Hengshui, Yingkou); (Beijing, Shenyang,Taiyuan, Hengshui, Yingkou); ... ;(Beijing, Shenyang,Taiyuan, Hengshui, Yingkou)

Fig. 5: Two patterns extracted from COVID dataset

while delays decreased when the hurricane became weaker at the end. Third, this pattern shows the evolution in terms of network (the set of three airports became four). It is observed that new flight routes via a new airport were added by airlines from 11/09/2005. Moreover, referring to map, we notice that the new added airport (for example, Chicago) in the airline network is usually located at the centre position of the previous airport network (for example, Kalamazoo, Detroit and Minneapolis), which ensures that airline connections are more convenient as it is close to all other airports. It is also observed from the occurrences of patterns that hurricanes have much stronger influence on close airports which are located in the East Coast of the United States, while western airports are much less affected.

In Fig. 5, we compare two extracted patterns to analyse the transmission of COVID and its variant Omicron (minvol =5, maxvol=8, minsup=15, mincom=5). This pattern appears 21 times in the dataset. First, the two chosen patterns highlight the transmission of COVID in a mixed city network which is composed of mega, medium and small cities. We note that the city size and new case numbers are strongly correlated. In 2020 and 2022, COVID spreads very quickly in medium and mega cities, while new cases in small cities shown almost zero growth. It is probably because in small cities, the transport connections are much easier to control. For example, a small city has in general only two train stations and one airport while in medium-sized and big cities, it could have up to 109 train stations and 12 airports. Moreover, the flow is in general 30 times higher than small cities, which makes it much more difficult to miss any positive case. Second, it is observed that the COVID caused many severe consequences in 2020, as the death began to increase in three days after the emergence of new COVID cases, and it took in average more than 10 days for recovery. While in 2022, the virulence of variant Omicron became much weaker, as there are almost no deaths, and the recoveries began to emerge only three days after new detected cases. This analysis is very useful for anti-epidemic measures. As many countries began to cancel isolation policies in 2022. To conclude, these patterns allow studying virus transmission in different scale of cities (among big cities, small cities, medium-sized or mixed city network).

## 6 Conclusion

This paper has proposed a novel type of patterns called *frequent sequential subgraph evolutions* (FSSE) in a dynamic attributed graph. Its main advantage is to represent evolutions of general groups of objects. To mine FSSE, we have proposed the *FSSEMiner* algorithm. The latter is based on a novel mining strategy, called *graph addition*, to reduce computation time considerably. Experiments on both benchmark and real-world datasets have shown the scalability of the algorithm and the interest of these patterns. In the short-term, we plan to make new applications by using new datasets and study the explainability of the found patterns. In the long-term, a distributed version of the algorithm could be developed to further improve the performance.

## References

1. Cheng, Z., Flouvat, F., Selmaoui-Folcher, N.: Mining Recurrent Patterns in a Dynamic Attributed Graph. In: Advances in Knowledge Discovery and Data Mining, vol. 10235, pp. 631–643. Springer (2017)
2. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.F.: Cohesive co-evolution patterns in dynamic attributed graphs. In: International Conference on Discovery Science. pp. 110–124. Springer (2012)
3. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.F.: Trend mining in dynamic attributed graphs. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 654–669. Springer (2013)
4. Fiedler, M., Borgelt, C.: Support computation for mining frequent subgraphs in a single graph. In: MLG (2007)
5. Fournier-Viger, P., Cheng, C., Cheng, Z., Lin, J.C.W., Selmaoui-Folcher, N.: Mining significant trend sequences in dynamic attributed graphs. Knowledge-Based Systems **182**, 104797 (2019)
6. Fournier-Viger, P., He, G., Cheng, C., Li, J., Zhou, M., Lin, J.C.W., Yun, U.: A survey of pattern mining in dynamic graphs. WIREs Data Mining and Knowledge Discovery **10**(6), e1372 (2020)
7. Fournier-Viger, P., He, G., Lin, J.C.W., Gomes, H.M.: Mining attribute evolution rules in dynamic attributed graphs. In: International Conference on Big Data Analytics and Knowledge Discovery. pp. 167–182. Springer (2020)
8. Kaytoue, M., Pitarch, Y., Plantevit, M., Robardet, C.: Triggering patterns of topology changes in dynamic graphs. In: International Conference on Advances in Social Networks Analysis and Mining. pp. 158–165. IEEE (2014)