Finding Strongly Correlated Trends in Dynamic Attributed Graphs

Philippe Fournier-Viger[1], Chao Cheng[2], Zhi Cheng[3],
Jerry Chun-Wei Lin[4], and Nazha Selmaoui-Folcher[3]

[1] School of Natural Sciences and Humanities,
Harbin Institute of Technology, Shenzhen, China
[2] School of Computer Sciences and Technology,
Harbin Institute of Technology, Shenzhen, China
[3] University of New Caledonia, ISEA, BP R4, F-98851 Noumea, New Caledonia
[4] Department of Computing, Mathematics and Physics, Western Norway University
of Applied Sciences (HVL), Bergen, Norway
philfv8@yahoo.com, tidescheng@gmail.com, zhi.cheng@univ-nc.nc,
jerrylin@ieee.org,nazha.selmaoui@univ-nc.nc

Abstract. Discovering patterns in dynamic attributed graphs allows to capture how attribute values and graph structures changes over time. This allows to understand how a graph has evolved and may change in the future, to support decision-making. But an important limitation of current studies is that they mainly select patterns based on their frequency. Thus, they may find many frequent but weakly correlated patterns. To discover strongly correlated patterns, this paper proposes a novel significance measure named *Sequence Virtual Growth Rate*. It allows evaluating if a pattern represents entities that are correlated in terms of their proximity in a graph over time. Based on this measure a novel type of graph patterns is defined called *Significant Trend Sequence*. To efficiently mine these patterns, an algorithm named TSeqMiner is proposed, which relies on a novel upper bound and pruning strategy to reduce the search space. Experiments show that the algorithm is efficient and can identify interesting patterns in social network and spatio-temporal data.

Keywords: Dynamic attributed graph · Significant sequential patterns

1 Introduction

In the last decades, analyzing graphs has received an increasing amount of attention from the data mining community. The reason is that graphs naturally capture the structure of data in many domains such as in social network and wireless sensor network analysis, and bioinformatics [1, 7, 12, 16, 18]. Discovering patterns in a dynamic attributed graph such as trends is useful to understand how the graph evolved in terms of attribute values and graph structure, and how it may change in the future. Several algorithms have been proposed for mining patterns in dynamic attributed graphs (graphs evolving other time where nodes

have multiple numeric attributes) [3–5, 14]. However, most algorithms select patterns based on their occurrence frequency. Using the frequency as main criterion to select patterns can filter some noise but it can result in discovering many weakly correlated patterns whose constituting elements apppeared together by chance. To illustrate this issue, consider Figure 1, which will be used as running example. It depicts the evolution of a dynamic attributed graph over six timestamps. In this graph, nodes are labelled using numbers from 1 to 5 and attributes are labelled as a_1, a_2 and a_3. Figure 1 a), b), c), d) and e) respectively depict the evolution of the graph from the 1st to 2nd, 2nd to 3rd, 3rd to 4th, 4th to 5th, and 5th to 6th timestamps. Attribute values are represented using the "+", or "-" trend symbols to indicate whether each attribute value has increased or decreased since the previous timestamp, respectively. Now consider that one wants to find patterns indicating that a vertex influences the trends of its neighbors for the following timestamp. If only the frequency is considered, patterns such as $\langle\{a_1+\},\{a_3+\}\rangle$ can be found. It indicates that an increase of attribute a_1 for a vertex is followed by an increase of a_3 for a neighbor. However, since $\{a_3+\}$ appears almost everywhere in the graph, the pattern $\langle\{a_1+\},\{a_3+\}\rangle$ is weakly correlated, and uninteresting. Conversely, the pattern $\langle\{a_1+, a_2+\},\{a_3-\}\rangle$ has a relatively low support but is a strongly correlated trend. In fact, the trend $\{a_3-\}$ does not frequently appear globally but it often locally appears, followed by the trend $\{a_1+, a_2+\}$. To filter the many spurious weakly correlated patterns and present only the strongly correlated ones to the user, a novel significance measure is needed. Such measure would allow to find that $\{a_3-\}$ is more likely to appear after the observation $\{a_1+, a_2+\}$ than $\{a_3+\}$ is likely to follow $\{a_1+\}$.

To address this issue, this paper proposes the novel problem of mining significant trend sequences in dynamic attributed graphs. It defines a novel significance measure called *Sequence Virtual Growth Rate* to identify correlated patterns. Furthermore, an algorithm named TSeqMiner is presented to find these correlated patterns efficiently using novel search space pruning techniques.

The rest of this paper is organized as follows. Section 2 reviews related work on pattern mining in dynamic attributed graphs. Section 3 presents the proposed problem. Section 4 describes the algorithm and its pruning techniques. Section 5 presents results of a performance evaluation and discusses patterns found in real-life social network and flight data. Finally, Section 6 draws the conclusion.

2 Related work

This section gives an overview of relevant related work about mining patterns in dynamic attributed graphs, emerging patterns, and spatio-temporal patterns.

Recently, much attention has been given to mining patterns representing trends in dynamic attributed graphs, which capture how entity relationships and entity attributes change over time. Identifying such trend patterns is useful for both the commercial and scientific communities for many applications such as social network analysis [2]. To analyze a vertex-weighted dynamic network (a dynamic attributed graph with a single attribute), Jin et al. [14] mined con-

nected subgraphs whose vertices show the same trend during a time interval. Such pattern can reveal important events occurring in a dynamic system. However, an important limitation of this work is that it considers a single attribute with only two trend types, that is an increase (+) or decrease (-), which restricts its applications. A second limitation is that it enforces a strict constraint that timestamps must be consecutive in each time interval. Desmier et al. [4, 5] generalized this work by considering patterns with multiple attributes and non-consecutive timestamps. This type of patterns is called *co-evolution patterns*. A co-evolution pattern is a graph representing attribute variations between two timestamps. A co-evaluation pattern may appear multiple times for different timestamps of a dynamic attributed graph. Besides, several constraints were introduced to obtain an efficient algorithm and find more useful patterns. Cheng et al. [3] then proposed a more general type of patterns called *recurrent patterns*, which are frequent sequences of graphs rather than a single graph. Each graph of a recurrent pattern may contain different vertices and trends. This type of patterns allows to capture the relationships between trends in a graph over time. Although finding such patterns is useful, many recurrent patterns may contain graphs that are weakly correlated. For example, a pattern containing a sequence of two graphs may be frequent only because the second graph appears very frequently over time (similarly to the example presented in the introduction with a_3+). Hence, a significance measure is required to find patterns containing correlated graph entities. Such measure will be presented in this paper.

Another related work is the discovery of *emerging patterns* in databases. In emerging pattern mining, the *growth rate* measure is used to find patterns having a frequency that is largely different from one database to another [6]. Kaytoue et al. [15] used the growth rate to measure the confidence that variations of attribute values will trigger topological changes. A triggering pattern is a sequence of attribute variations, followed by a single topological variation. However, a limitation of this work is that the growth rate is only calculated to assess the influence of the last attribute variations of a triggering pattern on the topology variation. Thus, this approach may find patterns where attribute variations are weakly correlated over time, and thus not meaningful. The novel significance measure presented in this paper addresses this limitation.

Spatio-temporal (ST) data mining is another related field. The relationship between graph mining and ST data mining is that (1) graphs are often used to model spatial information [18], and (2) graphs are often interpreted from a spatial perspective (e.g. the shortest path between two vertices can be considered as a spatial distance) [17]. In ST data mining, Huang et al. [13] proposed to mine significant sequential patterns in spatio-temporal data using a significance measure similar to the growth rate, called *sequence index*, to ensure that events in sequences are strongly correlated. Although the concept of pattern significance as defined by the sequence index is useful, it cannot be directly applied to mine patterns in a dynamic attributed graph. The main reason is that this work considers that two events cannot co-occur. In dynamic attributed graphs, time and locations are discrete and multiple trend types often appear simultaneously

for different vertices and timestamps. Allowing simultaneous events is desirable but greatly increases the size of the search space.

To address this problem, reasonable constraints and efficient pruning techniques must be designed to avoid exploring the whole search space. Another challenge for applying the sequence index in dynamic graph minng is that ST mining mainly considers the distance relationship between events as spatial information. But in a dynamic graph, vertices and edges provide much richer information as the topology of a graph may dynamically change over time. This enables users to adopt various neighborhood definitions suia_1 for specific applications.

In summary, pattern significance is an important concept in data mining but has not been fully introduced in dynamic attributed graph mining. In some related work, the concept of significant sequential patterns was defined for spatio-temporal mining [13]. But it is not trivial to extend these concepts for dynamic attributed graph mining.

3 Notations and problem definition

To address the aforementionned limitations of previous work, this section defines a novel problem of mining significant patterns in dynamic attributed graphs. Definitions are presented, and then the problem.

Dynamic attributed graph. Let there be a set of timestamps $\mathcal{T} = \{1, 2, \ldots, t_{max}\}$, a set of vertices $\mathcal{V}$ and a set of attributes $\mathcal{A}$. A dynamic attributed graph is a sequence of attributed graphs $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \langle E_1, E_2, ..., E_{t_{max}} \rangle, \langle \lambda_1, \lambda_2, ..., \lambda_{t_{max}} \rangle)$, where the attributed graph for timestamp $t \in \mathcal{T}$ is defined by a set of edges $E_t \subseteq \mathcal{V} \times \mathcal{V}$ and a function $\lambda_t : \mathcal{V} \times \mathcal{A} \rightarrow \mathbb{R}$ that associates a real value to each vertex-attribute pair. In the following, the attributed graph at timestamp t of a dynamic attributed graph $\mathcal{G}$ is denoted as $G_t = (\mathcal{V}, \mathcal{A}, E_t, \lambda_t)$. Notice that the vertices in the dynamic graph are not changing. Fig. 1 illustrates a dynamic attributed graph where trends were obtained by preprocessing an original dynamic attributed graph with six timestamps.

Trend set, trend sequence and point. A *trend* is an attribute variation, such as a_1+, a_2-. A *trend set* is a set of attribute variations, such as $\{a_1+, a_2+\}$, $\{a_1+, a_2-, a_3+\}$. A k-trend-sequence tss is an ordered list of k trend sets. Let the notation $tss[i]$ denotes the i-th trend set of tss. Moreover, let $tss[i:j]$ denotes the subsequence consisting of the i-th trend set to the (j-1)-th trend set. A point $p = (t, v)$ is a tuple consisting of a timestamp t and a vertex v. For example, $tss = \langle \{a_1+, a_2+\}, \{a_3-\}, \{a_2+, a_3+\} \rangle$ is a 3-trend-sequence. $tss[2] = \{a_3-\}$, $tss[2:4] = \langle \{a_3-\}, \{a_2+, a_3+\} \rangle$. In the graph of Fig. 1, there are 5 points, i.e. $\{(t_1, 1), (t_1, 2), (t_1, 3), (t_1, 4), (t_1, 5)\}$.

Neighboring space. The neighboring space $Ns(p)$ of a point $p = (t, v)$ in a dynamic attributed graph is defined as a set of points that are neighbors of p, i.e. $Ns(p) = \{p' \mid neighborhood(p', p) = true\}$, where *neighborhood* is a boolean function indicating if two points p' and p are neighbors. The neighboring space of a set of points ps is defined as the union of the neighboring spaces of each point in ps, i.e. $Ns(ps) = \{\ p' |\ \exists p, neighborhood(p', p) = true$ and $p \in ps\} =$

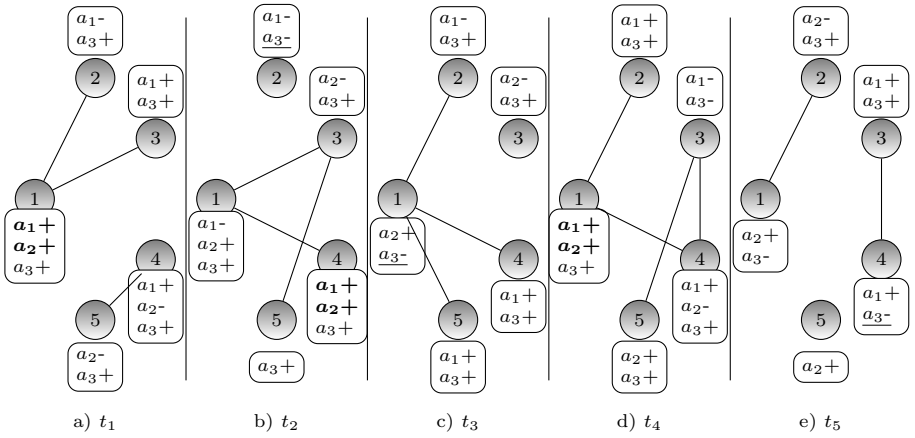

Fig. 1: Trend graphs obtained by preprocessing a dynamic attributed graph originally having 6 timestamps. Occurrences of the pattern $< \{a_1+, a_2+\}, \{a_3-\} >$ and $< \{a_3-\} >$ are highligthed in bold and with an underline, respectively.

$\bigcup_{p \in ps} Ns(p)$. The above definition depends on that of the *neighborhood* predicate, which can be parameterized for various applications. A simple setting is to consider $neighborhood(p', p) = true$ for two points p and p' if only if p' appears at the timestamp following p, and the vertex of p' is connected to that of p at previous timestamp. This is the definition that will be used in the rest of this paper. For the above setting, in Fig. 1, $Ns\left((t_1, 1)\right) = \{(t_2, 2), (t_2, 3)\}$, $Ns\left((t_4, 1)\right) = \{(t_5, 2), (t_5, 4)\}$, $Ns\{(t_1, 1), (t_4, 1)\}) = \{(t_2, 2), (t_2, 3), (t_5, 2), (t_5, 4)\}$.

Supporting points. Let $ts(p)$ denotes the trend set of a point p. The supporting points of a trend set ts are the points that have a trend set that is a superset of ts, i.e. $SP(ts) = \{p \mid ts \subseteq ts(p)\}$. For a k-trend sequence tss, if $k = 1$, the tail supporting points of tss are the supporting points of $tss[1]$. If $k > 1$, the tail supporting points of tss are the points that support $tss[k]$ in the neighboring space formed by all tail supporting points of $tss[1 : k]$, i.e. $TSP(tss) = \{p \mid p \in Ns(TSP(tss[1 : k]))$ and $tss[k] \subseteq ts(p)\}$ for $k \geq 2$. For example, $TSP(tss[1 : 2]) = SP(tss[1]) = \{(t_1, 1), (t_2, 4), (t_4, 1)\}$, $Ns(TSP(tss[1 : 2])) = \{(t_2, 2), (t_2, 3), (t_3, 1), (t_5, 2), (t_5, 4)\}$, $TSP(tss[1 : 3]) = \{(t_2, 2), (t_3, 1), (t_5, 2)\}$.

Significance measure. The support of a trend set ts in a neighboring space Ns is defined as: $supp(ts, Ns) = \frac{|\{p \mid p \in Ns \ \ and \ \ ts \subseteq ts(p)\}|}{|Ns|}$. For example, $supp(\{a_3-\}, Ws) = \frac{5}{25}$. Thus, the *Virtual Growth Rate(VGR)* of ts in Ns is defined as: $VGR(ts, Ns) = \frac{supp(ts, Ns)}{supp(ts, Ws)}$, where Ws is the whole space, i.e. set of all points in $\mathcal{G}$.

For example, $VGR(\{a_3-\}, Ns(\{a_1+, a_2+\})) = \frac{supp(\{a_3-\}, Ns(\{a_1+, a_2+\}))}{supp(\{a_3-\}, Ws)} = \frac{\frac{3}{5}}{\frac{5}{25}} = 3$. The above measure can be viewed as an adaptation of the growth rate measure used in traditional emerging pattern mining. Next we adapts the VGR

measure to obtain a more general significance measure that can be applied for a k-trend sequence tss.

The *Sequence Virtual Growth Rate (SVGR)* of a k-trend-sequence ($k \geq 2$) tss is defined as: $SVGR(tss) = \min_{i \in [2,k]} VGR(tss[i], Ns(TSP(tss[1:i])))$ $k \geq 2$. The proposed significance measure is partially anti-monotone, i.e. for any k-trend-sequence tss such that $k \geq 2$, it follows that $SVGR(tss[1:i]) \geq SVGR(tss)$ for $i \in [3, k+1]$. The proof is omitted due to space limitation. The anti-monotonicity property is important for reducing the search space. To filter some noise, two frequency constraints are applied. A k-trend-sequence $tss(k \geq 2)$ is said to be frequent if and only if $|SP(tss[1])| \geq minInitSup$ and $|TSP(tss[1:i])| \geq minTailSup$ for $\forall i \in [3, k+1]$, where $minInitSup$ and $minTailSup$ are user specified thresholds. Due to the anti-monotone property of the support, the frequency constraints can be used to prune the search space. Then, a k-trend-sequence is said to be significant if and only if tss is frequent and $SVGR(tss) \geq minSig$, where $minSig$ is a parameter specified by the user.

Problem setting. Let there be a dynamic attributed graph, a significance threshold $minSig$, and two support thresholds $minInitSup$ and $minTailSup$. The problem of discovering all significant k-trend sequences is to find each significant trend sequence $SigTSeq$ such that $SVGR(SigTSeq) \geq minSig$, $|SP(SigTSeq[1])| \geq minInitSup$ and $|TSP(SigTSeq[1:i])| \geq minTailSup$ for $\forall i \in [3, k+1]$ (where k is the length of $SigTSeq$).

4 The TSeqMiner algorithm

The proposed algorithm, named TSeqMiner, performs a depth-first search to find the significant trend sequences, and applies two pruning strategies. In the following, the dynamic attributed graph of Fig. 1 is considered as example, with $minInitSup = 3$, $minTailSup = 1$, $minSig = 3$, and the neighborhood definition presented in Section 3.

4.1 The search space

The proposed algorithm first applies a frequent itemset mining algorithm such as *Eclat* [19] with the parameter $minInitSup$ to find all frequent trend sets. They are used to build the search space, as shown in Fig. 2. The search space can be viewed as containing two parts called *inner-levels* and *outer-levels*. The i-th outer-level contains all i-trend sequences, where some of them share a same (i-1)-trend sequence as prefix. The inner-level contains all frequent trend sets (organized in a certain way, which will be explained in Section 4.3). For example, the first outer-level is depicted in the top of Fig. 2, for the running example. It consists of a single inner-level, represented by the topmost dashed box. This box contains all frequent trend sets (1-trend-sequences), represented as a tree. The second outer-level consists of all 2-trend-sequences. In the illustration, part of this level is represented by the two bottommost boxes. Each box represents an inner-level of the second outer-level. For example, the leftmost box of Fig. 2 shows

trend sets that can be appended to the prefix 1-trend sequence $\langle\{a_1+\}\rangle$ to form 2-trend sequences. Similarly, the rightmost bottom box of Fig. 2 shows trend sets that can be appended to the prefix 1-trend sequence $\langle\{a_3+, a_1-\}\rangle$ to form 2-trend sequences. In the illustration, each dashed arrow represent extension(s) of a k-sequence to form $(k+1)$-trend sequence(s) having a same prefix.

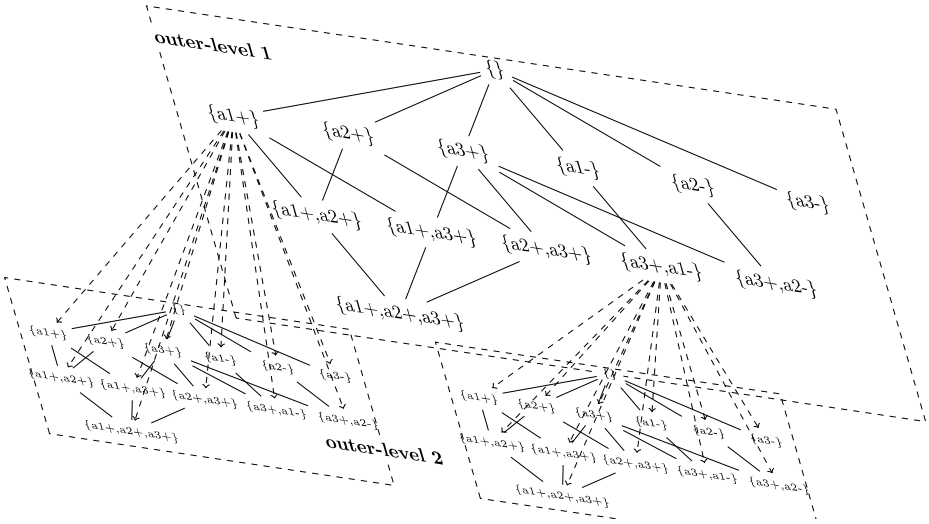

Fig. 2: The search space

4.2 Two pruning strategies

Outer-level pruning. Because $SVGR$ is anti-monotone, if a k-trend-sequence is insignificant, all the (k+1)-trend-sequences that extend that trend-sequence are also insignificant. Moreover, if a k-trend-sequence is infrequent, all its extensions are also infrequent.

Inner-level pruning. Pruning using $minTailSup$ can be done at the inner-level and is trivial. Hence, it is not discussed here and this paper focuses on inner pruning using $minSig$.

For the VGR measure, there is no pruning property. Consider a trend sequence tss and two trend sets R and T such that $T \subseteq R$. Let $VGR(tss \rightarrow T) = VGR(T, Ns(TSP(tss)))$. Then, $VGR(tss \rightarrow T) = \frac{supp(T, Ns(TSP(tss)))}{supp(T, Ws)}$. $VGR(tss \rightarrow R) = \frac{supp(R, Ns(TSP(tss)))}{supp(R, Ws)}$. Since $supp(R, Ws) \leq supp(T, Ws)$ and $supp(R, Ns(TSP(tss))) \leq supp(T,\ Ns(\ TSP(tss)))$ always hold, $VGR(tss \rightarrow T)$ can be larger than, equal to, or smaller than $VGR(tss \rightarrow R)$.

To address the lack of search space pruning property for the VGR measure, a lower bound on the VGR named lbs is proposed for pruning. Let there be a set

of trend sets I_s for which the support in Ws has already been measured. Besides note that in the following Ws is ignored to improve readability. Moreover, assume that $T \in I_s$. A lower bound on the support of T, named *lbs* is defined as: $lbs(T) = \min_{T \subset R \in I_s} lbs(R)$ if $\exists R \in I_s$ and $T \subset R$; $lbs(T) = supp(T)$ otherwise. The *lbs* measure is a lower bound on the support and is monotone. In other words, for two trend sets T_1, $T_2 \in I_s$ such that $T_1 \subseteq T_2$, $lbs(T_1) \leq supp(T_1)$ and $lbs(T_1) \leq lbs(T_2)$. *Proof of monotonicity*: if *a*) $T_1 = T_2$, $lbs(T_1) = lbs(T_2)$. *b*) else $lbs(T_1) = \min\{lbs(T_2), \min_{T_1 \subset R \in I_s, R \neq T_2} lbs(R)\} \leq lbs(T_2)$.

Proof of lower bound: *a*) if $\nexists R \in I_s$ and $R \supset T$, then $lbs(T) = supp(T)$. *b*) else there must $\exists R \in I_s, R \supset T$ such that $\nexists Q \in I_s, Q \supset R$. Then, $lbs(T) \leq lbs(R) = supp(R) \leq supp(T)$.

Next we introduce an Upper bound on the Virtual Growth Rate ($UVGR$), defined as $UVGR(tss, Ns) = \frac{supp(ts,Ns)}{lbs(ts,Ws)}$. According to properties of *lbs* and anti-monotonicity of *supp*, it is evident that $UVGR$ is an upper bound on the VGR and is anti-monotonous. Based on properties of $UVGR$, pruning techniques can be developed to accelerate the search for patterns.

4.3 A structure to support searching for patterns

During the search, quickly finding trend sets is necessary. To achieve this, a map is used. First, a total order $\prec$ of trend sets is defined. Without loss of generality, assume that the elements of any trend set ts are sorted according to the lexicographical order. Let $ts[i]$ denotes the i-th element of ts. For two trend sets $ts1$ and $ts2$, let $minL = \min\{|ts1|, |ts2|\}$. The trend set $ts1$ is said to precede $ts2$, denoted as $ts1 \prec ts2$, if and only if one of the following constraints is satisfied. 1) $\exists i \in [1, minL]$ such that $ts1[j] = ts2[j]$ for $j < i$ and $ts1[i] < ts2[i]$; 2) $\forall j \in [1, minL]$, $ts1[j] = ts2[j]$ and $|ts1| < |ts2|$. Then, $ts2$ is said to be a dominated superset of $ts1$ (and $ts1$ is said to be a dominating subset of $ts2$) if $ts1 \subset ts2$, $|ts1| + 1 = |ts2|$ and $ts1 \prec ts2$. For the running example, assume that the lexicographical order is $a_1+ < a_2+ < a_3+ < a_1- < a_2- < a_3-$. The search structure stores all dominance relationships, as shown in Fig. 3. The dominated supersets of $\{a_1+\}$ are $\{a_1+, a_2+\}$, $\{a_1+, a_3+\}$. When calculating $lbs(\{a_1+\}, Ws)$, the considered range is $Is = \{\{a_1+\}, \{a_1+, a_2+\}, \{a_1+, a_3+\}, \{a_1+, a_2+, a_3+\}\}$.

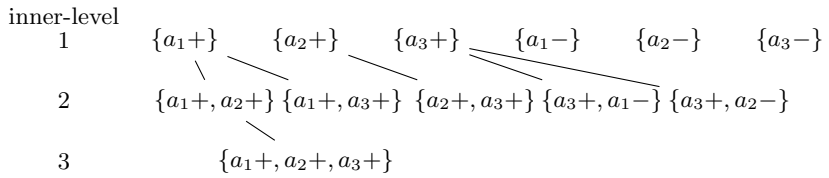

Fig. 3: The search structure.

4.4 The algorithm

The algorithm performs two steps, and consists of four main procedures presented in A.1, A.2, A.3 and A.4, where "A.x" denotes "Algorithm x".

Step 1. Generate 1-trend-sequence and maps. The algorithm scans the set of trend graphs once and acquires all trend sets with their supporting points. The *Eclat* algorithm is applied to find all frequent trend sets with their supporting points ($trendsetMapSP$ in A.1). Since this part is not the main contribution of this paper and can be easily implemented, details are omitted. Then, three maps are created. $levelMapTrendset$ maps each level to its trend sets. $trendsetMapDom$ maps each trend set to all its dominated supersets. $trendsetMapMin$ maps each trend set to its lbs value. Notice that to compute the lower bound on the support of ts, only trend sets in the subtree rooted at ts are considered. Calculating lbs in this way for pruning is called large-grained pruning. Another way of calculating lbs called medium-grained pruning is presented in Algorithm 4, to improve efficiency.

Step 2. Pattern extension. The algorithm performs a depth-first search. For each visited k-trend-sequence tss and its tail supporting points, a neighboring space $neighborSP$ is constructed based on the provided neighborhood definition $\mathcal{NH}$. Then, trend sets from this space are tested to generate (k+1)-trend-sequences with their supporting points. The algorithm starts the search from 1-trend-sequences, which are processed one by one (A.1 line 2-6). For each k-trend-sequence tss, if its size is larger than one, it is output (A.2 line 2-3). Then, the method $acquireNeighbors$ constructs the neighboring space based on the tail supporting points of tss (A.2 line 4-5). In this neighboring space, all trend sets with supporting points that have a large significance will be found (A.2 line 7). Then, the depth-first search is continued (A.2 line 8-9).

At the inner-level, trend sets are visited using a depth-first search on the search structure presented in Section 4.3 (A.3 1-3). The goal is to find trend sets that can extend a trend-sequence $prefix$ to generate larger trend-sequences. For each trend set ts, the method $computeLocalSup$ is applied to calculate the local support and supporting points of ts in the neighboring space (A.4 line 1). Then, the VGR of ts is derived (A.4 line 2-3). If the VGR of ts is not smaller than $minSig$, and the number of supporting points is not smaller than $minTailSup$, the extension $prefix + ts$ is saved (A.4 line 4-6). Otherwise, the $UVGR$ upper bound is calculated using the precalculated lower bound lbs on the support (A.4 line 8-9). Note that because the lbs of each trend set is precalculated, testing $UVGR$ is very fast. If this upper bound is smaller than the threshold, the subtree rooted at ts can be pruned (A.4 line 11). Otherwise, each dominated superset ts' of ts will be processed (A.4 line 19). If medium-grained pruning is activated (A.4 line 15), then lbs of ts is the lbs of ts' (A.4 line 16). It means that the trend sets of the subtree rooted at ts', together with ts, are used to calculate lbs. In that case, if $UVGR$ of ts is smaller than $minSig$, the subtree rooted at ts' can be pruned (A.4 line 17-18).

Algorithm 1: TSeqMiner$_{dfs}$

input : a set of trend graphs(tGs), $minSig$, $minInitSup$, $minTailSup$;
$\mathcal{NH}$: a neighborhood definition;
$trendsetMapSP$: map from each trend set to its set of supporting points;
$levelMapTrendset$: map from each level to its trend sets;
$trendsetMapDom$: map from each trend set to its dominated supersets;
$trendsetMapMin$: map from each trend set to its lbs lower bound on the support;
output: all significant trend sequences

```
1  addedTsList, addedSPList, prefix ← {};
2  foreach (ts, SP) ∈ trendsetMapSP do
3  |  addedTsList.add(ts);
4  |  addedSPList.add(SP);
5  end
6  outerDFSHelper(prefix, addedTsList, addedSPList);
```

Algorithm 2: OuterDFSHelper

input : $prefix$: a sequence of trend sets that form a sequence;
$newTrendsetList$: a list of trend sets to be appended to $prefix$;
$tailSPList$: a list of sets of supporting points corresponding to trend sets in $newTrendsetList$;

```
1  foreach ts ∈ newTrendsetList do
2  |  tss ← prefix + ts;
3  |  if tss.size > 1 then outputPattern(tss);
4  |  SP ← corresponding set of supporting points in tailSPList;
5  |  neighborSP ← acquireNeighbors(SP, NH);
6  |  addedTsList, addedSPList ← {};
7  |  innerDFS(neighborSP, addedTsList, addedSPList);
8  |  prefix ← tss;
9  |  outerDFSHelper(prefix, addedTsList, addedSPList);
10 |  remove the last trend set of prefix;
11 end
```

Algorithm 3: InnerDFS

input : $neighborSP$: neighboring space consisting of points;
$addedTsList$: a list of trend sets that are significant in the neighboring space;
$addedSPList$: supporting points of each trend set in $addedTsList$

```
1  foreach trend set ts ∈ levelMapTrendset.get(1) do
2  |  innerDFSHelper(ts, neighborSP, addedTsList, addedSPList)
3  end
```

4.5 Discussion

One of the main interest of discovering the type of patterns proposed in this paper is that patterns that contain strongly correlated entities can be found, while filtering other spurious uncorrelated patterns. Thus, a user can obtain a small subsets of correlated patterns that is significant.

A question about the proposed problem is why using two minimum frequency thresholds in addition to the minimum significance threshold. There are two reasons. First, by allowing multiple attribute variations to co-occur, the search space becomes quite large on real datasets. Thus, we cannot afford to explore all combinations of all possible variations. The minimum frequency thresholds thus act as a filter to avoid exploring the whole search space. Second, it is often unde-

Algorithm 4: InnerDFSHelper

```
input : ts: current processed trend set;
        neighborSP, addedTsList, addedSPList: same as in Algorithm 3.
1  localSup, SP ← computeLocalSup(neighborSP, ts);
2  supp ← trendsetMapSP.get(ts).size;
3  vGR ← localSup / supp;
4  if vGR ≥ minSig and |SP| ≥ minTailSup then
5  |  addedTsList.add(ts);
6  |  addedSPList.add(SP);
7  else
8  |  lbs ← trendsetMapMin.get(ts);
9  |  uVGR ← localSup / lbs;
10 |  // Large-grained pruning
11 |  if uVGR < minSig then return;
12 end
13 foreach ts' ∈ trendsetMapDom.get(ts) do
14 |  // Medium-grained pruning
15 |  if mGP then
16 |  |  lbs ← trendsetMapMin.get(ts');
17 |  |  uVGR ← localSup / lbs;
18 |  |  if uVGR ≥ minSig then
   |  |    innerDFSHelper(ts', neighborSP, addedTsList, addedSPList);
19 |  else innerDFSHelper(ts', neighborSP, addedTsList, addedSPList);
20 end
```

sirable to explore very low frequency patterns because some combinations are just noise and appear in very few situations. Therefore, it is suggested to set the two proposed frequency parameters to low values to filter noise to some extent, while not eliminating too many patterns. Unlike previous work who aim to mine frequent patterns using only a frequency threshold, the proposed algorithm allow to set the frequency thresholds quite low because the significance measure is also used for search space pruning using the proposed $UVGR$ upper-bound.

In [15], a dynamic attributed graph is transactionized to get a sequential database. Then, the classical PrefixSpan sequential pattern mining algorithm is applied to find sequential patterns [10] in the graph. For the proposed problem, we argue that this approach is inappropriate. The reason is that a pattern as in [15] can be described using a single transaction item. However, in this study, for the neighborhood definition of the running example, a pattern involve at least two transaction items. And this is the case for most complicated neighborhood definitions. Thus, it is not trivial to modify sequential pattern mining algorithms for the proposed problem. Besides, a research goal was to propose a general algorithm that is flexible (can be used with various neighborhood definitions).

The complexity of the TSeqMiner algorithm is analyzed as follows. The algorithm first discovers frequent trend sets using frequent itemset mining. Frequent itemset mining algorithms generally have a time complexity that is linear to the number of possible patterns (trend sets). If there are w possible trend values in the original database, in the worst case $2^w - 1$ possible trend sets are considered, although in real-life that number is generally much smaller than $2^w - 1$ because not all trends co-occur. After the set M of frequent trend sets has

been identified, the three mappings $levelMapTrendset$, $trendsetMapSuperset$, $trendsetMapMin$ are generated, which require $O(M)$, $O(M^2)$, $O(M^2)$ time, respectively. Suppose that the maximal depth of the tree is $\overline{D}$. For each node in the tree up to M extensions are attempted. The main cost to evaluate an extension is the intersection of its neighboring space and supporting points of a trend set, whose average cost is assumed to be $\overline{I}$. Therefore, the cost for considering all extensions is $(M + M^2 + ... + M^{\overline{D}}) \cdot M \cdot \overline{I}$. The total time cost is $O(M^{\overline{D}+1} \cdot \overline{I})$. In the worst case, $\overline{D} = T_{max} - 1$.

5 Experimental results

To evaluate the proposed TSeqMiner algorithm, quantitative and qualitative experiments have been conducted on two real world datasets. Note that because the proposed problem is considerably different from previous pattern mining problems, no algorithm can be directly compared with the proposed algorithm. Thus, performance experiments have been done by studying how parameter settings influence the algorithm's performance, and the algorithm was compared with a basic version without optimizations to show that optimizations are useful. The **DBLP dataset** consists of references published from 1990 to 2010. It involves 2723 authors and 43 conferences/journals(more than 10 publications). The corresponding dynamic attributed graph has 9 timestamps([1990-1994][1992-1996]...[2006-2010]). Each vertex denotes an author and has an attribute vector indicating the number of publications for 43 conferences or journals. The **Domestic US Flight dataset** contains records of US airport traffic from 01/08/2005 to 25/09/2005. Three hurricanes, namely Irene(04/08-18/08), Katrina(23/08-31/08) and Ophelia(06/09-17/09), occurred during this time. Data are aggregated by weeks for a total of 8 timestamps. Each attributed graph has 280 vertices (airports) having 8 attributes (e.g. number of arrivals/departures, average arrival/departure delay). Edges denote flight connections.

The proposed algorithm was implemented in Java. Experiments were conducted on a computer with Ubuntu 16.04 (Intel(R) Xeon(R) CPU E3-1270 3.60GHz) an 64 GB of RAM. Source code can be downloaded from the SPMF [9] software at `http://philippe-fournier-viger.com/spmf/`.

Quantitative Evaluation. In the following, "DFS-OP" denotes TSeqMiner, "NO-PRUNING" denotes TSeqMiner without pruning, and "DFS" denotes TSeqMiner without medium-grained pruning. The algorithm takes as input three parameters, $minInitSup$, $minTailSup$ and $minSig$. $minInitSup$ determines the number of frequent trend sets, and thus also the size of the inner search space. It is observed in Fig. 4 (a)(b) that when the number of frequent trend sets increases, the execution time and the number of patterns grow slowly (note that values of the x-axis are not equidistant). It is also observed in Fig. 4 (a)(b) that pruning and optimizations improve the execution time. Fig. 4 (c)(d) indicate that both $minSig$ and $minTailSup$ have great impact on pruning. If any of them is set to a large value (e.g. $minTailSup = 90$ or $minSig = 15$ in Fig. 4(c)), the search stops quickly. Fig. 4(e)(f) reports results about scalability. Number

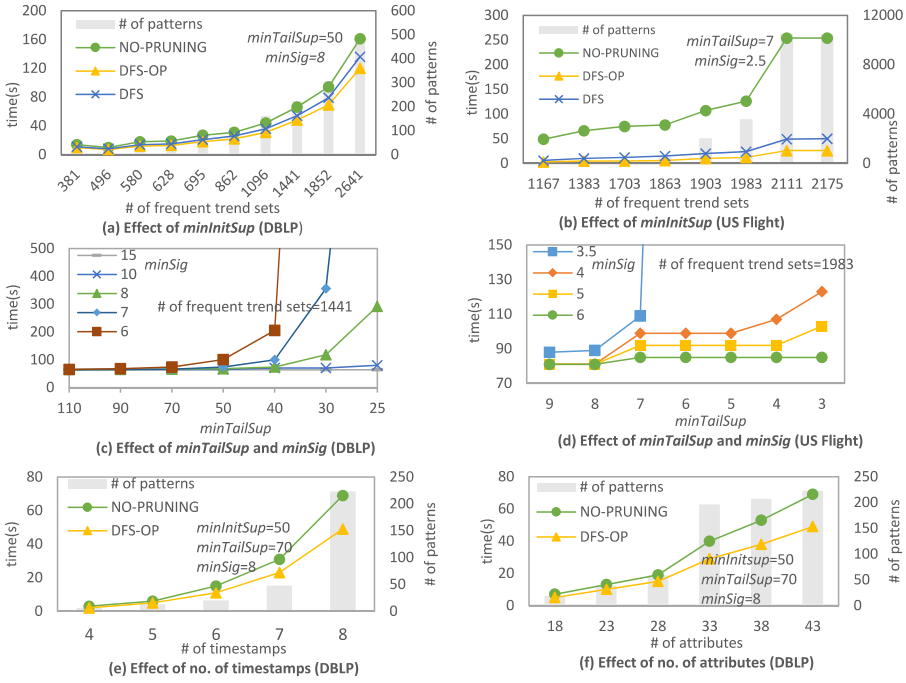

Fig. 4: Influence of parameters and scalability on performance

of found patterns and execution time grow relatively fast when the number of timestamps is increased, since it increases both the number of edges and vertices. However, the proposed algorithm have great scalability with respect to the number of attributes. Hence, TSeqMiner works well on graphs where entities have many attributes, which is one of the main challenges for mining dynamic attributed graphs.

Qualitative Evaluation. Consider the neighborhood definition of Section 3. For the DBLP dataset, let $minInitSup = 35$, $minTailSup = 100$, and $minSig = 10$. For these parameters, TSeqMiner generates 1610 frequent trend sets and 3 patterns (show in Table 1) in 45 s. The first pattern indicates that a co-author of someone who has published more papers in $ICDE$, $PVLDB$, and $ACMTransDBSys$ in recent years, is more likely to publish more papers in $PVLDB$ and less in $VLDB$. In this pattern, the first trend set is supported by 45 authors in the whole space and the second by 107 authors in the neighboring space of the first trend set. This pattern has a strong correlation as measured by the significance measure. The significance value is 11.5. This is a pattern that provides interesting insight on authorship behavior.

For the US Flight dataset, $minInitSup = 8$, $minTailSup = 7$ and $minSig = 6$, TSeqMiner$_{dfs}$returned 768 patterns in 9 s. A significant trend sequence is $\{\{NbCancel\text{--}, NbDivert\text{--}, DelayDepart\text{-}\}, \{NbDepart\text{-}, NbCancel\text{-}, NbDivert\text{-},$

Table 1: Three significant trend sequences mined from the DBLP dataset, with their number of supporting points and significance.

Patterns	supporting point count	signif.
$\{\{ICDE+, PVLDB+, ACMTransDBSys+\}, \{PVLDB+, VLDB-\}\}$	$\{45, 107\}$	$\{11.5\}$
$\{\{PVLDB+, VLDB+\}, \{PVLDB+, VLDB-\}\}$	$\{57, 120\}$	$\{11.5\}$
$\{\{JMLR=\}, \{JMLR-\}\}$	$\{283, 147\}$	$\{10.3\}$

DelayDepart-, *Delay Arriv+*}}. In the first trend set, *NbCancel--* and *NbDivert--* indicate a significant decrease of the number of cancelled and diverted flights after recovering from the damage caused by a hurricane. Most supporting points of that trend set are at timestamp 3 or 4 on the east coast where the most severe damage occurred. In the second trend set, *NbCancel-* and *NbDivert-* indicate moderate recovery from damage. The supporting points of the second trend are all at timestamp 5 and mostly not located on the east coast. That is, these airports experienced moderate problems caused by delays and diverted flights due to the hurricane on the east coast (more specifically at timestamp 4). Therefore, these airports are moderately influenced by the hurricane and recover moderately quickly after the recovery of east coast airports. *DelayArriv+* indicates an increase of the average delay time of arriving flights. It may indicate that rescheduled flights are not completely operating as usual. The above patterns capture an interesting aspect of spatio-temporal data. Thus, on overall, the proposed algorithm can be deemed as finding interesting patterns in real-life data. It can also be applied to other domains where data is modelled as graphs.

6 Conclusion

This paper proposed a novel type of pattern called significant trend sequence. It is a sequence of attribute variations, where correlations between elements are measured by a novel significance measure named *Sequence Virtual Growth Rate*. To efficiently mine these patterns, an algorithm named TSeqMiner is proposed, which relies on a novel upper bound and search space pruning strategy. Experiments on real-life datasets have shown that the proposed algorithm is efficient and can identify interesting patterns in social network and flight data.

For future work, we will extend the proposed model for mining sequences of significant subgraphs, and design methods to find other types of significant patterns [8, 11].

References

1. Aggarwal, C.C., Wang, H. (eds.): Managing and Mining Graph Data. Springer (2010)
2. Ahmed, R., Karypis, G.: Algorithms for mining the evolution of conserved relational states in dynamic networks. Knowl. Inf. Syst. **33**(3), 603–630 (2012)

3. Cheng, Z., Flouvat, F., Selmaoui-Folcher, N.: Mining recurrent patterns in a dynamic attributed graph. In: Proc. 21st Pacific-Asia Conf. on Knowledge Discovery and Data Mining. pp. 631–643. Springer (2017)
4. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.F.: Trend mining in dynamic attributed graphs. In: Proc. 24th European Conf. on Machine Learning and Knowledge Discovery in Databases. pp. 654–669. Springer (2013)
5. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.: Cohesive co-evolution patterns in dynamic attributed graphs. In: Proc. 15th Intern. Conf. of Discovery Science. pp. 110–124 (2012)
6. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. In: Proc. 5th ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining. pp. 43–52. ACM (1999)
7. Fassetti, F., Rombo, S.E., Serrao, C.: Discovering discriminative graph patterns from gene expression data. In: Proc. 31st Annual ACM Symposium on Applied Computing. pp. 23–30. ACM (2016)
8. Fournier-Viger, P., Li, X., Yao, J., Lin, J.C.W.: Interactive discovery of statistically significant itemsets. In: Proc. 31rd Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems. pp. 101–113. Springer (2018)
9. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Proc. 19th European Conf. Principles of Data Mining and Knowledge Discovery. pp. 36–40. Springer (2016)
10. Fournier-Viger, P., Lin, J.C.W., Kiran, U.R., Koh, Y.S.: A survey of sequential pattern mining. Data Science and Pattern Recognition **1**(1), 54–77 (2017)
11. Fournier-Viger, P., Zhang, Y., Lin, J.C.W., Fujita, H., Koh, Y.S.: Mining local and peak high utility itemsets. Information Sciences **481**, 344–367 (2019)
12. Holder, L.B., Cook, D.J., et al.: Learning patterns in the dynamics of biological networks. In: Proc. 15th ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining. pp. 977–986. ACM (2009)
13. Huang, Y., Zhang, L., Zhang, P.: A framework for mining sequential patterns from spatio-temporal event data sets. IEEE Transactions on Knowledge and Data Engineering (4), 433–448 (2007)
14. Jin, R., McCallen, S., Almaas, E.: Trend motif: A graph mining approach for analysis of dynamic complex networks. In: Proc. 7th IEEE Intern. Conf. on Data Mining. pp. 541–546. IEEE (2007)
15. Kaytoue, M., Pitarch, Y., Plantevit, M., Robardet, C.: Triggering patterns of topology changes in dynamic graphs. In: Proc. 6th IEEE/ACM Intern. Conf. on Advances in Social Networks Analysis and Mining. pp. 158–165. IEEE/ACM (2014)
16. Lv, T., Gao, H., Li, X., Yang, S., Hanzo, L.: Space-time hierarchical-graph based cooperative localization in wireless sensor networks. IEEE Transactions on Signal Processing **64**(2), 322–334 (2016)
17. Sanhes, J., Flouvat, F., Selmaoui-Folcher, N., Pasquier, C., Boulicaut, J.F.: Weighted path as a condensed pattern in a single attributed dag. In: Proc. of the 23rd International Joint Conference on Artificial Intelligence (2013)
18. Wen, Y.T., Fan, Y.Y., Peng, W.C.: Mining of location-based social networks for spatio-temporal social influence. In: Proc. 21st Pacific-Asia Conf. on Knowledge Discovery and Data Mining. pp. 799–810. Springer (2017)
19. Zaki, M.J.: Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering **12**(3), 372–390 (2000)