Granularity of Co-evolution Patterns in Dynamic Attributed Graphs

Élise Desmier[1], Marc Plantevit[2], Céline Robardet[1], and Jean-François Boulicaut[1]

[1] INSA Lyon, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, France
[2] Université Claude Bernard Lyon 1, LIRIS CNRS UMR 5205, F-69621 Villeurbanne

Abstract. Many applications see huge demands for discovering relevant patterns in dynamic attributed graphs, for instance in the context of social interaction analysis. It is often possible to associate a hierarchy on the attributes related to graph vertices to explicit prior knowledge. For example, considering the study of scientific collaboration networks, conference venues and journals can be grouped with respect to types or topics. We propose to extend a recent constraint-based mining method by exploiting such hierarchies on attributes. We define an algorithm that enumerates all multi-level co-evolution sub-graphs, i.e., induced sub-graphs that satisfy a topologic constraint and whose vertices follow the same evolution on a set of attributes during some timestamps. Experiments show that hierarchies make it possible to return more concise collections of patterns without information loss in a feasible time.

1 Introduction

Due to the success of social media and the ground-breaking discovery in experimental sciences, network data have become increasingly available in the last decade. Consequently, graph mining is recognized as being one of the most studied and challenging tasks for the data mining community. Two different and complementary ways have been considered so far: (1) analyzing graphs based on macroscopic properties (e.g., degree distribution, diameter) [9] or partitioning techniques [12], and (2) extracting more sophisticated properties within a pattern discovery setting. In particular, local pattern mining in graphs has received much attention, leading to the introduction of new problems (e.g., mining collections of graphs [17,22] or single graphs [5,7]). The graph vertices are generally depicted by additional information that form with the graph structure an attributed graph [15,16,18,20]. Such attributed graphs support advanced discovery processes providing insightful patterns.

However, there exists other types of augmented graphs such as evolving [1,4,19] or multidimensional graphs [2]. A growing body of literature has investigated augmented graphs by only considering one of the above types at a time. In [11], we proposed to tackle both dynamic and attributed graphs by introducing the problem of trend sub-graphs in dynamic attributed graph discovery. This new

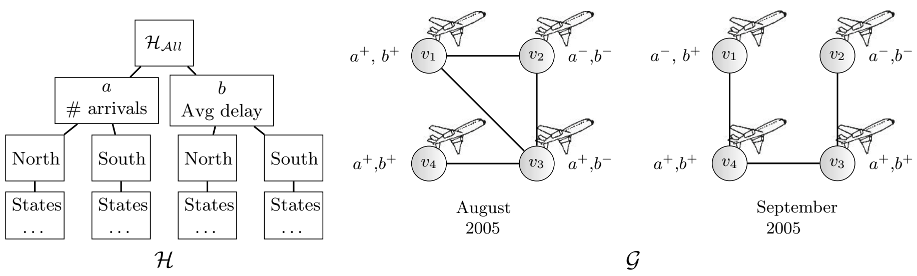

Fig. 1. US domestic flights dynamic graph

kind of patterns relies on the graph structure and on the temporal evolution of the vertex attribute values. In this paper, we go deeper in the analysis of dynamic attributed graphs by also examining the existence of a hierarchy over the vertex attributes. Indeed, we believe that the subsumption power of hierarchies is of most interest to summarize patterns and avoid unperceptive/useless/meaningless patterns. We propose to mine maximal dynamic attributed sub-graphs that satisfy some constraints on the graph topology and on the attribute values. To be more robust towards intrinsic inter-individual variability, we do not compare raw numerical values, but their trends, that is, their derivative at time stamp t. Let us consider the example in Fig. 1 that depicts a dynamic attributed graph describing the US domestic flights. The vertices stand for the airports and edges link airports that are connected by at least a flight during the time period of observation. Two attributes described the vertices of the graph: a is the number of flight arrivals and b is the average delay of arrival. At each time period of observation, we only consider the evolution or trend of the attribute values, and the value increases are denoted $+$, whereas their decreases is denoted $-$. The two attributes a and b can be specialized according to the geographical location of the airports where planes come from (see hierarchy $\mathcal{H}$ on Fig. 1). The incoming number of flights and their average delay can be decomposed into the ones coming from the North and South areas, as well as the distinct states of America. Hence, if a pattern describes a phenomenon that characterizes the whole airplane system, the most appropriate level of description is the first one (namely attributes a and b), whereas if the pattern is specific to a peculiar state, the involved attributes will be the ones at the leaves of the hierarchy.

The connectivity of the extracted dynamic sub-graphs is constrained by a maximum diameter value that limits the length of the longest shortest path between two vertices. Additional interestingness measures are used to assess the relevancy of the trend dynamic sub-graphs and guide their search with user-parametrized constraints. In this unified framework, these measures aim at evaluating (1) how the vertices outside the trend dynamic sub-graph are similar to the ones inside it; (2) the dynamic of the pattern through time; (3) the quality of the description of the pattern given the hierarchy over the vertex attributes. The algorithm designed to compute these patterns traverses the lattice of dynamic attributed sub-graphs in a depth-first manner. It prunes and propagates constraints that are fully or partially monotonic or anti-monotonic [8], and thus takes advantage of a large variety of constraints that are usually not exploited by standard lattice-based approaches. Our contributions are:

– The definition of hierarchical co-evolution sub-graphs: We define them as a suitable mathematical notion for the study of dynamic attributed graphs and introduce the purity and h-gain concepts (see Section 2).
– The design of an efficient algorithm `H-MINTAG` that exploits the constraints, even those that are neither monotonic nor anti-monotonic (see Section 3).
– A quantitative and qualitative empirical study. We report on the evaluation of the efficiency and the effectiveness of the algorithm on a real-world dynamic attributed graph (see Section 4).

2 Hierarchical Co-evolution Sub-graphs

A dynamic attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{T}, \mathcal{A})$ is a sequence over a time period $\mathcal{T}$ of attributed graphs $\{G_1, \ldots, G_{|\mathcal{T}|}\}$ where each attributed graph G_t is a triplet $(\mathcal{V}, E_t, A_t)$, with $\mathcal{V}$ a set of vertices that is fixed throughout the time, $E_t \subseteq \mathcal{V} \times \mathcal{V}$ a set of edges at timestamp t, and $\mathcal{A}$ a set of attributes common to all vertices at all times. $A_t(v) \in \mathbb{R}^{|\mathcal{A}|}$ are the values of vertex v at time t on $\mathcal{A}$.

A *vertex induced* dynamic sub-graph of $\mathcal{G}$ is an induced subgraph across a subsequence of $\mathcal{G}$, denoted by (V, T) with $V \subseteq \mathcal{V}$ and $T \subseteq \mathcal{T}$. In order to take into account both the fact that attributes can be expressed according to different levels of granularity and the end-user's prior knowledge, we assume that a hierarchy $\mathcal{H}$ is provided over the set of vertex attributes $\mathcal{A}$. A hierarchy $\mathcal{H}$ on $dom(\mathcal{H})$ is a tree whose edges are a relation *is_a*, a specialization (resp. generalization) relationship that corresponds to a path in the tree from the root node $\mathcal{H}_{\mathcal{A}ll}$ to the leaves, that are the attributes of $\mathcal{A}$ (resp. from the leaves to the root). Different functions are used to run through the hierarchy:

– $parent(x)$ returns the direct parent of the node $x \in dom(\mathcal{H})$
– $children(x)$ returns the direct children of the node $x \in dom(\mathcal{H})$
– $leaf(x)$ returns all the leaves down from $x \in dom(\mathcal{H})$

We aim at identifying relevant sub-graphs that rely on the graph structure, the temporal evolution of attributes and the associated hierarchy. To this end, we define a new kind of pattern, the so-called *hierarchical co-evolution sub-graphs.* Intuitively, a hierarchical co-evolution sub-graph $P = \{V, T, \Omega\}$ is such that $V \subseteq \mathcal{V}$, $T \subseteq \mathcal{T}$ and $\Omega \subseteq \{dom(\mathcal{H}) \times \{+, -\}\}$, a set of signed attributes. Such a dynamic sub-graph of $\mathcal{G}$ is induced by (V, T) and its vertices follow the same trends defined by Ω. Such dynamic sub-graphs, whose attribute values increase or decrease at the same timestamps, may be unconnected. Therefore, to support analysis based on the graph structure, we introduce a structural constraint that is based on the diameter of the induced dynamic subgraph and provides relevant patterns.

A hierarchical co-evolution sub-graph is then defined as follows:

Definition 1 (Hierarchical co-evolution Sub-Graph). *$P = (V, T, \Omega)$ is a sequence of graphs $G_t[V]$ induced*[1] *by the vertices of V in the graphs G_t, $t \in T$. The sets V, T and Ω are such that $V \subseteq \mathcal{V}$, $T \subseteq \mathcal{T}$ and $\Omega \subseteq \{dom(\mathcal{H}) \times \{+, -\}\}$. The pattern P has to satisfy the two following constraints:*

1. *Each signed attribute $(a, s) \in \Omega$ defines a trend that has to be satisfied by any vertex $v \in V$ at any timestamp $t \in T$. Thus, if (v, a, t) is the value of attribute a at time t for vertex v, $trend(v, a, t) = s$ with:*

$$trend(v,a,t) = + \text{ iff } \sum_{a_i \in leaf(a)} (v, a_i, t) < \sum_{a_i \in leaf(a)} (v, a_i, t+1)$$
$$trend(v,a,t) = - \text{ iff } \sum_{a_i \in leaf(a)} (v, a_i, t) > \sum_{a_i \in leaf(a)} (v, a_i, t+1)$$

   Thus, if $\forall v \in V$, $\forall t \in T$ and $\forall (a, s) \in \Omega$, we have $trend(v, a, t) = s$, then $coevolution(P)$ constraint is satisfied.
2. *Given Δ, a user-defined threshold, and $sp_G(v, w)$ the length of the shortest path between vertices v and w in graph G, the constraint $diameter(P)$ is satisfied iff $\forall t \in T$, $\max_{v,w \in V} sp_{G_t[V]}(v, w) \leq \Delta$.*

The maximum diameter constraint makes it possible to focus on some specific graph structure within the discovery of hierarchical co-evolution sub-graphs. Indeed, it allows to check how far the vertices are from each other. $\Delta = 1$ implies that the sub-graph is a clique, $\Delta = 2$ implies that the vertices of the sub-graph have at least one common neighbor. More generally, the higher the maximum diameter threshold Δ, the sparser the sub-graphs can be. Until $\Delta = |\mathcal{V}| - 1$, the sub-graphs have to be connected.

The attribute value of a parent node within the hierarchy is evaluated by adding the corresponding values of its children. Therefore, even if the trend conveyed by an attribute of the parent is true, it is important to check how this information is valid, i.e., if the trends associated to its children are similar. Indeed, if a children attribute has a large increase while the other children have a small decrease, the sum associated to the parent attribute may result in an increase that is not followed by most of its leaves. The purity measure evaluates the correlation between trends of the leaves of an attribute. Given the Kronecker function $\delta_{condition}$ and given a user-defined threshold $\psi \in [0, 1]$, the *purity* of a pattern returns the number of valid trends $trend(v, a, t) = s$ of the pattern compared to the total number of trends:

$$purity(V, T, \Omega) = \frac{\sum_{v \in V} \sum_{t \in T} \sum_{(a,s) \in \Omega} \sum_{\ell \in leaf(a)} \delta_{trend(v,\ell,t)=s}}{|V| \times |T| \times |leaf(\Omega)|}$$

[1] $G_t[V] = (V, E_t \cap \{V \times V\})$

	a	
	a_{north}	a_{south}
v_1	+	+
v_2	+	−
v_3	+	+
v_4	+	+

$P_1 = \{\{v_1, v_2, v_3, v_4\}, \{Aug.\ 2005\}, \{(a, +)\}\}$

	b	
	b_{north}	b_{south}
v_1	+	+
v_2	+	−
v_3	+	−
v_4	+	−

$P_2 = \{\{v_1, v_2, v_3, v_4\}, \{Aug.\ 2005\}, \{(b, +)\}\}$

Fig. 2. Illustration of the purity values of two patterns extracted from dynamic graph presented in Fig. 1

From this measure, we can derive the predicate $purityMin(P)$ which is true iff $purity(P) \geq \psi$. For example of Fig. 2, the purity of the pattern $P_1 = \{\{v_1, v_2, v_3, v_4\}, \{Aug.\ 2005\}, \{(a, +)\}\}$ is equal to $\frac{7}{8} = 0.875$ whereas the one of $P_2 = \{\{v_1, v_2, v_3, v_4\}, \{Aug.\ 2005\}, \{(b, +)\}\}$ is equal to $\frac{5}{8} = 0.625$.

One inconvenient of hierarchy is that it may introduce redundancy among the hierarchical co-evolution sub-graphs. An important issue is thus to avoid this redundancy by identifying the good level of granularity of a pattern. The question is thus to determine whether the pattern is worth to be specialized. Fig. 2 illustrates this problem with two patterns in two dimensions, i.e., lines depict vertices, columns are related to attributes. A cell is coloured if the trend of the attribute is +. Considering the pattern $P_3 = \{\{v_1, v_2, v_3, v_4\}, \{Aug.\ 2005\}, \{(a_{north}, +)\}\}$, its purity is equal to 1, and the one of pattern P_1 is of 0.875. There is no much interest in specializing pattern P_3 into P_1: end-users may prefer to consider the pattern P_3 as it is more synthetic while having a similar purity. On the other hand, the pattern $P_4 = \{\{v_1, v_2, v_3, v_4\}, \{Aug.\ 2005\}, \{(b_{north}, +)\}\}$ as a purity of 1 while its parent P_2 has a purity of 0.625. Then it seems much more interesting to keep the "parent" attribute instead of producing redundant pieces of information.

To this end, we introduce the *gain* of purity that evaluates whether the purity of the pattern would increase if it gets specialized or not. To this aim, we compare the purity of the a pattern P with respect to the purity of its "parent" patterns, that is, all the patterns made by generalizing one of the attributes of P. Given a user-threshold $\gamma \geq 1$, the gain of purity is defined as the purity of the "children" pattern compared to the purity of its "parent" patterns:

$$gainMin(P) \text{ iff } \frac{purity(P)}{max_{P_i \in parent(P)}(purity(P_i))} \geq \gamma$$

where $(V, T, \Omega_i) \in parent(V, T, \Omega)$ if $\exists (a_i, s_i) \in \Omega_i$ and $\exists (a, s) \in \Omega$ s.t. $a \in children(a_i)$ and $(\Omega_i \setminus a_i) = (\Omega \setminus children(a_i))$. From Fig. 2, the pattern P_4 has a gain equal to 1.6, whereas P_3 has a gain equal to 1.14.

Before ending this Section, let us formalize the general problem we want to solve as follows:

Problem 1 (Maximal hierarchical co-evolution sub-graph discovery). Let $\mathcal{G}$ be a dynamic attributed graph, $\mathcal{H}$ be a hierarchy over the set of vertex attributes $\mathcal{A}$, Δ be a maximum diameter threshold, and γ be a minimum gain threshold. Additional quality measures Q can be used, as defined in [11] (e.g., volume, vertex specificity, temporal dynamic). Given a conjunction of constraints $\mathcal{C}_Q$ over Q, the maximal hierarchical co-evolution sub-graph mining problem is to find the set of all the patterns that satisfy the constraints *coevolution*, *diameter*, *gainMin* and $\mathcal{C}_Q$.

3 H-MINTAG Algorithm

Algorithm 1 presents the main steps of H-MINTAG. The search space of the algorithm can be represented as a lattice which contains all possible tri-sets from $\mathcal{V} \times \mathcal{T} \times (dom(\mathcal{H}) \times \{+, -\})$, with bounds $\{\emptyset, \emptyset, \emptyset\}$ and $\{\mathcal{V}, \mathcal{T}, dom(\mathcal{H}) \times \{+, -\}\}$. The enumeration of all the patterns by materializing and traversing all possible tri-sets from the lattice is not feasible in practice. Therefore, in the algorithm, all possibly valid tri-sets are explored in a depth-first search manner which allows to extract the whole collection of hierarchical co-evolution sub-graphs and the constraints are used to reduce the search space by using their properties to not develop tri-sets that can not be valid patterns. The enumeration can be represented as a tree where each node is a step of the enumeration. A node contains two tri-sets P and C. P is the pattern in construction and C contains the elements not yet enumerated and that can potentially be added to the pattern. At the beginning, P is empty and C contains all the elements of $\mathcal{G}$, i.e., $P = \emptyset$ and $C = \{\mathcal{V}, \mathcal{T}, dom(\mathcal{H}) \times \{+, -\}\}$. The extracted patterns are the ones that respect the *diameter*, the *coevolution*, the *gainMin*, the *maximality* constraints, and the other possible constraints as defined in [11].

At each step of the enumeration, either an element of C is enumerated (vertex, timestamp or attribute) (lines 18-27) or an attribute of P is specialized (lines 5-10) and an attribute from C is enumerated (lines 11-15) while keeping the non specialized attribute. At the beginning of the algorithm, one vertex, one timestamp and one attribute are enumerated to allow a better use of the constraints to prune the search space. At each step, the elements of C (vertices, timestamps and attributes) are deleted if they can not be added to P without invalidating it, i.e., if they cannot respect the different constraints (line 1). If P does not respect the constraints, the enumeration is stopped.

The constraints *coevolution*, *diameter*, *purityMin* are not anti-monotonic considering the algorithm. They cannot be used directly to prune the search space. However, some piecewise monotonic properties of these constraints can be used to reduce the search space.

The *coevolution* constraint is not anti-monotonic considering the specialization of an attribute: If a vertex v does not respect the trend a^s at time t, no conclusions can be derived for the trends of any of its leaf attributes a_i. Indeed, the trend associated to a is computed while summing the values of the $a_i \in leaf(a)$, so some a_i can have an opposite trend. However, considering the enumeration

Algorithm 1. H-MINTAG

```
Require: $P = \varnothing$,$C = (\mathcal{V}, \mathcal{T}, children(\mathcal{H}_{All}))$,$attr$,$\mathcal{C}_{\mathcal{Q}}$
Ensure: Maximal hierarchical co-evolution sub-graphs
 1. Propagation(C)
 2. if $\neg empty(C)$ and $\mathcal{C}_{\mathcal{Q}}(P, C)$ then
 3.    if $attr \neq \varnothing$ then
 4.       $child \leftarrow children(attr)$
 5.       for i in $1..|child|$ do
 6.          if $gainMin(P.V \cup C.V, P.T \cup C.T, P.A \setminus attr \cup child[i])$ then
 7.             H-MINTAG($(P \setminus attr) \cup child[i]$, $C \cup child[i+1..|child|]$, $child[i]$)
 8.             $hasSon \leftarrow true$
 9.          end if
10.       end for
11.       if $hasSon$ then
12.          for i in $1..|C.A|$ do
13.             H-MINTAG($P \cup C.A[i]$, $C \setminus C.A[1..i]$, $i$)
14.          end for
15.       else
16.          $attr \leftarrow \varnothing$
17.       end if
18.    else
19.       $E \leftarrow ElementTypeToEnumerate(P,C)$
20.       for i in $1..|C.E|$ do
21.          if E = A then
22.             $attr \leftarrow C.E[i]$
23.          end if
24.          H-MINTAG($P \cup C.E[i]$, $C \setminus C.E[1..i]$, $attr$)
25.       end for
26.       H-MINTAG($P$, $C \setminus C.E$,$\varnothing$)
27.    end if
28. else if $\mathcal{C}_{\mathcal{Q}}(P)$ output $(P)$
29. end if
```

of the proposed algorithm, the *coevolution* can be pruned if the next step is not a specialization step. Indeed, if the attributes of the pattern are leaves of $\mathcal{H}$ or if the attributes have already passed the specialization step, the constraint is anti-monotonic. Then enumeration can be stopped if $coevolution(P)$ is false and elements e of C can be deleted if $coevolution(P \cup e)$ is false.

The *diameter* constraint is neither monotonic nor anti-monotonic. The addition of a vertex to a set of vertices can increase or decrease the diameter of the induced subgraph. Then, it is not possible to check strictly the diameter on P and C, however one can check if the induced graph can respect the *diameter* constraint while adding all or part of the vertices of C. Thus, during the algorithm, the following relaxed constraint $lightDiameter(P, C)$ is used:

$$\forall t \in T, \max_{v,w \in P.V} sp_{G_t[P.V \cup C.V]}(v, w) \leq \Delta$$

Otherwise, no valid pattern can be enumerated. Moreover only elements of C that can be added while respecting the diameter constraint are kept, i.e., $C.V = \{v \in C.V | \forall t \in T, \max_{w \in P.V} sp_{G_t[P.V \cup C.V]}(v, w) \leq \Delta\}$ and $C.T = \{t \in C.T | \max_{v,w \in P.V} sp_{G_t[P.V \cup C.V]}(v, w) \leq \Delta\}$.

The *purityMin* constraint is not anti-monotonic. Indeed, while specializing an attribute, the number of trends $trend(v, a, t) = s$, $v \in (P.V \cup Q.V), t \in (P.T \cup Q.T), (a, s) \in (P.\Omega \cup Q.\Omega)$ can increase or decrease if the leaf attributes do not follow the same trend as their parent. One must compute the number of trends that validate either s or $\overline{s}$ for at least one of the leaf attribute of a. Then the number of valid trends $\sum_{v \in P.V \cup C.V} \sum_{t \in P.T \cup C.T} \sum_{\ell \in leaf(P.\Omega \cup C.\Omega)} \delta_{trend(v,a,t)=s} + \delta_{trend(v,a,t) \neq s}$ is anti-monotonic and the number of possible trends $|P.V| \times |P.T| \times |leaf(P.\Omega)|$ is monotonic. The *lightPurity* relaxed constraint is anti-monotonic:

$$lightPurity(P, C) = \frac{\sum_{v \in P \cup C.V} \sum_{t \in P \cup C.T} \sum_{(a,s) \in leaf(P \cup C.\Omega)} \delta_{trend(v,a,t)=s} + \delta_{trend(v,a,t) \neq s}}{|P.V| \times |P.T| \times |leaf(P.\Omega)|}$$

If $lightPurity(P, C) < \psi$ is false, then the enumeration can safely be stopped.

4 Experiments

We carried out some experiments on a dynamic attributed graph built from the DBLP Computer Science Bibliography[2]. Vertices of the graph represent 2,145 authors who published at least 10 papers in a selection of 43 conferences and journals of the Data Mining and Database communities between January 1990 and December 2012. This time period is divided into 10 overlapping periods. A hierarchy over the 43 attributes is built considering the type of publications (e.g., journal, conference), the related area (e.g., database, machine learning,

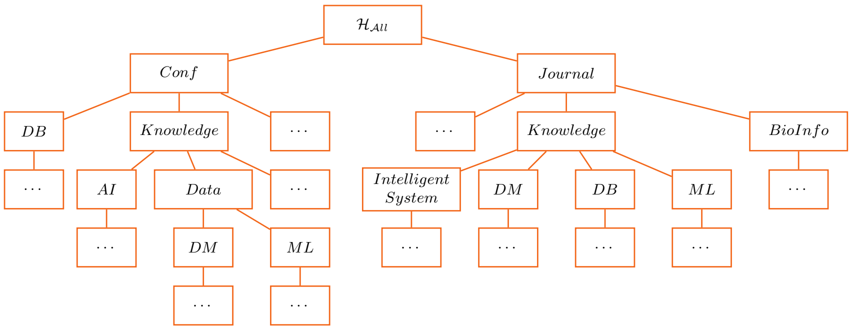

Fig. 3. Hierarchy of DBLP dataset

[2] http://dblp.uni-trier.de/db/

data mining, bioinformatics). This hierarchy contains 59 nodes and has a depth equal to 5, it is partly represented in Fig. 3. The default setting is $\Delta = 1, \Gamma = 1.1, \Psi = 0.2$ and two maximum threshold on the vertex specificity (κ) and the temporal dynamic (τ) set to 0.5.

Quantitative experiments. The impact of the hierarchy can be analyzed with respect to 3 parameters: the purity and the h-gain and the depth of the hierarchy. Fig. 4 reports the execution time of `H-MINTAG` and the number of patterns according to these parameters. The purity constraint has a significant and similar positive impact on both the execution time and the number of patterns. Increasing the h-gain enables to discard many patterns while the running time is marginally impacted. To study the impact of the depth of the hierarchy, we modified the hierarchy by deleting levels of abstractions. Hierarchy with depth equal to 0 is the dataset with no-hierarchy (i.e., only the 43 attributes). In our approach, the deeper the hierarchy, the lower the number of patterns. The execution time also decreases when the hierarchy becomes deeper.

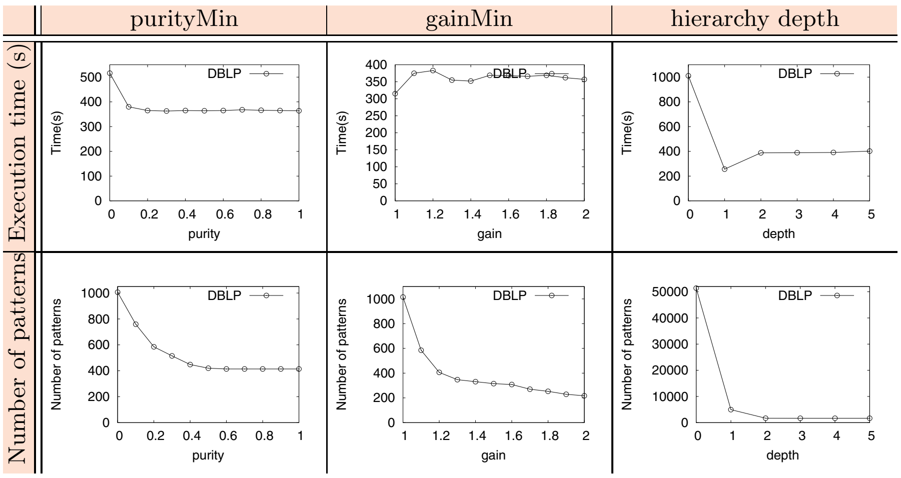

Fig. 4. Execution time and number of patterns with respect to ψ, γ and of the depth of the hierarchy

Qualitative experiments. We then look for connected hierarchical co-evolution sub-graphs (i.e., $\Delta = 2144$), with $\gamma = 1.1$ and $\psi = 0.35$. We also set some additional interestingness measures thresholds (a minimum volume threshold $\vartheta = 20$, a maximum vertex specificity threshold $\kappa = 0.2$ and a maximal temporal dynamic threshold $\tau = 0.4$). As this dataset has many attribute values equal to 0, it is not relevant to set the purity threshold too high. Considering the hierarchy, attributes too generalized as "conference" or "journal" are not really interesting, then γ was set to 1.1 to obtain patterns not too generalized. Two patterns were obtained in

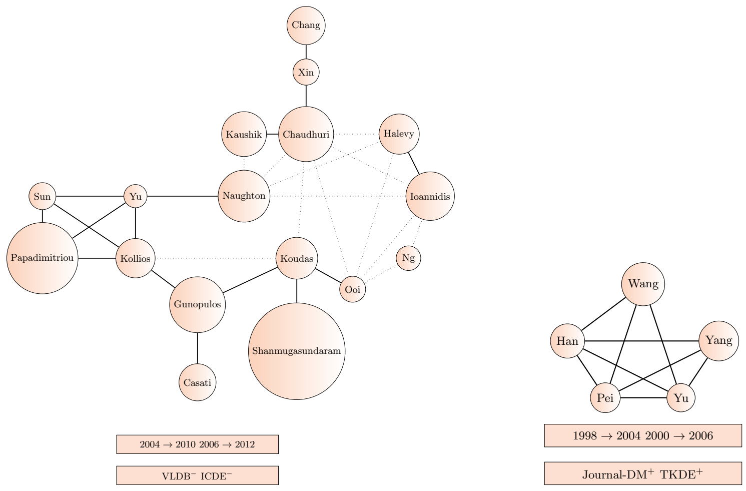

Fig. 5. First (on the left) and second (on the right) patterns extracted from DBLP with the parameters: $\vartheta = 20$, $\Delta = -1$, $\gamma = 1.1$, $\psi = 0.35$ $\kappa = 0.2$ and $\tau = 0.4$

this extraction. The first pattern is presented in Fig. 5 (left). This pattern concerns 17 authors who decreased their number of publications in VLDB and ICDE between 2004 and 2012. This pattern is relatively sparse, as the edges are dotted when they exist only at one of the two timestamps, for instance "Raymond T. Ng" is connected to "Beng Chin Ooi" at the first timestamp and to "Yannis E. Ioannidis" at the second timestamp, but he is connected to none author at both timestamps. It represents small groups of authors who work together occasionally. Moreover, if the decreasing of publication in VLDB seems logical considering the new publication policy of the "VLDB endowment" it is noteworthy that it is also true for the ICDE conference. This pattern has small outside densities with $VertexSpecificity = 0.126$ and $TemporalDynamic = 0.118$. Since the decreasing in "VLDB" concerns many authors at this timestamp (not only those involved in this pattern), we can conclude that the vertex specificity is mainly due to the decreasing in "ICDE". The low temporal dynamic specificity means that they do not decrease their number of publication in these conferences and that the pattern can show that this small community changed its publication policy.

The second pattern is illustrated in Fig. 5 (right). It involves 5 authors that increase their number of publication in the journal "IEEE-TKDE" and in the data-mining journals between 1998 and 2006. This pattern reflects that even if the journal "IEEE-TKDE" is considered as a database journal in the hierarchy, it has a high attractiveness in data mining. This pattern has a purity of 0.417, which means that they publish in a lot of data-mining journals; it seems to make sense since these authors are well-known in the data mining community. The vertex specificity is equal to 0.073 which depicts that this behavior is truly specific to these authors. And the temporal dynamic is equal to 0.4 which shows that their number of publications maybe oscillates. That points out that it is difficult to publish regularly in this type of journals.

5 Related Work

Recently, dynamic attributed graphs have received a particular interest. Boden et al. [3] mine sequences of attributed graphs. They propose to extract clusters in each attributed graph and associate time consecutive clusters that are similar. Jin et al. [14] consider dynamic graph with weights on the vertices. They extract groups of connected vertices whose vertex weights follow a similar increasing or decreasing evolution, on consecutive time stamps. Desmier et al. [10] discover subgraphs induced by vertices whose attributes follow the same trends. However, these propositions do not take into account additional user knowledge.

Hierarchies are not often used in the analysis of graphs. In [21], the authors propose subgraph querying in labelled graphs based on isomorphisms using an ontology on the labels. They use a similarity function such that the extracted subgraphs have labels similar to the query. Inokuchi [13] propose generalized frequent subgraphs in labelled graphs using a taxonomy on vertex and edge labels.The method is based on an isomorphic function and avoid the extraction of over-generalized patterns. The authors of [6] defines the taxonomy-superimposed graph mining problem. They compute frequency based on generalized isomorphism with a one-to-one mapping function. These propositions treat labelled graphs instead of attributed graphs and do not deal with dynamic aspect of the graphs.

6 Conclusion

We propose to extract hierarchical co-evolution sub-graphs from a dynamic attributed graph and a hierarchy. These patterns are sets of vertices that are connected and that follow the same trends over a set of attributes over time, with attributes that are either those of the dataset or of the hierarchy. We also define some constraints to reduce the execution time and increase the relevancy of the patterns, in particular according to hierarchy. We design an algorithm `H-MINTAG` to compute the complete set of patterns. Experiments on a real-world dataset prove that this method extracts, in a feasible time, interesting patterns based on the user parametrized constraints.

Acknowledgements. The authors thank ANR for supporting this work through the FOSTER project (ANR-2010-COSI-012-02).

References

1. Berlingerio, M., Bonchi, F., Bringmann, B., Gionis, A.: Mining graph evolution rules. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 115–130. Springer, Heidelberg (2009)
2. Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., Pedreschi, D.: As time goes by: Discovering eras in evolving social networks. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS, vol. 6118, pp. 81–90. Springer, Heidelberg (2010)

3. Boden, B., Günnemann, S., Seidl, T.: Tracing clusters in evolving graphs with node attributes. In: CIKM, pp. 2331–2334 (2012)
4. Borgwardt, K.M., Kriegel, H.P., Wackersreuther, P.: Pattern mining in frequent dynamic subgraphs. In: Int. Conf. on Data Mining (ICDM), pp. 818–822 (2006)
5. Bringmann, B., Nijssen, S.: What is frequent in a single graph? In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 858–863. Springer, Heidelberg (2008)
6. Cakmak, A., Özsoyoglu, G.: Taxonomy-superimposed graph mining. In: EDBT, pp. 217–228 (2008)
7. Calders, T., Ramon, J., van Dyck, D.: Anti-monotonic overlap-graph support measures. In: ICDM, pp. 73–82 (2008)
8. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.-F.: Closed patterns meet n-ary relations. TKDD 3(1), 3:1–3:36 (2009)
9. Chakrabarti, D., Faloutsos, C.: Graph mining: Laws, generators, and algorithms. ACM Comput. Survey 38(1) (2006)
10. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.-F.: Cohesive co-evolution patterns in dynamic attributed graphs. In: Discovery Science, pp. 110–124 (2012)
11. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.-F.: Trend mining in dynamic attributed graphs. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part I. LNCS, vol. 8188, pp. 654–669. Springer, Heidelberg (2013)
12. Ester, M., Ge, R., Gao, B.J., Hu, Z., Ben-moshe, B.: Joint cluster analysis of attribute data and relationship data. In: SIAM SDM, pp. 246–257 (2006)
13. Inokuchi, A.: Mining generalized substructures from a set of labeled graphs. In: ICDM, pp. 415–418 (2004)
14. Jin, R., McCallen, S., Almaas, E.: Trend Motif: A Graph Mining Approach for Analysis of Dynamic Complex Networks. In: ICDM, pp. 541–546. IEEE (2007)
15. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: SDM, pp. 593–604 (2009)
16. Mougel, P.N., Rigotti, C., Plantevit, M., Gandrillon, O.: Finding maximal homogeneous clique sets. Knowl. Inf. Syst. 39(3), 579–608 (2014)
17. Nijssen, S., Kok, J.N.: Frequent graph mining and its application to molecular databases. In: Systems, Man and Cybernetics (SMC), vol. 5, pp. 4571–4577 (2004)
18. Prado, A., Plantevit, M., Robardet, C., Boulicaut, J.F.: Mining graph topological patterns. IEEE TKDE, 1–14 (2013)
19. Robardet, C.: Constraint-based pattern mining in dynamic graphs. In: ICDM, pp. 950–955 (2009)
20. Silva, A., Meira Jr., W., Zaki, M.J.: Mining attribute-structure correlated patterns in large attributed graphs. PVLDB 5(5), 466–477 (2012)
21. Wu, Y., Yang, S., Yan, X.: Ontology-based subgraph querying. In: ICDE, pp. 697–708 (2013)
22. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: ICDM, pp. 721–724 (2002)