Mining Graph Evolution Rules

Michele Berlingerio[1], Francesco Bonchi[2],
Björn Bringmann[3], and Aristides Gionis[2]

[1] ISTI - CNR, Pisa, Italy
[2] Yahoo! Research, Barcelona, Spain
[3] Katholieke Universiteit Leuven, Belgium

Abstract. In this paper we introduce *graph-evolution rules*, a novel type of frequency-based pattern that describe the evolution of large networks over time, at a local level. Given a sequence of snapshots of an evolving graph, we aim at discovering rules describing the local changes occurring in it. Adopting a definition of support based on *minimum image* we study the problem of extracting patterns whose frequency is larger than a minimum support threshold. Then, similar to the classical association rules framework, we derive graph-evolution rules from frequent patterns that satisfy a given minimum confidence constraint. We discuss merits and limits of alternative definitions of support and confidence, justifying the chosen framework. To evaluate our approach we devise *GERM* (Graph Evolution Rule Miner), an algorithm to mine all graph-evolution rules whose support and confidence are greater than given thresholds. The algorithm is applied to analyze four large real-world networks (i.e., two social networks, and two co-authorship networks from bibliographic data), using different time granularities. Our extensive experimentation confirms the feasibility and utility of the presented approach. It further shows that different kinds of networks exhibit different evolution rules, suggesting the usage of these local patterns to globally discriminate different kind of networks.

1 Introduction

With the increasing availability of large social-network data, the study of the temporal evolution of graphs is receiving a growing attention. While most research so far has been devoted to analyze the change of global properties of evolving networks, such as the diameter or the clustering coefficient, not much work has been done to study graph evolution at a microscopic level. In this paper, we consider the problem of searching for patterns that indicate local, structural changes in dynamic graphs. Mining for such local patterns is a computationally challenging task that can provide further insight into the increasing amount of evolving-network data.

Following a frequent pattern-mining approach, we introduce the problem of extracting *Graph Evolution Rules* (*GER*), which are rules that satisfy given constraints of minimum support and confidence in evolving graphs. An example of a real *GER* extracted form the DBLP co-authorship network is given in Fig. 1: nodes are authors, with an edge between two nodes if they co-authored a paper.

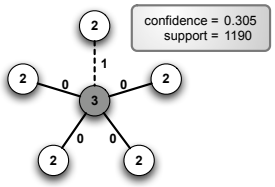


Fig. 1. A Graph Evolution Rule extracted from the DBLP co-authorship network.

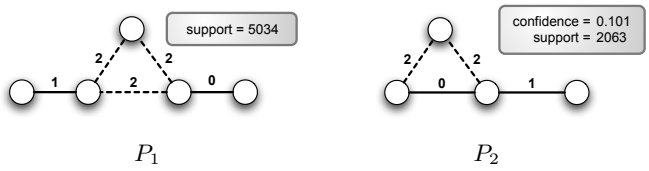
In this specific example, the node labels represent a class of degree of the node: the higher the label the higher the degree of the node. It is important to note that the label refers to the degree of the node in the input graph, not in the rule. In particular the label 3 indicates a node with degree > 50. In general, node labels may represent any property of a node. The labels on the edges instead are more important as they represent the (relative) year in which the first collaboration between two authors was established. Intuitively (later we will provide all the needed definitions) the rule might be read as a sort of local evidence of *preferential attachment*, as it shows a researcher with a large degree (label 3) that at time t is connected to four medium degree researchers (labels 2), and that at time $t + 1$ will be connected to another medium degree researcher. The definition, extraction and subsequent empirical analysis of such *Graph Evolution Rules* (*GER*) constitute the main body of our work.

The remainder of the paper is organized as follows: Section 2 describes the problem under investigation and defines the novel kind of pattern we are interested in. In Section 3 we describe the details of our algorithm. We report on our experimental results in Section 4 and present related work in Section 5. Finally, in Section 6 we discuss possible future research directions and in Section 7 we provide our conclusions.

2 Patterns of graph evolution

2.1 Time-evolving graphs

We start by describing how we *conceptually* represent an evolving graph, and subsequently discuss how to *actually* represent the graph in a more compact format. As usual the terminology $G = (V, E, \lambda)$ is used to denote a graph G over a set of nodes V and edges $E \subseteq V \times V$, with a labeling function $\lambda : V \cup E \rightarrow \Sigma$, assigning to nodes and edges labels from an alphabet Σ. These labels represent properties, and for simplicity we assume that they do not change with time. As an example, in a social network where nodes model its members, node properties may be *gender*, *country*, *college*, etc., while an edge property can be the kind of connection between two users. The evolution of the graph over time is conceptually represented by a series of undirected graphs $G_1, \ldots, G_T$, so that $G_t = (V_t, E_t)$ represents the graph at time t. Since $G_1, \ldots, G_T$ represent different snapshots of the same graph, we have $V_t \subseteq V$ and $E_t \subseteq E$. For simplicity of presentation, we assume that as the graph evolves, nodes and edges are only added and never deleted: i.e., $V_1 \subseteq V_2 \subseteq \ldots V_T$ and $E_1 \subseteq E_2 \subseteq \ldots E_T$.



P_1 P_2

Fig. 2. Relative time patterns extracted from two different samples of the DBLP co-authorship network: respectively 1992-2002 for (P_1), and 2005-2007 (P_2). Dataset details are given in Sec. 4.1.

It is worth noting that the number of edge deletions in social networks is so small to be negligible when analyzing the temporal evolution of networks. However, in our framework we can handle also deletions by slightly changing the matching operator as described in Section 6.

Our mining algorithm represents the dataset by simply collapsing all the snapshots $G_1, \ldots, G_T$ in one undirected graph G, in which edges are *time-stamped* with their first appearance. Thus, we have $G = (V, E)$ with $V = \bigcup_{t=1}^{T} V_t = V_T$ and $E = \bigcup_{t=1}^{T} E_t = E_T$. To each edge $e = (u, v)$ a time-stamp $t(e) = \arg\min_j \{E_j \mid e \in E_j\}$ is assigned. Note that time-stamps on the nodes may be ignored as a node always comes with its first edge and hence this information is implicitly kept in edge time-stamps. Overall, a time-evolving graph is described as $G = (V, E, t, \lambda)$, with t assigning time-stamps to the set of edges E.

2.2 Patterns

Consider a time-evolving graph G, as defined above. Intuitively a pattern P of G is a subgraph of G that in addition to matching edges of G also matches their time-stamps, and if present, the properties on the nodes and edges of G.

Definition 1 (Absolute-time pattern).
Let $G = (V, E, t, \lambda)$ and $P = (V_P, E_P, t_P, \lambda_P)$ be graphs, where G is the time-evolving dataset and P a pattern. We assume that P is connected. An occurrence *of P in G is a function $\varphi : V_P \to V$ mapping the nodes of P to the nodes of G such that for all $u, v \in V_P$:*
(i) $(u, v) \in E_P$ *it is* $(\varphi(u), \varphi(v)) \in E$,
(ii) $(u, v) \in E_P$ *it is* $t(\varphi(u), \varphi(v)) = t(u, v)$, *and*
(iii) $\lambda_P(v) = \lambda(\varphi(v)) \wedge \lambda_P((u, v)) = \lambda((\varphi(u), \varphi(v)))$

In case no labels are present for edges or nodes, the last condition (*iii*) is ignored. Two examples of patterns from the DBLP co-authorship network are shown in Fig. 2. Those examples motivate us to make two important decisions. First, since our goal in this paper is to study patterns of evolution we naturally focus on patterns that refer to more than one snapshots such as the examples in Fig. 2. In other terms we are not interested in patterns where all edges have the same time-stamp. The second decision is based on the following observation. Consider pattern P_1: arguably, the essence of the pattern is the fact that two distinct pairs of connected authors, one collaboration created at time 0, and one at time 1, are later (at time 2) connected by a collaboration involving one author from each pair, plus a third author. We would like to account for an occurrence

of that event even if it was taking place at times, say, 16, 17 and 18. To capture this intuition we define *relative-time patterns*.

Definition 2 (Relative-time Pattern). *Let G and P be a graph and pattern as in Definition 1. We say that P* occurs in G at relative time *if there exists a $\Delta \in \mathbb{R}$ and a function $\varphi : V_P \to V$ mapping the nodes of P to the nodes in G such that $\forall u, v \in V_P$*
(i) $(u, v) \in E_P$ *it is* $(\varphi(u), \varphi(v)) \in E$,
(ii) $(u, v) \in E_P$ *it is* $t(\varphi(u), \varphi(v)) = t(u, v) + \Delta$, *and*
(iii) $\lambda_P(v) = \lambda(\varphi(v)) \wedge \lambda_P((u, v)) = \lambda((\varphi(u), \varphi(v)))$

The difference between Definitions 1 and 2 is only in the second condition. As a result of Definition 2, we obtain naturally forming equivalence classes of structurally isomorphic relative time patterns that differ only by a constant on their edge time-stamps. To avoid the resulting redundancies in the search space of all relative time patterns we only pick one representative pattern for each equivalence class, namely *the one where the lowest time-stamp is zero.*

In the remainder of this paper we focus on relative time patterns, as they subsume the absolute time case: they are both more interesting and more challenging to mine.

2.3 Support

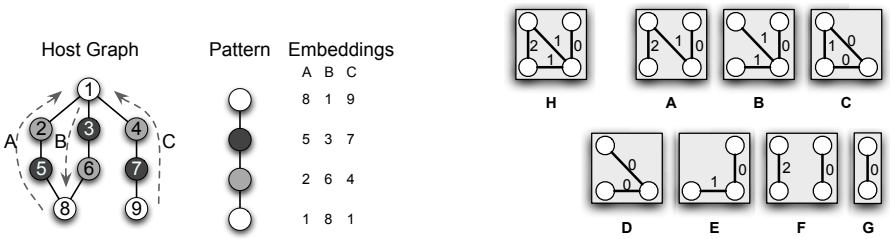
Next we discuss the support measure we use. Let $\mathcal{G}_\Sigma$ be the set of all graphs over an alphabet Σ. We define support as a function $\sigma : \mathcal{G}_\Sigma \times \mathcal{G}_\Sigma \to \mathbb{N}$. Given a host-graph G and a pattern P, the value of $\sigma(P, G)$ reflects the support of the pattern in the host-graph.

Defining a concept of support for the single graph setting is a non-trivial task, which has received attention recently [14, 8, 4, 5]. The most important property that a definition of support must satisfy is anti-monotonicity, that is, for all graphs G, P and P', where P is a subgraph of P', it must hold that $\sigma(P, G) \geq \sigma(P', G)$. This property is exploited by pattern miners to prune the search space. Anti-monotonicity holds trivially in the transactional setting, but is more tricky for the single-graph setting. For instance, while the total number of occurrences of a pattern is intuitively a meaningful measure, it is not anti-monotonic. As an example consider Fig. 4(b): the number of occurrences in the host graph Y of the pattern indicated as "body" is 1, while the number of occurrences of its supergraph indicated as "head" is 2, thus violating anti-monotonicity.

A first feasible support measure was proposed in [14] followed by a refinement published in [8]. Both measures rely on solving a maximum independent set problem *MIS* which is NP-complete. We employ the *minimum image based support* measure recently introduced in [4] which does not require solving a *MIS*. This measure is based on the number of unique nodes in the graph $G = (V_G, E_G)$ that a node of the pattern $P = (V_P, E_P)$ is mapped to, and defined as follows:

Definition 3 (Support).

$$\sigma(P, G) = \min_{v \in V_P} |\{\varphi_i(v) : \varphi_i \textit{ is an occurrence of } P \textit{ in } G\}|$$



(a) (b)

Fig. 3. (a): a graph with three different occurrences of a pattern evaluates to $\sigma = 2$. (b): a graph H with relative edge labels and all possible relative subgraphs A, B, C, D, E, F, G.

By taking the node in P as reference which is mapped to the least number of unique nodes in G, the anti-monotonicity of the measure is ensured. An example of minimum image based support is given in Fig. 3(a). Even if the pattern has 3 occurrences in the host graph, it has support $\sigma = 2$. In fact the lower white node of the pattern can only be mapped to nodes 1 and 8 in the host graph.

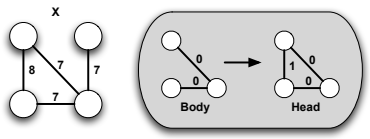
The advantage of this definition over other definitions introduced [14, 8] is twofold. From a practical point of view it is computationally easier to calculate since it does not require the computation of all possible occurrences of a pattern-graph in the host-graph. Additionally it does not require to solve a maximal independent set problem for each candidate pattern. From a theoretical perspective we know that this definition is an upper bound for the according overlap based definitions [4, 8]. Hence the support according to this definition is closer to the real number of occurrences in the graph.

2.4 Rules and Confidence Measure

The support of a pattern can provide insight into how often such an event may happen compared to other specific changes, but not how likely is a certain sequence of steps. To acquire this information we need to decompose a pattern into the particular steps and subsequently determine the *confidence* for each transition. Each step can be considered as a rule *body*→*head* with both *body* and *head* being patterns as defined in the previous section. Unfortunately, this does not yet solve our problem, but rather introduces two important questions:

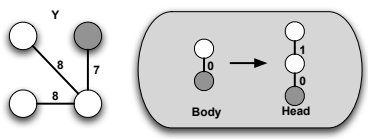
1. How to decompose a pattern into *head* and *body*?
2. What are reasonable definitions of confidence?

Regarding the decomposition consider pattern H in Fig. 3(b). An occurrence of H implies an occurrence of all its sub-patterns $A-G$. Similarly to the definition of association rules all $A - G$ can be considered candidate-*body* in order to form a graph evolution rule with pattern H as *head*. Fortunately, most of those possibilities can be discarded immediately. First, we are interested in evolution and hence only care about rules describing edges emerging in the future. This allows us to discard bodies A, C, D, E, and F thus only leaving B and G. Furthermore,



measure	head	body	confidence
minimum image support	1	1	1.00
number of occurrences	2	6	0.33

(a)



measure	head	body	confidence
minimum image support	1	1	1.00
number of occurrences	2	1	2.00

(b)

Fig. 4. Two example host-graphs X and Y illustrating different problems with support and confidence notions.

the step should be as small as possible to allow for a high granularity wherefore we would drop candidate-body G in the example, leaving B as body for the head H. Following the same reasoning, G would be the only choice as body for B as head. Similar the other rules in the example are $E \rightarrow A, D \rightarrow C, G \rightarrow E$. The natural body thus would be the head discarding all the edges from the last time-step of the target-pattern. More formally:

Definition 4 (Graph Evolution Rule). *Given a pattern head P_H the body P_B is defined as: $E_B = \{e \in E_H \mid t(e) < \max_{e^* \in E_H}(t(e^*))\}$ and $V_B = \{v \in V_H \mid deg(v, E_B) > 0\}$, where $deg(v, E_B)$ denotes the degree of v with respect to the edges in E_B. Moreover we constrain P_B to be* connected. *Finally, the support of a graph evolution rule is the support of its head.*

This definition yields a unique body for each head and therefore a unique confidence value for each head. This allows us to represent the rules by the head only. Note that the definition disallows disconnected graphs as body due to the lack of a support-definition for disconnected graphs. As a consequence *not all frequent patterns can be decomposed into graph evolution rules.*

Consider for instance pattern P_1 in Fig. 2: after removing all edges with the highest time-stamp, and discarding disconnected nodes, the graph that remains still contains two disconnected components (the one-edge component with label 1, and the one with label 0). Since the support is not defined for such disconnected pattern, P_1 can not be decomposed to be a *GER*. On the other hand, P_2 can be decomposed: in fact after removing all edges with the maximum time-stamp, and subsequently the disconnected node, we obtain a connected graph that will become the body of the rule for which P_2 is the head. Note that a *GER* can be represented in two different ways: either explicitly as two patterns (*body→head*), or implicitly by representing only the head as P_2 in Fig. 2. This is possible since there is a unique body for each head.

Finally, we have to choose a reasonable definition of *confidence* of a rule. Following the classic association rules framework, a first choice is to adopt the ratio of head and body supports as confidence. With the support being anti-monotonic this yields a confidence value which is guaranteed to be between zero and one. However, Fig. 4(a) shows that this definition may in some cases lack a reasonable semantic interpretation. In the upper host-graph X we find three possible ways to close a triangle given the edges from time-stamp 7. The confidence of 1 suggests that all of these will close to form triangles, while the graph shows

that only one actually does. To overcome this counterintuitive result, we investigated if the ratio of number of occurrences of head and body can be employed to solve this issue. While this definition of confidence allows for more reasonable semantics for the case in Fig. 4(a), it has the clear disadvantage that, due to the lack of anti-monotonicity, it may yield confidence values larger than 1, as in Fig. 4(b). In our experiments we compare the two alternative definitions showing that the minimum-image-based support is an effective and useful concept, while the occurrence-based definition has unpredictable behavior. Moreover, while the support is already available as it is computed for extracting the frequent patterns, the occurrence based confidence needs a separate and costly computation.

3 Mining graph evolution rules

GERM is an adaptation of the algorithm in [4], which was devised to prove the feasibility of the minimum image based support measure, and which in turn, was an adaptation of gSpan [22]. Thus, *GERM* inherits the main characteristics from those algorithms. In particular, *GERM* is based on a DFS traversal of the search space, which leads to very low memory requirements. Indeed, in all the experiments that we performed the memory consumption was negligible.

Algorithm 1 $SubgraphMining(\mathbb{GS}, \mathbb{S}, s)$

```
1: if s ≠ min(s) then return // using our canonical form
2: 𝕊 ← 𝕊 ∪ s
3: generate all s' potential children with one edge growth
4: Enumerate(s)
5: for all c, c is s' child do
6:    // using definition 3 based on definition 1 or definition 2
7:    if support(c)≥ minSupp then
8:       s ← c
9:       SubgraphMining(𝔾𝕊, 𝕊, s)
```

We next describe in detail how to adapt gSpan to *GERM* whereas the main changes are in the *SubgraphMining* method shown as Algorithm 1. The first key point is that we mine patterns in large single graphs, while gSpan was developed to extract patterns from sets of graphs. The part that is most involved in adapting gSpan is the *support computation* in line 7. Thus we start from the implementation of [4], where gSpan support calculation is replaced by the minimum image based support computation, without the need for changing the core of the algorithm.

One of the key elements in gSpan is the use of the *minimum DFS code*, which is a canonical form introduced to avoid multiple generations of the same pattern.

We need to change this canonical form in order to enable *GERM* to mine patterns with relative time-stamps (cf. line 1). As explained after Definition 2, we only want one representative pattern per equivalence class; namely the one with the lowest time-stamp being zero. This is achieved by modifying the canonical

Table 1. Dataset statistics: Number of nodes and edges and resulting average degree for the total graph as well as for the largest connected component (LCC) out of all connected components (CC). Further the growth rate in terms of edges: total growth as ratio between the graph size at the final and the initial time-stamps, and average growth rate per time-stamp.

							LCC			Growth Rates	
Dataset	**Date**	**|V|**	**|E|**	**avg deg**	**T**	**#CC**	**|V|**	**|E|**	**avg deg**	**total**	**avg**
flickr-month	03-05	147463	241391	1.64	24	16357	74792	182417	2.43	60347.8	2.83296
flickr-week	02-05	149863	246331	1.64	76	16661	76058	186504	2.45	246331	0.241055
y!360-month	04-05	177278	205412	1.16	10	17926	110627	155089	1.40	68470.7	5.15042
y!360-week	04-05	177278	205412	1.16	41	17926	110627	155089	1.40	68470.7	0.83434
arxiv92-01	92-01	70951	289226	4.08	10	6563	49008	260938	5.32	803.41	1.69114
dblp92-02	92-02	129073	277081	2.15	11	13444	83606	220098	2.63	25.52	0.408188
dblp03-05	03-05	109044	233961	2.15	3	14500	53370	153797	2.88	3.47	0.871401
dblp05-07	05-07	135116	290363	2.15	3	16333	72882	201468	2.76	3	0.749355

form such that the first edge in the canonical form is always the one with the lowest time-stamp, as compared to gSpan where the highest label is used as a starting node for the canonical form. Any pattern grown from such a pattern by extending the canonical form will have the same lowest time-stamp, which we set to zero by a simple constraint on the first edge. Hence we guarantee to extract only one pattern per equivalence class which dramatically increases performance and eliminates redundancy in the output.

Note that when matching a pattern to the host-graph we implicitly fix a value of Δ, representing the time gap between the pattern and the host graph. In order to complete the match all remaining edges must adhere to this value of Δ. If all the edges match with the Δ set when matching the first edge, the pattern is discovered to match the host-graph with that value of Δ.

Another important issue is to be able to deal with large real-world graphs, in which several nodes have high degree (the degree distribution in our datasets follows a power law). In typical applications of frequent-subgraph mining in the transactional setting, such as biology and chemistry, the graphs are typically of small size and they are not high-degree nodes. Dealing with large graphs and high degrees give rise to increased computational complexity of the search. In particular, having nodes with large degree increases the possible combinations that have to be evaluated for each subgraph-isomorphism test. We thus equip *GERM* with a user-defined constraint specifying the maximum number of edges in a pattern. This constraint allows to deal more efficiently with the DFS strategy by reducing the search space. Our experiments confirm that the total running time is much more influenced by the maximum-edge constraint than by the minimum support threshold.

4 Experimental Results

In this section, we report our experimental analysis. The *GERM* algorithm is implemented in C++. All the experiments are conducted on a Linux cluster equipped with 8 Intel Xeon processors at 1.8Ghz, 16Gb of RAM.

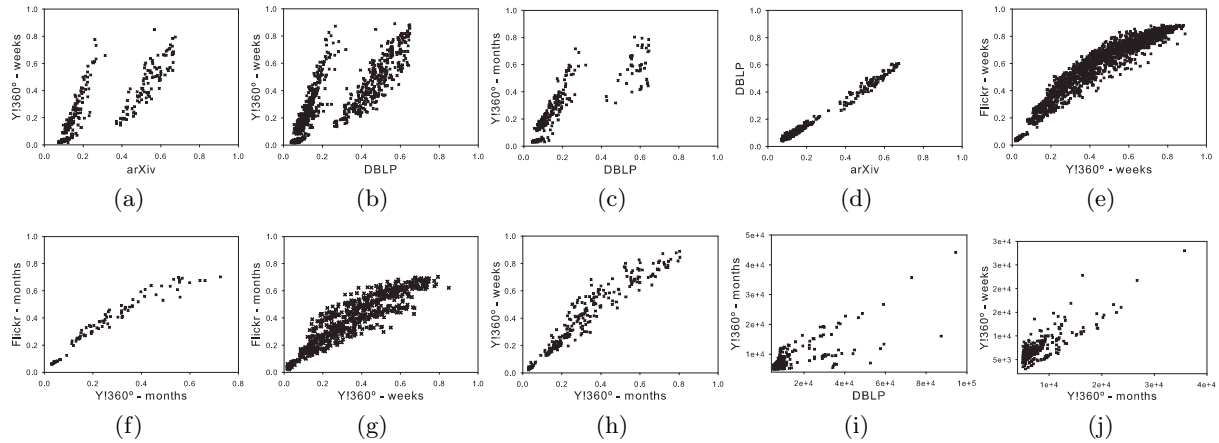


Fig. 5. (a)–(h): comparison of confidence of graph evolution rules in different networks. (i),(j): comparison of support of patterns in different networks.

4.1 Datasets

We conduct experiments on four real-world datasets: two social networks (Flickr and Y!360) and two bibliographic networks (DBLP and arXiv). Table 1 reports statistics of the resulting graphs.

Flickr (`http://www.flickr.com/`): Flickr is a popular photo-sharing portal. We sample a set of Flickr users with edges representing mutual friendship and edge time-stamp the moment when the bidirectional contact is established. We generate one graph with monthly and one with weekly granularity.

Y!360 (`http://360.yahoo.com/`): Yahoo! 360° is an online service for blogging. Again we sample a set of users and proceed as in the Fickr dataset. In this case the monthly and weekly datasets contain exactly the same time period.

DBLP (`http://www.informatik.uni-trier.de/~ley/db/`): This dataset is based on a recent snapshot of the DBLP which has a yearly time-granularity. Authors are represented by vertices with a connecting edge if they are co-authors. The assigned time-stamp on an edge represents the year of the first co-authorship. Three different samples are extracted each containing the edges created in the corresponding years. These three samples allow us to analyze and compare long and short term trends.

arXiv(`http://arxiv.org/`): Similar to the DBLP dataset a co-authorship graph from a sample of the arXiv repository considering only physics publications is extracted. The obtained graph *arxiv92-01* contains the co-authorships which emerged in the years 1992 to 2001 with a time granularity of years.

As discussed in Section 2, beside the time-stamp associated to each edge, our framework allows to have labels on both nodes and edges representing additional information. We experiment with node labels that are based on two graph-theoretic measures: the *degree* and the *closeness centrality*. These measures change as the graph evolves. To obtain static labels the measures are computed once on the whole graph, corresponding to the last time-stamp and then they are discretized in 5 bins.

4.2 Results

We analyze the experimental results with regard to the following questions:

Q1 Do the extracted patterns and rules characterize the studied network?
Q2 Do different time granularities influence the confidence of the rules?
Q3 How do the different confidence definitions compare?
Q4 How do the parameters and the type of dataset influence the number of derivable rules, the number of patterns obtained, and the run time?

Q1: Discriminative analysis. The first question is if the extracted patterns carry information that characterizes the analyzed network. In order to address this question graph evolution rules from the first six datasets in Table 1 were extracted with a minimum support threshold of 5000 for all but the "weeks" datasets where a minimum threshold of 3000 was used. Then we compared all pairs of datasets with respect to the rules confidences found in both datasets. We show the pair-wise comparison results in Fig. 5. The plots allow for several interesting observations. First, we see that the comparison between a co-authorship network (arXiv or DBLP) and a social network (Y!360) as in Fig. 5(a),(b) and (c) show different confidence values of the rules for each dataset (using Flickr instead of Y!360 gives the same results).

In contrast, the comparison of two co-authorship networks (arXiv and DBLP, in Fig. (d)) or two social networks (Flickr and Y!360, in figures (e) to (h)) reveals that all rules are in the proximity of the bisector, meaning that each rule has very similar confidence values in the both datasets. This observation confirms our claim: *graph evolution rules indeed characterize the different types of networks.*

Fig. 5 (i) and (j) compare the same two datasets as in (c) and (h) respectively. However, in (i) and (j) we plot the rules according to their support instead of their confidence. Contrary to (c) and (h), both plots (i) and (j) show similar results, indicating that the support of a rule can not be used to characterize different types of networks.

Q2: Different granularity analysis. Fig. 6(a) similarly to Fig. 5(h), focuses on the difference of confidence from rules originating in the same network but with different time granularity. We observe that confidences for the weekly granularity are larger than the corresponding monthly confidences. The colors/shapes in the plot correspond to the difference between maximum time-stamp on an edge in the head (MTH) and maximum time-stamp on an edge in the body (MTB) of the rule. This figure very clearly reveals the cause for the specific structure of the plot. First, the difference between the maximum time-stamp in head and body indicated by the shape perfectly models the confidence differences between monthly and weekly granularity: the rules form three clear clusters (with the corresponding regression lines reported in the plot).

The second observation is that a larger difference between the time-stamps corresponds to a higher difference in confidence towards weekly granularity. This is quite natural if we think about confidence trough the lenses of prediction: the difference between the time-stamps in head and body can be thought as the temporal gap that must be bridged by a prediction task, and clearly predicting

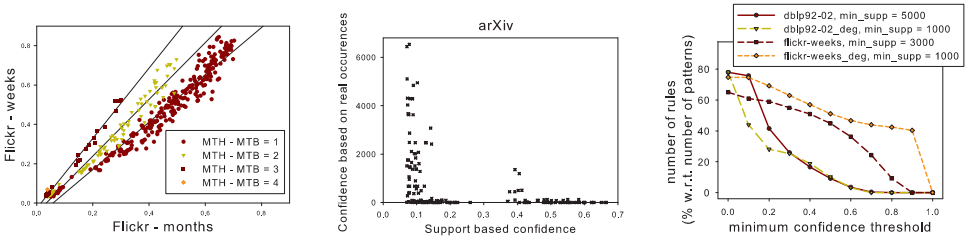
further in the future is more difficult (i.e., lower confidence). Hence clusters with higher time-difference have higher confidences in the weekly setting simply because three weeks are shorter than three months.

Finally, it is worth noting that only one rule in this plot has a difference of 4 between the maximum time-stamp in head and body: as expected it is in the left bottom corner, and is closer to the weekly axis than to the monthly axis.

Q3: Confidence and rules. Fig. 6(b) shows that the two confidence measures disagree. A more thorough investigation shows that all the rules with an occurrence-based confidence exceeding 200 have the most simple body: one single edge. Furthermore, all those rules span 3 or 4 time-steps from body to head. Given that they all share the same simplistic body, which can be matched anywhere, a correct prediction, especially 3 or 4 time-steps into the future is doomed to fail. The support-based confidence however, nicely assigns a confidence below 0.2 to all rules with the simplistic body, equivalent to declaring them almost meaningless, thus proving itself one more time fruitful being investigated and worthy being used.

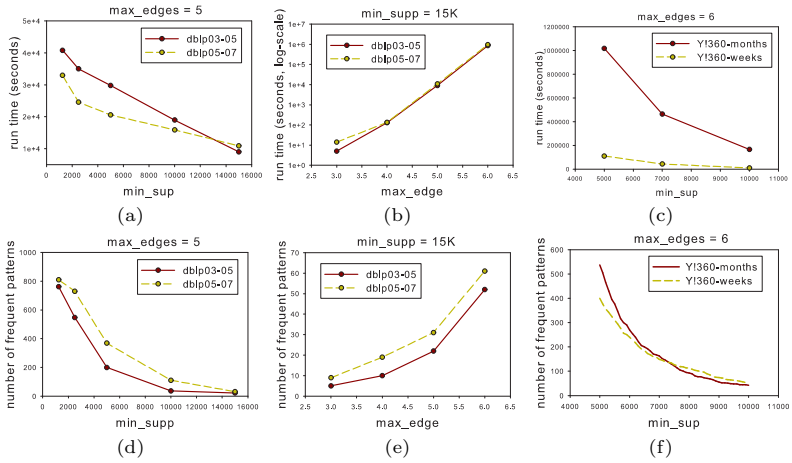
Q4: Influence of parameters and dataset. Further important insights can be gained from an analysis of the number of rules and patterns extracted as well as the run-times. To understand how many of the extracted relative time patterns are decomposable and thus can be interpreted as rules we calculated the ratio of valid rules over all extracted patterns. Fig. 6(c) reports the number of valid graph evolution rules as percentage of the number of frequent patterns found, for various minimum confidence thresholds. This is done on one bibliographic and one social network; each with and without node labels. In all cases, the number of rules is close to 80% of the number of frequent patterns. Besides the fact that a lower minimum confidence thresholds yields more rules, the results nicely reaffirm the observation from Fig. 6(a). Indeed, rules extracted from a dataset with weekly granularity enjoy a much higher confidence than rules extracted from a dataset with yearly granularity.

As intended, the size of the result of the mining task depends on the maximum edge and the minimum support constraint. With a higher number of edges exponentially more graphs are possible, thus the exponentially increased number of extracted patterns for larger number of edges in Fig. 7(e) comes at no



(a) (b) (c)

Fig. 6. (a): confidence comparison between monthly and weekly granularity. (b): scatter plot comparing the two different definitions of confidence discussed in Section 2.4. (c) number of valid rules as percentage of the number of frequent patterns, for varying confidence.



y!360 weeks

min_supp	*e*2	*e*3	*e*4	*e*5	*e*6
1000	16	241	1478	7253	34904
1100	16	215	1179	5337	23828
1200	16	196	963	4075	16777
1300	16	173	791	3119	12028
1400	16	155	665	2435	8814
1500	15	141	550	1933	6584
1600	15	130	487	1581	5011
1700	15	115	413	1272	3870
1800	15	106	358	1051	3004
1900	14	95	308	884	2405
2000	14	87	269	717	1903
2500	12	59	153	337	708
3000	11	43	99	180	315
3500	9	34	64	103	149
4000	8	26	43	61	76
5000	6	18	26	29	29
6000	5	12	15	16	16
7000	5	11	12	12	12
10000	3	5	5	5	5

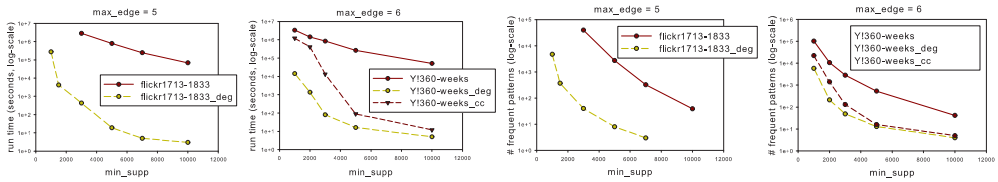
(g)

Fig. 7. (a)–(f): run time and number of patterns found with varying *min. support* and *max. edge* thresholds. (g): number of patterns of different size at various minimum support (*ei* denotes a pattern with $\leq i$ edges).

surprise. For lowering the minimal support Fig. 7(d) shows the typical result. Lowering the support threshold allows for more complex patterns which contain more edges and thus for the same reason as above the growth is exponential.

As Fig 7(a)-(b) show, the run time is affected much more by the maximum edge (max-edge) than the minimum support constraint (min-sup). While the increase is almost linear with decreasing minimum support, the run time grows exponentially with an increasing maximum edge size. Note the increase of two orders in magnitude in Fig. 7(b) from four to five and five to six edges.

A more interesting observation can be made from the Fig 7(c) and (f). The underlying graph structure is the same in both datasets with the only difference being the time-granularity of the edge time-stamps. The weekly graph with 41 edge labels is more *diverse* than the monthly graph with only 10. While the runtime between both datasets varies highly with changing minimum support, the number of patterns extracted is almost the same for each minimum support. With regard to the number of patterns a higher label-diversity allows for more different patterns (i.e., more possible combinations for a fixed number of edges) *if* the support is low enough for these to be considered frequent. However, a lower label-diversity means that patterns can be found more repeatedly since the host-graph is more homogeneous, but the amount of different patterns is limited. Thus for a fixed number of edges there are more patterns with a high support in the more homogeneous graph and more patterns with low support in the graph with a higher label diversity as confirmed in Fig. 7(f). Regarding the varying run times between the datasets, in the more diversified data more patterns of smaller size can be found. The subgraph-isomorphism for these patterns is easier to calculate simply because they are smaller. Furthermore, it is easier to find a non-matching edge in the more diversified graph earlier, thus being able to terminate a search-



(a) (b) (c) (d)

Fig. 8. Run time and number of patterns found on networks with labelled nodes with varying *min_sup*.

branch for the subgraph-isomorphism check earlier. These two reasons explain the much lower runtime on more diverse graphs.

A similar reasoning holds for graphs with labelled nodes (Fig. 8). Also in this case the diversity introduced by the node labels reduced the number of patterns found and the run time.

5 Related Work

Several papers have focussed on the global evolution of networks by an exploratory point of view. Leskovec et al. [16] discovered the shrinking diameter phenomena on time-evolving networks. Backstrom et al. [2] study the evolution of communities in social networks. Still from an exploratory perspective, Leskovec et al. [15] study the evolution of networks but at a more local level. Using a methodology based on the maximum-likelihood principle, they investigate a wide variety of network formation strategies, and show that edge locality plays a critical role in evolution of networks.

Other recent papers, present algorithmic tools for the analysis of evolving networks. Tantipathananandh et al. [20] focus on assessing the community affiliation of users and how this changes over time. Sun et al. [18], apply the MDL principle to the discovery of communities in dynamic networks, developing a parameter-free framework. This is the main difference to previous work such as [1, 19]. However, as in [20], the focus lies on identifying approximate clusters of users and their temporal change. No exact patterns are found, nor is time part of the results obtained with these approaches. Ferlez et al. [7] use the MDL principle for monitoring the evolution of a network.

While the aforementioned body of work studies the evolution in networks, it does not take a pattern mining approach as we do in this paper. Although, to the best of our knowledge, this is the first work on mining frequent subgraphs from evolving graphs that follows the classical graph mining paradigm in terms of considering exact structure matching we are aware of several studies focussing on mining evolving graphs, all using some form of approximation. Desikan and Srivastava [6] study the problem of mining temporally evolving web graphs. Three levels of interest are defined: single node, subgraphs and whole graph analysis, each of them requiring different techniques. Inokuchi and Washio [11] propose a fast method to mine frequent subsequences from graph sequence data defining a formalism to represent changes of subgraphs over time. However the

time in which the changes take place is not specified in the patterns. Liu et al. [17] identify subgraphs changing over time by means of vertex-importance scores and vertex-closeness changes in subsequent snapshots of the graphs. The most relevant subgraphs are hence not the most frequent, but the most significant based on the two defined measures. The paper that is most related to our work is [3]. Borgwardt et al. represent the history of an edge as a sequence of 0's and 1's representing the absence and presence of the edge respectively. Then conventional graph-mining techniques are applied to mine frequent patterns. However, there are several differences to our approach. First, the employed mining algorithm GREW is not complete, but heuristic. Further, the overlap-based support measure used requires solving an maximal independent set problem for which a greedy algorithm is used. Another computational issue with their approach is the extension of an edge in the so-called inter-asynchronous FDS case. Accordingly the size of the networks analyzed in the paper is rather small.

Various proposals for mining frequent patterns in the single graph context [14, 21, 8, 4] were discussed in Section 2. A recent paper by Calders et al. [5] introduces a new measure named *minimum clique partition*, which analogous to the maximal independent set is based on the notion of an overlap graph and thus requires solving an NP-complete problem. They prove that support measures based maximal independent set and minimum clique partition are the minimal and the maximal possible meaningful overlap measures, and show that [12] introduced a function which is sandwiched between these two measures; computable in polynomial time. However, any of those measures requires computing an overlap graph for each candidate pattern, which is a costly operation in itself due to requiring enumerating all occurrences of a pattern.

6 Extensions and future work

In this section we discuss briefly how to relax some of the assumptions of our problem definition.

Consider first the pattern H in Fig. 3(b). Imagine that in the data it is the case that when there is a star of size 3 an edge between two peripheral nodes appear. Pattern H captures partly this phenomenon, but is also too "specific" as it emphasizes that the star was formed in particular time instances before the appearance of the last edge. A more general pattern would be to replace the time-stamp of the last edge with T, and the time-stamp of all the edges in the star with the constraint "$< T$", which will have to be satisfied when tested with the time-stamps of the host graph. We plan to extend our algorithm to experiment with this idea as a continuation of our work.

For sake of presentation, in Section 2 we assumed that graphs can only grow in time. However, our approach can be easily extended to handle edge-deletions if an edge can appear and disappear at most once. The extension would consider two time-stamps t_I (time of insertion) and t_D (time of deletion) on each edge instead of the single time t. By modifying definitions 1 and 2 condition (ii) to $\forall (u, v) \in E_P$ it is $t_I(\varphi(u), \varphi(v)) = t_I(u, v) + \Delta$ and $t_D(\varphi(u), \varphi(v)) = t_D(u, v) + \Delta$.

We did no implement the above matching since two out of four datasets (arXiv and DBLP) are naturally only growing (thus, no deletions) and deletions are rare in the other two. As future work we plan to incorporate deletions and study networks with a higher likelihood of such events.

In our approach, we have not considered node or edge relabelling so far. Considering node and edge relabeling is very interesting, as in many graphs, such as social networks, the properties of nodes and edges change over time. For example, in social-network analysis it would be interesting to study the change of leadership in communities and its effects.

Besides all the above, which are possible extensions to the kind of patterns we are able to mine, we would like to go further by taking advantage of the just defined rules and confidence: such a paradigm enables us to put the basis for defining a framework that will allow us to *predict* graph evolution, and that, together with *GERM*, will provide helpful tools for social-network analysis and other fields of research where dynamic graphs are a good data representation.

7 Conclusions

Following a frequent pattern mining approach, we defined relative time patterns and introduced introduced the problem of extracting *Graph Evolution Rules*, satisfying given constraints of minimum support and confidence, from an evolving input graph. While providing the problem definition we discussed alternative definitions of support and confidence, their merits and limits. We implemented *GERM* an effective solution to mine Graph Evolution Rules, and we extensively test it on four large real-world networks (i.e., two social networks, and two co-authorship networks from bibliographic data), using different time granularities. Our experiments confirmed the feasibility and the utility of our framework and allowed for interesting insights. In particular we showed that Graph Evolution Rules with their associated concept of confidence, indeed characterize the different types of networks.

Availability. The executable code of the GERM software is freely available at: `http://www-kdd.isti.cnr.it/~berlingerio/so/gm/`.

References

1. C. C. Aggarwal and P. S. Yu. Online analysis of community evolution in data streams. *SIAM International Data Mining Conference (SDM)*, 2005.
2. Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. *Int. Con. on Knowledge Discovery and Data Mining (KDD)*, 2006.
3. Karsten M. Borgwardt, Hans-Peter Kriegel, and Peter Wackersreuther. Pattern mining in frequent dynamic subgraphs. *Int. Conf. on Data Mining (ICDM)*, 2006.
4. Björn Bringmann and Siegfried Nijssen. What is frequent in a single graph? *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2008.
5. T. Calders, J. Ramon, and D. Van Dyck. Anti-monotonic overlap-graph support measures. *International Conference on Data Mining (ICDM)*, 2008.

6. Prasanna Desikan and Jaideep Srivastava. Mining temporally changing web usage graphs. *WebKDD*, 2004
7. Jure Ferlez, Christos Faloutsos, Jure Leskovec, Dunja Mladenic, and Marko Grobelnik. Monitoring network evolution using MDL. *International Conference on Data Engineering (ICDE)*, 2008.
8. M. Fiedler and C. Borgelt. Subgraph support in a single graph. *Workshop on Mining Graphs and Complex Data (MGCS), 2007.*
9. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
10. Lawrence B. Holder, Diane J. Cook, and Surnjani Djoko. Substucture discovery in the SUBDUE system. *AAAI KDD Workshop*, 1994.
11. Akihiro Inokuchi and Takashi Washio. A fast method to mine frequent subsequences from graph sequence data. *International Conference on Data Mining (ICDM)*, 2008.
12. Donald E. Knuth. The sandwich theorem. *Electronic Journal of Combinatorics*, 1:1, 1994.
13. M. Kuramochi and G. Karypis. Frequent subgraph discovery. *Int. Conf. on Data Mining (ICDM)*, 2001.
14. Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery*, 11(3):243–271, 2005.
15. Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.
16. Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.
17. Zheng Liu, Jeffrey Xu Yu, Yiping Ke, Xuemin Lin, and Lei Chen. Spotting significant changing subgraphs in evolving graphs. *Int. Conf. on Data Mining (ICDM)*, 2008.
18. Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. *Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2007.
19. Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. *Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2006.
20. Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2007.
21. N. Vanetik, S. E. Shimony, and E. Gudes. Support measures for graph data. *Data Mining Knowledge Discovery*, 13(2):243–260, 2006.
22. Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. *International Conference on Data Mining (ICDM)*, 2002.
23. Feida Zhu, Xifeng Yan, Jiawei Han, and Philip S. Yu. gprune: A constraint pushing framework for graph pattern mining. *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2007.