Relational Mining for Discovering Changes in Evolving Networks

Corrado Loglisci[1,*], Michelangelo Ceci[1,*], Donato Malerba[1,*]

[a]*Dipartimento di Informatica, University of Bari "Aldo Moro", Via Orabona, 4, 70125, Bari, Italy*

Abstract

Networks are data structures more and more frequently used for modeling interactions in social and biological phenomena, as well as between various types of devices, tools and machines. They can be either static or dynamic, dependently on whether the modeled interactions are fixed or changeable over time. Static networks have been extensively investigated in data mining, while fewer studies have focused on dynamic networks and how to discover complex patterns in large, evolving networks. In this paper we focus on the task of discovering changes in evolving networks and we overcome some limits of existing methods i) by resorting to a relational approach for representing networks characterized by heterogeneous nodes and/or heterogeneous relationships, and ii) by proposing a novel algorithm for discovering changes in the structure of a dynamic network over time. Experimental results and comparisons with existing approaches on real-world datasets prove the effectiveness and efficiency of the proposed solution and provide some

*Corresponding authors

Email addresses: `corrado.loglisci@uniba.it` (Corrado Loglisci), `michelangelo.ceci@uniba.it` (Michelangelo Ceci), `donato.malerba@uniba.it` (Donato Malerba)

insights on the effect of some parameters in discovering and modeling the evolution of the whole network, or a subpart of it.
Keywords: Evolving networks. Discovery of evolution chains. Discovery of change patterns. Change mining in networked data.

1. Introduction

Network structures typically consist of entities, also of different types, which are associated to each other in the network via various explicit relations (or edges). Analyzing and mining networked data allows us to discover the properties of nodes, as well as to capture topological, geometric and other characteristics of the structure of the network in many contexts (such as social networks, biological networks, chemical compounds and hidden criminal networks)[1].

Most objects and data in the real world are of multiple types and are interconnected, forming complex and heterogeneous information networks [2]. However, researchers mainly focus on analyzing and mining homogeneous networks, without distinguishing different types of objects and links in them. Mining heterogeneous networks requires for attention not only on the attributes which may describe nodes and links, but also on the possibly different types of nodes (with different attributes) and different types of edges among them.

Moreover, most of the existing algorithms developed to learn or analyze networked data assume that the network is static, i.e., the structure of the network is unchangeable and known before the learning process starts. This assumption also seems to be too restrictive in real-world scenarios, where networks can be dynamic and can exhibit changes especially when modeling phenomena which evolve over time. In this case, the networks are observed at consecutive snapshots, so that a stream of data can be generated. In this stream, properties of both nodes and edges may change over time and both nodes and edges of the networks may appear and disappear. For instance, in social networks, nodes can denote users (or "users profiles"). Two users can be connected, at a certain time-point, through the relationship "participation in the same event", but at a different time-point they can be connected through the relationship "friendship". By analyzing network changes, we can follow variations, adapt tools and services to new demands, as well as capture and delay undesirable alterations. Moreover, time associated to changes represents a valuable source of information which should be modeled to better understand both the whole dynamics and each change in the course of the dynamics. For example, in social networks the time of appearance of links among the users may convey important information on the formation of social communities [3].

Heterogeneity and dynamicity of the networks require for a different class of data mining methods and different representation formalisms which are able to overcome limitations of current methods. In the literature, the task of change mining has been mainly explored for time-series, transactional data and tabular data, by focusing on the detection of significant deviations in the values of the attributes describing the data. However, research on network analysis has mainly investigated graph-theoretical approaches, which oversimplify the representation of networks. Indeed, graph-theory mainly investigates structural aspects, such as distance and connectivity, in homogeneous networks, while it dedicates less attention to heterogeneous networks.

This heterogeneity of nodes and edges requires for different representation formalisms and, consequently, a different class of data mining methods which are able to handle this further complexity in the data. It has been argued that the (multi-) relational setting [4] is the most suitable for data mining problems on complex objects, since it can deal with the heterogeneity, it can distinguish the different role of object types (target or non-target, that is, primary units of analysis or secondary units of analysis), it can naturally represent a large variety of relationships among objects, it can characterize the change in objects and it can accommodate temporal information associated to the change. However, very few attempts have been made to investigate this class of data mining methods in the case of dynamic and heterogeneous networks, and so the current work is intended to be a contribution in this direction.

We investigate the problem of change mining in dynamic and heterogeneous networks through an approach which exploits the representation formalism and reasoning techniques of the most common multi-relational framework, namely inductive logic programming (ILP)[5]. The formalism of first order logic available in ILP allows us to straightforwardly model relationships and properties of nodes and edges with logical predicates and, therefore, it introduces an articulated and sophisticated representation of a network. Indeed, since a network can always be represented in the first order logic formalism but, on the contrary, first order logic data cannot always be represented as a network, the proposed approach could also be applied to representations which are more complex than a (even heterogeneous) network[1]. Moreover, the ILP framework allows us to exploit (when available) some forms of background knowledge which facilitates the learning task. However, resorting to ILP solutions, while providing the benefits described before, can also lead to efficiency problems which we alleviate through an efficient search procedure.

In the task we consider, changes denote the evolution of relationships or properties which emerge in the network at consecutive time windows. They are expressed in the form of *change chains*, which are sequences of frequent patterns which accommodate temporal information associated to the change. Frequent patterns are discoveredalong time windows. In particular, each frequent pattern represents a portion of the network which is frequently observed in the state of the network along the time window. Changes are punctual differences exhibited by the frequent patterns over consecutive windows. Mining changes from the frequent patterns leads to several advantages:

- Frequent patterns allow us to search for changes in an abstract and summarized description of the network, with the final result of reducing the computational cost with respect to directly analyzing data.

- Since frequency denotes statistical evidence, frequent patterns provide arguments for the robustness of our method. Patterns are also justified by the fact that a dynamic network actually exhibits changes only in some aspects, while it keeps others unchanged, and, therefore, they

[1]For example, is-a relationships between objects cannot be represented in the network-based formalism.

turn out to be a suitable means to capture those regularities which are present over time.

- Resorting to a relational data mining solution allows us to identify patterns which, in addition to common features, represent both structural and topological properties of the network and, thus, structural and topological changes of the network over time. In particular, we can represent different aspects of the network such as nodes (*is_a(X,user)*, X is a node of the network which represents a user), nodes' properties (*age_of_the_user(X,20)*) and edges (*friendship(X,Y)*). The first and the third aspect allow us to represent the structure of the network and thus, its topological properties.

It is noteworthy that high-frequency patterns correspond to situations which are observed in several snapshots of the network. Therefore, the term "frequent" does not refer to properties which are common to many objects (nodes) of the network, but it refers to properties of the same network which are observed at different timestamps. This allows us to catch changes that are associated to a small set of objects in the network.

An example of a change chain which can be extracted in the context of social network analysis is $\langle P_1, P_2, P_3 \rangle$, where:

P_1: $network(N), subscribed\_to(X,N), is\_a(X,user), subscribed\_to(Y,N), is\_a(Y,user),$ $subscribed\_to(W,N), is\_a(W,group), \mathbf{participation\_to\_the\_same\_event}(\mathbf{X},\mathbf{Y}),$ $membership(X,W), membership(Y,W)$ [October_2010]

P_2: $network(N), subscribed\_to(X,N), is\_a(X,user), subscribed\_to(Y,N), is\_a(Y,user),$ $subscribed\_to(W,N), is\_a(W,group), \mathbf{membership\_to\_the\_same\_group}(\mathbf{X},\mathbf{Y}),$

$membership(X, W), membership(Y, W)$ [November_2010]

P_3: $network(N), subscribed\_to(Z, N), is\_a(Z, user), subscribed\_to(Y, N), is\_a(Y, user),$ $subscribed\_to(W, N), is\_a(W, group), \mathbf{friendship(Z, Y)},$ $membership(X, W), membership(Y, W)$ [December_2010]

P_1 states that two *users* (denoted with the variables X, Y) are (frequently) connected through the $participation\_in\_the\_same\_event(X, Y)$ relationship during the time window [October_2010]. Pattern P_1 also expresses the membership of the two users X and Y in the group denoted as W. P_2 and P_3 refer to different time windows and are similar to P_1, except that the relationship between the same users differs. $\langle P_1, P_2, P_3\rangle$ includes two *changes* expressed by the pairs of patterns (P_1,P_2) and (P_2,P_3) respectively. The first change is associated to the time window pair [October_2010],[November_2010] and concerns the edge *participation_in_the_same_event(X, Y)* in the pattern P_1 which becomes *membership_to_the_same_group(X, Y)* in pattern P_2. The second change is associated to the time-window pair [November_2010], [December_2010] and concerns the edge *membership_to_the_same_group(X, Y)* in pattern P_2 which becomes *friendship(Z, Y)* in pattern P_3.

The novelty of the proposal is clarified in the next section, where related works are introduced and discussed. Then the problem faced in the paper is formally stated in Section 3. In Section 4, the proposed computational solution is reported. We structure it in two main steps and describe the algorithmic details to implement them. Also, we discuss the importance of a user-defined background knowledge for the presented approach and finally we report a theoretical analysis of the time complexity. The experimental setting is detailed in Section 5, where the results are also reported and evaluated.

Finally, in Section 6, conclusions are drawn and future research directions are identified.

2. Related Work and Motivations

Although numerous contributions can be listed under the umbrella of change mining, the investigation of this problem in dynamic networks is rather recent. In this context, two approaches can be identified:

i) Clustering-based solutions, which work on the identification of the changes in the global properties of the network;

ii) Frequent pattern mining-based solutions, which focus on the characterization of changes of local properties.

Clustering-based solutions are based on the intuition that clusters provide a natural summary for understanding both the underlying network structure and the inherent changes during the evolution process [6]. For example, in the context of social networks, in [7] the authors propose a method to selectively store a subset of graphs, in order to approximate the entire graph stream and to find community changes in time-evolving graphs based on the user specified time interval and on the number of the communities. Sun et al. [8] propose a technique to discover communities (clusters) and detect changes in clusters extracted at different time points. Clusters are represented according to some encoding schemes and the algorithm exploits the MDL principle. A similar idea is used in [9] and [10] where the authors propose to incrementally and efficiently summarize tensors by exploiting tensor analysis. In [11] the authors focus on the problem of publication analysis, in order to identify changes and evolution of research communities. The algorithm is based on a specifically designed description language to compress publication information in a bipartite graph with time stamps. The goal is to operate on such a time-stamped graph and exploit the MDL principle, in order to automatically spot communities, their evolution and cut-points between epochs of stable community evolution.

A hierarchical clustering technique is used in [12] to identify periods of evolution (eras) of a dynamic network. A period is associated to a cluster and it is produced as a sequence of structurally similar temporal snapshots of the network, so a new cluster represents a structural change with respect to previously generated clusters and denotes the beginning of a new period. An interesting feature of this solution is the possibility to analyze the evolution of the network at different temporal granularity levels, thanks to the adoption of hierarchically related clusters.

Another research stream focuses on the aspect of using visualization techniques in order to present the evolution of the network. For example, in [3], the authors propose to cluster together nodes in order to identify subgroups. Once subgroups have been identified, it is possible to visualize both the evolution of a subgroup and the evolution of connections among the subgroups.

A different perspective of the problem that does not resort to the clustering task is given in [13], where the authors study the temporally evolving web graphs. The peculiarity of this work is that the mining problem is divided into three levels of interest: single node, subgraphs and whole graph analysis, each of which is faced with different techniques. The level of subgraphs, which is more related to the task considered in this paper, is solved by means of frequent pattern mining solutions.

In [14] the authors present one of the most recent works which extracts changes in the form of patterns. In this work the authors study the problem of analyzing the whole evolution of the network and propose a method which creates a graph (i.e. a set of patterns) where conserved states of the network are the vertices while the admissible transitions among those states are the edges. The conserved states correspond to sequences of consecutive time-stamped networks with structural similarity. They are represented as induced sub-graphs whose configuration of the relations (labelled edges) and nodes occurs frequently over the corresponding sequence. The transitions are associated to modifications on the nodes of a state and can determine the migration towards the next state. The paths of the graph of the states represent alternative courses which can characterize the whole evolution. Only those which are considered maximal are retained. Although the discovery of the conserved states is based on probabilistic evidence, the discovery of transitions relies on the mere (dis)similarity between the nodes of two states.

In [15], we investigated the task of characterizing the evolution by introducing a new notion of emerging patterns [16]. In that work, emerging patterns model changes on the frequency and topology of the sub-graphs, occurring over consecutive pre-defined time-periods. In [17] the authors propose a method to mine frequent subsequences from graph sequence data in the form of patterns. They also define a formalism to represent changes of subgraphs over time. However, as observed in [18], the patterns discard information on the time in which the changes take place. In [19] the authors identify subgraphs changing over time by means of vertex-importance scores and vertex-closeness changes in subsequent snapshots of the graphs. The basic intuition comes from the social network domain: if two (important) nodes A and B are connected, then the distance (closeness) between nodes which are connected to A and nodes which are connected to B decreases (increases). Consequently, the notion of subgraph does not depend on the frequencies, which we consider important for the robustness of the mining task, but on the importance scores (which are computed using random walks). In [20] and [18] the history of an edge (absence and presence) is represented as a sequence. Then graph-mining techniques are applied to mine frequent patterns. In [18] the authors propose several optimizations that lead to extracting "Graph Evolution Rules" from larger networks. These optimizations are implemented in the system GERM.

Lahiri et. al. [21] introduce an approach to predict the future structure in a dynamic network and mine periodic patterns using frequent subgraphs. The approach proposed in [22] follows the same principle, but a compression-based measure is used instead of the frequency-based approach, in order to discover patterns in a dynamic graph.

Dynamic spatio-temporal networks are analyzed in [23], in order to track the movements of objects recorded in video data. A sequence of graphs models the video by associating a graph to a video-frame: the nodes denote regions which contain examined objects and edges represent the adjacency relationship between nodes. The goal is to mine frequent plane subgraphs from a database of plane graphs. These subgraphs are then used to generate spatiotemporal patterns, where a spatiotemporal pattern is a set of occurrences of a given pattern, such that occurrences are not too far apart, for close time points. In the considered application domain, a spatiotemporal pattern is used to track an object in a video.

By comparing clustering-based approaches and pattern-based approaches, we conclude that, while the former provide us an enumeration of the entities which show similar changes, the latter provide us a more human-understandable *characterization* of changes in a dynamic network. Since we aim at both characterizing and describing changes, we focus on the problem of extracting patterns from dynamic networks. Moreover, differently from existing approaches that extract patterns in order to identify changes due to insertion/deletions of nodes/edges in the network (GERM is an example of approach that identifies single insertions), we consider the less investigated (but still important) problem of identifying changes in the same (frequent) subgraphs across different time windows. In other words, we concentrate on structural updates of the network rather than either insertions or deletions.

3. Basics and Problem Definition

Before formally stating the data mining problem, we introduce some basic definitions. We first provide basics and notations for data representation, then we provide formal definitions of change chains. Eventually, we provide a formal definition of the problem to be solved.

3.1. Data representation

Let $O = \langle O_1, O_2, \ldots, O_n \rangle$ be a sequence of time-ordered observations of the network, obtained at regular time-points. At each time-point t_i, the network is described by an observation $O_i = \langle \mathcal{N}_i, E_i \rangle$, where $\mathcal{N}_i$ denotes the sets of nodes, $E_i = \{(u'_j, u''_j, e_j) \mid u'_j, u''_j \in \mathcal{N}_i, e_j \in \mathcal{E}\}$ represents the edges and $\mathcal{E}$ denotes the set of all possible labels of the edges.

A *time-period* (or time window or, simply, *period*) τ in $\{t_1 \dots t_n\}$ is a sequence of consecutive time-points $\{t_i, \dots, t_j\}$ ($t_1 \leq t_i$, $t_j \leq t_n$). The width w of τ is the number of time-points in τ, i.e. $w = j-i+1$. Two periods τ and τ' are said to be *consecutive* if $\tau = \{t_i, \dots, t_{i+w}\}$ and $\tau' = \{t_{i+w+1} \dots t_{i+2w}\}$. Since we assume that all the periods have the same width w, we enumerate periods and use the notation τ_h and τ_{h+1} to indicate two consecutive periods.

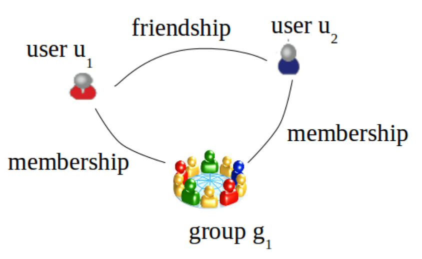
In the relational setting, when handling complex objects such as networks, different roles can be played by different *types* of data. More precisely, objects can be distinguished as target objects of analysis (TOs) and non-target objects of analysis (NTOs). In our context, TOs represent the whole network at a single time-point, that is O_i, while NTOs refer to nodes (of different *types*) of a network. This distinction, which comes from a usual practice in statistics of distinguishing between units of analysis and units of observation, allows us to generalize on the units of analysis, i.e. on the state of the network. It is noteworthy that TOs play a crucial role since they are used in the computation of the support of a pattern (this aspect will be discussed later). We denote the unique set of TOs as S and the multiple sets of NTOs as R_k ($1 \leq k \leq M$), where M denotes the number of sets of NTOs. For example, in the description of the state of the network in Figure 1, a_{t_i} is the TO which represents the whole network at time t_i, while u_1 (user), u_2 (user) and g_1 (group) are NTOs. It is noteworthy that constants u_1, u_2 and g_1 are local to the snapshot of the network at time t_i. Thus, they can represent different objects in two distinct snapshots.

Both target objects and non-target objects can be represented in Datalog [24] as sets of ground atoms[2] which populate the extensional part D^E of a deductive database D. Since we can populate D^E with the ground atoms of TOs and NTOs observed in a specific time-period τ_h, we can actually associate a deductive database D_h to each time-period τ_h.

Some predicate symbols are introduced in order to express both properties of TOs and NTOs and relationships between them. They can be categorized into four classes:

1. *key predicate*: it identifies the TOs in D_h^E (e.g., *network(a_{t_i})* in the example in Figure 1);
2. *property predicates:* they are binary predicates which define the values taken by an attribute of a TO or an NTO (e.g., *age_of_the_user(u_1,20)*, in Figure 1);
3. *structural predicates:* they are binary predicates which relate an NTO or a TO with another NTO (e.g., *friendship(u_1,u_2)* in Figure 1);
4. *is_a predicate*: it is a binary predicate which associates an NTO with its *type* (in the example in Figure 1 there are two users and one group). In our definition of the problem, an NTO can be represented at different levels of granularity since it is possible to define hierarchies on each single type. In this way, it is possible to represent relationships between both individual objects and sets of objects, for instance we can repre-

[2]A ground atom is an n-ary logic predicate symbol applied to n constants. We assume that the reader is familiar with some basic notions of computational logic, such as term, atom, literal, clause and substitution. Readers unfamiliar with this terminology should consult [25].

a)	b)
network(a_{t_i}), subscribed_to(u_1,a_{t_i}), subscribed_to(u_2,a_{t_i}), subscribed_to(g_1,a_{t_i}), is_a(u_1,user), age_of_the_user(u_1,20), is_a(u_2,user), age_of_the_user(u_2,41), is_a(g_2,group), friendship(u_1,u_2), membership(u_1,g_1), membership(u_2,g_1)	 friendship user u$_2$ user u$_1$ membership membership group g$_1$

friendship(john, james);
friendship(graduatedstudent, teacher);
friendship(user, user);

Figure 1: The state of the network at the time point t_i: a) Datalog representation and b) graphical representation.

(a)	(b)

Figure 2: Given the hierarchy on the *type* “users” (in (b)), we represent the same relationship at different granularity levels (in (a)).

sent the hierarchical relationship between an individual object (e.g., ”James”) and a set (e.g., ”teacher”, see Figure 2).

A structural predicate which relates two NTOs (also belonging to different types) represents the label of the edge connecting the nodes, which correspond to the two NTOs in the network.

The intensional part D_h^I of the deductive database D_h allows the user to define a graph which models the background knowledge on the labels of the edges of the network[3]. More precisely, this graph allows us to express

[3]In order to avoid confusion, in the paper, the terms network and graph are not used interchangeably.

the pair-wise dissimilarity between the labels of the network in the form of Datalog facts.

For instance

$$\textit{friendship} \xrightarrow{0.88} \textit{membership\_in\_the\_same\_group}$$

states that the dissimilarity between the labels *friendship(·, ·)* and *membership_in_the_same_group(·, ·)* is 0.88. More generally, it represents an undirected weighted link between two vertices v_i, v_j (e.g., between *friendship* and *membership_in_the_same_group*) with weight w_{ij} (e.g., 0.88) and it is denoted as $l(v_i, v_j, w)$. A finite sequence of undirected links $l_1, l_2, \ldots, l_m$ which connects two vertices v_i, v_j is called $SPath$ and denoted as $\rho(v_i, v_j)$. More specifically, $SPath$ is the shortest path which connects v_i and v_j. The complete list of such undirected links represents a user-defined background information on the dissimilarity between the labels of the network and, accordingly, allows us to quantify the change between two patterns. All deductive databases D_h share the same intensional part $D_h^I{=}D^I$.

While defining the background knowledge can be a problem in some application domains, it can also be an opportunity since it can also be profitably used by domain experts in order to adequately configure the system and extract useful and actionable knowledge. The effort in defining the background knowledge would mostly lie in the identification of the values of dissimilarity between pairs of the labels of the edges. Since in our approach the number of different labels is rather limited, the definition of the background is not a demanding task for the expert. When this background knowledge is not provided, all dissimilarities between labels of edges will be considered equal.

3.2. Change Patterns and Change Chains

Relational patterns consist of Datalog non-ground atoms and are expressed by means of a set notation. A Datalog non-ground atom is an n-ary predicate symbol applied to n terms (either constants or variables), at least one of which is a variable. A formal definition of pattern is reported in the following:

Definition 1. *Relational pattern*
Let P be a set of atoms, P is a *relational pattern* iff

$$P = p_0(t_0^1), p_1(t_1^1, t_1^2), p_2(t_2^1, t_2^2), \ldots, p_k(t_k^1, t_k^2),$$

where p_0 is the key predicate, while p_i, $i = 1, \ldots, k$ is either a structural predicate or a property predicate or an *is_a* predicate. Moreover, all variables are connected to the variable used in the key predicate (according to the linkedness property). ■

Terms t_i^j are either constants, which correspond to values of property predicates, or variables, which identify target objects or non-target objects. Each p_i is a predicate occurring in D_h^E (extensionally defined predicate).

A relational pattern P is characterized by a statistical parameter, namely the *support* (denoted as $supp(P)$), which denotes the relative frequency, computed on TOs, of P in a time-period τ_h. When the support exceeds a minimum user-defined threshold, P is said to be *frequent.*

The following definitions are crucial for this work:

Definition 2. *Stable pattern*

Let P be a relational pattern and τ_h, τ_{h+1} be two consecutive time-periods. If P is frequent both in τ_h and τ_{h+1} then P is *stable* in $[\tau_h, \tau_{h+1}]$. ■

Definition 3. *Change pattern*

Let:

- τ_h, τ_{h+1} be two consecutive time-periods;
- $P' = p_0({t'}^1_0), p_1({t'}^1_1, {t'}^2_1), \ldots, p_{k-1}({t'}^1_{k-1}, {t'}^2_{k-1}), p'_k({t'}^1_k, {t'}^2_k), p_{k+1}({t'}^1_{k+1}, {t'}^2_{k+1}) \ldots$ be a frequent relational pattern in τ_h and a non-frequent relational pattern in τ_{h+1};
- $P'' = p_0({t''}^1_0), p_1({t''}^1_1, {t''}^2_1), \ldots, p_{k-1}({t''}^1_{k-1}, {t''}^2_{k-1}), p''_k({t''}^1_k, {t''}^2_k), p_{k+1}({t''}^1_{k+1}, {t''}^2_{k+1}) \ldots$ be a frequent relational pattern in τ_{h+1} and a non-frequent relational pattern in τ_h.

Then:

$$P^{(c)} = p_0(t^1_0), p_1(t^1_1, t^2_1), \ldots, p_{k-1}(t^1_{k-1}, t^2_{k-1}), (p'_k(t^1_k, t^2_k) \to p''_k(t^1_k, t^2_k)), p_{k+1}(t^1_{k+1}, t^2_{k+1}) \ldots$$

$$[\tau_h, \tau_{h+1}]$$

is a change pattern in $[\tau_h, \tau_{h+1}]$ iff

- p'_k and p''_k are *different* structural predicates which correspond to labels of edges in the network;
- $P^{(c)}$, P' and P'' are equal, except for the k-th atom and up to a re-denomination of the variables. ■

Since p'_k and p''_k express the change across two consecutive time-periods, we use the following notation:

$$P^{(c)} = p_0(t_0^1), p_1(t_1^1, t_1^2), \ldots, (p_k^{\tau_h}(t_k^1, t_k^2) \rightarrow p_k^{\tau_{h+1}}(t_k^1, t_k^2)), \ldots \qquad [\tau_h, \tau_{h+1}],$$

where the symbol " $\rightarrow$ " in $(p_k^{\tau_h}(\cdot,\cdot) \rightarrow p_k^{\tau_{h+1}}(\cdot,\cdot))$ indicates that the predicate $p_k^{\tau_h}(\cdot,\cdot)$, observed in the period τ_h, becomes $p_k^{\tau_{h+1}}(\cdot,\cdot)$ in the period τ_{h+1}. A change pattern is characterized by a value γ which quantifies the modelled change (further details on γ will be provided in the following).

An example of a change pattern between the time-periods $October\_2010$ and $November\_2010$ is:

$$\begin{aligned} P_4^{(c)}:\ & network(N), \\ & subscribed\_to(X,N), is\_a(X, user), \\ & subscribed\_to(Y,N), is\_a(Y, user), \\ & \big(\mathbf{participation\_in\_the\_same\_event^{October\_2010}(X,Y)} \rightarrow \\ & \mathbf{membership\_in\_the\_same\_group^{November\_2010}(X,Y)}\big)\ . \quad [October\_2010, November\_2010], \end{aligned}$$

where the variable N denotes the target object, variables X, Y denote some non-target objects, while the predicate $network(\cdot)$ identifies the key predicate and $participation\_in\_the\_same\_event(\cdot,\cdot)$ and $membership\_in\_the\_same\_group(\cdot,\cdot)$ are structural predicates. All variables are implicitly existentially quantified. Intuitively, change patterns are obtained by joining relational patterns extracted at consecutive time-periods. In this case, $P_4^{(c)}$ is derived from P_5 and P_6:

$$
\begin{aligned}
P_5 :\ & network(N),\\
& subscribed\_to(X,N), is\_a(X,user),\\
& subscribed\_to(Y,N), is\_a(Y,user),\\
& participation\_in\_the\_same\_event(X,Y). \qquad [October\_2010]
\end{aligned}
$$

$$
\begin{aligned}
P_6 :\ & network(N),\\
& subscribed\_to(Z,N), is\_a(Z,user),\\
& subscribed\_to(Y,N), is\_a(Y,user),\\
& membership\_in\_the\_same\_group(Z,Y). \qquad [November\_2010]
\end{aligned}
$$

Details on how joining is performed are provided in the next Section.

As stated before, this definition of change pattern allows us to identify changes in the same subgraphs (of the same size) across different time windows. However, if a pattern of length n is frequent and the same pattern with an additional literal is frequent as well, they are both considered in the change patterns. Since that point, they are processed independently, although their frequencies still remain strictly related and variations on one pattern will significantly influence variations on the other pattern.

Once we have defined the concepts of stable and change patterns, we can define the concept of a change chain.

Definition 4. *Change Chain*

Let:

- P_1, P_2,..., P_n be a list of relational patterns which are frequent in the time-periods $\tau_1, \tau_2, \ldots, \tau_n$, respectively;
- $P^{(c)}_{1,2}$ be a change pattern for $[\tau_1, \tau_2]$ derived from P_1 and P_2,

- $P^{(c)}_{n-1,n}$ be a change pattern for $[\tau_{n-1}, \tau_n]$ derived from P_{n-1} and P_n,
- $P_{i,i+1},\ i = 2, \ldots, n-2$ be either change patterns or stable patterns for $[\tau_i, \tau_{i+1}]$ derived from P_i and P_{i+1}.

Then:
$C = \langle P^{(c)}_{1,2}; P_{2,3}; \ldots; P_{n-2,n-1}; P^{(c)}_{n-1,n} \rangle$ is a *change chain*. ■

Intuitively, a change chain collects the (most frequent) changes that the network exhibits in pairs of consecutive time-periods, possibly alternating with stable time-periods. The changes are modelled in the form of change patterns. An example of a change chain with only change patterns is:

$\langle$
$network(N), subscribed\_to(X, N), is\_a(X, user), subscribed\_to(Y, N), is\_a(Y, user),$
$(\mathbf{participation\_in\_the\_same\_event^{October\_2010}(X, Y)}$
$\quad \rightarrow \mathbf{membership\_in\_the\_same\_group^{November\_2010}(X, Y)})$
$[October\_2010, November\_2010];$
$network(N), subscribed\_to(X, N), is\_a(X, user), subscribed\_to(Y, N), is\_a(Y, user),$
$(\mathbf{membership\_in\_the\_same\_group^{November\_2010}(Z, Y)}$
$\quad \rightarrow \mathbf{friendship^{December\_2010}(Z, Y)})$
$[November\_2010, December\_2010]\ \rangle$

An example of a change chain which includes both stable and change patterns is:

$\langle$
$network(N), subscribed\_to(X, N), is\_a(X, user), subscribed\_to(Y, N), is\_a(Y, user),$

$$(\mathbf{participation\_in\_the\_same\_event}^{\mathbf{October\_2010}}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{membership\_in\_the\_same\_group}^{\mathbf{November\_2010}}(\mathbf{X},\mathbf{Y})) \quad [October\_2010, November\_2010];$$

$network(N), subscribed\_to(X,N), is\_a(X,user), subscribed\_to(Y,N), is\_a(Y,user),$
$membership\_in\_the\_same\_group(Z,Y) \quad [November\_2010, December\_2010];$

$network(N), subscribed\_to(X,N), is\_a(X,user), subscribed\_to(Y,N), is\_a(Y,user),$

$$(\mathbf{membership\_in\_the\_same\_group}^{\mathbf{December\_2010}}(\mathbf{Z},\mathbf{Y}) \rightarrow \mathbf{friendship}^{\mathbf{January\_2011}}(\mathbf{Z},\mathbf{Y})) \quad [December\_2010, January\_2011]$$

$\rangle$

3.3. Formal definition of the problem

We can now give a formal statement of the problem of discovering change patterns and change chains:

Given:

- A sequence of n observations $\langle O_1, \ldots, O_n \rangle$;
- the width w of the time-periods;
- a threshold $minSup \in [0;1]$, which represents the minimum support value for mining relational frequent patterns;
- a threshold $min\Gamma \in [0;1]$, which defines the minimum dissimilarity value (between the labels of the edge) allowed to detect the change between two different structural predicates;
- two thresholds $minP$, $maxP$, which determine the minimum and maximum number of change patterns in a change chain, respectively;

- a threshold $maxS$, which determines the maximum number of stable patterns in a change chain.[4]

Find:

- The set Υ of change patterns. They are built by using only patterns whose support is greater than $minSup$, and by considering only changes where structural predicates differ at least by $min\Gamma$.
- The set Ψ of change chains generated by Υ. They are built by satisfying the constraints set by $minP$, $maxP$, and $maxS$.

4. The Algorithm

The computational solution which we propose for the problem formalized in the previous section operates in three steps: *i)* discovering a set of frequent relational patterns $\mathcal{P}_h$ from each deductive database D_h built on the TOs and $NTOs$ of the time-period τ_h; *ii)* generating change patterns from the frequent patterns; *iii)* generating change chains from both the discovered change patterns and the stable patterns.

4.1. Discovering Frequent Relational Patterns

We define as *units of analysis* the target objects on which patterns are determined and which contribute to compute the support of a pattern. The non-target objects contribute to define the units of analysis and can be involved in a pattern. The support $supp_h(P)$ of a pattern P is the percentage

[4]Implicitly, $minP$, $maxP$, and $maxS$ define the minimum and the maximum length of a change chain.

of *units of analysis* in D_h *covered by* P. More precisely, the set of units of analysis $D_h[s]$ of a target object $s \in S$ in the time-period τ_h is a subset of ground atoms in D_h^E defined as follows:

$$D_h[s] = is\_a(R_h(s)) \cup D_h[s|R_h(s)] \cup \bigcup_{r_i \in R_h(s)} D_h[r_i|R_h(s)], \qquad (1)$$

where $R_h(s)$ is the set of NTO directly or indirectly related to s in τ_h, $is\_a(R_h(s))$ is the set of is_a atoms which define the types of $r_i \in R_h(s)$, $D_h[s|R_h(s)]$ contains properties of s and relations between s and some $r_i \in R_h(s)$ in τ_h, $D_h[r_i|R_h(s)]$ contains properties of r_i and relations between r_i and some $r_j \in R_h(s)$ in τ_h. By assigning a pattern P with an existentially quantified conjunctive formula $eqc(P)$, obtained by transforming P into a Datalog query, the units of analysis $D_h[s]$ are *covered by* a pattern P if $D_h[s] \models eqc(P)$, namely $D_h[s]$ logically entails $eqc(P)$.

Frequent patterns are mined with SPADA[26, 27], which enables the discovery of relational patterns (at different levels of granularity) whose support exceeds *minSup*. SPADA performs a breadth-first search of the space of patterns, from the most general to the more specific ones, and prunes portions of the space which contain only non-frequent patterns. The pruning strategy guarantees that all non-frequent patterns are removed and, to this aim, uses a generality ordering based on the notion of θ-subsumption [28]:

Definition 5. P' is *more general* than P'' under θ-subsumption ($P' \succeq_\theta P''$), if and only if P' θ-subsumes P'', i.e. a substitution θ exists, such that $P'\theta \subseteq P''$.

For instance, given the following relational patterns:

$P_7 \equiv network(N), subscribed\_to(X, N), is\_a(X, user)$

$P_8 \equiv network(N), subscribed\_to(X, N), is\_a(X, user), subscribed\_to(Y, N)$

$P_9 \equiv network(N), subscribed\_to(X, N), is\_a(X, user), subscribed\_to(Y, N), is\_a(Y, user)$

we observe that P_7 θ-subsumes P_8 ($P_7 \succeq_\theta P_8$) and P_8 θ-subsumes P_9 ($P_7 \succeq_\theta P_9$) with substitutions $\theta_1 = \theta_2 = \oslash$. The generality order is monotonic with respect to the pattern support, so whenever P_7 is non-frequent, its more specific patterns (e.g., P_8, P_9) will be non-frequent too.

The search is based on the level-wise method and implements a two-stepped procedure:

- *i)* generation of candidate patterns with k atoms (k-th level) by considering the frequent patterns with $k - 1$ atoms ((*k-1*)-th level);
- *ii)* evaluation of the frequency of the patterns with k atoms. So, the patterns whose support does not exceed *minSup* will not be considered for the next level.

Since in real-world applications a large number of frequent patterns can be generated, SPADA also offers a declarative language to express some pattern constraints which are then used to filter out uninteresting patterns [29].

SPADA has been recently extended in order to handle very large data sets [30]. This extension resorts to data sampling and distributed computation in Grid environments, and generates a set of frequent patterns which approximates of the set of exact solutions. In this work experiments could still be performed by applying the original serial version of SPADA. Nevertheless, for very large data sets, the parallel, distributed version of SPADA should be considered. Obviously, the use of SPADA for mining frequent relational patterns does not exclude the possibility of using other methods in this initial processing step.

4.2. Generating Change Patterns

This step is in charge of generating change patterns by combining the sets of frequent patterns $\mathcal{P}_h$, $\mathcal{P}_{h+1}$ extracted from data of the two consecutive time-periods τ_h and τ_{h+1}, respectively. Each change represents differences between the atoms of a pattern in $\mathcal{P}_h$ and the atoms of a pattern in $\mathcal{P}_{h+1}$.

The atoms considered are those whose predicates correspond to the labels on the edges $\mathcal{E}$ of the network, while the difference between the atoms is quantified by the dissimilarity value between the labels of the edges (according to the background knowledge D^I). A change pattern is the result of the combination of two patterns which differ in only one atom.

Algorithm 1 describes how frequent patterns in $\mathcal{P}_h$ and $\mathcal{P}_{h+1}$ are combined. In particular, the algorithm first creates a bipartite graph $\mathcal{G}_{\mathcal{D}}$, which represents the candidate patterns to be combined (lines 1-6) and then uses the graph to construct change patterns $\Upsilon_{h,h+1}$ (lines 7-16).

Concerning the first part, $\mathcal{G}_{\mathcal{D}}$ is a *bipartite* graph where vertices are partitioned into those representing patterns in $\mathcal{P}_h$ and those representing patterns in $\mathcal{P}_{h+1}$. As in classical bipartite graphs, only inter-partition links are allowed. For each pair of patterns which have the same length (namely, at the same level of the level-wise search method) the system checks whether they differ in only one atom and share the remaining atoms up to a re-denomination of variables (line 2). Let α and β be the two atoms which differentiate $P' \in \mathcal{P}_h$ from $P'' \in \mathcal{P}_{h+1}$ (α in P', β in P''), then the shortest

```
    Data: $\mathcal{P}_h, \mathcal{P}_{h+1}, D^I, min\Gamma$
    Result: $\Upsilon_{h,h+1}$
 1  for $(P', P'') \in \mathcal{P}_h \times \mathcal{P}_{h+1}$ do
 2      if $length(P') = length(P'')$ and $check\_atoms(P', P'')$ then
 3          $(\alpha, \beta) := atoms\_diff(P', P'')$; // $\alpha, \beta$ atoms differentiating $P', P''$
 4          $\omega := compute\_distance(\alpha, \beta, D^I)$;
 5          if $\omega \geq min\Gamma$ then
 6              $addVertex(P', \mathcal{G}_\mathcal{D})$; $addVertex(P'', \mathcal{G}_\mathcal{D})$; $addLink(P', P'', \omega, \mathcal{G}_\mathcal{D})$;
 7  $\mathcal{L}_\mathcal{D} \leftarrow$ links of $\mathcal{G}_\mathcal{D}$;
 8  $\Upsilon_{h,h+1} := \emptyset$;
 9  for $\langle P', P'', \omega \rangle \in \mathcal{L}_\mathcal{D}$ // list of links ordered in descending mode w.r.t. $\omega$
10  do
11      $P''' \leftarrow combine(P', P'')$;
12      $set\_\gamma(P''', \omega)$;
13      $\Upsilon_{h,h+1} := \Upsilon_{h,h+1} \cup P'''$;
14      $removeVertex(P', \mathcal{G}_\mathcal{D})$;
15      $removeVertex(P'', \mathcal{G}_\mathcal{D})$;
16      $\mathcal{L}_\mathcal{D} \leftarrow$ links of $\mathcal{G}_\mathcal{D}$;
```
Algorithm 1: Mining Change Relational Patterns

path ρ which connects α and β (or viceversa) is searched among the weighted links in D^I. If the sum ω of the weights (dissimilarities) found in the path (see Algorithm 2) is higher than the minimum change $min\Gamma$, the vertices P' and P'' are inserted into $\mathcal{G}_\mathcal{D}$ and connected through a link with weight ω (lines 3-6, Algorithm 1). Intuitively, at the end of the first sub-procedure, $\mathcal{G}_\mathcal{D}$ will contain, as vertices, the patterns which meet the condition in line 2 (Algorithm 1), and it will contain, as links, the weights associated to the path linking the atoms which differentiate the patterns. The minimum change threshold $min\Gamma$ is considered to prevent the generation of uninteresting change patterns. This allows us to prune the search space.

Once $\mathcal{G}_\mathcal{D}$ is built, in the second part of the algorithm the vertices, which are connected by means of a link, are transformed into change patterns. In particular, a list $\mathcal{L}_{\mathcal{D}}$ is populated with the vertices and links of $\mathcal{G}_{\mathcal{D}}$: an element of $\mathcal{L}_{\mathcal{D}}$ is a triple $\langle P', P'', \omega \rangle$ composed of a pair of patterns (P', P'') with their relative weight. Elements in $\mathcal{L}_{\mathcal{D}}$ are ranked in descending order with respect to the values of ω (line 9). This guarantees that change patterns with less similar atoms will be preferred to the others. For each element in $\mathcal{L}_{\mathcal{D}}$, the two patterns P' and P'' are combined, in order to generate a pattern P''', composed of the same atoms in common to P' and P'', as well as of the atom formed by the composition of two different atoms (lines 11-12). The value γ associated to P''' is exactly ω, computed according to Algorithm 2.

```
  Data: $\alpha, \beta, D^I$
  Result: $\omega$
1 $v_i = getLabel(\alpha)$ $v_j = getLabel(\beta)$;
2 if $\rho(v_i, v_j) \neq \oslash$ // Shortest path between $v_i$ and $v_j$ in $D^I$
3 then
4     $\omega := \sum_{l(v_k, v_q, w_{kq}) \in \rho(v_i, v_j)} w_{kq}$;
5 else
6     $\omega := +\infty$;
```

Algorithm 2: Dissimilarity between patterns according to D^I

This combination procedure allows us to build $\Upsilon_{h,h+1}$. At each iteration, the triple for the patterns P' and P'' is removed from $\mathcal{L}_{\mathcal{D}}$ (lines 14-15), as already considered in $\Upsilon_{h,h+1}$.

In Figure 3 we report a toy example for the generation of change patterns. Two frequent patterns discovered in two consecutive time-periods ($April\_1990$ and May_1990) are combined to form a change pattern if they differ in only one predicate. More precisely, they are combined if the predicates in which the patterns differ correspond to two different labels of edges which are dissimilar more than the threshold $min\Gamma$. By supposing $min\Gamma =$

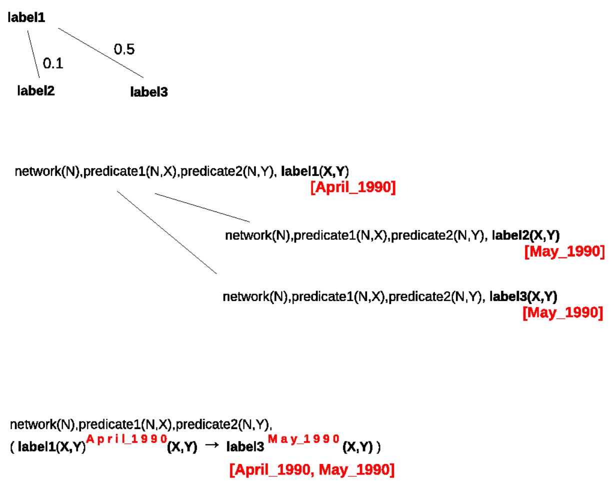

label1
0.1
0.5
label2
label3
network(N),predicate1(N,X),predicate2(N,Y), label1(X,Y)
[April_1990]
network(N),predicate1(N,X),predicate2(N,Y), label2(X,Y)
[May_1990]
network(N),predicate1(N,X),predicate2(N,Y), label3(X,Y)
[May_1990]
network(N),predicate1(N,X),predicate2(N,Y),
(label1(X,Y)^April_1990 (X,Y) → label3^May_1990 (X,Y))
[April_1990, May_1990]

Figure 3: Change patterns are created by combining frequent patterns which are discovered in two consecutive time-periods and which differ in only one predicate. The graph at the top of the picture represents the dissimilarity between labels.

0.3, we can combine only the patterns

- $network(N), predicate1(N, X), predicate2(N, Y), \mathbf{label1(X, Y)}[April\_2010]$
- $network(N), predicate1(N, X), predicate2(N, Y), \mathbf{label3(X, Y)}[May\_2010]$

since the dissimilarity between $label1$ and $label3$ is greater than 0.3.

A more complex real world example is illustrated in Figure 4. Consider the background knowledge D^I on the dissimilarity among four possible edge labels in social networks (Figure 4a), $min\Gamma$=0.3 and the sets of frequent patterns mined in the time-periods $\tau_h = April\_1990$ and $\tau_{h+1} = May\_1990$, respectively (Figure 4b). First, the bipartite graph $\mathcal{G}_\mathcal{D}$ is created, then two change patterns are generated (squares 1 and 2 in Figure 4c), the first of length 8 and the second of length 5. Note that the first pattern (square 1) is generated from the combination of the atoms *friendship(·, ·)* and *kinship(·, ·)*,

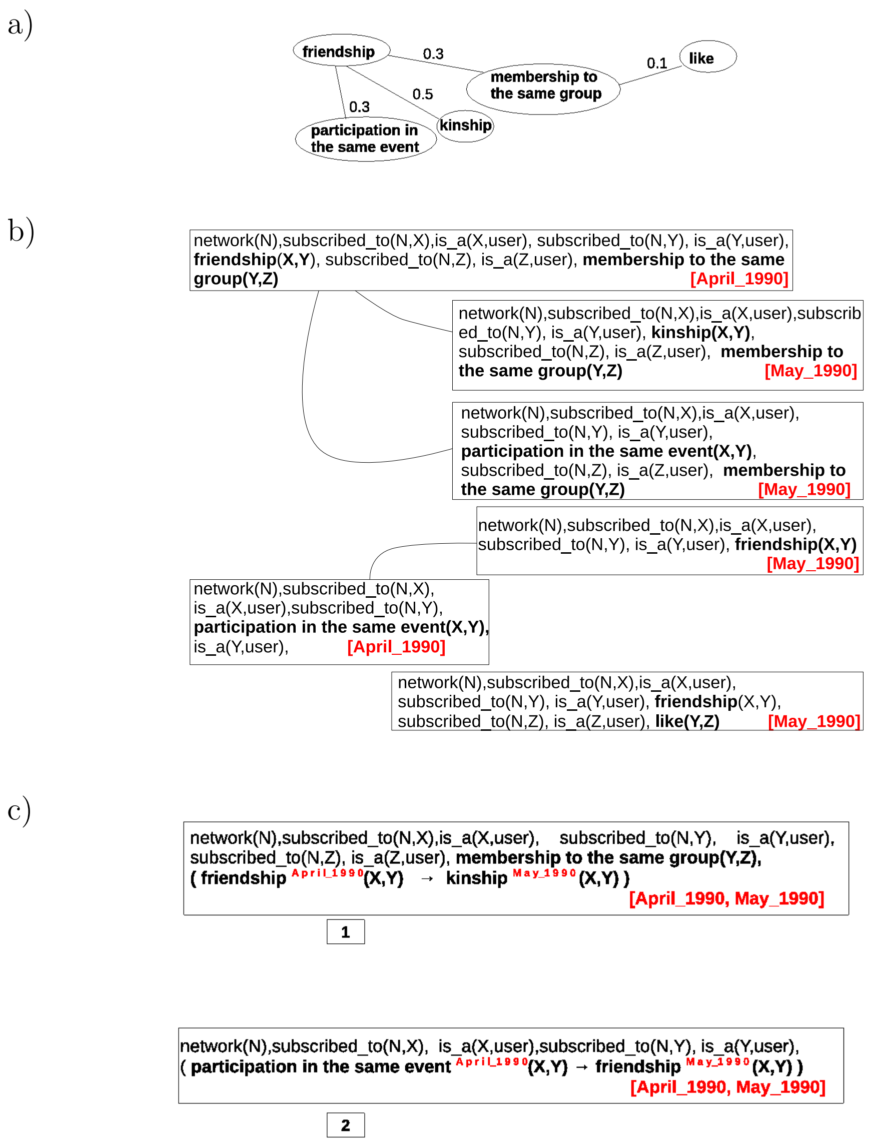

a)
friendship
0.3
membership to the same group
0.1
like
0.5
0.3
kinship
participation in the same event
b)
network(N),subscribed_to(N,X),is_a(X,user), subscribed_to(N,Y), is_a(Y,user), friendship(X,Y), subscribed_to(N,Z), is_a(Z,user), membership to the same group(Y,Z) [April_1990]
network(N),subscribed_to(N,X),is_a(X,user),subscribed_to(N,Y), is_a(Y,user), kinship(X,Y), subscribed_to(N,Z), is_a(Z,user), membership to the same group(Y,Z) [May_1990]
network(N),subscribed_to(N,X),is_a(X,user), subscribed_to(N,Y), is_a(Y,user), participation in the same event(X,Y), subscribed_to(N,Z), is_a(Z,user), membership to the same group(Y,Z) [May_1990]
network(N),subscribed_to(N,X),is_a(X,user), subscribed_to(N,Y), is_a(Y,user), friendship(X,Y) [May_1990]
network(N),subscribed_to(N,X), is_a(X,user),subscribed_to(N,Y), participation in the same event(X,Y), is_a(Y,user), [April_1990]
network(N),subscribed_to(N,X),is_a(X,user), subscribed_to(N,Y), is_a(Y,user), friendship(X,Y), subscribed_to(N,Z), is_a(Z,user), like(Y,Z) [May_1990]
c)
network(N),subscribed_to(N,X),is_a(X,user), subscribed_to(N,Y), is_a(Y,user), subscribed_to(N,Z), is_a(Z,user), membership to the same group(Y,Z), (friendship April_1990(X,Y) → kinship May_1990(X,Y)) [April_1990, May_1990]
1
network(N),subscribed_to(N,X), is_a(X,user),subscribed_to(N,Y), is_a(Y,user), (participation in the same event April_1990(X,Y) → friendship May_1990(X,Y)) [April_1990, May_1990]
2

Figure 4: a) An example of the background knowledge D^I in form of graph: a link between two vertices expresses the dissimilarity between the labels associated to the edges. b) The bipartite graph $\mathcal{G}_{\mathcal{D}}$, in its initial form, created from the patterns discovered in $April\_1990$ and May_1990 respectively ($min\Gamma$=0.3). c) Two change patterns discovered in [$April\_1990$, May_1990]: they are originally generated by combining the frequent patterns illustrated in the Figure b).

which is preferred to other combinations, due to their higher value of dissimilarity (0.5).

4.3. Discovering Change Chains

Once the sets of change patterns $\Upsilon_{1,2}, \Upsilon_{2,3}, \ldots, \Upsilon_{m-1,m}$ are identified, they are used to determine possible change chains. The algorithm operates with three sets: the set Ψ'' which contains the candidate incomplete chains (that is, chains that terminate with a stable pattern), the set Ψ' which contains the set of change chains and the set Ψ which contains the set of change chains to be returned. The algorithm proceeds iteratively and, at the h-th iteration ($h = 1, \ldots, m-1$), it uses $\Upsilon_{h,h+1}$ and $\mathcal{P}_{h+1}$ to update the sets Ψ'' and Ψ'.

Chains are processed in four different ways:

1. Let $P \in \mathcal{P}_{h+1}$ and $C \in \Psi'$, such that *i)* the number of stable patterns in C is less than $maxS$ and *ii)* the pattern associated to τ_h of the last change pattern in C is equal to P. Then a new chain which adds to C the stable pattern P is created and stored in Ψ''_{new}. C is removed from Ψ'.
2. Let $P \in \mathcal{P}_{h+1}$ and $C \in \Psi''$, such that *i)* the number of stable patterns in C is less than $maxS$ and *ii)* the pattern associated to τ_h of the last stable pattern in C is equal to P. Then a new chain which adds to C the stable pattern P is created and stored in Ψ''_{new}. C is removed from Ψ''.
3. Let $P^{(c)} \in \Upsilon_{h,h+1}$ and $C \in \Psi'$, such that *i)* the number of change patterns in C is less than $maxP$ and *ii)* the pattern associated to τ_h of the last change pattern in C is equal to $P^{(c)}$. Then a new chain which

adds to C the change pattern $P^{(c)}$ is created and stored in Ψ'_{new}. C is removed from Ψ' and $P^{(c)}$ is removed from $\Upsilon_{h,h+1}$.

4. Let $P^{(c)} \in \Upsilon_{h,h+1}$ and $C \in \Psi''$, such that *i)* the number of change patterns in C is less than $maxP$ and *ii)* the pattern associated to τ_h of the last stable pattern in C is equal to $P^{(c)}$. Then a new chain which adds to C the change pattern $P^{(c)}$ is created and stored in Ψ'_{new}. C is removed from Ψ'' and $P^{(c)}$ is removed from $\Upsilon_{h,h+1}$.

These four cases are considered in this order. This means that we give priority to stable patterns and not to change patterns (according to Definition 3). The removal of the used change patterns from the set $\Upsilon_{h,h+1}$ guarantees the discovery of maximal chains, namely the algorithm does not generate chains which are contained in other chains. In cases 3 and 4, if more than one change pattern is a candidate to be added to C, the one with the greatest γ is preferred.

At the end of each iteration, all the chains remaining in Ψ' are added to Ψ if the number of change patterns is greater than $minP$. At the next iteration, Ψ' is initialized with Ψ'_{new} and Ψ'' is initialized with Ψ''_{new}. Finally, after the last iteration, all the chains remaining in Ψ'_{new} are added to Ψ, if the number of change patterns in these chains is greater than $minP$.

The algorithmic description is reported in Algorithm 3. In order to clarify how it works, we report an explanatory example in Figure 5 which uses the change patterns and stable patterns mined in the time-periods $\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ (Table 1).

Let us consider $minP$=2, $maxP$=3 and $maxS$=2. The generation of the change chains begins from the change patterns mined in the time-periods τ_1

```
**Data**: $(\{\Upsilon_{1,2}, \Upsilon_{2,3}, \ldots, \Upsilon_{m-1,m}\}, minP, maxP, maxS)$
**Result**: $\Psi$
$h := 3$; $\Psi' := \Upsilon_{1,2}$ ;
**while** $h \leq m$ **do**
    $\Psi'_{new} := \emptyset$;$\Psi''_{new} := \emptyset$;
    **for** $P \in \mathcal{P}_h$ **do**
        **for** $C \in \Psi'$ **do**
            // C.sCounter: no. of stable patterns in C; C.nCounter: no. of change patterns
                in C
            **if** $C.sCounter \leq maxS$ **then**
                $L \leftarrow getLastPattern(C)$ // the last pattern in the last change pattern in $C$
                **if** $equal(P, L)$ **then**
                    $remove(\Psi', C)$; $insert(\Psi''_{new}, join(C, P))$; $C.sCounter++$;
        **for** $C \in \Psi''$ **do**
            **if** $C.sCounter \leq maxS$ **then**
                $L \leftarrow getLastPattern(C)$// the pattern in the last stable pattern in $C$
                **if** $equal(P, L)$ **then**
                    $remove(\Psi'', C)$; $insert(\Psi''_{new}, join(C, P))$; $C.sCounter++$;
    $\Psi'_{temp} := \emptyset$; $\Psi''_{temp} := \emptyset$;
    **for** $P^{(c)} \in \Upsilon_{h-1,h}$ **do**
        $P' \leftarrow getFirstPattern(P^{(c)})$ // the first pattern in $P^{(c)}$
        **for** $C \in \Psi'$ **do**
            **if** $C.nCounter \leq maxP$ **then**
                $L \leftarrow getLastPattern(C)$ // the last pattern in the last change pattern in $C$
                **if** $equal(P', L)$ **then**
                    $insert(C.candidates, P^{(c)})$; $update(\Psi'_{temp}, C)$; $remove(\Upsilon_{h-1,h}, P^{(c)})$;
    $\Psi'_{new} := \Psi'_{new} \cup select\_change\_patterns(\Psi'_{temp})$; $\Psi' \leftarrow removeExtendedChains(\Psi', \Psi'_{new})$;
    **for** $P^{(c)} \in \Upsilon_{h-1,h}$ **do**
        $P' \leftarrow getLastPattern(C)$ // the last pattern in the last change pattern in $C$
        **for** $C \in \Psi''$ **do**
            **if** $C.nCounter \leq maxP$ **then**
                $L \leftarrow getLastPattern(C)$ // the pattern in the last stable pattern in $C$
                **if** $equal(P', L)$ **then**
                    $insert(C.candidates, P^{(c)})$; $update(\Psi''_{temp}, C)$; $remove(\Upsilon_{h-1,h}, P^{(c)})$;
    $\Psi'_{new} := \Psi'_{new} \cup select\_change\_patterns(\Psi''_{temp})$; $\Psi' \leftarrow removeExtendedChains(\Psi', \Psi'_{new})$ ;
    $\Psi \leftarrow \Psi \cup check\_for\_minP(\Psi')$; $\Psi'_{new} \leftarrow \Psi'_{new} \cup \Upsilon_{h-1,h}$ ;
    $\Psi' \leftarrow \Psi'_{new}$; $\Psi'' := \Psi''_{new}$; $h++$;
$\Psi \leftarrow \Psi \cup check\_for\_minP(\Psi')$;
```

Algorithm 3: Discovering Change Relational Chains.

Table 1: The stable patterns and change patterns used in the example of Figure 5.

$P^{(c),1}_{1,2} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}))$	$([\tau_1,\tau_2])$
$P^{(c),2}_{1,2} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label3}^{\tau_2}(\mathbf{X},\mathbf{Y}))$	$([\tau_1,\tau_2])$
$P^{(c),3}_{1,2} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label4}^{\tau_2}(\mathbf{X},\mathbf{Y}))$	$([\tau_1,\tau_2])$
$P^{(c),1}_{2,3} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label5}^{\tau_3}(\mathbf{X},\mathbf{Y}))$	$([\tau_2,\tau_3], \gamma{=}0.8)$
$P^{(c),2}_{2,3} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label6}^{\tau_3}(\mathbf{X},\mathbf{Y}))$	$([\tau_2,\tau_3], \gamma{=}0.7)$
$P^{3}_{2,3} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), \mathit{label}3(X,Y)$	$([\tau_2,\tau_3])$
$P^{(c),4}_{2,3} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label6}^{\tau_2}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label4}^{\tau_3}(\mathbf{X},\mathbf{Y}))$	$([\tau_2,\tau_3])$
$P^{(c),1}_{3,4} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label6}^{\tau_3}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label7}^{\tau_4}(\mathbf{X},\mathbf{Y}))$	$([\tau_3,\tau_4])$
$P^{2}_{3,4} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), \mathit{label}5(X,Y)$	$([\tau_3,\tau_4])$
$P^{3}_{3,4} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), \mathit{label}3(X,Y)$	$([\tau_3,\tau_4])$
$P^{(c),1}_{4,5} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), (\mathbf{label5}^{\tau_4}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label8}^{\tau_5}(\mathbf{X},\mathbf{Y}))$	$([\tau_4,\tau_5])$
$P^{2}_{4,5} = \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}), \mathit{label}3(X,Y)$	$([\tau_4,\tau_5])$

and τ_2 ($\Psi' : \{P^{(c),1}_{1,2},\ P^{(c),2}_{1,2},\ P^{(c),3}_{1,2}\},\ \Psi''{=}\oslash$). The algorithm proceeds (in the next time-periods) by evaluating first the stable patterns (lines 3-14) and then the change patterns (lines 15-30). The stable patterns which are not used to extend existing chains will be discarded when considering next time-periods, while the unused change patterns will be used for further analysis.

At the first iteration (h=3), the stable pattern $P^{3}_{2,3}$ is considered and is used to extend the chain

$$C_1 = \langle P^{(c),2}_{1,2} \rangle = \langle \mathit{network}(N), \mathit{predicate}1(N,X), \mathit{is\_a}(X,\mathit{user}), \mathit{predicate}1(N,Y), \mathit{is\_a}(Y,\mathit{user}),$$

$$(\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label3}^{\tau_2}(\mathbf{X},\mathbf{Y})) \qquad [\tau_1,\tau_2]\rangle,$$

so that the following (incomplete) chain is generated:

$$C_2 = \langle network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$$
$$(\mathbf{label1^{\tau_1}(X,Y)} \rightarrow \mathbf{label3^{\tau_2}(X,Y)}); \qquad [\tau_1, \tau_2]$$
$$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user), label3(X,Y)$$
$$[\tau_2, \tau_3]\ \rangle.$$

This is possible because the number of stable patterns already inserted into C_2 (i.e. $C_2.sCounter$) is less than the threshold $maxS$ (line 6).

The chain C_1 is removed from Ψ', while the chain C_2 is inserted into Ψ''_{new} (line 9). The analysis continues with the change patterns $P^{(c),1}_{2,3}$, $P^{(c),2}_{2,3}$ and $P^{(c),4}_{2,3}$. For each of these, we consider the patterns related to τ_2 (P', line 17) and check the equality with the pattens related to τ_2 of the chains remaining in Ψ' (L, line 20). The candidate chains are those obtained by combining $P^{(c),1}_{2,3}$ or $P^{(c),2}_{2,3}$ with the chain $C_3 = \langle P^{(c),1}_{1,2} \rangle$ (line 22). From these two alternatives, the algorithm prefers the one with the highest γ (Algorithm 4).

```
Data: $\Psi_{temp}$: set of change chains with candidate change patterns
Result: $\Psi_{final}$: set of change chains containing the extended chains
1 for $Temp \in \Psi_{temp}$ do
2     $selected\_P^{(c)} := \underset{candidate \,\in\, Temp.candidates}{\text{argmax}} \; get\_\gamma(candidate)$;
3     $insert(\Psi_{end}, join(Temp, selected\_P^{(c)}))$;
4     $C.nCounter := C.nCounter + 1$;
```

Algorithm 4: select_change_patterns

For this reason, the following chain is mined:

$C_4 = \langle network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}));$ $[\tau_1,\tau_2]$

$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$

$(\mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label5}^{\tau_3}(\mathbf{X},\mathbf{Y}))$ $[\tau_2,\tau_3]\ \rangle$

At the end of the iteration (h=3) we have in Ψ' the chain composed of $P_{1,2}^{(c),3}$ only, which, since it has not been extended and does not fulfill the $minP$ constraint, is discarded. In the next iteration (h=4), the set Ψ' is composed of C_4 and of chains built with the remaining change patterns of $\Upsilon_{2,3}$, namely $C_5 = \langle P_{2,3}^{(c),2}\rangle$ and $C_6 = \langle P_{2,3}^{(c),4}\rangle$ (lines 32-33). Among the stable patterns $P_{3,4}^2$ and $P_{3,4}^3$, we select the $P_{3,4}^3$ for the extension of the chain C_2 in Ψ'' (lines 12-14), so that the following (incomplete) chain is built:

$C_7 = \langle network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label3}^{\tau_2}(\mathbf{X},\mathbf{Y}));$ $[\tau_1,\tau_2]$
$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user), label3(X,Y);$
$[\tau_2,\tau_3]$

$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user), label3(X,Y)$

$[\tau_3,\tau_4]\ \rangle.$

The stable pattern $P_{3,4}^2$ is used instead to extend the chain C_4 into C_8 (lines 12-14):

$C_8 = \langle network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}));$ $[\tau_1,\tau_2]$
$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label5}^{\tau_3}(\mathbf{X},\mathbf{Y}));$ $[\tau_2,\tau_3]$
$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user), label5(X,Y)$
$[\tau_3,\tau_4]\ \rangle.$

Both C_7 and C_8 are stored in Ψ''_{new} (line 14). When considering the change patterns, it is possible to extend $C_5 = \langle P^{(c),2}_{2,3} \rangle$ with $P^{(c),1}_{3,4}$ (lines 20-22), so that the following chain is obtained:

$C_9 = \langle\ network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label6}^{\tau_3}(\mathbf{X},\mathbf{Y}));$ $[\tau_2,\tau_3]$
$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label6}^{\tau_3}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label7}^{\tau_4}(\mathbf{X},\mathbf{Y}))\ \rangle$ $[\tau_3,\tau_4]$.

Now, we have C_9 in Ψ' (C_6 remains unused and therefore discarded), while C_7 and C_8 are in Ψ'' (lines 32-33). At the last iteration (h=5), C_7 cannot be extended with $P^2_{4,5}$, since the number of stable patterns in C_7 ($C_7.sCounter$=2) reaches the maximum threshold $maxS$ (line 11). The only operation we can complete is the extension of the chain C_8 with $P^{(c),1}_{4,5}$, which generates the chain C_{10} (lines 19-22):

$C_{10} = \langle network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label1}^{\tau_1}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}));$ $[\tau_1,\tau_2]$
$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$
$(\mathbf{label2}^{\tau_2}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label5}^{\tau_3}(\mathbf{X},\mathbf{Y}));$ $[\tau_2,\tau_3]$
$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user), label5(X,Y);$
$[\tau_3,\tau_4]$

$network(N), predicate1(N,X), is\_a(X,user), predicate1(N,Y), is\_a(Y,user),$

$(\mathbf{label5}^{\tau_4}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{label8}^{\tau_5}(\mathbf{X},\mathbf{Y}))$ $[\tau_4,\tau_5]\ \rangle$.

Finally, the set Ψ is composed of the chains $\{C_7, C_9, C_{10}\}$: the chain C_7 is removed in the light of Definition 4, while C_9 and C_{10} are returned, since they meet both Definition 4 and the threshold constraints (line 34).

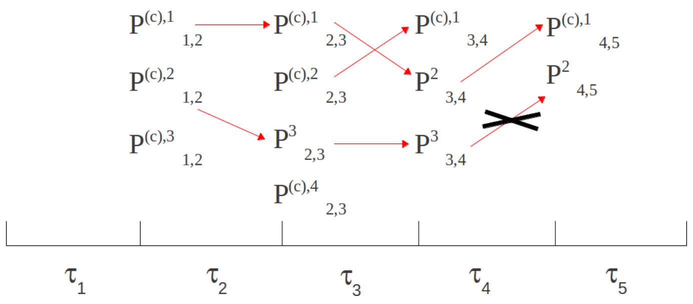


Figure 5: The algorithm joins the stable and change patterns to the chains created in the previous time-periods.

4.4. Time complexity

The time complexity of the whole algorithm depends on the computational complexity of SPADA. The complexity of SPADA leads to the notorious trade-off between expressiveness and efficiency in first order representations. Indeed, it is well known that a simple matching of two expressions with commutative and associative operators (such as the logical OR of atoms in a clause) is NP-complete. Therefore, any known algorithm that checks the coverage of an atom set or that equivalently evaluates a query with respect to a relational database has an exponential complexity. Nevertheless, queries with up to k atoms, where each atom contains at most j terms, can be evaluated in polynomial time [31]. This is the case of our algorithm, where j is limited by k.

Denoting as l_1 the time complexity of SPADA (necessary to generate each set of patterns for each time-period $\mathcal{P}_h$), we can define the complexity of the whole algorithm. For simplicity, we assume that $l_2 = \mathcal{P}_h = \mathcal{P}_{h+1}$ (with $h = 1, \ldots, n-1$). In this case, the worst case complexity is $O(n \times l_1 + n \times l_2^2)$

where $O(n \times l_2^2)$ is the time complexity of the generation of the chains, which is quadratic in the number of the average number of patters extracted for each time-period.

5. Experiments

In order to prove the viability of the proposed approach, we performed experiments on four real-world datasets with different characteristics in terms of size of the network and number of observations. Experiments aim at qualitatively and quantitatively evaluating chains and change patterns extracted by the proposed approach. In particular, we report some interesting chains and study the influence of parameters on the obtained results. We also report a scalability analysis and compare our approach with GERM [18], which, as stated before, extracts "Graph Evolution Rules", which can be directly compared with the chains we extract.

In the following subsections, we first present the datasets and the evaluation measures considered and then present the obtained results.

5.1. Dataset Description

The first dataset (*KEDS*) concerns the geographic-social-political network derived from the news reports[5] and collects data on the social and political relationships among nations and world-wide organizations. As in [32], we consider this dataset as a network where nations and world-wide organizations represent the nodes and social and political relationships correspond to the edges between nodes (structural predicates). Nations and

[5] `http://web.ku.edu/~keds/data.html`

world-wide organizations are the non-target objects and the networks at single time-points represent target objects. In KEDS, the set $\mathcal{N}$ contains 228 nodes and there are 20 different labels of the edges (in D^I), which are listed in the following:

make_public_statement, *disapprove*, *appeal*, *express_intent_to_cooperate*, *consult*, *engage_diplomatic_cooperation*, *engage_material_cooperation*, *provide_aid*, *yield*, *investigate*, *demand*, *reject*, *threaten*, *protest*, *exhibit_military_posture*, *reduce_relations*, *coerce*, *assault*, *fight*, *attack_with_weapons_of_mass_destruction*.

The dissimilarity in D^I between the labels of the edges is set as their pairwise semantic distance computed by means of the linguistic tool presented in [33]. The networks are collected day by day from April 1979 to July 2009, therefore the time-points are in the format year/month/day (one time-point represents one day). We have on average 12.82 edges per time-point. For this dataset, we defined on the individual objects the hierarchy represented in Figure 6.

The second dataset (*DAYS*) collects all stories released by the news agency Reuters concerning the September 11 attack on the U.S.[6]. As in [34], we consider this dataset as a network. In our case, the nodes of the network denote the relevant terms in the news, while the edges denote the discretized frequency with which the two connected terms co-occur in the same sentence of the text. In DAYS, the set $\mathcal{N}$ contains 13332 nodes. The edges used in

[6]`http://vlado.fmf.uni-lj.si/pub/networks/data/CRA/terror.htm`

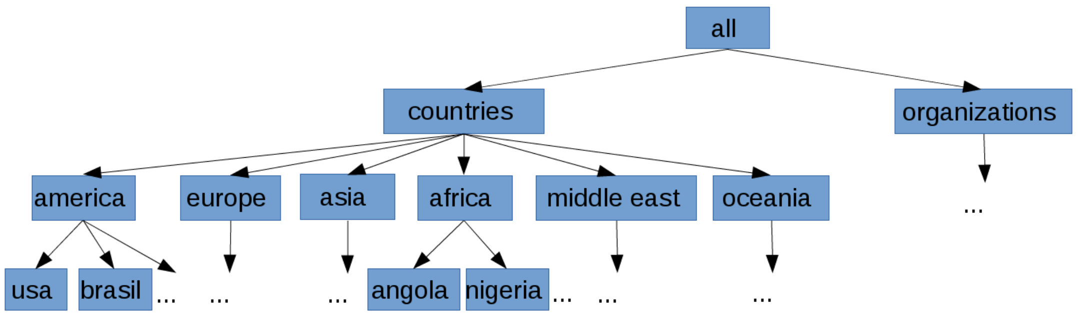


Figure 6: The hierarchy defined on the examples represented in the nodes of the dataset KEDS.

the D^I are 4 and result from the application of an equal-frequency discretization technique to the values of frequency of the co-occurrence of two nodes: each label represents one of the four quartiles (low_range, $middle\_low\_range$, $middle\_high\_range$, $high\_range$). The dissimilarity values are defined as follows: labels of consecutive quartiles (e.g., $middle\_high\_range$-$high\_range$) have a dissimilarity of 0.25, labels of quartiles at distance 2 have a dissimilarity of 0.5 (e.g., $middle\_low\_range - high\_range$), finally, labels of quartiles at distance 3 have a dissimilarity of 0.75. In this way, it is possible to study the evolution of the co-occurrence of terms. The networks are collected day-by-day from September 11th 2001 for 66 days, therefore the time-points are in the format month/day (one time-point represents one day). We have 28.59 edges per time-point, on average.

The third dataset (*INFECTIOUS*) contains the dynamic contact networks collected during the Infectious SocioPatterns event that took place at the Science Gallery in Dublin, Ireland, during an art-science exhibition[7]. As in [35] the nodes represent visitors to the Science Gallery while the edges represent the close-range of face-to-face contact between visitors. In particular, in our case, edges represent discretized duration (in seconds) of contacts. In INFECTIOUS, the set $\mathcal{N}$ contains 28 nodes. The edges used in the D^I are 10 and result from the application of an equal-frequency discretization technique to the duration of the contact (in seconds) associated to two nodes: each label represents one of ten ranges of seconds returned by the discretization. The dissimilarity values are defined as follows: labels of consecutive ranges have a dissimilarity of 0.1, labels of ranges at distance 2 have a dissimilarity of 0.2 and so on. The networks are collected day-by-day from April 28th 2009 to July 16th 2009, therefore the time-points are in the format month/day (one time-point represents one day). We have 9.5 edges per time-point, on average.

The last dataset (*DBLP*) refers to the collaboration network based on the co-authorship of scientific papers in computer science stored in the DBLP bibliographic database. Originally, it contains co-authorship entries collected with yearly time granularity from January 1988 to September 2013 (one time-point represents one year). From this original dataset, we discarded papers with only one author. Moreover, we concentrated only on the one hundred more productive (in the number of published papers) authors which are supposed to be the "influencers" of the network. As in [36], nodes represent authors and edges represent co-authorships. There are two edge types which

[7]`http://www.sociopatterns.org/datasets/`

represent the co-authorship: co-authorship in conference papers and co-authorship in journal papers. Edges are labeled on the basis of the number of co-authored papers (i.e. 1=low,2=medium or 3 or more= high). In practice, given two authors a_1 and a_2, one of the following predicates for conference paper co-authorship can be used to connect them: $conference\_low(a_1, a_2)$, $conference\_medium(a_1, a_2)$, $conference\_high(a_1, a_2)$ and one of the following predicates for journal paper co-authorship can be used to connect them: $journal\_low(a_1, a_2)$, $journal\_medium(a_1, a_2)$, $journal\_high(a_1, a_2)$.

The dissimilarity values are defined as follows:
$journal\_low\ \underline{0.2}\ journal\_medium$; $journal\_medium\ \underline{0.2}\ journal\_high$; $journal\_low\ \underline{0.4}\ journal\_high$; $conference\_low\ \underline{0.2}\ conference\_medium$; $conference\_medium\ \underline{0.2}\ conference\_high$; $conference\_low\ \underline{0.4}\ conference\_high$; $journal\_low\ \underline{0.6}\ conference\_low$; $journal\_low\ \underline{0.8}\ conference\_medium$; $journal\_low\ \underline{1.0}\ conference\_high$; $journal\_medium\ \underline{0.6}\ conference\_medium$; $journal\_medium\ \underline{0.8}\ conference\_high$; $journal\_high\ \underline{0.2}\ conference\_low$; $journal\_high\ \underline{0.4}\ conference\_medium$; $journal\_high\ \underline{0.6}\ conference\_high$.

In this way, it is possible to study the evolution of the collaborations in the publishing activity. In the final dataset, we have 93.2 edges per time-point, on average.

5.2. Evaluation measures

As previously mentioned, the first experiment is performed to test the influence of the input parameters on the final change chains and to study the characteristics of the extracted chains. In this case, we manually tune the minimum threshold of support $minSup$ and the minimum dissimilarity value between the labels $min\Gamma$ and we collect the results in terms of the statistics listed in Table 2. In addition, we associate to each chain two quantitative parameters: the value of the average change of the chains ($avg\ chains\ \gamma$), which corresponds to the average of the dissimilarity values used to mine change patterns of the final chains, and the value of the average support of the chains (*avg supp*), defined as follows:

Table 2: Collected statistics

Value	Description
times	Running times
#chains	Number of discovered chains
#joins	Total number of change and stable patterns in the final chains
avg length	Average number of change patterns involved in the final chains
avg cp	Average number of mined change patterns, including those not used in the chains
avg periods γ	Average dissimilarity values between labels observed in the change patterns. It is computed as the mean of the dissimilarity values of the change/stable patterns mined in all the time-periods

Consider the change chain $C = \langle P^{(c)}_{h,h+1}; P_{h+1,h+2}; \ldots ; P_{h+q-2,h+q-1}; P^{(c)}_{h+q-1,h+q}\rangle$ and let *supp*$(C, h+i)$, $i = 0, .., q$ be the support defined as follows:

$$supp(C, h+i) = \begin{cases} supp_{h+i}(getLastPattern(\langle P_{h+i-1,h+i}\rangle)) & \text{if } i = 1, \ldots, q \\ supp_h(getFirstPattern(\langle P_{h,h+1}\rangle)) & \text{if } i = 0 \end{cases} \quad (2)$$

then:

$$avgsupp = \frac{\sum_{i=0}^{q} supp(C, h+i)}{\sum_{i=0}^{q} |\tau_{h+i}|} \tag{3}$$

which, intuitively, is the microaverage relative support of the frequent patterns used in C, computed on the respective time-periods.

5.3. Results: Influence of the parameters

The results reported in this section clarify the effect of $minSup$ and $min\Gamma$, which we consider to be the parameters which significantly influence the obtained results. In order not to introduce a bias on the temporal discretization and to report results which are not affected by w, we report the average values computed on three different values of w for each dataset. More precisely, the time-periods span 3, 6 and 12 months ($w = \{90, 180, 360\}$) for KEDS; 7, 10 and 15 days ($w = \{7, 10, 15\}$) for DAYS; 10, 15 and 20 days ($w = \{10, 15, 20\}$) for INFECTIOUS; 4, 5 and 6 years ($w = \{4, 5, 6\}$) for DBLP.

The results reported in Figures 7, 8, 9 and 10 show that when the threshold $minSup$ increases, the values of $times$, $\#chains$ and $\#joins$ decrease. Indeed, as expected, high values of support lead to the generation of a small set of frequent patterns and the reduction of the running times ($times$). Consequently, we have a small set of stable and change patterns (generated from the frequent ones, see *avg cp* in Figures 7b, 8b, 9b) and 10b) and, therefore, a smaller number of change and stable patterns that can be added to the chains ($\#joins$), thus reducing the number of final chains ($\#chains$) (see Figures 7a, 8a, 9a) and 10a). The overall decrease of the change patterns also implies the reduction of the length of the chains (*avg length*).

Another observation is inspired by the influence of *minSup* on the values of *avg periods* γ and *avg chains* γ, which allow us to quantify the change of the network. As we can see, when increasing *minSup* the changes captured from the change patterns (*avg periods* γ) and from the final chains (*avg chains* γ) tend to be milder, meaning that the strongest changes are not particularly frequent. Moreover, we notice that the values of *avg periods* γ are lower than those of *avg chains* γ (especially in Figures 7b and 9b), although they exhibit the same behaviour. This can be explained by the observation that *avg periods* γ considers the changes in all the time-periods, while *avg chains* γ represents the changes only in the time-periods considered in the chains. The use of the change patterns with relatively high values of γ in the process of change pattern discovery allows us to highlight the capability of the chains to represent significant changes.

Also, it is noteworthy that the different characteristics of the datasets determine different responses of the algorithms: the results obtained on DAYS seem to be less influenced by the variation of *minSup* than the results obtained with KEDS, INFECTIOUS and DBLP. Indeed, although DAYS present a higher number of data per time-period (28.59 edges per time-point), the number of labels is lower than that of other datasets. This results in a restricted variability of the edges, which gives a limited dynamicity over time. Another observation can be done on the number of change patterns and change chains. In particular, the results obtained from DBLP present smaller sets of change patterns and change chains than those obtained with KEDS, DAYS and INFECTIOUS. This can be motivated by the relatively small number of networks (at most 6) collected in each time-period which ((a))

Figure 7: Results produced from KEDS when tuning *minSup* (*min*Γ=0.2, *minP*=2, *maxP*=8, *maxS*=5).

((a))

Figure 8: Results produced from DAYS when tuning *minSup* (*min*Γ=0.5, *minP*=2, *maxP*=7, *maxS*=4).

makes it difficult the identification of changes and, consequently, the discovery of frequent evolutions. Finally, the tendency of *avg supp* to increase as the threshold increases is obvious.

Figures 11, 12, 13 and 14 show the effect of *min*Γ on the obtained results. High values of *min*Γ lead to change patterns with high values of γ, that is, they concentrate the search only on the frequent patterns with very high dissimilarity. As a consequence, we have less and shorter chains (see *#chains* and *avg length*, respectively). This explains also the reduction of the values of *#joins*, *#avg cp* and *times*. Finally, as expected, the higher the *min*Γ, the higher the *avg chains* γ and *avg periods* γ (we underline that the charts reported in the figures use the logarithmic scale). However, for very high

((a))

Figure 9: Results produced from INFECTIOUS when tuning *minSup* (*min*Γ=0.1, *minP*=2, *maxP*=5, *maxS*=4).

(a)

Figure 10: Results produced from DBLP when tuning $minSup$ ($min\Gamma$=0.2, $minP$=2, $maxP$=4, $maxS$=1).

(a)

Figure 11: Results produced from KEDS when tuning $min\Gamma$ ($minSup$=0.07, $minP$=2, $maxP$=8, $maxS$=5).

(a)

Figure 12: Results produced from DAYS when tuning $min\Gamma$ ($minSup$=0.8, $minP$=2, $maxP$=7, $maxS$=4).

(a)

Figure 13: Results produced from INFECTIOUS when tuning $min\Gamma$ ($minSup$=0.05, $minP$=2, $maxP$=5, $maxS$=4).

(a)

Figure 14: Results produced from DBLP when tuning $min\Gamma$ ($minSup$=0.7, $minP$=2, $maxP$=4, $maxS$=1).

values of $min\Gamma$, the algorithm is not able to extract change patterns (this is the case of DAYS).

5.4. Results: Scalability

Specific experiments are performed in order to test the computational properties of the approach. In particular, the scalability is empirically evaluated by increasing the width (namely, the number of included time-points) of the time-periods and by increasing the number of the time-periods along which the chains are discovered.

In Figure 15, we show the scalability on the whole dataset KEDS. Obviously, the higher the width w, the lower the *total* number of time-periods. The first observation is that, as expected, the running time exponentially increases when decreasing w. This is due to the linear increase of the number of edges per time-period, which produces an exponential increase of $\#joins$. However, setting $w \leq 10$ leads to a huge amount of frequent patterns (and therefore a huge amount of chains) which do not have a significant statistical motivation. On the contrary, a small number of change patterns leads to the reduction of the join operations ($\#joins$), to the reduction of change patterns used in the chains (*avg length*) and therefore to the reduction of the number of chains ($\#chains$). It is noteworthy that for $w \geq 25$ there are no significant variations in the changes detected in the network (*avg period* γ, *avg chains* γ).

Figure 16, which reports results obtained from a subset of KEDS (1995-2009), shows that the computational cost linearly grows with the number of periods, while the overall number of chains ($\#chains$) is quite constant (except in [10,15]). This is not unexpected since, although new change patterns are created (*avg cp* remains quite identical), they seem to be unsuitable for the extension of chains (*#joins* is quite constant when the number of periods increases). This reduces the possibility to discover new chains or extend those already generated, and consequently motivates the behaviour of *avg length* and of the values of *avg chains* γ and *avg periods* γ.

5.5. Qualitative evaluation

In this subsection we report some examples of change chains extracted by our approach. We report also the change γ captured by each change pattern included in the chains.

For instance, the following change chain[8] has been extracted from the DAYS dataset (with *minSup*=0.7, *min*Γ=0.2, *w*=7 days):

$$\begin{aligned}
&\langle P^{(c),1} = network(N), is(X, afghanistan), is(Y, attack),\\
&\quad (\mathbf{high\_range}^{\mathbf{Sep.11-Sep.17,2001}}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{low\_range}^{\mathbf{Sep.18-Sep.24,2001}}(\mathbf{X},\mathbf{Y}))\\
&\qquad\qquad ([Sep.11 - Sep.17, 2001, Sep.18 - Sep.24, 2001], \gamma = 0.5)\\
&P^{2} = network(N), is(X, afghanistan), is(Y, attack), low\_range(X, Y))\\
&\qquad\qquad ([Sep.18 - Sep.24, 2001, Sep.25 - Oct.01, 2001])\\
&P^{(c),3} = network(N), is(X, afghanistan), is(Y, attack),\\
&\quad (\mathbf{low\_range}^{\mathbf{Sep.25-Oct.01,2001}}(\mathbf{X},\mathbf{Y}) \rightarrow \mathbf{middle\_low\_range}^{\mathbf{Oct.02-Oct.08,2001}}(\mathbf{X},\mathbf{Y}))\\
&\qquad\qquad ([Sep.25 - Oct.01, 2001, Oct.02 - Oct.08, 2001], \gamma = 0.25)\ \rangle
\end{aligned}$$

This chain shows the evolution of the frequency of the co-occurrence of

[8]For the sake of simplicity, in the description of the pattern we omit some predicates which help to link the variables, namely *term_occurring_in*/2, *nation_in*/2, *author_present_in*/2, where the first argument denotes the network (e.g., N), while the second argument denotes a term (e.g. X).

the terms “attack” and “afghanistan” in the Reuters news in the period [September 11th, 2001 - October 8th, 2001]. As it is possible to see while in the first week the co-occurence of the terms is very high, in the following two weeks the number of news with both terms significantly decreases. In the fourth week, the frequency increases again.

The following change chain has been extracted from the DBLP dataset (with $minSup$=0.75, $min\Gamma$=0.4, w=4 years):

$$\langle P^{(c),1} = network(N), author(X, lastnameA\_firstnameA), author(Y, lastnameB\_firstnameB),$$
$$(\mathbf{conference\_high^{2001-2005}(X,Y) \rightarrow conference\_low^{2005-2008}(X,Y)})$$
$$([2001-2005, 2005-2008], \gamma = 0.4)$$

$$P^{(c),2} = network(N), author(X, lastnameA\_firstnameA), author(Y, lastnameB\_firstnameB),$$
$$(\mathbf{conference\_low^{2005-2008}(X,Y) \rightarrow journal\_medium^{2008-2012}(X,Y)})$$
$$([2005-2008, 2008-2012], \gamma = 0.8)\ \rangle$$

This chain describes the evolution of the collaboration between the authors $lastnameA\_firstnameA$ and $lastnameB\_firstnameB$ (authors have been anonymized for privacy reasons). This collaboration moves from a large number of co-authored conference papers to a small number of co-authored conference papers and, subsequently, to a medium number of co-authored journal papers.

The arrangement of the nodes in a hierarchy (as in Figure 6) allows us to discover change chains with nodes collocated at different levels of granularity and which, therefore, express information at different levels of specialization. For instance, the following change chains have been extracted from the dataset KEDS with the hierarchy drawn in Figure 6 ($minSup$=0.05, $min\Gamma$=0.2, w=180 days):

$\langle P^{(c),1} = network(N), is(X, africa), is(Y, america),$

$(\mathbf{consult^{June\_2008}(X, Y) \rightarrow express\_intent\_to\_cooperate^{December\_2008}(X, Y)})$

$([June\_2008, December\_2008], \gamma = 0.287)$

$P^{(c),2} = network(N), is(X, africa), is(Y, america),$

$(\mathbf{express\_intent\_to\_cooperate^{December\_2008}(X, Y) \rightarrow make\_public\_statement^{June\_2009}(X, Y)})$

$([December\_2008, June\_2009], \gamma = 0.287)$

$(supp = 0.0519)\ \rangle$

while at the second level of the hierarchy, we have

$\langle P^{(c),1} = network(N), is(X, angola), is(Y, usa),$

$(\mathbf{consult^{June\_2008}(X, Y) \rightarrow express\_intent\_to\_cooperate^{December\_2008}(X, Y)})$

$([June\_2008, December\_2008], \gamma = 0.287)$

$P^{(c),2} = network(N), is(X, angola), is(Y, usa),$

$(\mathbf{express\_intent\_to\_cooperate^{December\_2008}(X, Y) \rightarrow make\_public\_statement^{June\_2009}(X, Y)})$

$([December\_2008, June\_2009], \gamma = 0.287)$

$(supp = 0.0519)\ \rangle$

These chains describe the same evolution expressed by the sequence of relationships *consult*, *express_intent_to_cooperate*, and then *make_public_statement*. In particular, in the first chain, the evolution holds on two objects identified as $africa$ and $america$, while the second chain provides a more specific information and holds on two objects identified as $angola$ and usa, which are descendants of $africa$ and $america$ respectively. Also, it is noteworthy that in this particular case both chains have the same frequency (*supp*), which means that the evolution modeled by the two chains is not replicated by other nodes different from $angola$ and usa, but it is the result of the particular behavior of the nodes $angola$ and usa in the time-periods $[June\_2008, December\_2008]$ and $[December\_2008, June\_2009]$.

5.6. Comparative evaluation

A comparative evaluation was performed between the proposed approach and the system GERM [18]. As introduced in Section 2, GERM discovers patterns (evolution rules) able to characterize the more frequent evolutions of the network over time. In particular, a pattern reflects the same evolution in its multiple occurrences. The first difference, with respect to our approach, is the representation of the data which, in GERM, tends to over-simplify the network. Indeed, the network is modeled as a cumulative graph, where the nodes and the edges can be only added and never deleted. The consequence of this is a partial analysis of the evolution, which considers as topological changes only insertions and disregards deletions. Moreover, in GERM two nodes can be connected by only one edge labelled with the time-point in which the edge first appears. This allows the system to neither model the variety of the relationships which can exist in the real-world networks nor consider the cases in which two nodes can be connected by more than one edge at the same time.

In Figure 17, we report the results of the comparison. In the case of our approach, the reported values are averages of the results obtained with two different widths, $w = 90$ and $w = 180$ (3 and 6 months). In the case of GERM, the data associated to each time-point are obtained by collecting the edges observed in the periods of 3 and 6 months. In this way, it is possible to guarantee a fair comparison between the two approaches. Experiments were performed by tuning the threshold *minSup*, which is the input parameter common to both algorithms. In Figure 17, we can see that our approach outperforms GERM in terms of running times for all values of *minSup*. In particular, for our approach, the time consumption significantly decreases when *minSup* increases from 0.05 to 0.13, while for GERM it remains unchanged since we set to 24 hours of uninterrupted execution the maximum running time for the experiments. This behaviour can be explained with an algorithmic difference of the two approaches. GERM operates directly on the cumulative graph from which it mines frequent sub-graphs that express the evolutions. On the contrary, our approach does not extract changes directly from the network data, but it works on the set of discovered frequent patterns.

In Figure 17, we can also notice the difference in the number of discovered patterns: the set of *#evolution rules* is several orders of magnitude larger than the set of *#chains*. This is due to the different modeling of the network. Indeed, GERM uses a cumulative graph in which only insertions are counted, since edge removals are not allowed. This means that the algorithm has to take into account the existence of a higher number of nodes and edges, thus resulting in a larger set of frequent sub-graphs. Therefore, it is more difficult for sub-graphs to model changes due to removal operations. Instead, we work on networks observed by time-periods, where nodes and edges existing in one period could disappear in the other, hence the change chains can model both insertions and deletions equally well.

In Figure 17(a), we compare the two algorithms on the length of the chains (our approach) and evolution rules (GERM), and on the average support associated to them. The length corresponds to the number of time-periods covered by the chains (our approach) and to the number of time-points covered by the evolution rules (GERM). We have to consider that the evaluation on GERM is relative to the evolution rules discovered in the 24 hours of execution. The different behaviour can be motivated, basically, by the different way of representing the change. In our approach the change from one time-period to the next is determined by the edges, while in GERM the change is due to some insertion in the network which, in a cumulative graph, facilitates the generation of longer evolutions. Moreover, while in our approach a change in a chain is associated to two consecutive time-periods, in GERM the same change (modeled by an evolution rule) can be associated to different consecutive pairs of time-points, which increases the absolute frequency of the rule, thus resulting in longer evolutions. Finally, as expected, by increasing the threshold *minSup* we observe a higher average support of both chains and evolution rules.

GERM has anyway the advantage of not necessarily relying on a background knowledge which is manually defined by the users. Indeed, in our case, the evolutions modelled in the change chains are generated thanks to the availability of domain information which quantifies the pairwise dissimilarity of the labels of the edges. Without such background knowledge changes could not be captured, even if the network evolves. Obviously, background knowledge can also be profitably used by domain experts in order to adequately configure the system and extract useful and actionable knowledge.

6. Conclusions

In this paper we have investigated the task of discovering changes in evolving networks and we have proposed a novel method for the discovery ((a))

Figure 15: Scalability on KEDS when tuning w ($min\Gamma$=0.1, $minP$=2, $maxP$=8, $maxS$=5, $minSup$=0.2).

((a))

Figure 16: Scalability on KEDS when the number of time-periods increases ($min\Gamma$=0.1, $minP$=2, $maxP$=8, $maxS$=5, $minSup$=0.2, w=6).

of relational patterns which characterize such changes. The method is motivated by real-world scenarios, such as social networks, where the evolution of a network mainly involves the type of interaction between the nodes. It traces the evolution of the network as a succession of states (time-periods) of the network and discovers statistically evident changes which occur in the form of variations at the level of edges. It operates in three steps. Initially, frequent patterns are discovered at consecutive time-periods. Then, change patterns are generated from the frequent patterns. Finally, change chains are generated by combining incrementally change patterns. This computational solution permits to separate the identification of the states of the network from the discovery of statistically evident changes. Hence, tuning parameters used to filter either change patterns or change chains requires only re-running

((a))

Figure 17: Comparison with the system GERM

the second and third steps, which are the less computationally demanding.

We evaluated our method on an set of real-world networks, characterized by different heterogeneities and different sizes, coming from the areas of social, political, multi-media and collaboration networks. Empirical results allowed us to draw some conclusions on the computational features of the proposed method.

As to the influence of the input parameters, the results show the influence of the minimum support threshold on the number of frequent patterns, on the number of change patterns and on the computational performances. On the contrary, the minimum dissimilarity threshold seems to affect only the number of change patterns with no consequence on the computational performances.

Scalability has been evaluated with respect to the number of time-periods and to the width of the time-periods. The results empirically show that the running times grow linearly in the number of time-periods and grow exponentially in the width of the time-periods. This suggests careful tuning of the width of the time-periods. Indeed, a small width may lead to a higher number of time-periods and thus may help discovering evolutions (chains) at a small temporal granularity, without incurring in high computational costs.

Comparative experiments have highlighted the efficiency of the proposed method with respect to another state-of-the-art method without loss in statistical evidence of the patterns. Also, they provide an empirical proof of two basic choices of our proposal: the use of the relational setting to handle heterogeneity and complexity, and the analysis of the evolving data with an approach based on an abstract and summarized description (patterns) of the data.

For future work, we plan to extend our proposal in five directions: *i)* automatic determination of the optimal widths of the time-periods on the basis of the underlying distribution of the data, *ii)* use of solutions of big data analytics to discover approximate frequent pattern sets [30], *iii)* extraction of change chains from biomedical literature in order to identify terminological/topic evolutions in research papers [37], *iv)* application to biological data in order to understand the evolution of relations between biological entities, and *v)* discovery of time series of patterns in order to model regularities in ongoing processes.

Acknowledgements

The authors would like to acknowledge the support of the European Commission through the project MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944). The authors would like also to thank Lynn Rudd for proof reading the article.

References

[1] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000, pp. 649–690.

[2] Y. Sun, J. Han, Mining heterogeneous information networks: a structural analysis approach, SIGKDD Explorations 14 (2012) 20–28.

[3] T. Falkowski, J. Bartelheimer, M. Spiliopoulou, Mining and visualizing the evolution of subgroups in social networks, in: Proc. of the 2006

IEEE/WIC/ACM Int. Conf. on Web Intelligence, WI '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 52–58.

[4] S. Džeroski, N. Lavrač, Relational Data Mining, Springer-Verlag, 2001.

[5] S. Muggleton (Ed.), Inductive Logic Programming, volume 38 of *The APIC Series*, Academic Press, 1992.

[6] C. Aggarwal, H. Wang, Graph data management and mining: A survey of algorithms and applications, in: C. C. Aggarwal, H. Wang (Eds.), Managing and Mining Graph Data, volume 40 of *Advances in Database Systems*, Springer US, 2010, pp. 13–68.

[7] C. C. Aggarwal, P. S. Yu, Online analysis of community evolution in data streams, in: Proc. of the 2005 SIAM Int. Conf. on Data Mining, SDM 2005, 2005.

[8] J. Sun, C. Faloutsos, S. Papadimitriou, P. S. Yu, Graphscope: parameter-free mining of large time-evolving graphs, in: Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD '07, ACM, New York, NY, USA, 2007, pp. 687–696.

[9] J. Sun, D. Tao, C. Faloutsos, Beyond streams and graphs: dynamic tensor analysis, in: Proc. of the 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mmining, KDD '06, ACM, New York, NY, USA, 2006, pp. 374–383.

[10] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, C. Faloutsos, Colibri: fast mining of large static and dynamic graphs, in: Proc. of the 14th ACM

SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD '08, ACM, New York, NY, USA, 2008, pp. 686–694.

[11] J. Ferlez, C. Faloutsos, J. Leskovec, D. Mladenic, M. Grobelnik, Monitoring network evolution using mdl, in: Proc. of the 2008 IEEE 24th Int. Conf. on Data Engineering, ICDE 2008, IEEE Computer Society, Washington, DC, USA, 2008, pp. 1328–1330.

[12] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, D. Pedreschi, Evolving networks: Eras and turning points, Intelligent Data Analysis 17 (2013) 27–48.

[13] P. K. Desikan, J. Srivastava, Mining temporally changing web usage graphs, in: B. Mobasher, O. Nasraoui, B. Liu, B. M. Masand (Eds.), Advances in Web Mining and Web Usage Analysis, 6th International Workshop on Knowledge Discovery on the Web, WebKDD 2004, volume 3932 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 1–17.

[14] R. Ahmed, G. Karypis, Algorithms for mining the evolution of conserved relational states in dynamic networks, Knowledge Information Systems 33 (2012) 603–630.

[15] C. Loglisci, M. Ceci, D. Malerba, Discovering evolution chains in dynamic networks, in: A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, Z. W. Ras (Eds.), NFMCP, volume 7765 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 185–199.

[16] M. Ceci, A. Appice, D. Malerba, Discovering emerging patterns in spatial databases: A multi-relational approach, in: Proc. of the 11th Eu-

ropean Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2007, volume 4702 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 390–397.

[17] A. Inokuchi, T. Washio, A fast method to mine frequent subsequences from graph sequence data, in: Proc. of the 2008 Eighth IEEE Int. Conf. on Data Mining, ICDM 2008, IEEE Computer Society, Washington, DC, USA, 2008, pp. 303–312.

[18] M. Berlingerio, F. Bonchi, B. Bringmann, A. Gionis, Mining graph evolution rules, in: Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I, ECML PKDD '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 115–130.

[19] Z. Liu, J. X. Yu, Y. Ke, X. Lin, L. C. 0002, Spotting significant changing subgraphs in evolving graphs, in: Proc. of the 2008 Eighth IEEE Int. Conf. on Data Mining, ICDM 2008, IEEE Computer Society, 2008, pp. 917–922.

[20] K. M. Borgwardt, H.-P. Kriegel, P. Wackersreuther, Pattern mining in frequent dynamic subgraphs, in: Proc. of the 6th Int. Conf. on Data Mining, ICDM '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 818–822.

[21] M. Lahiri, T. Y. Berger-Wolf, Mining periodic behavior in dynamic social networks, in: Proc. of the 2008 Eighth IEEE Int. Conf. on Data Mining, ICDM 2008, IEEE Computer Society, Washington, DC, USA, 2008, pp. 373–382.

[22] C. h. You, L. B. Holder, D. J. Cook, Learning patterns in the dynamics of biological networks, in: Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD '09, ACM, New York, NY, USA, 2009, pp. 977–986.

[23] A. Prado, B. Jeudy, É. Fromont, F. Diot, Mining spatiotemporal patterns in dynamic plane graphs, Intelligent Data Analysis 17 (2013) 71–92.

[24] S. Ceri, G. Gottlob, L. Tanca, Logic Programming and Databases, Springer, 1990.

[25] J. W. Lloyd, Foundations of Logic Programming, second ed., Springer-Verlag, Berlin, 1987.

[26] A. Appice, M. Ceci, A. Lanza, F. A. Lisi, D. Malerba, Discovery of spatial association rules in geo-referenced census data: A relational mining approach, IDA 7 (2003) 541–566.

[27] F. A. Lisi, D. Malerba, Inducing multi-level association rules from multiple relations, Machine Learning 55 (2004) 175–210.

[28] G. D. Plotkin, A note on inductive generalization, Machine Intelligence 5 (1970) 153–163.

[29] A. Appice, M. Berardi, M. Ceci, D. Malerba, Mining and filtering multi-level spatial association rules with ARES, in: M.-S. Hacid, N. V. Murray, Z. W. Ras, S. Tsumoto (Eds.), Foundations of Intelligent Systems, 15th International Symposium, ISMIS 2005, volume 3488 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 342–353.

[30] A. Appice, M. Ceci, A. Turi, D. Malerba, A parallel, distributed algorithm for relational frequent pattern discovery from very large data sets, Intelligent Data Analysis 15 (2011) 69–88.

[31] L. De Raedt, S. Džeroski, First-order jk-clausal theories are pac-learnable, Artificial Intelligence 70 (1994) 375–392.

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

[33] V. Veksler, A. Grintsvayg, R. Lindsey, W. Gray, A proxy for all your semantic needs, in: Proc. of the 29th Annual Meeting of the Cognitive Science Society, CogSci 2007, 2007.

[34] V. Batagelj, A. Mrvar, Density based approaches to network analysis, Analysis of Reuters Terror News Network. University of Ljubljana (2001).

[35] L. Gauvin, A. Panisson, C. Cattuto, A. Barrat, Activity clocks: spreading dynamics on temporal networks of human contact, Scientific Reports 3 (2013) 3099.

[36] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, Arnetminer: extraction and mining of academic social networks, in: Y. Li, B. Liu, S. Sarawagi (Eds.), Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, ACM, 2008, pp. 990–998.

[37] C. Loglisci, M. Ceci, Discovering temporal bisociations for linking concepts over time, in: D. Gunopulos, T. Hofmann, D. Malerba, M. Vazirgiannis (Eds.), ECML/PKDD (2), volume 6912 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 358–373.