COPYRIGHT NOTICE

DOI: `10.1007/978-3-319-46307-0_23`.

Anomaly Detection in Networks with Temporal Information*

Fabrizio Angiulli, Fabio Fassetti, and Estela Narvaez

DIMES, University of Calabria, Italy
email: {f.angiulli,f.fassetti,e.narvaez}@dimes.unical.it

Abstract. We present a technique for node anomaly detection in networks where arcs are annotated with time of creation. The technique aims at singling out anomalies by taking simultaneously into account information concerning both the structure of the network and the order in which connections have been established. The latter information is obtained by timestamps associated with arcs. A set of temporal structures is induced by checking certain conditions on the order of arc appearance denoting different kinds of user behaviors. The distribution of these structures is computed for each node and used to detect anomalies. The anomaly score measures the deviation from the expected number of structures associated with each node on the basis of the correlation between nodes degree and numerousness of exhibited structures. The resulting algorithm has low computational cost and is applicable to large networks. We present experimental results on some real-life networks showing the reliability of the approach.

1 Introduction

The large use of social networks supplies a huge amount of data which provides much information about individuals and individual behaviors reflecting human relationship in the real world. Such behaviors can be model as relational structures among the actors of the social network.

Among the interesting hidden knowledge that can be mined by analyzing node behaviors, a relevant role is played by the anomaly discovery, where the aim is to find those individuals that can be considered as outliers, since they assume exceptional behaviors. The problem of finding malicious nodes in networks is of interest in many areas such as fake account detection, spammer node detection, ddos attacks in computer networks, and many others. Much work has been made to detect anomalous nodes mostly based on detecting anomalous structures around the individual [1].

However, in many scenarios, the exceptional behavior of an individual has not to be searched only in the structural composition of its neighborhood but the exceptional behavior is characterized by the temporal sequence of connection establishments. Thus, taking into account the time dimension sheds interesting lights on individuals' behaviors. As such, our approach is orthogonal to the works aimed at mining structural properties of large static networks.

* This research has been partially supported by the PRIN project 20122F87B2 titled "Compositional Approaches for the Characterization and Mining of Omics Data" co-financeby the Italian Ministry of Education, University and Research.

Consider, for example, the individuals registered to the Facebook social network and arcs between them defined as follows: if a marks b as a friend there is an arc from a to b and, vice versa, there is an arc from b to a if b marks a as a friend. Thus, the arc from a to b represents that either a sends a Facebook request to b or a accepts a Facebook request coming from b.

Consider, now, an individual a with five hundreds friends then with five hundreds other individuals there is a connection from a and a connection towards a. Clearly, it is not anomalous since such a number of friends is not so exceptional. But, if a is always the first to send Facebook friend requests and all the five hundreds just accept this request (and, then, for any b friends of a the arc from a to b always precedes the arc from b to a), a becomes a clear outlier.

The problem tackled with in the rest of the paper can be defined as follows:

Anomaly detection in timed networks. *Given a timed network, that is a network where each arc is equipped with a timestamp denoting the time of creation of the corresponding link, find the nodes in the network that are considerably dissimilar with respect to the rest of the network nodes when both the structure of their neighborhood and the order in which the structure has been established are taken into account.*

Different approaches have been proposed in the literature that search for anomalies in dynamic networks, among them [6, 4, 3, 5, 2].

We point out that the approach here proposed is substantially different from techniques dealing with dynamic networks. Indeed, our aim is not to determine the points in time in which a certain portion of the networks (typically a community or a subgraph) exhibited a significant change, as usually done by dynamic-graph anomaly detection techniques. Rather, our primary aim is to analyze each single node by taking simultaneously into account its temporal footprint.

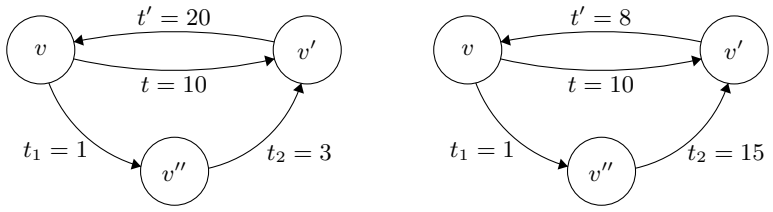
In this sense our approach can be regarded as a static-graph anomaly detection technique in which temporal information has a privileged role in characterizing the behavior of network nodes.

The rest of the work is organized as follows. Section 2 reports preliminary notions and describe how the individual behavior is modeled; subsequent Section 3 illustrates the specific behavior models we retrieve to detect outliers; Section 4 is devoted to discuss the outlier score and its properties; Section 5 presents the several phases of the mining algorithm; Section 6 depicts the experiments we conduct on real datasets; finally, Section 7 concludes the work.

2 Behaviors on timed networks

In this section, we report the preliminary definitions and the notations employed throughout the paper. We aim at modeling the behavior of a node in the network through the way the node has interacted with its neighborhood during the time. Thus, first of all we introduce the model of network equipped with time information tackled by the proposed technique.

Definition 1 (Timed Network). *A* timed network *(or, simply,* network*) is a triple* $\mathcal{N} = (V, E, \tau)$*, where* $V = \{v_1, \ldots, v_n\}$ *is a set of* nodes*,* $E = \{e_1, \ldots, e_m\}$ *is a set of* arcs*,*



(a) (b)

Fig. 1: Example

with each $e_i = \langle s_i, d_i \rangle$ an ordered pair of nodes in V, and τ a function associating each arc $\langle s, d \rangle$ in E with a timestamp representing the instant of time in which the connection from s to d is established.

Moreover, given a node v, we refer to the set of nodes v' such that there is an arc from v to v' as the set of *outgoing neighbors* of v (or, simply, neighbors) and we denote it as $\overrightarrow{N}(v)$. Vice versa, the set of nodes v' such that there is an arc from v' to v is referred to as the set of *ingoing neighbors* of v and we denote it as $\overleftarrow{N}(v)$. Finally, the total number of outgoing and ingoing neighbors is denoted as $\deg(v)$, then $\deg(v) = |\overrightarrow{N}(v)| + |\overleftarrow{N}(v)|$.

Next, we provide formal definition of *contact* and *awareness* between nodes that are exploited for modeling interactions.

Given a network $\mathcal{N} = (V, E, \tau)$ and two nodes v and v'in V, we say that v *contacts* v' at time t if $\langle v, v' \rangle \in E$ and $\tau(\langle v, v' \rangle) = t$. Also, in this case, we say that v' is a contact of v starting from the instant of time t.

For example, in Figure 1a, v contacts v' at time $t = 10$ and, hence, starting from that time, v' is a contact of v.

An *interaction* between two nodes v and v' is fired (or established) at time t if either v contacts v' at time t or v' contacts v at time t. In such a case, the established contact is said to be the *contact associated with the interaction*.

Given an interaction i between two nodes v and v', the inverse interaction of i is the establishment of the contact inverse with respect to the contact associated with i.

Thus, in Figure 1a and 1b, there is an interaction between v and v' fired at time $t = 10$ having as associated contact the arc from v to v' and, then, the inverse interaction between v' and v is fired at time $t = 20$, having as associated contact the arc from v' to v.

Next we provide the definition of *awareness* which intends to model the intuition that an individual knows another individual, which is not one of its contacts, due to the presence of a common friend.

Definition 2 (Awareness). *Given a network $\mathcal{N} = (V, E, \tau)$ and two nodes v and v' in V, we say that the node v is* mediately aware *(or, simply, aware) of v' at time t_a if there exists a node v'' in the network $\mathcal{N}$, such that v contacts v'' at time t_1 , v'' contacts v'*

at time t_2, $\max\{t_1, t_2\} \leq t_a$ *and (i) either* $\langle v, v' \rangle$ *is not in* E *or (ii)* $t_a \leq \tau(\langle v, v' \rangle)$. *Moreover, we call* intermediary *the node* v'' *responsible of the awareness.*

Note that, according to our definition, v is no more aware of v' once v' becomes a contact of v, since with the awareness we want to model the mediated knowledge between individuals.

In Figure 1a, v is aware of v' at each instant of time in the range $[3, 9]$. Starting from the instant of time $t = 10$, v' becomes a contact of v and, then, v is no more aware of v'. Conversely, in Figure 1b, v is never aware of v' and, starting from the instant of time $t = 10$, v' becomes a contact of v.

The notions of *contacts* and *awareness* are next exploited to model the behavior of a node within its neighborhood and, in particular, we distinguish between two families of behaviors: *action behavior* and *reaction behavior*.

Definition 3 (Action behavior). *Let* $\mathcal{N}$ *be a network, an* action *is an interaction between two nodes* v *and* v' *of the network fired before that the inverse interaction is fired.*

Let v be a node of a network $\mathcal{N}$ and let v' be one of its neighbor. According to the above definition, the action behaviors involving v can be both the establishing of a connection from v to v' *preceding* a possibly connection from v' to v and the establishing of a connection from v' to v *preceding* a possibly connection from v to v'.

Consider Figure 1a. There are two actions involving v: (i) the contact from v to v' which is accomplished before that v' contacts v and (ii) the contact from v to v''. Consider, now, Figure 1b. There are again two actions involving v: (i) the contact from v' to v which is accomplished before that v contacts v' and (ii) the contact from v to v''.

Definition 4 (Reaction behavior). *Let* $\mathcal{N}$ *be a network, a* reaction *is an interaction between two nodes* v *and* v' *of the network fired after that the inverse interaction is fired.*

The reaction behaviors involving v are both the establishing of a connection from v to v' *succeeding* a connection from v' to v and the establishing of a connection from v' to v *succeeding* a connection from v to v'.

Consider Figure 1a. There is one reaction involving v: the contact from v' to v which is accomplished after that v' contacts v. Consider, now, Figure 1b. There is again one reaction involving v: the contact from v' to v which is accomplished after that v contacts v'.

After having defined the concepts of actions and reactions, we can distinguish among different kinds of actions and reactions on the basis of the properties holding at the instant of time in which they are performed.

For example, Figures 1a and 1b depict two different kinds of actions:

i. the node (v) is aware of the other node (v') when performs the action of contacting it (Figure 1a);

ii. the node (v') is not aware of the other node (v) when performs the action of contacting it (Figure 1b).

In the following Section 3 we will describe in details which kinds of actions and reactions are addressed in this work.

The technique we propose aims at detecting outliers on the basis of their behavior taking simultaneously into account and suitably combining actions and reactions. Specifically, each action–reaction couple models a different *scenario* and we will denote it with the expression $A \leftrightarrow R$, where A is the action and R the reaction.

For example, consider the Twitter social network and consider the scenario in which an individual v starts to follow another individual v' and v' does not follow back v. There, we can individuate an action performed by v and received by v' and a reaction performed by v' (the decision of not following back v) and received by v.

For each scenario there are several involved performers: there is the performer who makes the action, the performer who receives the action, the performer who makes the reaction, the performer who receives the reaction and, in some cases, the performer involved as intermediary. Thus, on each scenario a node can play different roles. We call *structure* the coupling of role and scenario. Each structure defines a precise role played on a precise scenario and is referred to a single node called *actor* of the structure.

The previous example induces, then, four structures:

- s_1: node making action (who decides on its own initiative to follow another node);
- s_2: node receiving action (who is followed by another node on its own initiative);
- s_3: node making reaction (who decides to do not follow back a node by which it was followed);
- s_4: node receiving reaction (who is not followed back by a followed node).

For structures s_1 and s_4 the actor is v while for the structures s_2 and s_3 the actor is v'.

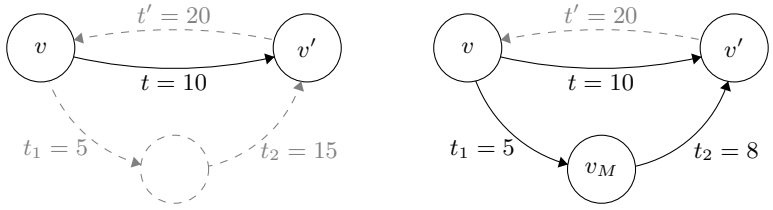
Hence, once actions and reactions have been defined, those draw several scenarios and relative roles played. Scenarios and associated roles induce a set of $\mathcal{S}$ structures which encodes the node behavior. In particular, evaluating how frequently a node v plays each possible role on each scenario (then, how frequent v is the actor of each structure) leads to the building of the vector $\phi(v) = (\phi_{s_1}(v), \ldots, \phi_{s_k}(v))$ which represents the distribution of the roles played by the node on the different scenarios and $\phi_{s_i}(v)$ represents how frequently v is the actor of the structure s_i. This distribution semantically encodes the *behavior* of the node in the network and can be effectively exploited to find anomalous individuals, as detailed in the Section 4.

3 Modeled behaviors

This section is devoted to present the behaviors considered. In particular, we present the kinds of actions and reactions we capture to gather information for modeling the overall node behavior. However, the approach is easily extensible to cover other kinds of actions/reactions.

Given a node v and one of its neighbor v', next we present the actions taken into account to model the behavior of v and start by summarizing the notation employed:

- t the instant of time $\tau(\langle v, v' \rangle)$ associated with $\langle v, v' \rangle$;
- t' the instant of time $\tau(\langle v', v \rangle)$ if $\langle v', v \rangle \in E$; if $\langle v', v \rangle$ is not in E then t' is set to -1, meaning that it is not defined;



(a) Action 1 (b) Action 2

Fig. 2: Action behaviors

t_M the greatest instant of time smaller than t such that v is aware of v' at time t_M due to the intermediary v'', and we refer as v_M the node v''; if v was not aware of v' when the connection from v and v' has been established then t_M is set to -1, meaning that it is not defined.

For the sake of readability, we employ $\texttt{def}(t)$ for indicating that $t \neq -1$ and $\texttt{undef}(t)$ for indicating that $t = -1$.

For each considered action/reaction, we discuss the semantic behavior associated with it together with the conditions to be checked in order to verify if the behavior under analysis is actually assumed.

(Action 1) **a node contacts another node on its own initiative**.
This action represents that v contacts v' before that v' contacts v and without v being aware of v' (see Fig. 2a).

Condition: $(\texttt{undef}(t') \vee t \leq t') \bigwedge \texttt{undef}(t_M)$

(Action 2) **a node contacts another node due to an intermediary**.
This structure means that v contacts v' before that v' contacts v but after that v becomes aware of v' (see Fig. 2b).

Condition: $(\texttt{undef}(t') \vee t \leq t') \bigwedge \texttt{def}(t_M) \bigwedge t_M < t$

As for the reactions, we capture four kinds of reactions for v. Note that, since these are reactions, we assume that either a connection from v to v' or a connection from v' to v has already been fired.

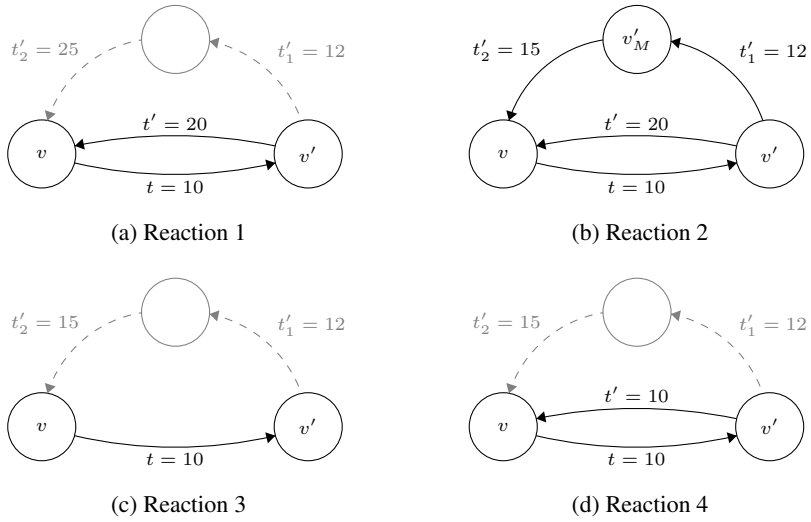
(Reaction 1) **a node directly replies to the node who contacts him**:
This reaction models that v' contacts v since v contacts v' and not because v' becomes aware of v (see Fig. 3a).

Condition: $\texttt{def}(t) \bigwedge \texttt{def}(t') \bigwedge t < t' \bigwedge (\texttt{undef}(t'_M) \vee t'_M < t)$

(Reaction 2) **a node replies to the node who contacts him due to an intermediary**:
This reaction represents that v' contacts v after that v contacts v' but only after that v' becomes aware of v (see Fig. 3b).

Condition: $\texttt{def}(t) \bigwedge \texttt{def}(t') \bigwedge \texttt{def}(t'_M) \bigwedge t < t'_M < t'$

(Reaction 3) **a node does not reply to the node who contacts him**:
This is not an actual reaction since it represents that v' does not react to



(a) Reaction 1

(b) Reaction 2

(c) Reaction 3

(d) Reaction 4

Fig. 3: Reaction behaviors

the action performed by v (see Fig. 3c).

Condition: $\texttt{def}(t) \bigwedge \texttt{undef}(t')$

(Reaction 4) **a node contacts the node who contacts him independently**:
This is not an actual reaction since it represents that v' contacts v on its own initiative (see Fig. 3d).

Condition: $\texttt{def}(t) \bigwedge \texttt{def}(t') \bigwedge t' = t \bigwedge \texttt{undef}(t'_M)$

Once actions and reactions are defined, we can analyze which scenarios are modeled and which structures are induced. In particular, we have eight different scenarios and for each scenario two roles are definable, the node who acts and the node who reacts. Moreover, for scenarios involving action A_2 and/or reaction R_2, also the role of intermediary is definable.

Thus, focusing on a single node v, we can define seventeen structures for it:

structures $s_1 \dots s_4$: v plays the role of performing action A_1 and receiving one of the possible four reactions;

structures $s_5 \dots s_8$: v plays the role of performing action A_2 and receiving one of the possible four reactions;

structures $s_9 \dots s_{12}$: v plays the role of receiving action A_1 and performing one of the possible four reactions;

structures $s_{13} \dots s_{16}$: v plays the role of receiving action A_2 and performing one of the possible four reactions;

structure s_{17}: v plays the role of being the intermediary of a couple of interacting nodes.

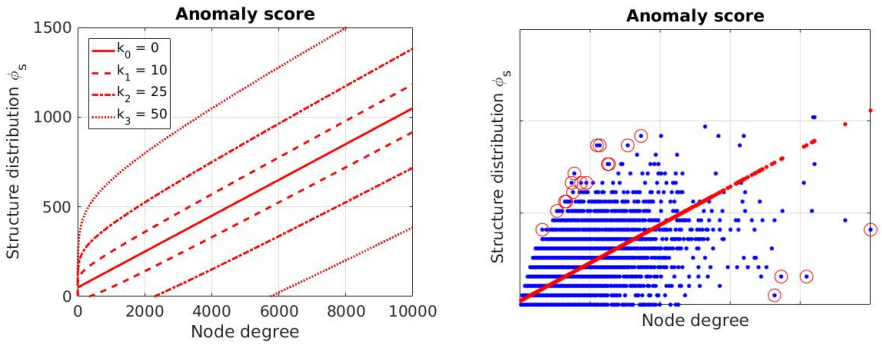


Fig. 4: Example of bandwidths associated with the score of a structure (left) and example highlighting the top twenty anomalies on a real dataset (right).

For example, consider the scenario depicted in Figure 1a again: *v contacts v' after viewing that v' is a contact of v'' which is one of its contact. v' reacts to this contacting back v*. The scenario is then $A_2 \leftrightarrow R_1$ and the roles are: (1) who makes the action A_2 and receives the action R_1, (2) who receives the action A_2 and makes the reaction R_1, and (3) who plays the role of intermediary. Thus, after analyzing this scenario, for the node v we have to update the structure associated with (1), that is s_5; for the node v' the structure associated with (2), that is s_{13}; and for the node v'' the structure associated with (3), that is s_{17}.

4 Anomaly Score

The distribution $\phi_s(v)$ encodes the behavior of the node v in terms of how much frequently it is involved in the different structures. We can then exploit the distributions ϕ_s in order to determine how typical is the behavior of each node with respect to the whole population. With this aim, an anomaly score is assigned to each node.

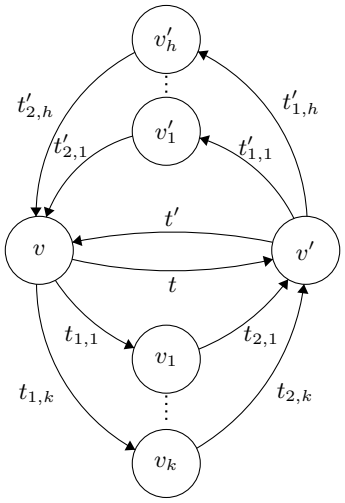
Given the network $\mathcal{N} = (V, E, \tau)$, for each structure $s \in \mathcal{S}$ the regression line of the set of points $P_s(\mathcal{N}) = \{(\deg(v_i), \phi_s(v_i)) \mid v_i \in V\}$ is computed.

Let α_s and β_s denote be the parameters of the estimated line. The anomaly score of the node v_i with respect the structure s is defined as:

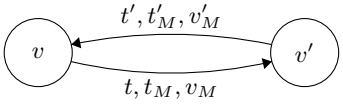
$$\mathrm{sc}_s(v_i) = \left| \frac{\phi_s(v_i) - [\alpha_s \cdot \deg(v_i) + \beta_s]}{\log_2 (1 + \deg(v_i))} \right| \tag{1}$$

The numerator of Equation (1) represents the deviation of the observed number of structures $\phi_s(v_i)$ from the expected one y_i, according to the value predicted by the regression curve $y_i = \alpha_s \cdot \deg(v_i) + \beta_s$. As for the denominator, it serves the purpose of taking into account the cardinality of the neighborhood of v_i, while the absolute value is needed to capture both the upper tail and the lower tail of resulting distribution.

Figure 4 on the left reports a regression line (the solid curve for $k_0 = 0$, having parameters $\alpha = 0.1$ and $\beta = 50$) and the bandwidths associated with different values of anomaly score (specifically, for $k_1 = 10$, $k_2 = 25$, and $k_3 = 50$; e.g., the score associated with points falling within the dashed bandwidth will be not greater than k_1).



(a) analyzed pattern



where $t_M = \max\{t_i = \max\{t_{1,i}, t_{2,i}\} \mid t_i < t\}$
$v_M = v_i$ s. t. $\max\{t_{1,i}, t_{2,i}\} = t_M$
$t'_M = \max\{t'_i = \max\{t'_{1,i}, t'_{2,i}\} \mid t'_i < t'\}$
$v'_M = v'_i$ s. t. $\max\{t'_{1,i}, t'_{2,i}\} = t'_M$

(b) annotated graph

Fig. 5: Graph annotation (Phase 1)

Figure 4 on the right reports the structure distribution associated with a real dataset and the top twenty anomalies according to Equation (1).

Scores associated with each single structure are then normalized in order to make them homogeneous

$$\widehat{\mathrm{sc}}_s(v_i) = \frac{\mathrm{sc}_s(v_i)}{\mathrm{std}(\{\mathrm{sc}_s(\mathbf{v})\})} \tag{2}$$

and, hence, expressed in terms of number of standard deviations.

The anomaly score of a node v_i is

$$\mathrm{sc}(x_i) = \sum_s \mathrm{sc}_s(x_i), \tag{3}$$

that is obtained by combining the scores computed with respect to the single structures.

5 Algorithm

In this section the algorithm we designed to mine outliers is presented and its properties are discussed. The algorithm consists in three main phases each accounted next.

Phase 1. This phase has the intent of enriching the information associated with the arcs (see Figure 5). In order to retrieve behaviors illustrated in Section 3 we need to find, for each arc $\langle v, v' \rangle$ with associated timestamp t, if v is aware of v' at time t, namely we have to search for a node $\hat{v}$ such that both edge $e_1 = \langle v, \hat{v} \rangle$ and edge $e_2 = \langle v, \hat{v} \rangle$ exist and the timestamps t_1 and t_2 associated with these edges are both strictly smaller than t. Among these nodes, we are interested in the node v_M which is the most recent

responsible of the fact that v is aware of v'. Finally, the edge $e = \langle v, v' \rangle$ is annotated with the node v_M^e and the time t_M^e which is the instant of time starting by which v is aware of v' due to v_M^e; in formula t_M^e is the maximum between the time associated with the arc $\langle v, v_M^e \rangle$ and the time associated with the arc $\langle v_M^e, v' \rangle$. Concluding, given a network $\mathcal{N} = (V, E, \tau)$, the phase returns the annotated network $\mathcal{N}^+ = (V, E, \tau^+)$ where $\tau^+(e)$ returns the triple $(\tau(e), t_M^e, v_M^e)$.

Computational complexity of Phase 1. As for the cost of this phase, let $\mathcal{N} = (V, E, \tau)$ be the analyzed network, let $n = |V|$ and let $m = |E|$. We iterate over the set of edges and for each arc $e = \langle v, v' \rangle$ in E we iterate over the set $\overrightarrow{N}(v)$ of neighbors of v in order to search v_M and, then, for each neighbor $\hat{v}$ of v we have to check if there exists an arc from it to v'. This latter operation can be performed through a binary search in the list of the outgoing arcs of $\hat{v}$. Thus, the overall cost is

$$\sum_{\langle v, v' \rangle} \sum_{\hat{v} \in \overrightarrow{N}(v)} \log \overrightarrow{N}(\hat{v}) = O(m \cdot \overline{n} \cdot \log \overline{n}) \tag{4}$$

where $\overline{n}$ denotes the mean number of neighbors of nodes in the networks.

Phase 1: Network information enrichment

Input: A network $\mathcal{N} = (V, E, \tau)$
Output: The annotated network $\mathcal{N}^+$

foreach edge $e = \langle v, v' \rangle$ in E **do**
  let $t = \tau(e)$ be the timestamp associated with the edge from v to v';
  set t_M to -1;
  set v_M to $\emptyset$;
  foreach edge $\hat{e} = \langle v, \hat{v} \rangle$ **do**
    let $t_1 = \tau(\hat{e})$ be the timestamp associated with the edge from v to $\hat{v}$;
    if $\langle \hat{v}, v' \rangle$ belongs to E **then**
      let $t_2 = \tau(\langle \hat{v}, v' \rangle)$ be the timestamp associated with the edge from $\hat{v}$ to v';
      let $\hat{t}$ be $\max\{t_1, t_2\}$;
      if $t_M < \hat{t} < t$ **then**
        set t_M to $\hat{t}$;
        set v_M to $\hat{v}$;
  associate t_M and v_M with the e;
  // then substitute $\tau(e) = t$ with $\tau^+(e) = (t, t_M, v_M)$– see Fig. 5

Phase 2. This phase has the intent of mining the behavior of each individual in the network, starting from the annotated network coming from the previous phase. Then, given an annotated network $\mathcal{N}^+ = (V, E, \tau^+)$ we iterate over the set of nodes V and for each node v in V we iterate over the set of neighbors and for each neighbor v' in $\overrightarrow{N}(v)$ the behaviors depicted in Section 3 are evaluated. In particular, through the information provided by $\mathcal{N}^+$ the conditions associated with the behaviors are checked

and, according to the result of the check, the behavior counters are updated. The result of this phase is then the distribution of the behaviors for each node.

Computational complexity of Phase 2. As for the cost of this phase, let $\mathcal{N}^+ = (V, E, \tau^+)$ be the analyzed network, let $n = |V|$ and let $m = |E|$. Iterating over the set of nodes and for each node v iterating over the set of neighbors $\overrightarrow{N}(v)$ corresponds to iterating over the set of edges. For each edge, in constant time we can obtain the required information by $\mathcal{N}^+$ and we can evaluate all the conditions. Since the number of conditions is fixed, also this latter step can be accomplished in constant time. Thus, the overall cost of this phase is $O(m)$.

Phase 2: Structures computation

Input: An annotated network $\mathcal{N}^+ = (V, E, \tau^+)$
Output: The distribution of structures ϕ_v for each node v

```
1  foreach node v in V do
2      set φ_s(v) to 0 for each structure s;
3      foreach neighbor v' of v do
4          extract tuple T = (τ+(⟨v, v'⟩), τ+(⟨v', v⟩));
           // T contains then (t, t_M, v_M, t', t'_M, v'_M)– see Fig. 5b
5          foreach action a do
6              foreach reaction r compatible with a do
7                  let s be the structure associated with the pair a ↔ r;
8                  if C_a(T) and C_r(T) then
9                      update φ_s(v);
10                 if a ↔ r involves node v_M then
11                     update φ_ŝ(v_M);
```

Phase 3. This phase has the intent of detecting outlier individuals. Starting from the distribution of behaviors computed by the previous phase, we have to compute the outlier score as defined in Section 4. The first step consists in computing, for each considered structure, the regression line. The second step consists in computing for each structure s and for each node v the score $\mathrm{sc}_s(v)$ achieved by node v on the structure s by means of Equation (1). Next the standard deviation of the scores assumed by nodes on structure s is computed and, then, this value is exploited to normalize the scores (lines 6–7).

After that all the structures have been analyzed, the outlier score of each node v is computed by properly aggregating the score achieved by v on each single structure by means of Equation (3).

Computational complexity of Phase 3. As for the cost of this phase, let $\mathcal{N}^+ = (V, E, \tau^+)$ be the analyzed network, let $n = |V|$ and let $m = |E|$. Computing the regression line has a cost linear with respect to the number n of nodes. Next, for each node we had to compute the score. Since Equation (1) is computable in constant time, also this step has a cost linearly dependent from n. Normalizing the scores costs $O(n)$ as well and, finally, also computing the overall outlier score costs $O(n)$ since Equation (3) iterates over a fixed number of structures.

Phase 3: Outlier mining

Input: The distribution of structures ϕ_v for each node v
Output: The overall outlier score for each node v

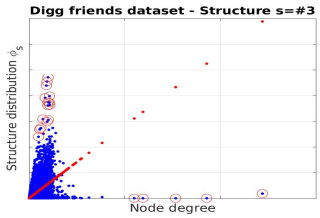
foreach structure s **do**
    compute the regression line of the observations $(\deg(\mathbf{v}), \phi_{\mathbf{s}}(\mathbf{v}))$;
    foreach node v in $\mathcal{N}$ **do**
        compute the score $\mathrm{sc}_s(v)$ of v for structure s through Eq. (1);
    compute the standard deviation of the score $\mathrm{std}(\{\mathrm{sc}_s(\mathbf{v})\})$;
    foreach node v in $\mathcal{N}$ **do**
        compute the normalized score of v for structure s through Eq. (2);
foreach node v in $\mathcal{N}$ **do**
    compute the overall score of v through Eq. (3);

6 Experimental results

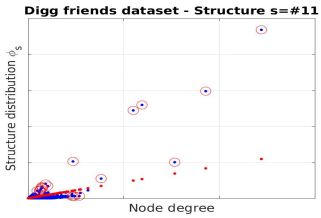
In this section experimental results concerning the introduced technique are presented. All the datasets employed are from the *Online Social Networks Research*. All the dataset underwent a preprocessing during which multiple and self links have been removed. The *Digg friends* dataset[1] contains data about stories promoted to Digg's front page[2] over a period of a month in 2009. The dataset contains Digg users who have voted for a story. We considered the voters' friendship links, where a link `user_id` $\rightarrow$ `friend_id` means that `user_id` is watching the activities of (is a fan of) `friend_id`. User identifiers are available already anonymized. The network analyzed consists of $279,631$ nodes and $2,251,166$ arcs. The *Facebook wall* dataset[3] contains a list of all of the wall posts from the Facebook New Orleans network. Each line contains two anonymized user identifiers, meaning the second user posted on the first user's wall. The third column is the times of the wall post. The network analyzed consists of $45,813$ nodes and $264,004$ arcs. The *Wikipedia growth*[4] dataset contains links between Wikipedia pages and the time when these links were first created. The dataset represents the complete history of the network over a period of 826 days, between January 1st, 2005 and April 6th, 2007. The datasets is anonymized to protect the privacy of page authors. The network analyzed consists of $1,870,709$ nodes and $39,953,145$ arcs.

The following table reports the total number of the structures mined, for each of the structures s_i in the set $\mathcal{S}$. Notice that the total counts associated with the pairs of structures (s_i, s_{i+8}) are identical, since these structures are induced by symmetric roles in the same scenario.

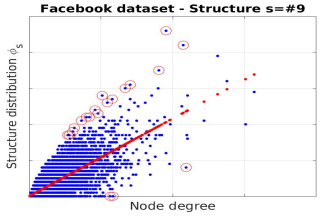
[1] `http://www.isi.edu/integration/people/lerman/downloads.html`
[2] *Digg* is a news aggregator (`http://digg.com`) aiming to select stories for the Internet audience such as science, trending political issues, and viral Internet issues. It allows people to vote web content up or down.
[3] `http://socialnetworks.mpi-sws.org/data-wosn2009.html`
[4] `http://socialnetworks.mpi-sws.org/data-wosn2008.html`



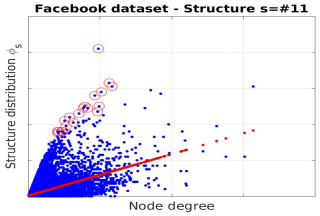
(a) Digg dataset – s_3



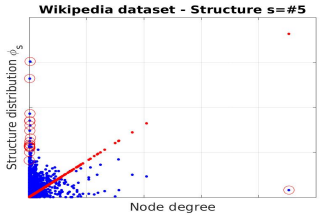
(b) Digg dataset – s_{11}



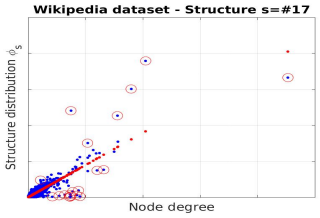
(c) Facebook dataset – s_9



(d) Facebook dataset – s_{11}



(e) Wikipedia growth dataset – s_5

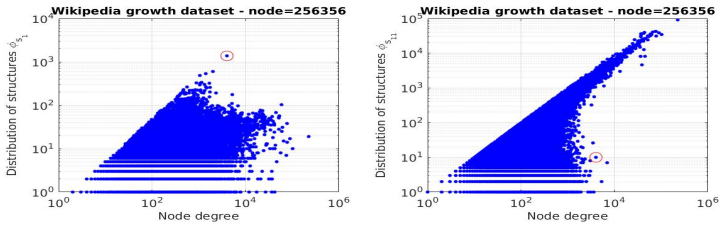


(f) Wikipedia growth dataset – s_{17}

Fig. 6: Structure count distribution.

Structure	*Digg friends*	*Facebook wall*	*Wikipedia growth*
s_1, s_9	$61,122$	$57,476$	$2,030,550$
s_2, s_{10}	$8,307$	$4,167$	$641,677$
s_3, s_{11}	$683,282$	$74,268$	$22,416,947$
s_4, s_{12}	$6,294$	0	$7,596$
s_5, s_{13}	$65,844$	$16,319$	$451,128$
s_6, s_{14}	$44,301$	$2,629$	$293,043$
s_7, s_{15}	$681,317$	$28,552$	$10,694,970$
s_8, s_{16}	152	0	56
s_{17}	$1,009,571$	$80,042$	$14,052,936$
Total	$7,574,125$	$446,864$	$87,124,870$

Clearly, this does not mean that they are redundant, since the respective counts per node differ in general, being different the number of times in which the single node plays each role in the same scenario. In the *Facebook wall* dataset the structures s_4, s_8, s_{12}, and s_{16}, capturing the simultaneity of the response, are not present since timestamps are almost all different (only 846 timestamps appear more than once in the dataset) and it is never the case that two users simultaneously make a post on the re-



(a) (b)

Fig. 7: Notable structure distributions for a top outlier node

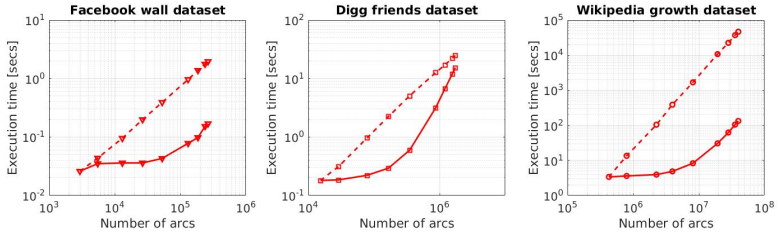


Fig. 8: Scalability analysis.

spective wall. In general, these structures are among the less numerous in these datasets due to the fine granularity of the temporal scale.

Figure 6 shows the scatter plots of the node degree versus the structure count for some of the structures mined. Each plot reports also the regression line of the points (represented by the red points) and highlights the top 20 anomalies (the red circled points) according to the anomaly score of Equation (1).

Next, we discuss on the knowledge mined for one of the top outlier in a dataset in order to highlight how the proposed technique is able to provide not only the outlier score but also an in intelligible interpretation of the score, shedding light on semantic underlying the decision made by the technique of signaling a node as anomalous.

Specifically, we focus on the *Wikipedia growth* dataset and on the node *out* having id $256,356$. We consider the two structures that mostly contribute to the large score achieved by *out*. Figure 7 reports how the number of times in which *out* is actor of structure s_1 and s_{11} places the node with respect to the number of times in which the other nodes perform as actors of those structures.

It is clear by the plots that *out* is located in both cases at the margin of the distribution. In particular, *out* performs as actor of structure s_1 much more times than other nodes, while performs as actor of structure s_{11} much less times than nodes having a similar neighborhood cardinality.

From a semantic point of view, these plots naturally lead to a description of the outlierness that could explain the exceptionality of the node. Since *out* very often plays as actor for structure s_1, the associated Wikipedia page has a high number of links and, exceptionally, almost always the linked page links back the page associated with *out*.

Moreover, since *out* rarely plays as actor for structure s_{11}, whenever *out* is linked by a page p, rarely a link to p does not appear in *out*.

Finally, Figure 8 shows the scalability of the method. We varied the size of the datasets, from the 1% to the 100% of the original data, by randomly sampling nodes and retaining the arcs linking only pairs of nodes in the selected sample. The solid lines in the plot show the total execution time versus the number of arcs of the network. The dashed lines represents the cost of the method in the average case, as reported in Equation (4), with a constant prefactor computed in a way such that the two curves start from the same point. The curves show that the actual cost of the method is generally below that predicted by the cost analysis, thus confirming the applicability of the method to large networks. To illustrate, the full *Wikipedia growth* dataset was processed in about 120 seconds on a Intel Core i7 2.40GHz equipped machine under the Linux operating system.

7 Conclusions

We considered the anomaly detection in timed networks problem whose goal is to single out anomalies by taking into account simultaneously information concerning both the structure of the network and the order in which connections have been established. Our primary aim is to analyzing each single node by taking simultaneously into account its temporal footprint. We defined a set of spatio-temporal structures is induced by checking certain conditions on the order of arc appearance denoting different kinds of user behaviors and exploited their distribution to detect anomalies. We presented a scalable algorithm and experimental results showing the peculiarity of the knowledge mined by our technique and its applicability to the analysis of large networks.

References

1. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Mining and Knowledge Discovery 29(3), 626–688 (2015), http://dx.doi.org/10.1007/s10618-014-0365-y
2. Chen, Z., Hendrix, W., Samatova, N.F.: Community-based anomaly detection in evolutionary networks. Journal of Intelligent Information Systems 39(1), 59–85 (2012)
3. Gupta, M., Gao, J., Sun, Y., Han, J.: Community trend outlier detection using soft temporal pattern mining. In: Machine Learning and Knowledge Discovery in Databases, pp. 692–708. Springer (2012)
4. Ji, T., Yang, D., Gao, J.: Incremental local evolutionary outlier detection for dynamic social networks. In: Machine Learning and Knowledge Discovery in Databases, pp. 1–15. Springer (2013)
5. Mongiovi, M., Bogdanov, P., Ranca, R., Singh, A.K., Papalexakis, E.E., Faloutsos, C.: Netspot: Spotting significant anomalous regions on dynamic networks. In: Proceedings of the 13th SIAM international conference on data mining (SDM), Texas-Austin, TX. SIAM (2013)
6. Wang, T., Fang, C.V., Lin, D., Wu, S.F.: Localizing temporal anomalies in large evolving graphs. In: Proceedings of the 2015 SIAM International Conference on Data Mining. pp. 927–935. SIAM