Snapshot Setting for Temporal Networks Analysis

Ahmed Ould Mohamed Moctar[1](✉), Idrissa Sarr[1], and Joel Vaumi Tanzouak[2]

[1] Department of Mathematics and Computer Science, Cheikh Anta Diop University, Dakar - Fann, BP 5005, Dakar, Senegal
{ahmed.ouldmoctar,idrissa.sarr}@ucad.edu.sn

[2] Department of Mathematics and Computer Science, University of Ngaoundere, BP 454, Ngaoundere, Cameroun
joel.tanzouak@univ-ndere.cm

Abstract. Temporal networks can be used to model systems that evolve over longer time scales such as networks of disease spread, for instance, HIV/AIDS disease that is propagated within the population over a relatively long period. Analyzing temporal networks can be done by considering the network either as a series of snapshots (aggregation over a time window) or as a dynamic object whose structure changes over time. The first approach is used in this paper and requires specifying a size of time window that delimits snapshot size. To our best knowledge, there is not yet studies on setting the size of the window in a methodical basis. In real, existing works rely on a static or a regular value of time window size to capture snapshots over time.

This work is conducted to identify dynamically snapshots over time in a directed and weighted network. That is, we aim to find out the right time to start and to end capturing a new snapshot. To this end, we define a quality function to evaluate the network state at anytime. Then, we rely on time series to predict the quality scores of the network over time. A significant changes of the network state is interpreted as the start and/or end of a snapshot. Our solution is implemented with R and we use a real dataset based on geographical proximity of individuals to demonstrate the effectiveness of our approach.

Keywords: Time window size · Temporal networks · Quality function

1 Introduction

Most of the networks such as the one of Facebook, LinkedIn and YouTube deal with hundreds of millions of users that interact hugely day in day out. The creation or the birth of these networks is a dynamic process in which the network evolves over time. Analyzing such a dynamic process unveils the intrinsic temporal aspects of the network. To analyze a temporal network, one may see the network as a static view in which all the links in the final network are present

G. Mendy et al. (Eds.): AFRICOMM 2018, LNICST 275, pp. 98–107, 2019.
https://doi.org/10.1007/978-3-030-16042-5_10

throughout the study. This is a very simplifying assumption for a network which is built instantly and does not evolve frequently over time. However, if the network evolves over longer time scales, it is worthwhile to take into account the fact that ties may be temporary, and the network structure can change at many points of time and have an impact on the final network status. For example, an individual with zero or very few contacts at a single time t may see his contacts growing significantly at time $t + 1$. Therefore, temporal analysis is challenging at many points such as how to track and/or represent changes over time, how to manage the time in order to report the right times within which the network undergoes through the main phases of its creation. To analyze the network evolution, one may consider either a series of network snapshots and assess whether changes occur between snapshots or track continuously the changes that occur over time. The first choice that we use in this work requires to define a time interval, called "time window", allowing to determine the time from which network updates are studied. Even though many studies point out the impact of the time window size on the quality of snapshots, they don't address particularly such an issue. Rather, existing works used to rely on a fixed time interval to capture snapshots.

The goal of this paper is to propose a strategy of setting time window for capturing a snapshot. In opposite to existing solutions, our approach does not define a fixed time window but a dynamic one based on the amount of reported changes. The reason of doing so is to make an earlier detection of the main phases of the network creation process. That is, our solution eases an on-time follow-up that alerts once significant changes occur rather than postponing them till a given time is reached. Such an approach sounds well for monitoring applications like dealers group monitoring, which requires rapid and instant reaction based on the organization size and/or status. This is also the case for epidemic surveillance systems. The main contributions of this work can be summarized as follows:

- a time-based quality function that evaluates the network quality at any given time t. The quality function takes into account the cohesion aspect as well as the communication intensity of nodes during a period of time.
- a prediction strategy that estimates scores of the network quality at time $t+1$ based on network composition at t. Two time series are used. The first one predicts the next quality score of the temporal network while the second one is used to correct the prediction by minimizing the error of prediction;
- a snapshot setting based on the prediction of the quality score values over time. Our solution work by supposing that two snapshots have to be different so as their quality scores. Therefore, we compute the variation of the scores at t and $t + 1$ and if it is beyond a given threshold, thus, $t + 1$ is the start of a new snapshot.
- an implementation of our solution using R to assess and validate our algorithms over a real data set. The quality of the predictions as well the snapshot bounding show the feasibility and performances of our approach.

The remainder of this paper is as follows. Firstly, we review the related works in Sect. 2. Secondly, we propose our quality function in Sect. 3. Thirdly, we explain in Sect. 4 our methodology for decomposing the network evolution. To validate our solution, we present our experiments in Sect. 5 before concluding in Sect. 6.

2 Related Work

Even though several studies explain the effect of time window size on snapshot setting [3,4,6]. To our best knowledge, there is not yet work focusing on estimating the time window size based on a methodological reasoning. In existing works, the time window size is fixed in a static manner without providing any argumentation. Let us give the following examples of periods: one day [2], one month [6] or multiple timescales [3].

Indeed, time window management can be done by using:

1. A static method, which decomposes the network evolution into several snapshots having all the same size.
2. A dynamic method that subdivides the network evolution into several snapshots whose size of each one may be different from that of other.

The static method has the advantage of being simple and easy to implement. However, if the network does not change during a given period, the snapshots obtained over this period will contain exactly the same structure. Thus, we are wasting time and resources searching for new changes while network has not changed. In addition, if network structure evolves irregularly, choosing a static time window size can be problematic: a small size may lead to snapshots that don't capture important connections, while a big size would hide the precise moments of significant changes of network structure.

To avoid the problem of time window size balancing as well as favor relevant captures that incorporate enough changes, we propose a strategy able to capture dynamically snapshots throughout the network evolution. To this end, we define a quality function allowing to quantify the network changes at a given time. Our method captures a new snapshot if the quality difference of network between two moments exceeds a given threshold. We use a time series to predict the moments from which the quality score reaches the defined threshold.

3 Quality Function

In this section, we propose a quality function allowing to evaluate a snapshot relevance in term of cohesion aspect as well as the communication intensity between nodes. We consider that a snapshot is a static network including all nodes/links that have appeared at least once during a time interval. If an interaction appears several times during the time internal, we represent it by a single link whose weight is equal to sum of links weights that correspond to different appearances.

The quality function we present here is a reformulation of the one we proposed in [1]. The difference between these two functions is that the first one evaluates the quality of a local community in a static network while the reformulated function evaluates the quality of a temporal network at a given instant t.

Let $\mathcal{N}$ a temporal network and $\mathcal{N}_t$ the network snapshot at instant t. To get an idea of the communication intensity between nodes within $\mathcal{N}_t$, we calculate the inverse of all links weights sum $\frac{1}{\sum_{w_{\mathcal{N}_t}}}$. The intuition behind is that more communication is intense within snapshot $\mathcal{N}_t$ (high values of links weights), more $\frac{1}{\sum_{w_{\mathcal{N}_t}}}$ tends to 0.

Regarding the internal cohesion of $\mathcal{N}_t$, we consider that more the topological structure of a snapshot at time t is similar to a clique, more the snapshot is considered cohesive. Therefore, the proportion $\frac{|V_{\mathcal{N}_t}|}{|E_{\mathcal{N}_t}|}$ allows to evaluate at what level the snapshot $\mathcal{N}_t$ is cohesive. $|V_{\mathcal{N}_t}|$ is the number of nodes in $\mathcal{N}_t$ and $|E_{\mathcal{N}_t}|$ the number of links.

Our quality function is defined as follows:

$$\psi(\mathcal{N}_t) = \frac{1}{\sum_{w_{\mathcal{N}_t}}} \times \frac{|V_{\mathcal{N}_t}|}{|E_{\mathcal{N}_t}|} \tag{1}$$

4 Methodology

We consider a network represented by a weighted digraph where a link weight states for the intensity (number of exchanges for instance) of two nodes and the orientation indicates which node has initiated the interaction. While assuming that the network is a temporal one, the overall structure shaping is assimilated as an evolution process within which a series of moments of significants changes are reported. A specific moment bringing numbers of changes can be seen as the delimitation of two snapshots. We name such a moment a "*switch moment*" since it indicates a precise time where changes make a great impact on the overall structure. Actually, the purpose of this work is to characterize, and moreover, to predict "*switch moment*". In this respect, we define a time-based function that measures the quality of the network at anytime. Hence, if the network quality score varies beyond a given threshold between t and $t+1$, thus, we consider that we reach a "*switch moment*" at $t+1$. In other words, the score of the quality remain almost the same for a network without and/or with a few significant changes during a period of time. We notice the quality function is a continuous one so that it affords the possibility to detect a"*switch moment*" once it happens. Furthermore, to be able predicting whether a new "*switch moment*" will occur, we need to foresee the quality scores evolution over time. To this end, we rely on exponential smoothing that helps us estimating upcoming score values based on the previous and detecting any variation beyond the fixed threshold. Finally, since the quality scores predictions give us "*switch moment*", we can monitor where one has to start or ending capturing a snapshot. In the following sections we portray the definition and prediction method of our quality function.

The optimal size of time window should enable us to decompose the network evolution into several snapshots, each one includes a considerable number of changes. To this end, we propose a strategy that works in two steps:

1. measure network changes during a time period using the quality function;
2. model the changes rate across a threshold η. If the network quality difference between two instants exceeds the threshold value, that means that the network has been considerably modified.

Formally, we consider that the network has undergone a considerable change if the difference between its quality at instant t and that at instant $t+1$ becomes higher than threshold η:

$$|\psi(\mathcal{N}_t) - \psi(\mathcal{N}_{t+1})| \geqslant \eta \tag{2}$$

Note that more higher the threshold value, more the changes number increases between each two successive snapshots.

4.1 Quality Scores Prediction

The principle of our method is to predict the quality score at instant $t+1$ from instant t. In other words, we try to predict the next quality score before the network reaches the next instant. To predict the next quality score, we use a time series built from quality scores over time. The prediction model we used is the simple exponential smoothing. Formally, our time series is defined as:

$$\hat{P}(t) = \alpha Q(t) + (1-\alpha)\hat{P}(t-1) \tag{3}$$

Such as:

- $\hat{P}(t)$ means the predicted quality score at instant $t+1$;
- $\alpha \in\ ]0,1[$ represents the smoothing coefficient;
- $Q(t)$ indicates the observed quality score at instant t;
- $\hat{P}(t-1)$ is the predicted quality score at instant t.

If α is closest to 0 (respectively if α is closest to 1), it means that to predict the next quality score, the most oldest (respectively most recent) predicted values will be taking into account.

By using the expression 3, the algorithm will learn based on the history of predicted quality scores over time. In order to improve these predictions, we used a second series that allows to correct the prediction of $\hat{P}$ based on the history of prediction errors.

4.2 Correction of Our Predictions

To correct the prediction, our algorithm first calculates the prediction error, defined by the difference between observed quality score and predicted quality score. Then, it predicts the next prediction error using a time series based on the history of prediction errors.

4.2.1 Prediction Error

The prediction presented in the previous section sometime could be inaccurate because of important fluctuation of quality scores. To improve the future predictions, we try to correct our prediction using a second time series that learns from past prediction errors. To this end, we define the prediction error given by the expression below:

$$E(t) = Q(t) - \hat{P}(t-1) \tag{4}$$

We defined also a second time series to estimate the predicted error in the future according to the error recorded in the past:

$$\hat{L}(t) = \beta E(t) + (1-\beta)\hat{L}(t-1) \tag{5}$$

Such as:

- $\hat{L}(t)$ means the predicted error at instant $t+1$;
- $\beta \in]0,1[$ represents the predicted error coefficient;
- $E(t)$ is the prediction error at instant t;
- $\hat{L}(t-1)$ is the predicted error at instant t.

If β is closest to 0 (respectively if β is closest to 1), it means that to estimate the next predicted error, the most oldest (respectively most recent) predicted values will be taking into account.

4.2.2 Correction of Predicted Quality Scores

After predicting the error, we are then able to make some correction of the prediction according to the context which is materialized by the value of the error. Our algorithm considers that the corrected prediction is equal to the sum of predicted score and predicted error:

$$\hat{C}(t) = \hat{P}(t-1) + \hat{L}(t-1) \tag{6}$$

Finally, it should be noted that results of our time series $\hat{L}(t)$ et $\hat{C}(t)$ improve by learning from past predicted values. Thus, the wider the history, the better the predicted futures will be.

5 Experimentation

The purpose of this section is to evaluate the effectiveness of our solution. To this end, we implemented our solution with R platform. The dataset we used includes more than 2 million mobile phone interactions between 80 students who lived in undergraduate dormitory [5]. These interactions were collected between September 05, 2007 and July 16, 2009. In the following, we present two experiments. The objective of the first one is to show that our predictions provide very close scores, often identical to observed quality scores. The second experiment aims to present some detected snapshots during the network evolution. These two experiences are presented in Sects. 5.1 and 5.2.

5.1 Quality Scores Prediction

The use of the time series P and L requires two parameters, namely, the prediction coefficient α and the coefficient of predicted error β. In our experiments, we chose the coefficient $\alpha = 0.5$ for the next prediction to be influenced, equitably, by oldest past values and the near past values. Regarding the β coefficient, we chose it so that momentary fluctuations do not have a significant impact on the predicted error of the next prediction. Thus, $\beta = 0.1$. The goal of the first experiment we conducted is to assess the accuracy of short-term prediction. To this end, we predicted network quality scores between September 05, 2007 and April 17, 2008, a period of approximately 7 months and two weeks. During this period, there were 2000 interactions.

Figure 1 presents the observed quality scores (black colored curve), the predicted quality score (red colored curve) and the corrected predictions (green colored curve). The x-axis indicates the times at which the network has undergone a change. The y-axis represents the corresponding quality scores. On Fig. 1, we noticed that the prediction is acceptable when there are fluctuations. However, the prediction becomes good if the quality scores curve is somehow linear. On the curves, we remark that prediction are sometime more accurate than correction. It can be explained by the fact that, since the correction of prediction is base on error prediction, it means that if the prediction of the error is bad, then the corrected prediction will be inaccurate. When the quality function values have important fluctuation in a short time, the error predicted becomes bad, so this inaccuracy is propagated to correction.

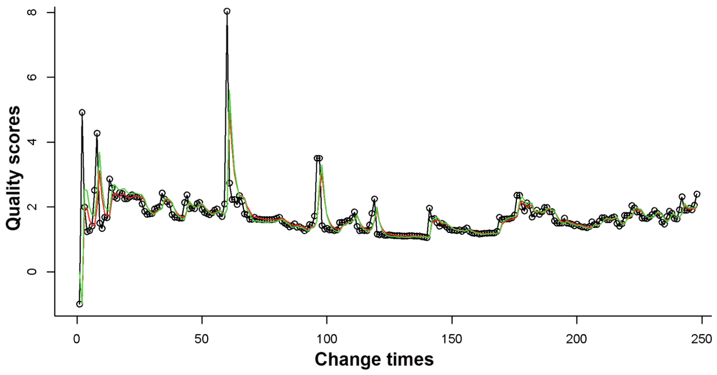


Fig. 1. Predicted and observed quality scores between September 05, 2007 and April 17, 2008 (Color figure online).

To evaluate the effectiveness of our long-term prediction, we conducted a second experiment. This time, we observed all the interactions that took place between September 05, 2007 and October 03, 2008, a period of one year and four weeks. This period includes 10000 interactions. The number of times the

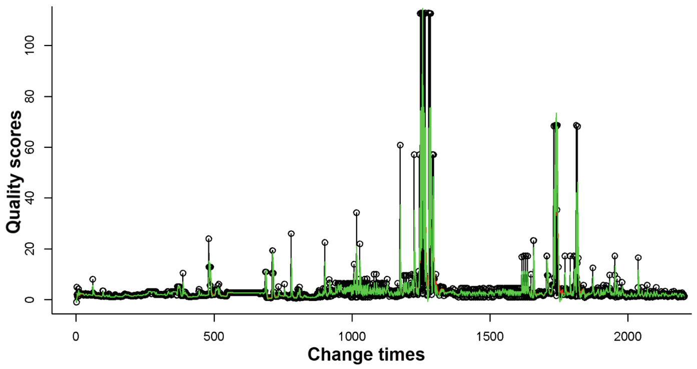


Fig. 2. Predicted and observed quality scores between September 05, 2007 and October 03, 2008.

network has undergone a change is equal to 2205. Figure 2 presents the curves of this experiment.

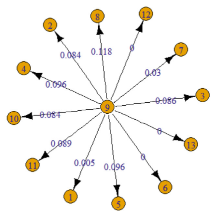
Note that in long-term, correction of prediction becomes almost identical with observed quality scores. This proves the effectiveness of our prediction algorithm.

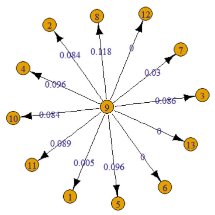
5.2 Examples of Detected Snapshots

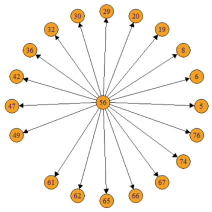
The objective of this experiment is to show the difference between our solution and the static method that decomposes the network evolution using a regular time interval. To this end, we consider that the time interval is 30 min. Thus, we make a new snapshot every 30 min.

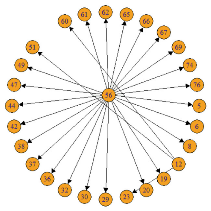
Figure 3a, b, c and d show the network state, respectively, after 30 min, one hour, one hour and a half and two hours. The value displayed on each link represents its weight during the time window. For readability reasons, we did not display the link weights in Figs. 3c and d. We find out that the network did not change during the first two snapshots and a slight change occurred during last two snapshots. This experiment illustrates the irrelevance of the static method since it can capture identical snapshots over time, which involves a waste of time and resources. To overcome this weakness, we present in the following a few snapshots captured by our solution. To this end, we consider that rate of quality changes needed to capture a new snapshot is $\eta = 0.8$. Remember that our method that allows to decompose the network evolution according to the degree of changes that takes place.

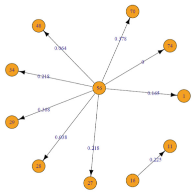
Figures 3e, f, g and h show four snapshots captured at different times. The value displayed on each link represents its weight during the time window. For readability reasons, we did not display the link weights in Figs. 3g and h. From these figures, we clearly see that the time window size is dynamic. It varies according to changes degree of network. The considerable difference between the size of these windows is due to the network changes degree over time.

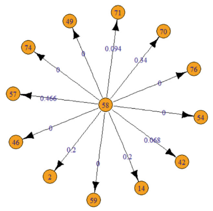

(a) Captured snapshot after 30 minutes.

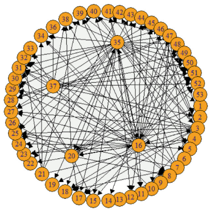

(b) Captured snapshot after 60 minutes.


(c) Captured snapshot after 90 minutes.


(d) Captured snapshot after 120 minutes.


(e) First time window representing the initial state of temporal network. This time window is captured on September 05, 2007 at 14:02:11.


(f) Second time window captured on September 05, 2007 at 14:12:33. The interval of this window is 10 minutes.


(g) Third time window captured on January 23, 2008 at 14:27:42. The size of this window is 4 months.

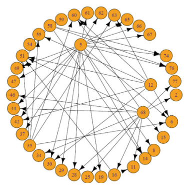

(h) Fourth time window captured on January 26, 2008 at 06:37:50. The size of this time window is 3 days.

Fig. 3. Examples of captured snapshots by the static method VS those one captured by our solution.

6 Conclusion

In this paper, we proposed a solution that answers two fundamental questions, namely:

1. how to measure the temporal network quality during a given period?
2. how to determine at what moment we need to capture a new snapshot?

To answer the first question, we proposed a quality function allowing to evaluate the internal cohesion and the communication intensity between nodes in a temporal network. Regarding the second question, we proposed a new strategy based on time series to predict the next moment corresponding to a new snapshot.

In perspectives, we are interested in making an experiment in order to determine the optimal threshold of a given dataset. In addition, we intend to study the time window size in relation to the local changes of some ego-communities. Finally, we will also experiment our solution on several datasets to determine the impact of the network kind on the time window size.

References

1. Ould Mohamed Moctar, A., Sarr, I.: Ego-centered community detection in directed and weighted networks. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, pp. 1201–1208, New York, NY, USA, 2017. ACM (2017)
2. Génois, M., Vestergaard, C.L., Fournet, J., Panisson, A., Bonmarin, I., Barrat, A.: Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers. Network Sci. **3**(3), 326–347 (2015)
3. Holme, P.: Modern temporal network theory: a colloquium. Eur. Phys. J. B **88**(9), 234 (2015)
4. Krings, G., Karsai, M., Bernhardsson, S., Blondel, V.D., Saramäki, J.: Effects of time window size and placement on the structure of an aggregated communication network. EPJ Data Sci. **1**(4), 1–16 (2012)
5. Madan, A., Cebrian, M., Moturu, S., Farrahi, K., et al.: Sensing the "health state" of a community. IEEE Pervasive Comput. **11**(4), 36–45 (2012)
6. Psorakis, I., Roberts, S.J., Rezek, I., Sheldon, B.C.: Inferring social network structure in ecological systems from spatio-temporal data streams. J. R. Soc. Interface **9**, 3055–3066 (2012)