Finding simple temporal cycles in an interaction network

Rohit Kumar[1] and Toon Calders[1,2]

[1] Université Libre de Bruxelles, Belgium
rohit.kumar@ulb.ac.be**
[2] Universiteit Antwerpen, Belgium
toon.calders@uantwerpen.be

Abstract. Interaction networks differentiate themselves from the traditional networks in the sense that nodes interact continuously and repeatedly. Hence, when studying patterns in interaction networks it is essential to take into account the temporal nature of the data. In this paper, we present preliminary work on the problem of finding cyclic interaction patterns; for instance: one person transfers money to a second person, who transfers it to a third person transferring the money back to the first person. It is important here that the cycle occurs in the right temporal order, and that the time interval between the first and the last interaction of the cycle does not exceed a given time window. Cyclic patterns represent highly useful information; they are for instance used to detect specific types of fraud in financial transaction networks. Furthermore, as our results show, datasets from different domains show different behavior in terms of number and size of cycles. As such, cycles capture essential differences in temporal behavior of interaction networks.

1 Introduction

With the advancement of the current digital era, very large temporal graphs are becoming quite common. Examples include the re-tweet network, online transactions, and phone or SMS communication. Most studies that aim at finding local or global patterns in such networks concentrate on static networks only, in which the links are relatively stable, such as for instance friendship links in social networks. In this paper, we are interested in the interactions that take place in such networks themselves, not the static structure they create. In order to study the dynamics of these networks, it is essential to take the temporal nature of the interactions into account [3].

Recently, Paranjape et al. [5] introduced an algorithm for counting the number of occurrences of a given temporal *motif* in a temporal network. In their paper the authors show that datasets from different domains have significantly different motif counts, showing that temporal motifs are useful for capturing differences in temporal behavior. Our paper can be seen as an extension of this

** Supported by Fonds de la Recherche Scientifique under Grant no T.0183.14 PDR

work to cyclic patterns, of any length, and constrained by a time window. Cycles appear naturally in many problem settings. For instance, in logistics if the interactions represent resources being moved between facilities, a cycle may indicate an optimization opportunity by reducing excessive relocation of resources; in stock trading cyclic patterns may indicate attempts to artificially create high trading volumes; and in financial transaction data cycles could be associated to specific types of fraud. The potential of cycles in the context of fraud detection has already been acknowledged [1].

Detecting or enumerating cycles in a directed static graph has already been studied for decades [2]. The existing algorithms for enumerating cycles in static graphs, however, do not directly apply to temporal networks. This paper is the first one to extend the problem of enumerating all cycles to temporal networks. We propose an algorithm for mining cycles without repeating nodes that occur within a given time window, in one scan over a given set of interactions ordered by interaction time. In our experiments we observe that cycle length and frequency distribution varies based on the characteristics of the network.

2 Preliminaries

Let V be a set of nodes. An interaction is defined as a triplet (u, v, t), where $u, v \in V$, and t is a natural number representing the time the interaction took place. Interactions are directed and could denote, for instance, the sending of a message in a communication network. Multiple edges can appear at the same time. A temporal network $G(V, \mathcal{E})$ is a set of nodes V, together with a set $\mathcal{E}$ of interactions over V. We will use $n = |V|$ to denote the number of nodes in the temporal graph, and $m = |\mathcal{E}|$ to denote the total number of interactions.

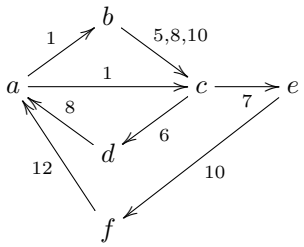


Fig. 1. Example temporal network

Definition 1. *(Simple Temporal Cycle) A temporal path between two nodes $u, v \in V$ is defined as a sequence of edges $p = \langle (u, n_1, t_1), (n_1, n_2, t_2), .., (n_{k-1}, v, t_k) \rangle$ such that $t_1 < t_2 < .. < t_k$ and all edges in p appears in $\mathcal{E}$. $dur(p) := t_k - t_1$ denotes the duration of the path. Often we use the more compact notation $u \xrightarrow{t_1} n_1 \xrightarrow{t_2} n_2 \ldots \xrightarrow{t_k} v$.*

A temporal cycle with root node r is a temporal path from r to itself. The cycle is called simple if each internal node in the cycle occurs exactly once.

Simple Cycle Detection(SCD) Problem: Given a temporal network $G(V, \mathcal{E})$ and a time window ω, find all simple cycles C with $dur(C) \leq \omega$.

Example 1. Consider the interaction network given in Figure 1. This interaction network contains 4 simple cycles, all with root node a. The cycles are: $a \xrightarrow{1} c \xrightarrow{6} d \xrightarrow{8} a$, $a \xrightarrow{1} b \xrightarrow{5} c \xrightarrow{6} d \xrightarrow{8} a$, $a \xrightarrow{1} c \xrightarrow{7} e \xrightarrow{10} f \xrightarrow{12} a$, and $a \xrightarrow{1} b \xrightarrow{5} c \xrightarrow{7} e \xrightarrow{10} f \xrightarrow{12} a$. If $\omega = 10$, the set of the first two cycles form the solution to the SCD Problem.

Algorithm 1 Simple algorithm to find all cycles

Require: Threshold ω, interactions $\mathcal{E}$
Ensure: All simple cycles.

1: $L \leftarrow \{\}$
2: **for** $(u, v, t) \in \mathcal{E}$, ordered ascending w.r.t. t **do**
3: **for** $p = v_1 \xrightarrow{t_1} v_2 \ldots \xrightarrow{t_{k-1}} v_k \xrightarrow{t_k} u \in L$ **do**
4: **if** $t_1 > t - \omega$ **then**
5: **if** $v_1 = v$ **then**
6: **Output** $v \xrightarrow{t_1} v_2 \ldots \xrightarrow{t_{k-1}} v_k \xrightarrow{t_k} u \xrightarrow{t_1} v$
7: **else if** $v \notin \{v_1, \ldots, v_k\}$ **then**
8: $L \leftarrow L \cup \{v_1 \xrightarrow{t_1} v_2 \ldots \xrightarrow{t_{k-1}} v_k \xrightarrow{t_k} u \xrightarrow{t_1} v\}$
9: **else**
10: $L \leftarrow L \setminus \{p\}$ ▷ prune paths exceeding the window duration
11: $L \leftarrow L \cup \{u \xrightarrow{t_1} v\}$

3 Enumerating All Simple Temporal Cycles

We present a one pass algorithm to enumerate all cycles in a given temporal network in Algorithm 1. We assume that the interactions are stored in chronological order. The key idea behind the algorithm is to maintain a list(L) of all *valid temporal paths* for a given window length ω. A temporal path $p = v_1 \xrightarrow{t_1} v_2 \ldots \xrightarrow{t_{k-1}} v_k \xrightarrow{t_k} u$ is considered valid at time stamp t if $t - t_1 < \omega$. If the path is not valid it is removed from the list as it can never be extended in future to form a valid cycle (line 10). For each interaction (u, v, t), all valid paths that end in u and do not contain v are extended to create a new temporal path (line 8). Furthermore, all valid simple paths from v to u form a cycle with v as the root node for interaction (u, v, t) (line 6).

4 Experiments

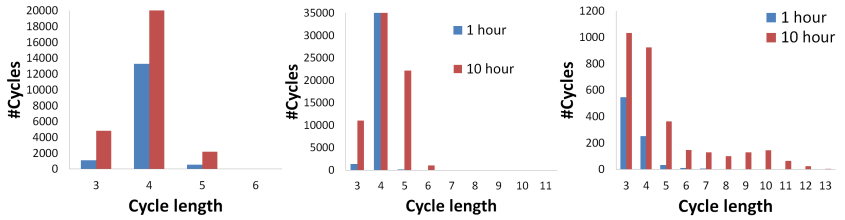
We evaluated the run-time and memory requirements of Algorithm 1 on three different datasets (Higgs-twitter,Facebook-WSON and SMS-A) [4] for ω = 1hr and 10hrs. We also analyzed the number and distribution of lengths of cycles in the different datasets.

Runtime and Memory. As shown in Table 1 both time and memory requirements increase with increasing ω, because the temporal paths remain valid for a longer duration. The time and memory required for SMS-A is higher than Facebook-WSON though SMS-A has fewer interactions because of the large number of temporal paths and cycles in SMS-A as compared to Facebook-WSON.

Analysis of Cycles found. The maximal length of the cycles found increases as we increase ω. Facebook-WSON and SMS-A exhibit a similar cycle length distribution pointing to the fact that both datasets capture messaging

Dataset	Nodes	Edges	Time(min)		Memory(MB)	
			$\omega = 1$	$\omega = 10$	$\omega = 1$	$\omega = 10$
Facebook-WSON	46 952	876 993	6.4	7.2	15	23
Higgs-twitter	304 691	526 167	39.5	198.5	158	1821
SMS-A	44 100	545 000	12	156.9	34	7905

Table 1. Time and memory requirement for different ω to find all the simple cycles.



(a) Facebook-WSON (b) SMS-A (c) Higgs-twitter

Fig. 2. Distribution of the number of cycles in function of the cycle length (a-c).

behavior over time. Higgs-twitter has much higher cycle lengths as compared to the other two datasets. These results are shown in Figure 2.

5 Conclusion

We introduced a new problem in temporal pattern mining in interaction graphs: finding all temporal cycles. We introduced a one-pass algorithm for this problem based on maintaining all paths that can still become cycles. The results show that the number and length of cycles differ substantially between the Twitter dataset and the other two, social network related datasets. Future work includes the development of more efficient algorithms for finding all simple cycles.

References

1. Hoffmann, F., Krasle, D.: Fraud detection using network analysis (2015), eP Patent App. EP20,140,003,010
2. Johnson, D.B.: Finding all the elementary circuits of a directed graph. SIAM Journal on Computing (1975)
3. Kumar, R., Calders, T., Gionis, A., Tatti, N.: Maintaining sliding-window neighborhood profiles in interaction networks. In: ECML/PKDD (2) (2015)
4. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014)
5. Paranjape, A., Benson, A.R., Leskovec, J.: Motifs in temporal networks. In: WSDM (2017)