TimeRank: A Random Walk Approach for Community Discovery in Dynamic Networks

Ilias Sarantopoulos[1,2(✉)], Dimitrios Papatheodorou[2,4], Dimitrios Vogiatzis[1,3], Grigorios Tzortzis[1], and Georgios Paliouras[1]

[1] National Centre for Scientific Research (NCSR) "Demokritos", Athens, Greece
isaranto@aueb.gr, dimitrv@iit.demokritos.gr, gtzortzi@iit.demokritos.gr, paliourg@iit.demokritos.gr
[2] Athens University of Economics and Business, Athens, Greece
dimitrispapatheodorou95@gmail.com
[3] The American College of Greece, Athens, Greece
[4] Aalto University, Espoo, Finland

Abstract. In this work we consider the problem of discovering communities in time evolving social networks. We propose TimeRank, an algorithm for dynamic networks, which uses random walks on a tensor representation to detect time-evolving communities. The proposed algorithm is based on an earlier work on community detection in multi-relational networks. Detection of dynamic communities can be be done in two steps (segmentation of the network into time frames, detection of communities per time frame and tracking of communities across time frames). Alternatively it can be done in one step. TimeRank is a one step approach. We compared TimeRank with Non-Negative Tensor Factorisation and Group Evolution Discovery method on synthetic and real world data sets from Reddit.

1 Introduction

Community detection in networks represented as static graphs, is a well-studied problem [7], but real world networks demonstrate temporal activity incurring changes in their structure over time. Along with the network, the structure and the composition of the communities also change, i.e. they *evolve*. Given a network that is observed over a time period, it can be divided into equal T time intervals or timeframes, then the social network can be represented as a sequence of graphs $\{G_1, G_2, \ldots, G_t\}$, where each graph includes the interactions between the nodes observed in the corresponding timeframe. A *dynamic community* is made of a chain of corresponding static communities along the time frames of the network. Some of the issues that are related to the problem of dynamic community detection, is the static community detection per time frame, the tracking of static communities along time frames, as well as the granularity of the segmentation.

L. M. Aiello et al. (Eds.): COMPLEX NETWORKS 2018, SCI 812, pp. 338–350, 2019.
https://doi.org/10.1007/978-3-030-05411-3_28

The motivating idea in the current work is to map a temporal network to a weighted static network, where the weights capture temporal relations. Subsequently any existing algorithm for community detection can be applied on the static network, and finally the communities of the static network can be mapped back to the original temporal network.

The main contributions of this paper are:

- TimeRank, a random walk algorithm for dynamic community detection that is based on an earlier algorithm for community detection on multi-relational networks.
- Experimental investigation of the merits of TimeRank on synthetic and real world data sets and comparison with two other methods.

The rest of the paper is organised as follows; Sect. 2 describes the related work; Sect. 3 introduces the background method MutuRank, an algorithm for community discovery in multi-relational networks; Sect. 4 describes the proposed method TimeRank; experimental results are presented in Sect. 5; finally, conclusions are drawn in Sect. 6.

2 Related Work

Dynamic community discovery is represented by two main approaches: two step and one step methods. Two step methods tend to see community detection and tracking as two discrete procedures. During the first step, time is divided into possibly overlapping frames and a community detection method is applied to the static graph of each frame (usually the Louvain method is chosen [5]). During the second step, communities between neighbouring time-frames are compared and matched using a similarity measure and depending on the degree of similarity an event is assigned on each evolution between two consecutive time-frames (form, dissolve, merge, split, grow, shrink, continue). Mostly the Jaccard similarity is being used. These two-step approaches do not tend to focus on the first step (community detection) but rather on modelling the tracking of the evolution of the detected communities. Such methods are described in [6,9–11,20,21]. One step approaches do not separate community detection from tracking of their evolution but perform both in one step [8,23]. An interesting instance of one step methods, performs both community tracking and community prediction in a parameter free way and it is based on tensor decomposition and the minimum description length principle [3]. Also approaches for detection and prediction have been proposed that are based on link and content analysis [2].

The ***Group Evolution Discovery (GED)*** method is a two step method (i.e. community tracking) [6]. For the second step a new measure is introduced called inclusion, which measures both the quality and quantity of the inclusion of one group in another. $I(G_1, G_2) = \frac{|G_1 \cap G_2|}{|G_1|} \cdot \frac{\sum_{x \in (G_1 \cap G_2)} SP_{G_1}(x)}{\sum_{x \in (G_1)} SP_{G_1}(x)}$, where the first fraction measures the group quantity and the second the quality, and $SP_{G_1}(x)$ is the value of social position of x in group G_1. Social position is a centrality measure.

For each of the matched communities we want to discover its evolution. Group evolution is a sequence of changes succeeding each other that happen in sequential timeframes. In [6] the authors extended the events proposed in [4,19]. They identified seven type of rule-based events that describe the change in the state of a group or groups between two timeframes (T_i and T_{i+1}): continuing, shrinking, growing, splitting, merging, dissolving, forming.

Non-Negative Tensor Factorisation (NNTF). The interaction among users across time can be represented by a 3-dimensional tensor $T \in \mathbb{R}^{N \times N \times S}$, where N is the number of nodes/users of the network and S the number of network snapshots. Tensor factorisation techniques and PARAFAC in particular [12] has been used in the detection of dynamic communities [8]. In tensor factorisation the following optimization problem is solved: $\min_{\mathbf{A,B,C}} ||T - \mathbf{A,B,C}||_F^2$, where matrices $\mathbf{A}$ and $\mathbf{B}$ are square of size $N \times k$ and associate a weight for the membership of each node to each community structure, while $\mathbf{C}$, which is of size $S \times k$, associates the communities with time intervals and k, which is provided as input, is the number of latent factors or communities that are discovered. In the case of undirected networks $\mathbf{A} = \mathbf{B}$. This way clustering and tracking is performed in one step, with matrix $\mathbf{A}$ representing the clustering output and matrix $\mathbf{C}$ the tracking output.

3 Background: MutuRank

We start by describing MutuRank, an algorithm for discovering community structure in multi-relational networks, which we later transform to fit the problem of dynamic community finding.

A multi-relational network (MRN), is a network where there can be many types of edges between any two nodes, for example in a social network an edge between two users can have multiple meanings: follow, like, retweet etc. A well known problem in MRNs is discovering community structure. The problem is similar to discovering communities in single-relational networks (SRNs). Because of the homogeneity of the nodes across the relations, we want to find a good k-way partition $\mathscr{P} = \{C_1, \ldots, C_k\}$, where C_k is the kth community and $C_1 \cup \ldots \cup C_k \subseteq V$ and V is the set of nodes that exist in the network, by considering all relations. The MRN is represented by a three dimensional $n \times n \times m$ affinity tensor, where n is the number of nodes and m is the number of relations. Let $\mathscr{A} = [a_{i,j,d}]$ be the affinity tensor and $a_{i,j,d} \in \mathbb{R}$ denote the relation strength between nodes i and j under the dth relation. Then $a_{i,j,d} = 1$ if i and j share an edge in d, otherwise $a_{i,j,d} = 0$.

In this context, co-ranking frameworks such as MutuRank [22] simultaneously rank the n nodes and the m relations in order to get a weighted importance p_i for each node i and q_d relation d and thus produce an SRN calculated as follows:

$$w_{i,j} = \sum_{d=1}^{m} q_d \cdot a_{i,j,d}, \tag{1}$$

where $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_m)$ are the weight/probability distribution vectors of nodes and relations respectively. In SRNs obtaining the distribution $\mathbf{p}$ which models node importance can be achieved with well known algorithms like HITS [13] and Pagerank [18]. But in the case of MRNs nodes and relations yield mutual influence, thus there is the need to co-rank nodes and relations simultaneously, hence $\mathbf{p}$ is implicitly used in the calculation of $\mathbf{q}$ in Eq. (3) (see below).

MutuRank uses the mutual influence in order to rank relations and eventually transform the multi-relational network into a single-relational network (SRN), where there exists only one type of interaction between nodes, and communities can be detected using any community detection algorithm applicable to SRNs.

As we envision a random walk applied on a MRN, we need to construct two probability tensors $\mathscr{O} = [o_{i,j,d}]$ and $\mathscr{R} = [r_{i,j,d}]$ with respect to nodes and relations by normalizing the entries of $\mathscr{A}$ as follows $o_{i,j,d} = \frac{a_{i,j,d}}{\sum_{l=1}^{n} a_{l,j,d}}$, $r_{i,j,d} = \frac{a_{i,j,d}}{\sum_{e=1}^{m} a_{i,j,e}}$

MutuRank also considers influence exerted by prior distributions of nodes and relations. To conclude, we have the following iterative equations for computing the ranking scores of nodes and relations simultaneously:

$$p_i^t = \alpha \sum_{j=1}^{n} \sum_{d=1}^{m} p_j^{t-1} \cdot o_{i,j,d} \cdot Prob^{t-1}[d|j] + (1-\alpha) p_i^*, \tag{2}$$

$$q_d^t = \beta \sum_{i=1}^{n} \sum_{j=1}^{n} p_j^{t-1} \cdot r_{i,j,d} \cdot Prob^{t-1}[i|j] + (1-\beta) q_d^* \tag{3}$$

where $\mathbf{p}^* = (p_1^*, p_2^*, \ldots, p_n^*)$ and $\mathbf{q}* = (q_1^*, q_2^*, \ldots, q_m^*)$ are the prior distributions of nodes and relations respectively, and α, β are two parameters to balance the knowledge coming from network structure and the prior knowledge and are also called damping factors. Ideally prior distributions are obtained from a domain expert who can provide us with prior knowledge which quantifies the importance of nodes and relations. Such prior knowledge is very difficult to obtain so in most cases we can simply set $p_i^* = 1/n$, $1 \le i \le n$ and $q_d^* = 1/m$, $1 \le d \le m$. Incorporating the prior knowledge is helpful to deal with dangling nodes (that have zero out-degree) [18]. More precisely, given a dangling node i, $\forall 1 \le j \le n$, $a_{i,j,d} = 0$ leads to $Prob\big[i|j\big] = Prob\big[d|j\big] = 0$. Therefore, the weight of the sink node cannot be updated by the iterative process (Fig. 1).

In the case where we have one relation, so $m = 1$ and $p_i^* = 0$, $1 \le i \le n$, Eq. (2) becomes $p_i^t = \sum_{j=1}^{n} p_j^{t-1} \cdot \frac{1}{k_j}$, where k_j is the degree of node j. In this case we have exactly the same computation as in Pagerank [18], thus Pagerank can be regarded as a special case of MutuRank. Upon running the above iterative process we can transform the MRN into a SRN with the help of Eq. (1). Afterwards we can employ a clustering algorithm on the graph described by two-dimensional matrix w to extract community structure.

4 TimeRank

In this section we describe the proposed method, TimeRank, a random walk algorithm for dynamic community detection that is based on MutuRank.

Our intention is to transform the problem of finding communities in MRNs to that of dynamic community finding and employ a similar procedure to that of MutuRank to get the dynamic communities. To do that we use a tensor representation for our dynamic network. The first two dimensions refer to the nodes and the third dimension to the timeframes, in essence treating the timeframes as relations. A tensor "slice" for a particular timeframe represents the adjacency matrix of the network at that timeframe. The difference in our representation is that in each slice, node i is represented by a set of distinct nodes, one for each timeframe which we call node images. Let N be the numbers of nodes, and T the number of timeframes. Then node i will actually be a set of the different images of the node, one for each timeframe, $i = \{i_1, i_2, \ldots, i_T\}$. This way the number of distinct nodes in our representation is $N \times T$. The dimensions of the new tensor are $(N \times T) \times (N \times T) \times (T)$.

$$\mathscr{A}[:,:,t] = \begin{pmatrix} a_{11t} & a_{12t} & a_{13t} & \cdots & a_{1(N\times T)t} \\ a_{21t} & a_{22t} & a_{23t} & \cdots & a_{2(N\times T)t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{(N\times T)1t} & a_{(N\times T)2t} & a_{(N\times T)3t} & \cdots & a_{(N\times T)(N\times T)t} \end{pmatrix}$$

Given the above representation the dynamic communities we want to discover will include node images from separate timeframes.

Inter-Timeframe Edges Community detection algorithms analyse the structure of the network, attempting to produce communities, which are sets of

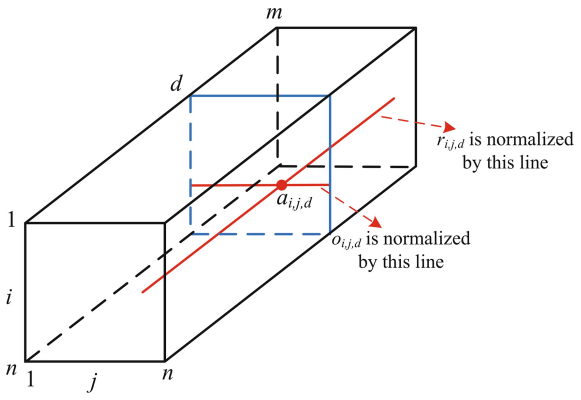

Fig. 1. Depiction of the normalisation of tensors $\mathscr{O}$ and $\mathscr{R}$ in MutuRank and TimeRank (source [22]).

nodes that are more densely connected with each other than the other nodes in the network. With the current representation it would not be possible for clusters from different timeframes to be placed in the same dynamic community, because neither the tensor "slices" for the timeframes nor the node images in theses "slices" are connected between them in any way. In this context we introduce inter-timeframe edges, which connect a node with its image in other timeframes. Dynamic communities contain separate groups (timeframe clusters) within them, which originate from different timeframes. These groups belong to the same dynamic community because they share some common nodes. The reasoning behind adding the inter-timeframe edges is that by connecting each node with its image we bring these groups closer to each other, thus allowing dynamic patterns to be more easily unveiled.

Given node i which exists in timeframe t_1 of a dynamic network, we propose two ways of connecting the node with its occurring images in the other timeframes $\{i_2, \ldots, i_T\}$. If a node does not appear in a specific timeframe, it will not be connected with its corresponding node images. Apart from the inter-timeframe edges there are also the original edges that exist in the graphs. Since each tensor slice represents a timeframe, it will comprise the edges that exist in the corresponding timeframe as they appear on the initial graph as well as the added inter-timeframe edges. The two variations of TimeRank rely on the following two representations:

- **One to All Connection (OAC)** In "One to All Connection" we connect each node i_t with all its occurrences in other timeframes. For each node i_t we add the following pairs of edges in the network: $\{(i_1, i_t), \ldots, (i_{t-1}, i_t), (i_{t+1}, i_t), \ldots, (i_T, i_t)\}$.
- **Next Occurrence Connection (NOC)** In "Next Occurrence Connection" we connect each node i_t with its images in the previous and next timeframes in which this node exists. For each node i_t we add the following pairs of edges in the network: $\{(i_{t-1}, i_t), (i_{t+1}, i_t)\}$.

Algorithm 1 TimeRank Algorithm

1: Create $(N \times T)$ adjacency matrices for each timeframe
2: Add inter-time edges
3: Compose tensor $\mathscr{A} \in \mathbb{R}^{(N\times T)\times(N\times T)\times(T)}$
4: Apply MutuRank algorithm on $\mathscr{A}$ and get ranking of timeframes
5: Create time-weighted network with $(N \times T)$ nodes
6: Perform clustering on this network and extract dynamic communities

In our implementation of TimeRank, in the final step we create an $N \times T$ matrix in the same fashion like MutuRank does, and we apply spectral clustering on that matrix to obtain the k dynamic communities, where the number of dynamic communities k is provided as input. In (1) Step 4 can be omitted, and

instead of running MutuRank we can use a uniform distribution for $\mathbf{q}$ during step 5. In MutuRank [22] irreducibility is a widely-used assumption. We also utilise irreducibility in TimeRank as an important assumption in order to derive the existence of the two equilibrium distributions.

5 Experimentation

5.1 Data Description and Engineering

Synthetic Data We constructed two different networks with two different event types, composed from 1000 nodes that span over 5 timeframes. We used the Dynamic Benchmark Network Generator [11], which is based on [15]. We set the seed to 10 in order for our dataset to be easily reconstructed. Nodes have a mean degree of 20 and a maximum degree of 40. The networks begin at $t = 1$ with approximately 32 communities, with each of the communities having a size in the range $[15, 50]$. The above parameters are common in the two datasets, with each having additional parameters regarding the events it implements. Specifically in the *Expand/Contract* dataset 10 randomly selected communities expand or contract by 25% of their previous size, while in the *Hide/Appear* dataset 10% of the communities hide and reappear between timeframes.

Real World Data: Reddit. Reddit is a popular social network that has the interesting construct of the *subreddits* which are theme-defined sub-communities of the network, e.g. "cats", "AskReddit", "Philosophy". The subreddits are explicitly used as the ground truth of our community tracking and the network is constructed through employing the users as nodes and their interactions/replies in the comment section as edges. We used the JSON comment dumps that can be found here[1], cleaned them, preprocessed them and sampled from them in order to have our final dataset based on the comments of the months 9/2010 and 10/2010. We removed inactive users and small, inactive or very large subreddits. The timeframe length we used was 1 week and the experiments were performed for 4 and 8 non-overlapping timeframes. Thus across a month there are 4 static graphs of communities that macroscopically compose a dynamic graph.

5.2 Evaluation Measures

To evaluate the results of the experiments we use a set of extrinsic metrics. These metrics require the actual community grouping to be known, commonly referred to as the ground truth. A detected community structure is considered good if it is close to the ground truth. There are different approaches in how to quantify the term "close". We use set matching methods (Normalised Mutual Information [14]) and pairwise evaluation measures (Omega Index [16], BCubed [1]). Let $\mathscr{T} = \{T_1, T_2, \ldots, T_k\}$ be the ground truth community structure of the network and $\mathscr{C} = \{C_1, C_2, \ldots, C_n\}$ be the community structure detected by an algorithm. Evaluation in this work is done in the context of dynamic communities, i.e.

[1] http://files.pushshift.io/reddit/comments.

ground truth dynamic communities are compared to the dynamic communities generated by the tested algorithms. The omega index as described in [16] can be found in GitHub[2] and the Python Package Index[3].

5.3 Results

Below there are the results of some interesting experiments using the synthetic and the Reddit data. The methods used were TimeRank-AOC (TR-AOC) and TimeRank-NOC (TR-NOC) with MutuRank and with uniform timeframe distributions (-U) for q (Table 2), Non Negative Tensor Factorisation (NNTF) and Group Evolution Discovery (GED). Overlapping versions were also tested, but showed no promising results. All code is available in GitHub.[4,5]

Benchmark Dynamic Networks In the synthetic *Expand/Contract* dataset, from the existing 32 communities, 10 expand and another 10 contract. Both expansion and contraction happen with a rate of 25%. The results, as seen in Table 1 establish that TimeRank succeeds in tracking the dynamic communities. In the case of NOC performance is boosted with the use of MutuRank, while AOC performance is inferior to NOC. In general the connection of each image with all of its nodes in this case does not help MutuRank uncover a relative ranking for the timeframes. While NNTF performs satisfactorily, GED fails to track the communities, because it identifies some expanded or contracted communities as new ones, instead of adjusted versions of the ones it has previously encountered.

Table 1. Tables for Expand/Contract and Hide/Appear datasets

	Expand/Contract					Hide/Appear				
Method	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.866	0.874	0.905	0.882	0.893	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
TR-NOC-U	0.908	0.919	0.944	0.921	0.933	0.880	0.890	0.912	0.963	0.937
TR-AOC	0.849	0.864	0.890	0.883	0.886	**1.000**	**1.000**	**1.000**	**1.0000**	**1.000**
TR-NOC	**0.923**	**0.954**	**0.964**	**0.944**	**0.953**	0.910	0.918	0.935	0.963	0.949
NNTF	0.805	0.8445	0.842	0.864	0.853	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
GED	0.464	0.659	0.924	0.572	0.707	0.531	0.700	0.901	0.662	0.763

[2] https://github.com/isaranto/omega-index.
[3] https://pypi.python.org/pypi/omega_index/.
[4] https://github.com/isaranto/community-tracking.
[5] https://github.com/NightmareNyx/CommunityTracking.

Table 2. Values for **q** distribution for the *Expand/Contract* and *Hide/Appear* datasets

	Expand/Contract					Hide/Appear				
Method	t_1	t_2	t_3	t_4	t_5	t_1	t_2	t_3	t_4	t_5
TR-AOC	0.212	0.196	0.198	0.196	0.199	0.214	0.189	0.199	0.191	0.207
TR-NOC	0.217	0.189	0.191	0.196	0.207	0.218	0.183	0.193	0.193	0.213

The output of NNTF does not support dynamic communities that are made up of communities whose composition differs in different timeframes. The composition of communities is obtained from the factors **A** and **B**, and factor **C** declares the timeframes in which each community exists, thus defining dynamic communities that comprise of identical communities in different timeframes. With the above taken into consideration it seems as the Hide/Appear dataset , with its evaluation results presented in 1 is tailored to the NNTF method, as communities in this dataset have exactly the same composition across time, but 10% of them disappear in some timeframe and appear again in another timeframe. TimeRank AOC captures these changes because it connects with all the images of a node, thus being able to place the same community in the corresponding dynamic one. TimeRank NOC on the other hand, although it achieves high performance, it does not provide such strong connections between the node images as in AOC, hence it can misplace nodes in datasets with behaviour similar to this one. In all the changes GED identifies death events in the case where a community is hidden introducing a new dynamic community when the community appears again, accompanied by a birth event.

Reddit Each table showcases the results in the basic (4 timeframes) and extended setting (8 timeframes) which tested the scalability of the methods.

Experiment 1 This experiment has 17 different-sized communities with temporal behaviours such as delayed birth, death, non-uniform size distribution across time, overlaps, death and re-birth. MutuRank-oriented TimeRank algorithms ended up with non-uniform weights and scoring marginally lower than uniform versions. Meanwhile, GED takes the lead managing to detect the sequential behaviour of the communities in the timeframe, especially the birth/death incidents. In the 8 timeframes setting, GED couldn't keep up and lowered its scores making TR-NOC the winner, which was better at differentiating between deaths/births and expands/contracts in the long run. Results in Table 3 below. At this point we can mention that MutuRank cause a significant drop in execution time performance, being more than 100 times slower than TimeRank with uniform q distributions.

Table 3. Experiment 1 results for 4 and 8 timeframes

	4 Timeframes					8 Timeframes				
Method	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.385	0.316	0.557	0.56	0.558	0.319	0.243	0.547	0.543	0.545
TR-NOC-U	0.48	0.428	0.639	0.619	0.629	0.377	0.455	0.612	0.580	0.596
TR-AOC	0.390	0.373	0.576	0.605	0.591	0.295	0.217	0.531	0.521	0.526
TR-NOC	0.457	0.473	0.633	0.623	0.628	**0.435**	**0.537**	0.610	**0.654**	**0.631**
NNTF	0.447	0.496	0.627	**0.642**	0.634	0.395	0.480	0.590	**0.650**	0.619
GED-T	**0.584**	**0.776**	**1.000**	0.625	**0.769**	0.323	0.432	**1.000**	0.377	0.548

Experiment 2 Here we have three big communities that change size in each timeframe with minor overlap. Results in Table 4 show that it was an easy win for TR-NOC in the 4-timeframe setting, which employed MutuRank correctly catching the underlying temporal structures. Interestingly, the tables are turned when four more timeframes are added, where everything fails except for NNTF, which scores much higher than before, due to the small number communities that express membership homogeneity through this longer period.

Table 4. Experiment 2 results for 4 and 8 timeframes

	4 Timeframes					8 Timeframes				
Method	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.050	0.068	0.480	0.544	0.510	0.028	−0.007	0.438	0.438	0.438
TR-NOC-U	0.380	0.487	0.692	**0.905**	0.784	0.032	0.032	0.435	0.536	0.480
TR-AOC	0.038	0.032	0.471	0.542	0.500	0.028	−0.007	0.438	0.439	0.439
TR-NOC	**0.456**	**0.604**	0.741	**0.951**	**0.833**	0.031	0.030	0.435	0.536	0.480
NNTF	0.276	0.389	0.723	0.652	0.686	**0.637**	**0.772**	0.849	**0.933**	**0.890**
GED-T	0.210	0.280	**1.000**	0.269	0.424	0.099	0.147	**1.000**	0.132	0.233

Experiment 3 This experiment contains 20 different sized communities that have big overlaps. Unsurprisingly, it proved to be the hardest one so far, especially for TimeRank, as the low scores indicate in Table 5 below. At this point we tried using the Fuzzy C-Means algorithm instead of K-Means in the Spectral Clustering process of TimeRank, to achieve an overlapping version of TimeRank and also used the overlapping version of NNTF. We tested these versions in the first 4 timeframes that showed the biggest overlap. Neither the overlapping versions nor the simple ones managed to score high; former losing points from high overlap generosity and the latter from the inability to predict overlaps. Moreover, the FCM clustering on the spectral embedding returned very uniform assignment matrices which led to predicting very overlapping communities. The highest scores came from GED and simple NNTF, except for the BCubed-Recall in which NNTF-Overlap scored relatively high, because of the

high amount of true positives. TimeRank's overlapping versions scored poorly and are not included.

Table 5. Experiment 3 results for 4 and 8 timeframes

	4 Timeframes					8 Timeframes				
Method	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.043	0.091	0.241	0.512	0.327	0.028	0.029	0.198	0.431	0.271
TR-NOC-U	0.056	0.075	0.256	0.501	0.338	0.080	0.088	0.227	**0.566**	0.324
TR-AOC	0.044	0.094	0.243	0.520	0.331	0.029	0.031	0.199	0.431	0.272
TR-NOC	0.071	0.114	0.264	0.495	0.345	0.053	0.086	0.228	0.420	0.296
NNTF	**0.277**	**0.410**	0.551	0.451	**0.496**	**0.170**	**0.344**	0.582	0.386	**0.464**
GED-T	**0.277**	0.398	**1.000**	0.307	0.470	0.088	0.234	**1.000**	0.178	0.302
NNTF-Ovlp	0.109	0.134	0.171	**0.859**	0.286					

6 Conclusions and Future Work

In this work we proposed TimeRank, with its two variations, as a new one-step method that performs both tracking and community detection in dynamic networks in one step. TimeRank represents a dynamic network as a tensor and then performs a random walk on the tensor in order to efficiently expose dynamic community structure. We compared TimeRank with two other methods. There seems to be no universal solution that can guarantee the best outcome. The GED method, could be characterised as a local method, as it can only discover changes between sequential timeframes, while NNTF is more suitable when communities show a membership homogeneity across time. On the other hand, TimeRank is robust to disruptions of homogeneity and is able to discover non-local temporal structure.

Future work involves experiments with more datasets. To speed up execution time we could relax the probabilities' estimation by adopting the idea from [17]. Finally, the inter-timeframe edges in AOC could have weights, proportional to the distance in terms of timeframes.

References

1. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. Inf. Retr. **12**(4), 461–486 (2009)
2. Appel, A.P., Cunha, R.L., Aggarwal, C.C., Terakado, M.M.: Temporally evolving community detection and prediction in content-centric networks. arXiv:1807.06560 (2018)
3. Araujo, M., Papadimitriou, S., Günnemann, S., Faloutsos, C., Basu, P.,Swami, A., Papalexakis, E.E., Koutra, D.: Com2: fast automatic discovery oftemporal ('comet') communities. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 271–283. Springer (2014)

4. Asur, S., Parthasarathy, S., Ucar, D.: An event-based framework for characterizing the evolutionary behavior of interaction graphs. ACM Trans. Knowl. Discov. Data (TKDD) **3**(4), 16 (2009)
5. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. (10), P10,008 (2008)
6. Bródka, P., Saganowski, S., Kazienko, P.: Ged: the method for group evolution discovery in social networks. Soc. Netw. Anal. Mining **3**(1), 1–14 (2013)
7. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
8. Gauvin, L., Panisson, A., Cattuto, C.: Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. PloS One **9**(1), e86,028 (2014)
9. Gliwa, B., Saganowski, S., Zygmunt, A., Bródka, P., Kazienko, P., Kozak, J.: Identification of group changes in blogosphere. In: 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1201–1206. IEEE (2012)
10. Goldberg, M.K., Magdon-Ismail, M., Nambirajan, S., Thompson, J.: Tracking and predicting evolution of social communities. In: SocialCom/PASSAT, pp. 780–783. Citeseer (2011)
11. Greene, D., Doyle, D., Cunningham, P.: Tracking the evolution of communities in dynamic social networks. In: Advances in Social Networks Analysis and Mining, pp. 176–183. IEEE (2010)
12. Harshman, R.A.: Parafac: an "explanatory" factor analysis procedure. J. Acoust. Soc. Amer. **50**(1A), 117–117 (1971)
13. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM (JACM) **46**(5), 604–632 (1999)
14. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. New J. Phys. **11**(3), 033,015 (2009)
15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78**(4), 046,110 (2008)
16. Murray, G., Carenini, G., Ng, R.: Using the omega index for evaluating abstractive community detection. In: Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization, pp. 10–18. Association for Computational Linguistics (2012)
17. Ng, M.K.P., Li, X., Ye, Y.: Multirank: co-ranking for objects and relations in multi-relational data. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1217–1225. ACM (2011)
18. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Technical Report, Stanford InfoLab (1999)
19. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. Nature **446**(7136), 664 (2007)
20. Tajeuna, E.G., Bouguessa, M., Wang, S.: Tracking the evolution of community structures in time-evolving social networks. In: IEEE International Conference on Data Science and Advanced Analytics (DSAA). 36678 2015, pp. 1–10. IEEE (2015)
21. Takaffoli, M., Fagnan, J., Sangi, F., Zaïane, O.R.: Tracking changes in dynamic information networks. In: 2011 International Conference on Computational Aspects of Social Networks (CASoN), pp. 94–101. IEEE (2011)
22. Wu, Z., Cao, J., Zhu, G., Yin, W., Cuzzocrea, A., Shi, J.: Detecting overlapping communities in poly-relational networks. World Wide Web **18**(5), 1373–1390 (2015)

23. Yang, J., Leskovec, J.: Overlapping community detection at scale: a non negative matrix factorization approach. In: Proceedings of the Sixth ACM International Conference on Websearch and Data Mining, pp. 587–596. ACM (2013)