Data Mining and Knowledge Discovery (2018) 32:1397–1420
https://doi.org/10.1007/s10618-018-0583-9

Dynamic graph summarization: a tensor decomposition approach

Sofia Fernandes[1] · **Hadi Fanaee-T**[2] · **João Gama**[1]

Received: 27 January 2018 / Accepted: 2 July 2018 / Published online: 12 July 2018

Abstract
Due to the scale and complexity of todays' social networks, it becomes infeasible to mine them with traditional approaches. A possible solution to reduce such scale and complexity is to produce a compact (lossy) version of the network that represents its major properties. This task is known as graph summarization, which is the subject of this research. Our focus is on time-evolving graphs, a more complex scenario where the dynamics of the network also should be taken into account. We address this problem using tensor decomposition, which enables us to capture the multi-way structure of the time-evolving network. This property is unique and is impossible to obtain with other approaches such as matrix factorization. Experimental evaluation on five real world networks implies promising results demonstrating that tensor decomposition is quite useful for summarizing dynamic networks.

Keywords Graph summarization · Time-evolving networks · Tensor decomposition

1 Introduction

With the emerging of more complex and large-scale networks, namely from online sources (social networks, exchanged emails, …), new challenges arise: for example,

Responsible editors: Jesse Davis, Elisa Fromont, Derek Greene, Björn Bringmann.

✉ Sofia Fernandes
sdsf@inesctec.pt

Hadi Fanaee-T
hadift@medisin.uio.no

João Gama
jgama@fep.up.pt

1 LIAAD, INESC TEC, University of Porto, Porto, Portugal

2 Department of Biostatistics, University of Oslo, Oslo, Norway

the visualization of such networks, which, in other cases, would provide insights on the global behavior of the network, besides being computationally demanding, is no longer enlightening. Moreover, the storage of such networks is also an issue due to memory limitations, especially in dynamic scenarios.

In this context, the replacement of the original network by a compact representation may be a reliable approach to tackle such problems. Such idea encompasses the scope of graph summarization. In more detail, the problem of graph summarization consists of finding a succinct/concise representation of the original graph, which captures its general structure (Liu et al. 2016).

Most of the research on this field has focused on static networks. However, in this work we address the problem of graph summarization for time-evolving networks. In particular, we process the networks using a sliding time window of network snapshots, thus taking into account the temporal occurrence of the network edges. Our work falls into the category of structural pattern-based summarization, since, for each time window, the summary result is a (smaller) graph, referred to as supergraph, whose supernodes represent groups of structurally similar nodes in the original graph and whose edges reflect the interactions between those groups.

Briefly, the problem addressed in this work is given as follows: *given a large time-evolving network, how can we represent it in a more concise way so that the global structural properties are captured?* We note that this type of summarization is different from traditional community detection algorithms. Although in some cases the result of both approaches may be similar, in community detection the goal is to group nodes which are more densely connected, while in this type of summarization the goal is to group nodes which have similar connection patterns, that is, connect to a similar subset of nodes at the same instants (Liu et al. 2016).

To address this problem, we introduce tenClutS, a method which explicitly models the time dimension of dynamic graphs, thus taking advantage of the multi-way structure of evolving networks driven from a tensor decomposition model. We also propose a simplification of the kC method (Tsalouchidou et al. 2016), which is meant to work as a baseline.

Our main contributions are as follows:

– We propose tenClustS, a tensor decomposition-based method for real time structural pattern summarization of time-evolving networks. According to our empirical study, tenClustS generates summaries in considerably less time than its competitors (especially in large networks), without compromising the quality of the summaries.
– We study the impact of the clustering distance metric on the summarization results and provide evidence that this parameter is critical, having a high (structural) impact on the summarization results, including the ones obtained using the baseline methods: the cosine clustering captures the global behavior of the network, while the euclidean clustering captures local patterns.

The rest of the paper is organized as follows. In Sect. 2 we overview the related work and background theory. We formalize the problem in Sect. 3. The proposed method is described in Sect. 4 and the experiments results are depicted in Sect. 5. We present the future work and conclude the paper in Sect. 6. Thorough the paper, the terms graph and network are used interchangeably.

2 Background and related work

2.1 Tensors

Tensors may be regarded as multi-dimensional arrays. Given a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_M}$, M is referred to as order (or number of modes) of the tensor and its size is given by $N_1 \times N_2 \times \cdots \times N_M$. It is noteworthy that vectors and matrices are a particular case of tensors, with one and two dimensions, respectively. The rate of non-zero entries in a tensor is called density and its Frobenius norm is computed as follows: $||\mathcal{X}|| = \sqrt{\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_M=1}^{N_M} [\mathcal{X}(i_1, i_2, \ldots, i_M)]^2}$, with $\mathcal{X}(i_1, i_2, \ldots, i_M)$ being the value of the entry $(i_1, i_2, \ldots, i_M)$ in the tensor.

The formal operation of rearranging the tensor into a matrix is known as unfolding or matricizing. The mode-d matricization of tensor $\mathcal{X}$ is denoted by $\mathbf{X}_{(d)}$, has size $N_d \times (\prod_{i \neq d} N_i)$ and is obtained by fixing each index of mode-d and varying the other modes indexes.

2.2 CP decomposition

CANDECOMP/PARAFAC (CP) decomposition (Kolda and Bader 2009) is one of the most popular tensor decomposition methods. For the sake of simplicity, let us consider a 3-order tensor, $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$. Then the goal of CP is to find vectors $\mathbf{a}_r \in \mathbb{R}^{N_1}$, $\mathbf{b}_r \in \mathbb{R}^{N_2}$, $\mathbf{c}_r \in \mathbb{R}^{N_3}$, with $1 \leq r \leq R$, such that the approximation error, given by $||\mathcal{X} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r||$, is minimized. In this context, $\circ$ denotes the vector outer product and $R \in \mathbb{N}$ is called number of components (or factors).

The matrices containing the vectors associated with each mode are called factor matrices and are defined as follows: $\mathbf{A} = [\mathbf{a}_1 | \mathbf{a}_2 | \ldots | \mathbf{a}_R] \in \mathbb{R}^{N_1 \times R}$, $\mathbf{B} = [\mathbf{b}_1 | \mathbf{b}_2 | \ldots | \mathbf{b}_R] \in \mathbb{R}^{N_2 \times R}$ and $\mathbf{C} = [\mathbf{c}_1 | \mathbf{c}_2 | \ldots | \mathbf{c}_R] \in \mathbb{R}^{N_3 \times R}$.

CP is traditionally computed using the alternating least squares algorithm (CP-ALS) (Kolda and Bader 2009). The idea exploited in this algorithm consists in considering the minimization (sub)problems obtained by fixing all but one factor matrix in the approximation error formula and solving such (sub)problem for each factor matrix sequentially and repeatedly, until some stopping criteria is met. The initial factor matrices used in the algorithm may be obtained randomly or based on the tensor. In this work, we considered the SVD-based initialization, in which the initial factor matrix associated to mode-d is computed according the singular value decomposition (SVD) of $\mathbf{X}_{(d)} \mathbf{X}_{(d)}^T$.

2.3 Graph summarization

Static graph summarization Briefly, the goal of graph summarization is to find a compact representation of the original graph, which preserves some of its key properties. Depending on both the type of the graph and the application domain, the key features may differ.

While graphs may be represented by their canonical forms, which correspond to their class of isomorphisms (Piperno 2008), this type of representations is lossless and, therefore, equivalent to the original graph, on the contrary to our approach.

Block modeling (Doreian et al. 2005) and its variants encompass a popular technique for generating graph compact representations. Briefly, its goal consists of discovering blocks of equivalent nodes in the network and constructing a supergraph summary in which the supernodes represent the blocks and the superedges map the interactions between the nodes of the blocks. The equivalence relation usually refers to structural equivalence. Within the scope of community detection, those blocks correspond to dense sub-graphs (Abbe 2017). It is noteworthy that, for most of real-world networks, the notion of equivalence is too strict and consequently relaxed/approximated variations should be considered in order to account for deviations. For example, in the traditional approaches a node either belongs to a block or not, however, in relaxed block modeling approaches, such as in Brandes and Lerner (2010), different levels of membership are allowed. Similarly, the idea of grouping identical nodes into supernodes and constructing a summary supegraph describing the relations between those supernodes is also exploited considering other types of similarity measures and grouping strategies. In LeFevre and Terzi (2010) and Riondato et al. (2017) the nodes are also grouped based on their connection patterns, while in Gansner et al. (2005) the idea is to group nodes connected by a path of short length so that the graph global structure is not compromised. Another approach consists of replacing the frequent patterns by a single structure (Buehrer and Chellapilla 2008). Additionally, role analysis aims at unveiling the nodes/edges having a similar role in the network (for example, central or bridge nodes) (Breiger and Pattison 1978; Henderson et al. 2012; Rossi and Ahmed 2015b). In this type of analysis, nodes may be assigned to the same role even if they are (geodesic) distant.

In Shen et al. (2006) and Spielman and Teng (2011), the authors propose sparsification strategies which consist in discarding nodes and/or edges that do not provide relevant information according to some criteria. For example, in a directed network one may be interested in mining the diffusion patterns (Mathioudakis et al. 2011).

The compression of the network adjacency matrix using matrix decomposition techniques may also be regarded as a method for generating network compact representations. Traditional algorithms such as SVD have efficiency issues. However, one can find alternatives such as Adaptive Cross Approximation (ACA) (Bebendorf and Rjasanow 2003), which approximates the matrix using its rows and columns, thus allowing for a more efficient processing. In this type of methods, the resulting representation is not a graph, on the contrary to our approach.

Other related mining tasks include the mining of the most relevant structures in the network using the minimum description length (MDL) (Navlakha et al. 2008) and the representation of graphs using feature vectors, referred to as fingerprints, which are used to compute similarity between graphs (Ralaivola et al. 2005).

Time-evolving graph summarization In the context of time-evolving networks, the goal is to find a summary that describes the state of the network at a given time period. Besides the naive approaches, which consist in applying the static summarization techniques to each timestamp, we already find in literature works exploiting the dynamics of this type of networks. Shah et al. (2015) proposed a compression-oriented approach based on MDL. Moreover, in Tsalouchidou et al. (2016) the authors apply clustering techniques to the sequential concatenation of the adjacency matrices in the time period considered and use the clustering result to define the supernodes of the summary supergraph.

Similarly to the work of Tsalouchidou et al. (2016), our proposed method, tenClustS, also considers a structural pattern-based summarization for time-evolving networks: nodes are grouped according to their connectivity patterns into a supernodes of the summary supergraph. However, the work of Tsalouchidou et al. fails at preserving the natural multi-way structure of the dynamic networks. The clustering strategy used in our approach is similar to Kolda and Sun (2008), nonetheless, to the best of our knowledge, the application of such clustering result to this type of summarization has not been exploited yet.

2.3.1 Structural-pattern summarization

One of the most relevant works in the context of structural-pattern summarization in static settings was introduced by LeFevre and Terzi (2010), for query efficiency purposes. In that work, given the original static graph, the goal is to generate a summary graph in which the nodes are grouped into supernodes according to their connectivity patterns. Such a graph is referred to as supergraph. In this context, nodes sharing neighbors are more similar and, therefore, are expected to be grouped in the same supernode. Given the supernodes, the superedges weights are computed as follows:

$$A_{G'}(S_i, S_j) = \begin{cases} \frac{\sum_{l\in S_i, m\in S_j} A_G(l,m)}{|S_i||S_j|}, & \text{if } S_i \neq S_j \\ \frac{\sum_{l\in S_i, m\in S_j} A_G(l,m)}{|S_i|(|S_j|-1)} & \text{if } S_i = S_j \end{cases}, \tag{1}$$

where A_G and $A_{G'}$ are the adjacency matrices of the original graph and the supergraph, respectively.

Since distinct node groupings lead to distinct supergraphs then, given the set of possible supergraphs, the goal is to find the node partition that minimizes the reconstruction error, which is defined as:

$$RE = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} |A_G(i,j) - \bar{A}_G(i,j))|}{N^2} \tag{2}$$

where N is the number of nodes in the original graph, $s : V \rightarrow V'$ is the function which assigns a node to its supernode and $\bar{A}_G$ is the adjacency matrix of the reconstructed graph, defined so that:

$$\bar{A}_G(i,j) = \begin{cases} A_{G'}(s(i), s(j)), & \text{if } i \neq j \\ 0, & \text{if } i = j. \end{cases} \tag{3}$$

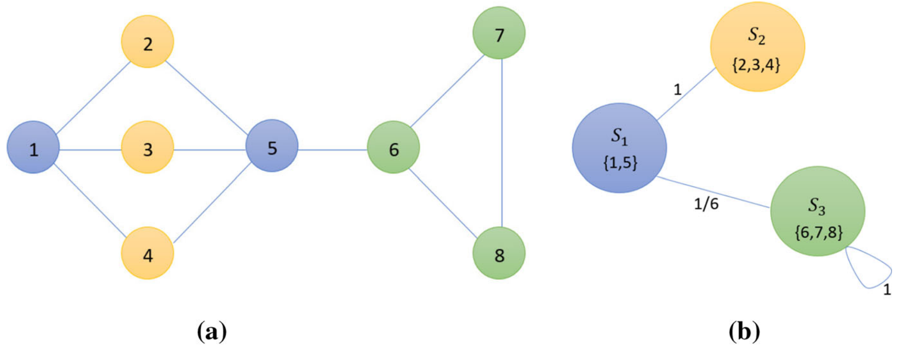


Fig. 1 Illustrative example: **a** original network and **b** its summary

Fig. 2 Adjacency matrices of the original graph (left) and the reconstructed graph (right)

$$A_G = \begin{bmatrix} 0&1&1&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&0&0&0&1&0&0&0\\ 0&1&1&1&0&1&0&0\\ 0&0&0&0&1&0&1&1\\ 0&0&0&0&0&1&0&1\\ 0&0&0&0&0&1&1&0 \end{bmatrix} \quad \bar{A}_G = \begin{bmatrix} 0&1&1&1&0&1/6&1/6&1/6\\ 1&0&0&0&1&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&0&0&0&1&0&0&0\\ 0&1&1&1&0&1/6&1/6&1/6\\ 1/6&0&0&0&1/6&0&1&1\\ 1/6&0&0&0&1/6&1&0&1\\ 1/6&0&0&0&1/6&1&1&0 \end{bmatrix}$$

As an illustrative example of this process, let us consider the static network in Fig. 1a and let us suppose that we are interested in generating a supergraph summary with 3 supernodes. Then, a possible node grouping into supernodes, S_i, could be: $S_1 = \{1, 5\}$ (in blue); $S_2 = \{2, 3, 4\}$ (in yellow); $S_3 = \{6, 7, 8\}$ (in green). Note that nodes 2, 3 and 4 are linked to the same nodes and, therefore, have exactly the same connection patterns.

Based on such grouping, the resulting supergraph would be the one illustrated in Fig. 1b. The adjacency matrix of the reconstructed network is shown in Fig. 2. In this example, the summary has a reconstruction error of ≈ 0.05.

This approach has been extended to a dynamic setting in the work of Tsalouchidou et al. (2016). The idea exploited in that adaptation was to account for the connection patterns over time, so that the nodes exhibiting constantly similar patterns over the time span are grouped into the same supernode. To meet this end, the supernodes are generated by applying clustering (k-means with cosine distance) on the sequential concatenation of the adjacency matrices in the given time window.

In more detail, given the time-evolving network, $\{A_G^t\}_1^L$, in which A_G^t is the adjacency matrix of graph G at time $t \in \{1, \ldots, L\}$, then, according to their method, the adjacency matrix of the supergraph summary, $A_{G'}$, is generated as follows:

$$A_{G'}(S_i, S_j) = \begin{cases} \frac{\sum_{t=1}^{L} \sum_{l \in S_i, m \in S_j} A_G^t(l,m)}{L|S_i||S_j|} & \text{if } S_i \neq S_j \\ 2\frac{\sum_{t=1}^{L} \sum_{l \in S_i, m \in S_j} A_G^t(l,m)}{L|S_i|(|S_j|-1)}, & \text{if } S_i = S_j \end{cases}, \tag{4}$$

which can be seen as the generalization of (1) to more than one timestamp.

Similarly, (2) was adapted to the dynamic setting:

$$RE = \frac{\sum_{t=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N}|A_G^t(i,j) - \bar{A}_G(i,j)|}{LN^2}. \tag{5}$$

where $\bar{A}_G$ is the adjacency matrix of the reconstructed window graph and satisfies $\bar{A}_G(i,j) = A_{G'}(s(i), s(j))$.

Despite being a generalization of the work of LeFevre and Terzi, there are some differences. First, in the static version, it is assumed that the graph is unweighted, however, in the Tsalouchidou et al. extension, the edges are assumed to have a weight in the interval [0, 1]. Moreover, the original graph is assumed to be simple in the static approach; nonetheless, in this dynamic extension such assumption is not imposed: self-loop edges are allowed and, as a consequence, the second case of (3) is not considered in the Tsalouchidou et al. dynamic approach.

3 Problem formulation

Formally, the problem addressed in this work may be formulated as follows:

Given an undirected, unweighted simple time-evolving graph, G, characterized by the sequence of its adjacency matrices over time, $\{A_G^t\}_1^L$, find a static weighted summary supergraph, G', characterized by the adjacency matrix, $A_{G'}$, that succinctly describes the original graph, in such a way that:

- *the supernodes of G' are homogeneous groups of nodes of the original graph in the sense that nodes in the same supernode exhibit similar connection patterns;*
- *the superedges weights reflect the level of interaction in the original graph between the nodes in the corresponding supernodes.*

The resulting supergraph summary may describe a single time stamp of the graph ($L = 1$) or cover a time window ($L > 1$).

As previously exposed, in this type of summarization, the supernodes are expected to be homogeneous, that is, nodes in the same supernode are expected to link to a similar set of nodes at the same time periods. In this context, the more homogeneous the supernodes, the better the supergraph approximates the original network (because the superedges weights will more accurately reflect the interactions between the nodes in the original network). Based on this, the level of how well the summary approximates the original network is an indicator of the quality of the summary. On the other hand, a good summary should also be a compact representation of the original network. Thus, a quality summary should be compact but approximate reasonably well the original network (more details on how to measure such indicators can be found in Sect. 5.3).

4 Proposed method: tenClustS

In the proposed method, we process the time-evolving network in real-time using a sliding-window, $\mathcal{W}$, so that at a given timestamp T, we only consider the current state of the network and the previous $L-1$ states. In other words, we generate a summary for each sequence of L consecutive adjacency matrices. The details of the method are depicted as follows.

Idea The idea exploited in this method consists in taking advantage of the 3-way structure, $nodes \times nodes \times timestamps$, of time-evolving graphs: it is expected that the tensor decomposition result unveils the hidden connection patterns in the network.

Data type and modeling The current time window, $\mathcal{W}$, of the dynamic graph is modeled as a 3-order tensor formed by the sequence of adjacency matrices over that time period so that:

$$\mathcal{W}(i, j, t) = A_G^t(i, j),$$

for $i, j \in \{1, \dots, N\}$ and $t \in \{1, \dots, L\}$.

Method Given the current time window of the dynamic graph, $\mathcal{W}$, tenClustS consists in:

1. *Generating nodes representation* Apply CP-ALS to the current tensor window, $\mathcal{W}$, using R number of components:

$$\mathcal{W} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r,$$

   where $\mathbf{a}_r$ and $\mathbf{b}_r$ are associated with node dimensions and $\mathbf{c}_r$ is associated with the time dimension. Based on this, set the nodes representation as one of the factor matrices associated to the nodes, for example, $\mathbf{A} = [\mathbf{a}_1|\mathbf{a}_2|\dots|\mathbf{a}_R]$. Recall that $\mathbf{A} \in \mathbb{R}^{N\times R}$, and therefore, node i is described by a R-dimension vector corresponding to the ith row of matrix $\mathbf{A}$. Note also that, since the networks are assumed to be undirected (and therefore, the corresponding adjacency matrices are symmetric), considering either factor matrices $\mathbf{A}$ or $\mathbf{B}$ should lead to similar results.
2. *Grouping nodes into supernodes* Generate the supernodes assignment by applying k-means, with euclidean distance, on the rows of $\mathbf{A}$.
3. *Building supergraph adjacency matrix* Generate supergraph adjacency matrix, $A_{G'}$, according to

$$A_{G'}(S_i, S_j) = \begin{cases} \frac{\sum_{t=1}^{L} \sum_{l\in S_i, m\in S_j} A_G^t(l,m)}{L|S_i||S_j|}, & \text{if } S_i \neq S_j \\ \frac{\sum_{t=1}^{L} \sum_{l\in S_i, m\in S_j} A_G^t(l,m)}{L|S_i|(|S_j|-1)} & \text{if } S_i = S_j \end{cases}. \tag{6}$$

   We note that this assignment differs from (4) only in the self-loop weight computation ($S_i = S_j$): we do not multiply the sum by two, because since the adjacency matrix is symmetric, each edge is already accounted twice.

In the nodes grouping stage, we considered only the euclidean distance in k-means because the small magnitude of some entries in the nodes representation, resulting from the application of the CP-ALS tensor decomposition algorithm, would compromise the distance computation. We leave this issue to be addressed in future work.

5 Experiments

The experiments were carried out using MATLAB along with Tensor Toolbox (Bader et al. 2015; Bader and Kolda 2007) in a machine with 2.7 GHz processor and 12 GB RAM. The code is available at https://github.com/ssfernandes/tenClustS.

5.1 Datasets

In these experiments we considered five real world dynamic networks publicly available, which are summarized in Table 1. The size of each network is represented in $nodes \times nodes \times timestamps$ format.

Each dataset consisted of a time-evolving network in which there was a link from entity i to entity j at time t if entities i and j interacted during such instant. The type of entity and interaction depended on the dataset. In all networks, but Hepth, entities represented people, in Hepth they represented papers. Moreover, the interactions represented (i) emails exchanged in Enron network; (ii) exchanged phone calls in Friends&Family network; (iii) co-authorship in DBLP; (iv) human contact interaction in InfectiousPatterns; and (v) citations in Hepth network.

In a pre-processing stage, the nodes in Enron dataset having no metadata were discarded. In the Friends&Family, we discarded the missed calls and all the calls involving individuals not under study at the time the data was collected. Moreover, all the time-evolving networks were subjected to a pre-processing step in which the edges weights and the self-loops edges were discarded. Finally, since the number of links was extremely small in some of the timestamps in the datasets, we discarded the timestamps having a small number of links. In particular, we truncated Enron to monthly timestamps 21–40; Friends&Family to monthly timestamps 9–16; InfectiousPatterns to the first 20 timestamps and, finally, Hepth to timestamps 53–72.

Table 1 Datasets summary

Network	Acronym	Content	Size	Density
Enron Priebe et al. (2005)	ERN	Email exchange	$130 \times 130 \times 21$	1.44E−2
Friends&Family (Aharony et al. 2011)	F&F	Phone calls	$129 \times 129 \times 8$	1.78E−2
DBLP (Desmier et al. 2012)	DBLP	Co-authorship	$2723 \times 2723 \times 9$	1.64E−3
Infectious patterns (Isella et al. 2011)	INFPTRN	Contacts	$10{,}970 \times 10{,}972 \times 20$	7.73E−6
Hepth (Leskovec et al. 2005; Rossi and Ahmed 2015a)	HPH	Citations	$22{,}906 \times 22{,}906 \times 20$	1.05E−5

5.2 Design of experiments

The networks were processed using a sliding-window, $\mathcal{W}$, of length L so that when a new adjacency matrix was available, at time $T+1$, we slid the previous window to include the new arriving matrix and discarded the oldest matrix from the previous window (that is, consecutive time windows had an overlap of $L-1$ instants). By considering a sliding window, we not only discarded outdated data, but also reduced the amount of data to process and store. Moreover, we considered the maximal overlap, $L-1$, between consecutive windows, to guarantee more stability between two consecutive windows. For each time window, a summary was generated and evaluated according to the evaluation metrics described in Sect. 5.3.

5.3 Evaluation metrics

Reconstruction error (RE) To assess the quality of the summaries we considered the summary reconstruction error (RE) (Tsalouchidou et al. 2016), given by expression (5). The lower the reconstruction error, the better the summary approximates the original graph.

Compression cost (CC) Each summary is associated with a compression level which depends on both the number of nodes and edges of the supergraph. In order to measure the "complexity" of the generated summaries, we measure the number of bits needed to store their topology. Based on this, given the summary, we compute its compression cost as the number of bits needed to store it (Mitchell 1997). In particular, given the number of supernodes of the summary, $|V_{summary}|$, and corresponding number of edges, $|E_{summary}|$, the compression cost is computed as follows: $CC = 2 \times \lceil \log_2(|V_{summary}|) \rceil \times |E_{summary}|$. Low compression values are preferable, specially in scenarios with few storage resources.

Running time Additionally, we also considered the average running time (in seconds) of the methods over the several time windows.

We set a soft running time limit of 10 min per time window for generating the supernodes assignment (after reaching 10 min, the search for the best assignment is stopped and the best assignment found so far is the one considered for the summary of the current time window).

5.4 Baseline methods for comparison

Proposed baselines kM_{euc} and $\mathrm{kM_{cos}}$ These methods may be described as simplifications of the kC method (Tsalouchidou et al. 2016) (described in Sect. 2.3.1). The idea is to use the sum of the adjacency matrices over the time window. Thus, the current time window, $\mathcal{W}$, of the dynamic graph is modeled by a single (adjacency) matrix, $\bar{A}$, which results from summing the adjacency matrices of the network in the L timestamps:

$$\bar{A}(i, j) = \sum_{t=1}^{L} A_G^t(i, j), \tag{7}$$

by recalling that A_G^t denotes the adjacency matrix of graph G at time t and L is the number of timestamps in the time window. The grouping of nodes into supernodes is carried out by applying k-means to the rows of $\bar{A}$. Finally, the supergraph adjacency matrix is constructed according to (6).

The difference between the two variants of this method is the distance metric used in k-means. As the name suggests, kM_{euc} and kM_{cos} employ, respectively, the euclidean and cosine distances.

kC Since our approach may be classified as a structural-pattern oriented approach, we considered a structural-pattern oriented approach as baseline. In particular, we considered the clustering-based method (kC) proposed by Tsalouchidou et al. (2016) (described in Sect. 2.3.1).

WSBM We also considered a stochastic block modeling approach which groups nodes with similar connection-patterns. In particular, we considered the stochastic block model for weighted networks proposed by Aicher et al. (2014).

Since this method was designed for static networks, in order to apply it to our setting, we collapsed each network window using (7). The blocks detected corresponded to the supernodes and the superedges were computed according to (6).

5.5 Parameter setting

Window length (WL) In order to have a more complete understanding of the behavior of the models, we considered window lengths of 3, 6, 9 and 12 timestamps. The maximum window length considered in each dataset depended on the number of timestamps available.

Number of tensor decomposition components selection criteria We carried out the estimation of the best number of components in CP using AUTOTEN (Papalexakis 2016), a scalable implementation of CORCONDIA (Bro and Kiers 2003).

In particular, the number of components was chosen so that the data was correctly modeled in the majority of the windows: the estimated number of components (see Table 2) was the average of the values obtained by AUTOTEN for the several windows. *Number of supernodes selection criteria* Since for kC, kM_{euc}, kM_{cos} and tenClustS, the nodes grouping into summary supernodes may be interpreted as a clustering problem, we applied the Elbow method (Kodinariya and Makwana 2013) as criteria to select the number of supernodes.

Given the maximum number of clusters allowed, $Kmax$, the Elbow method consists in applying k-means sequentially by varying the number of clusters from 2 to $Kmax$ and computing, for each clustering generated, the overall within-cluster sum of point-to-centroid distances. According to this method, the best number of clusters is associated with the "elbow" of the curve of such values.

Since the data representation differs in each approach, we applied the elbow method for each of such representations. The results are depicted in Table 3. The number of

Table 2 Average of the estimated CP number of components over the windows using AUTOTEN

Dataset	WL	Ncomps
Enron	3	4.11 ± 2.56
	6	3.40 ± 0.95
	9	4.67 ± 0.78
	12	4.67 ± 1.00
Friends&Family	3	2.00 ± 0.00
	6	2.67 ± 0.47
DBLP	3	8.29 ± 4.23
	6	7.00 ± 5.87
Hepth	3	13.83 ± 0.37
	6	10.08 ± 8.75
	9	11.58 ± 8.58
	12	8.11 ± 5.40
InfectiousPatterns	3	13.59 ± 1.78
	6	9.31 ± 1.20
	9	10.92 ± 4.66
	12	9.22 ± 1.30

clusters was estimated considering only the first data window, such number was used for the remaining windows.

In the case of the WSBM method, we estimated the number of supernodes for this method as the rounded mean of the values estimated in the other approaches. The goal of considering this strategy is to guarantee that the summaries generated by all the methods have similar complexity (the values are shown in Table 3).

5.6 Illustrative examples of summaries

With the goal of understanding the differences between the summaries generated by the various methods, we illustrate some of the summarization results.

For brevity and simplicity purposes, we only consider the first window of the (small) `Enron` network, however, the behavior here illustrated was also observed in the generality of the other time windows and datasets as well. In order to facilitate the visualization of the network window, we collapsed the network window into a single static network, which we refer to as view. The view was obtained as follows:

– the nodes in the view encompass all the nodes that had at least a link in the time window;
– two nodes are linked by an edge in the view if they were linked in at least one timestamp of the window.

In Fig. 3, we present 5 views of the first window of `Enron` dataset. Each view illustrates the supernodes assignment obtained by one of the five methods under study. The number of supernodes was set according to Table 3. The colors of nodes represent the supernodes so that if two nodes have the same color in the view, then they were

Table 3 Number of supernodes estimated

Dataset	WL	WSBM	kC	kM_{euc}	kM_{cos}	tenClustS
Enron	3	15	17	17	16	9
	6	13	15	16	13	7
	9	12	13	15	12	9
	12	12	13	12	13	10
Friends&Family	3	10	12	9	10	8
	6	9	10	9	10	8
DBLP	3	15	14	16	17	11
	6	15	15	17	17	11
Hepth	3	17	19	14	17	16
	6	15	17	14	19	11
	9	13	16	11	11	15
	12	14	16	12	15	11
InfectiousPatterns	3	14	13	15	13	16
	6	15	17	14	17	12
	9	17	20	16	21	12
	12	17	16	19	21	11

grouped in the same supernode by the corresponding method. For the sake of simplicity, we did not include in the view two other connected components having less than four nodes each.

We observed that the kC result captured some community structure of the network, as it was visible, for example, in the supernodes associated with colors orange (nodes 46, 49, 84, 104, 105 and 119) and dark green (nodes 47, 88, 94 and 100). This result was, in some way, expected since, within communities, the nodes are more likely to share neighbors, and therefore, to have similar connectivity patterns. A similar result was observed when considering kM_{cos}.

The results of methods kM_{euc} and tenClustS were quite different from the previous two. In these two approaches, we verified that the supernodes reflected also the activity level of the node. For example, the nodes of the supernode associated with the grey color were usually nodes of small degree. On the other hand, the most active nodes, 21 and 64, which were associated with superior roles in the company, were defined as singleton supernodes.

When comparing the results of these four approaches, we also verified that the supernodes size in kC and kM_{cos} summaries, was balanced, on the contrary, to the other two approaches. In particular, in kM_{euc} and tenClustS, we obtained a large supernode containing the less active nodes, while the remaining supernodes size was small.

Regarding WSBM, its behavior was similar to kC and kM_{cos}, in the sense that in these three approaches the size of the supernodes was balanced. However, the community-like grouping observed in kC and kM_{cos} was not modeled by WSBM.

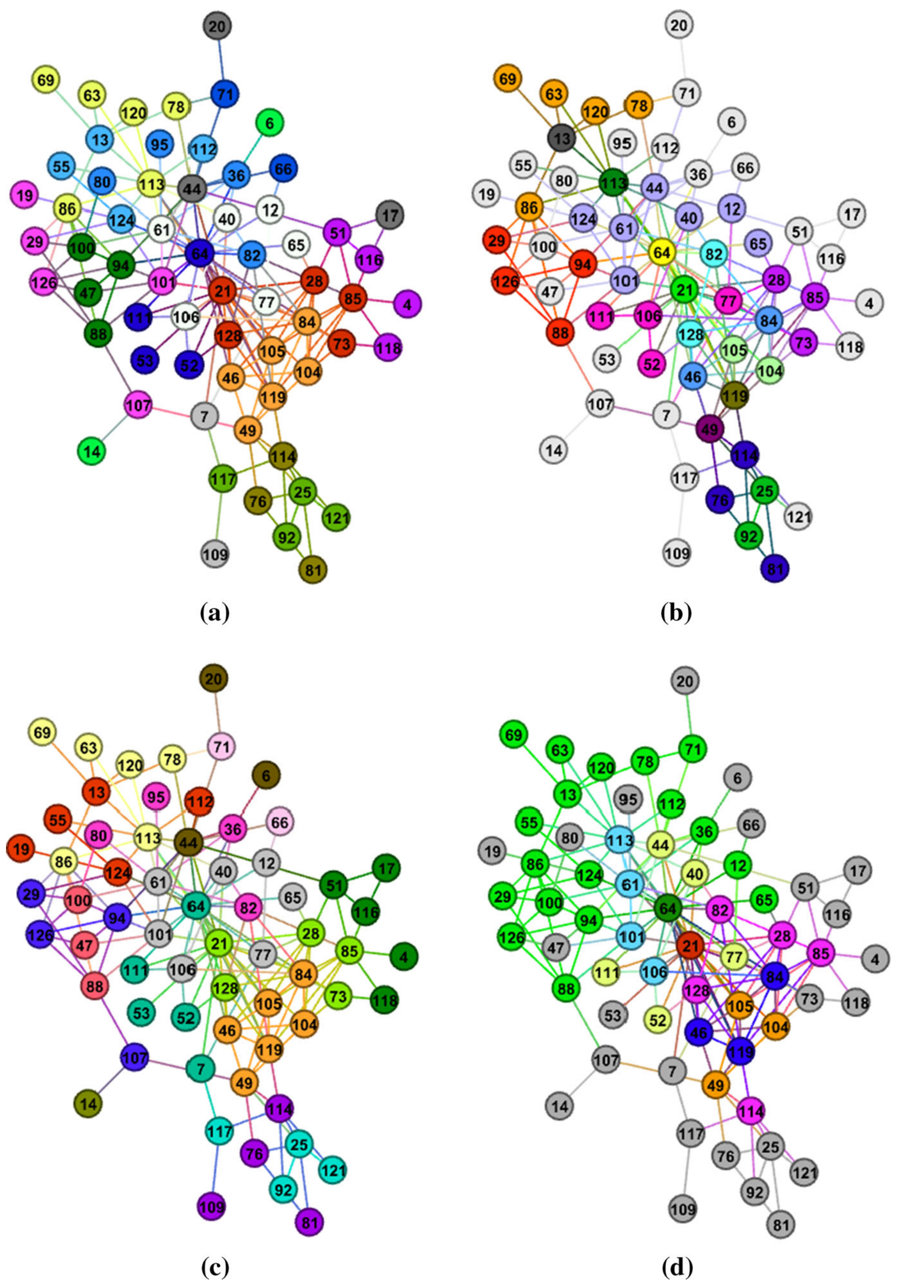
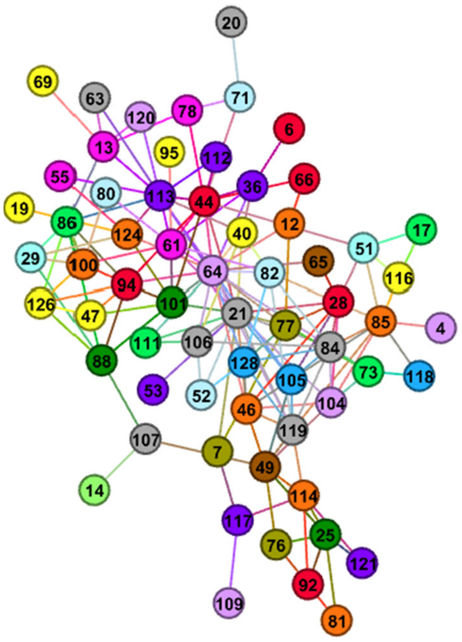


Fig. 3 Enron first window views with nodes colored according to their supernode grouping using: **a** kC, **b** kM_{euc}, **c** kM_{cos}, **d** tenClustS and **e** WSBM

Fig. 3 continued



(e)

In order to have a deeper understanding of the result of the several methods, we analyzed the adjacency matrices of the summaries generated (the corresponding supergraphs are illustrated in Fig. 4). In this context, Fig. 5 shows the distribution of the non-zero values in the adjacency matrices of the summaries generated by the four methods. Once more we observe a pattern: clustering-based methods using the same distance metric exhibited similar properties. In particular, we verified that the non-zero weights associated to the kC and kM$_{\text{cos}}$ summaries ranged in]0; 0, 6[, while, in kM$_{\text{euc}}$ and tenClustS, the values ranged in]0, 1]. This may be explained by the grouping obtained in each method. Note that, in kM$_{\text{euc}}$ and tenClustS, the size of most supernodes was smaller than in the other two approaches, thus, allowing to capture locally stronger connection patterns. For example, since nodes 46 and 84 were linked to node 21 in the original graph, then, the supernodes S_3 and S_8 in the kM$_{\text{euc}}$ summary, corresponding to {21} and {46, 84}, were connected by a superedge of weight 1. Finally, in the WSBM summary, we observed that the interval of values assumed by the superedges weights was the most compact (with all the values being less than 0.2).

We also carried out this analysis in other windows and datasets and the behavior here observed was also verified in such settings, in particular:

- the size of supernodes generated by WSBM, kC and kM$_{\text{cos}}$ was balanced, while in kM$_{\text{euc}}$ and tenClustS, we obtained a large supernode and the remaining were small;
- the non-zero superedges weights assumed a wider range of values in kM$_{\text{euc}}$ and tenClustS.

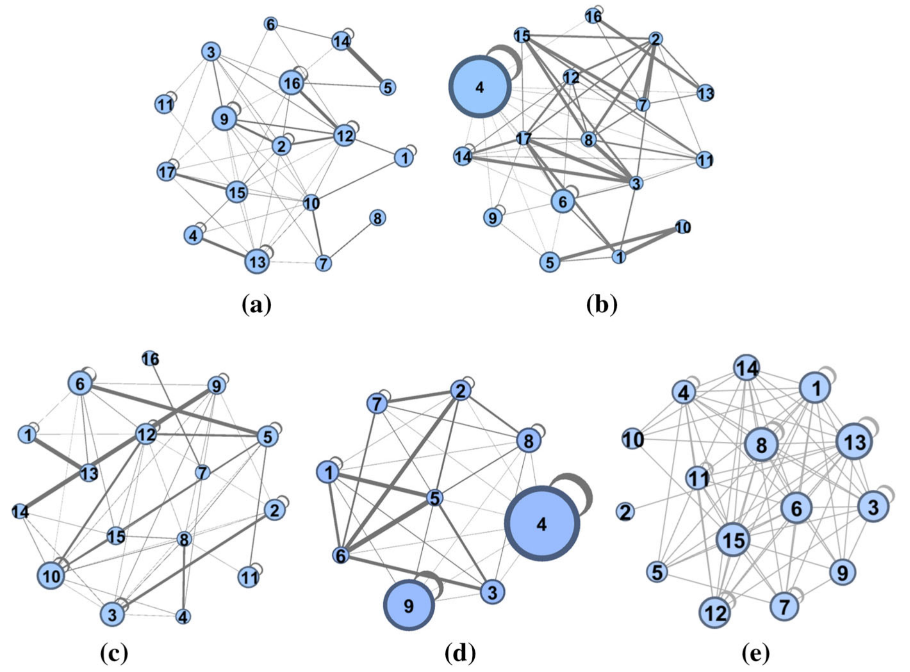


Fig. 4 Enron first window summaries generated using: **a** kC, **b** kM_{euc}, **c** kM_{cos}, **d** tenClustS and **e** WSBM. The size of the supernodes was set according to the number of nodes composing it so that a large size was assigned to supernodes composed by a large number of nodes

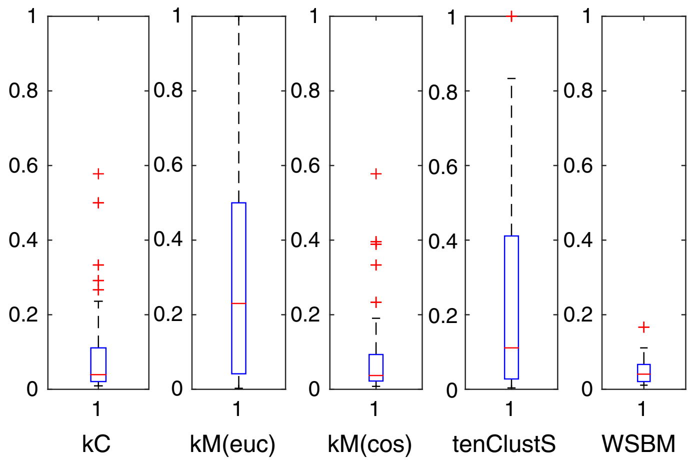


Fig. 5 Boxplot of the non-zero weights in the summaries adjacency matrices obtained by the methods under study

This analysis suggests that the distance metric considered in k-means has a strong impact on the structure of the summary: on the one hand, the summaries generated using cosine distance (kC and kM_{cos}) captured the global connection patterns; while on the other hand, the summaries generated using euclidean distance (kM$_{\text{COS}}$ and tenClustS) allowed to capture local patterns.

In other words, the summaries generated using the cosine distance approximated all the network "evenly". When considering the euclidean distance, we observed that there were small sub-regions which were considerably more well approximated than the remaining, which were neglected. In this sense the summarization results obtained by each metric may be seen as complementary.

5.7 Results

A quality method should generate good summaries in few time. Based on this, we proceeded with our experiments with the goal of understanding how good was the performance of the proposed procedures regarding the three evaluation metrics. In order to facilitate the understanding of the results, we analyze the performance of the methods regarding each metric separately.

Reconstruction error The reconstruction values of the summaries generated by the five methods are shown in Table 4. As it can be seen, kM$_{\text{euc}}$ was the method exhibiting the lowest error in almost all settings, followed by tenClustS. Among the clustering-based approaches kM$_{\text{COS}}$ exhibited the highest errors.

These results may be justified by the behavior observed in Sect. 5.6: when considering the euclidean metric, strong patterns were well approximated in the summary thus reducing the reconstruction error, however, little information was kept regarding nodes having less activity. Not preserving the information of less active nodes has less impact on the approximation quality than not preserving the strong connectivity patterns. Because of this, the summarization results when considering cosine distance, usually lead to a higher error: both weak and strong connection patterns were approximated in the same manner.

It is important to observe that in the `InfectiousPatterns` dataset, the summaries generated by kC and kM$_{\text{COS}}$ were equal when considering window lengths of 3 and 6.

Compression cost With respect to the compression cost (Table 5), we verified that, globally, tenClustS exhibited the lowest values. However, it is important to note that the summaries generated by such method usually had a smaller number of supernodes (recall Table 3).

In more detail, by analyzing Tables 3 and 5, we verified that the lowest compression cost was always associated with the summary with fewer supernodes, as it would be expected.

Time Regarding the approaches running time (Table 6), we verified that the tenClustS was the fastest method in almost all settings. It is noteworthy that, in the large datasets, tenClustS running time was considerably smaller than WSBM and kM$_{\text{euc}}$.

Moreover, we also observed that, despite generating the same summaries in `InfectiousPatterns` dataset with window lengths of 3 and 6, the time required by kM$_{\text{COS}}$ was considerable smaller than the one required by kC. Such time reduction

Table 4 Average reconstruction error (RE) results (with best performance marked in bold)

Dataset	WL	WSBM	kC	kM_{euc}	kM_{cos}	tenClustS
ERN	3	2.83 ± 0.61(E−2)	2.36 ± 0.50(E−2)	**1.97** ± **0.44(E−2)**	2.41 ± 0.50(E−2)	2.24 ± 0.49(E−2)
	6	2.93 ± 0.41(E−2)	2.57 ± 0.38(E−2)	**2.21** ± **0.39(E−2)**	2.63 ± 0.37(E−2)	2.48 ± 0.40(E−2)
	9	2.80 ± 0.45(E−2)	2.65 ± 0.36(E−2)	**2.34** ± **0.37(E−2)**	2.70 ± 0.35(E−2)	2.49 ± 0.40(E−2)
	12	2.72 ± 0.32(E−2)	2.70 ± 0.29(E−2)	**2.47** ± **0.31(E−2)**	2.74 ± 0.24(E−2)	2.54 ± 0.32(E−2)
F&F	3	3.58 ± 0.18(E−2)	3.62 ± 0.18(E−2)	**2.63** ± **0.08(E-2)**	3.69 ± 0.18(E−2)	2.93 ± 0.16(E−2)
	6	3.70 ± 0.07(E−2)	3.72 ± 0.11(E−2)	**2.75** ± **0.05(E−2)**	3.73 ± 0.14(E−2)	2.94 ± 0.08(E−2)
DBLP	3	3.16 ± 1.48(E−3)	3.22 ± 1.52(E−3)	**3.02** ± **1.43(E−3)**	3.21 ± 1.52(E−3)	3.07 ± 1.45(E−3)
	6	3.19 ± 0.85(E−3)	3.22 ± 0.86(E−3)	**3.10** ± **0.83(E−3)**	3.22 ± 0.86(E−3)	3.14 ± 0.84(E−3)
HPH	3	3.89 ± 1.05(E−5)	3.78 ± 1.09(E−5)	3.59 ± 1.04(E−5)	3.84 ± 1.14(E−5)	**3.50** ± **1.01(E−5)**
	6	4.15 ± 1.05(E−5)	4.18 ± 1.08(E−5)	**4.02** ± **1.03(E−5)**	4.16 ± 1.07(E−5)	4.05 ± 1.05(E−5)
	9	4.19 ± 0.84(E−5)	4.20 ± 0.84(E−5)	4.11 ± 0.82(E−5)	4.22 ± 0.85(E−5)	**4.09** ± **0.82(E−5)**
	12	4.22 ± 0.48(E−5)	4.23 ± 0.48(E−5)	**4.15** ± **0.47(E−5)**	4.24 ± 0.48(E−5)	4.17 ± 0.48(E−5)
INFPTRN	3	2.52 ± 0.81(E−5)	2.53 ± 0.82(E−5)	2.45 ± 0.81(E−5)	2.53 ± 0.82(E−5)	**2.44** ± **0.81(E−5)**
	6	2.69 ± 0.39(E−5)	2.68 ± 0.39(E−5)	2.66 ± 0.40(E−5)	2.68 ± 0.39(E−5)	**2.66** ± **0.39(E−5)**
	9	2.61 ± 0.18(E−5)	2.60 ± 0.18(E−5)	**2.58** ± **0.18(E−5)**	2.60 ± 0.18(E−5)	2.59 ± 0.18(E−5)
	12	2.62 ± 0.19(E−5)	2.62 ± 0.19(E−5)	**2.60** ± **0.19(E−5)**	2.62 ± 0.19(E−5)	2.61 ± 0.19(E−5)

Table 5 Average compression cost (CC) results (with best performance marked in bold)

Dataset	WL	WSBM	kC	kM_{euc}	kM_{cos}	tenClustS
ERN	3	1384 ± 157	1016 ± 162	1331 ± 197	754 ± 102	$\mathbf{510 \pm 60}$
	6	1202 ± 82	883 ± 95	1190 ± 129	728 ± 68	$\mathbf{278 \pm 9}$
	9	903 ± 227	843 ± 96	1180 ± 122	713 ± 51	$\mathbf{595 \pm 45}$
	12	789 ± 159	940 ± 58	827 ± 62	876 ± 69	$\mathbf{713 \pm 51}$
F&F	3	407 ± 26	599 ± 55	453 ± 31	452 ± 50	$\mathbf{284 \pm 16}$
	6	355 ± 26	552 ± 29	456 ± 0	520 ± 24	$\mathbf{324 \pm 21}$
DBLP	3	1437 ± 301	1545 ± 34	1597 ± 170	2761 ± 173	$\mathbf{895 \pm 54}$
	6	1632 ± 111	1796 ± 8	2323 ± 60	2870 ± 16	$\mathbf{960 \pm 9}$
HPH	3	1702 ± 476	1483 ± 333	$\mathbf{1057 \pm 163}$	1139 ± 263	1408 ± 218
	6	1443 ± 168	1877 ± 228	1329 ± 141	1850 ± 329	$\mathbf{895 \pm 49}$
	9	1221 ± 67	1745 ± 152	943 ± 24	$\mathbf{720 \pm 127}$	1729 ± 47
	12	1460 ± 64	1758 ± 175	1111 ± 53	1185 ± 186	$\mathbf{952 \pm 21}$
INFPTRN	3	470 ± 64	$\mathbf{356 \pm 51}$	600 ± 87	$\mathbf{356 \pm 51}$	700 ± 122
	6	575 ± 43	567 ± 45	453 ± 38	567 ± 45	$\mathbf{419 \pm 45}$
	9	729 ± 67	683 ± 50	489 ± 38	740 ± 76	$\mathbf{347 \pm 16}$
	12	1152 ± 61	521 ± 56	732 ± 34	746 ± 73	$\mathbf{305 \pm 30}$

is justified by the usage of a smaller dimension data representation when considering kM_{cos}.

Finally, we also observed that WSBM exhibited the highest running times, reaching the maximum running time allowed in some of the largest datasets.

Global observations Based on the previous experiments, we observed that, under this experimental setting: (i) kM_{euc} exhibited the lowest reconstruction values, but required considerable more time than kM_{cos} and tenClustS in the larger datasets; (ii) tenClustS exhibited the lowest running times and compression costs without substantially compromising the quality of the summary in terms of reconstruction error.

6 Conclusions

In this work we propose tenClustS, which is, to the best of our knowledge, the first tensor decomposition-based method for the structural-pattern summarization of dynamic graphs. The idea exploited in this work consists of resorting to tensor decomposition to simultaneously capture the dynamics of such networks, while reducing the dimensionality of the networks representation.

The experimental evaluation results show that the proposed method exhibits a trade-off performance between reconstruction error and running time, with tenClustS being the fastest method considered in this study, without compromising dramatically the quality of the summaries in terms of reconstruction error.

Table 6 Average running time results (with best performance marked in bold)

Data	WL	WSBM	kC	kM$_{euc}$	kM$_{cos}$	tenClustS
ERN	3	3.16 ± 2.62(E+0)	4.27 ± 0.32(E−1)	1.30 ± 0.15(E+0)	2.71 ± 0.20(E−1)	**2.12 ± 1.00(E−1)**
	6	3.15 ± 1.35(E+0)	3.21 ± 0.25(E−1)	8.93 ± 0.82(E−1)	2.16 ± 0.11(E−1)	**1.51 ± 0.69(E−1)**
	9	1.63 ± 1.13(E+1)	3.60 ± 0.54(E−1)	8.98 ± 0.86(E−1)	**2.03 ± 0.09(E−1)**	3.32 ± 0.55(E−1)
	12	3.05 ± 1.60(E+1)	4.08 ± 0.17(E−1)	7.02 ± 0.52(E−1)	**2.35 ± 0.09(E−1)**	3.58 ± 0.66(E−1)
F&F	3	1.00 ± 0.26(E+1)	4.19 ± 1.17(E−1)	5.15 ± 0.91(E−1)	1.59 ± 0.07(E−1)	**1.12 ± 0.05(E−1)**
	6	1.19 ± 0.10(E+1)	2.27 ± 0.10(E−1)	4.61 ± 0.29(E−1)	1.68 ± 0.03(E−1)	**1.32 ± 0.07(E−1)**
DBLP	3	6.17 ± 0.07(E+2)	1.49 ± 0.52(E+1)	3.38 ± 2.02(E+2)	6.39 ± 1.92(E+0)	**5.13 ± 2.35(E−1)**
	6	6.20 ± 0.07(E+2)	3.19 ± 0.37(E+1)	4.84 ± 0.81(E+2)	7.74 ± 1.00(E+0)	**7.93 ± 1.95(E−1)**
HPH	3	2.53 ± 1.27(E+2)	1.01 ± 0.26(E+0)	1.22 ± 0.44(E+1)	**6.78 ± 1.49(E−1)**	9.86 ± 4.17(E−1)
	6	4.99 ± 1.00(E+2)	3.62 ± 0.86(E+0)	4.40 ± 2.35(E+1)	1.62 ± 0.35(E+0)	**1.14 ± 0.59(E+0)**
	9	5.49 ± 0.52(E+2)	8.60 ± 1.63(E+0)	7.68 ± 2.52(E+1)	2.26 ± 0.73(E+0)	**2.03 ± 1.26(E+0)**
	12	6.17 ± 0.14(E+2)	1.78 ± 0.23(E+1)	2.49 ± 1.14(E+2)	5.17 ± 0.68(E+0)	**2.35 ± 1.02(E+0)**
INFPTRN	3	3.78 ± 1.27(E+2)	6.06 ± 1.30(E−1)	8.70 ± 3.73(E+0)	3.82 ± 0.57(E−1)	**3.78 ± 0.35(E−1)**
	6	6.01 ± 0.21(E+2)	1.88 ± 0.30(E+0)	3.45 ± 1.01(E+1)	1.05 ± 0.15(E+0)	**2.71 ± 0.24(E−1)**
	9	6.12 ± 0.04(E+2)	5.15 ± 0.88(E+0)	9.79 ± 2.75(E+1)	2.20 ± 0.26(E+0)	**3.27 ± 0.35(E−1)**
	12	6.21 ± 0.07(E+2)	1.13 ± 0.22(E+1)	2.35 ± 0.44(E+2)	3.97 ± 0.69(E+0)	**3.10 ± 0.20(E−1)**

Our analysis of the summarization results suggests that the distance metric used in the clustering phase is a critical parameter, regardless of the network representation considered. In particular, we observed that the summaries generated using each metric captured complementary patterns: while the cosine clustering captured the global behavior of the network, the euclidean clustering captured local patterns.

Future research directions include the extension of these methods to an online/incremental fashion and an adaptation of tenClustS for cosine distance metric.

Acknowledgements This work is financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013. Sofia Fernandes also acknowledges the support of FCT via the PhD scholarship PD/BD/114189/2016. The authors would also like to acknowledge the SocioPatterns collaboration for making the Infectious Patterns dataset available.

7 Appendix

The results of the signed Wilcoxon rank tests are shown in Table 7. We did not apply the tests on `Friends&Family` and `DBLP` datasets, when using a window length of 6, because the number of available windows was extremely small (3 and 4, respectively).

Table 7 *p* Values of the signed Wilcoxon rank tests involving tenClustS, with respect to: reconstruction error (top table), compression cost (middle table) and running time (bottom table)

Dataset	WL	SBM versus tCS	kM_e versus tCS	kC versus tCS	kM_c versus tCS
Enron	3	**1.96E−04**	1.96E−04	**1.96E−04**	**1.96E−04**
	6	**6.10E−05**	3.05E−04	**6.10E−05**	**6.10E−05**
	9	**4.88E−04**	4.88E−04	**4.88E−04**	**4.88E−04**
	12	**3.91E−03**	3.91E−03	**3.91E−03**	**3.91E−03**
Friends&Family	3	**3.13E−02**	3.13E−02	**3.13E−02**	**3.13E−02**
DBLP	3	**1.56E−02**	1.56E−02	**1.56E−02**	**1.56E−02**
Hepth	3	**1.96E−04**	**1.96E−04**	**1.96E−04**	**1.96E−04**
	6	**6.10E−05**	6.10E−05	**6.10E−05**	**6.10E−05**
	9	**4.88E−04**	**4.88E−04**	**4.88E−04**	**4.88E−04**
	12	**3.91E−03**	3.91E−03	**7.81E−03**	**3.91E−03**
InfectPatterns	3	**1.96E−04**	**1.96E−04**	**1.96E−04**	**1.96E−04**
	6	**6.10E−05**	**6.10E−05**	**6.10E−05**	**6.10E−05**
	9	**4.88E−04**	4.88E−04	**4.88E−04**	**4.88E−04**
	12	**3.91E−03**	3.91E−03	**3.91E−03**	**3.91E−03**

Table 7 continued

Dataset	WL	SBM versus tCS	kM$_e$ versus tCS	kC versus tCS	kM$_c$ versus tCS
Enron	3	**1.96E−04**	**1.96E−04**	**1.96E−04**	**1.95E−04**
	6	**6.10E−05**	**6.10E−05**	**6.10E−05**	**6.10E−05**
	9	**2.93E−03**	**4.88E−04**	**4.88E−04**	**4.88E−04**
	12	**2.97E−01**	**3.91E−03**	**2.34E−02**	**7.81E−03**
Friends&Family	3	**3.13E−02**	**3.13E−02**	**3.13E−02**	**3.13E−02**
DBLP	3	**1.56E−02**	**1.56E−02**	**1.56E−02**	**1.56E−02**
Hepth	3	**1.48E−02**	2.76E−01	**1.95E−04**	**4.93E−04**
	6	**6.10E−05**	**6.10E−05**	**6.10E−05**	**6.10E−05**
	9	4.88E−04	3.88E−01	**4.88E−04**	4.88E−04
	12	**3.91E−03**	**3.91E−03**	**3.91E−03**	**1.17E−02**
InfectPatterns	3	1.95E−04	**1.95E−04**	1.95E−04	1.95E−04
	6	**6.10E−05**	**6.10E−05**	**6.10E−05**	**6.10E−05**
	9	**4.88E−04**	**4.88E−04**	**4.88E−04**	**4.88E−04**
	12	**3.91E−03**	**3.91E−03**	**3.91E−03**	**3.91E−03**
Enron	3	**1.96E−04**	**2.33E−04**	**1.96E−04**	**4.97E−03**
	6	**6.10E−05**	**1.22E−04**	**6.10E−05**	**8.36E−03**
	9	**4.88E−04**	**4.88E−04**	**4.88E−04**	4.88E−04
	12	**3.91E−03**	**3.91E−02**	**3.91E−03**	3.91E−03
Friends&Family	3	**3.13E−02**	**3.13E−02**	**3.13E−02**	**3.13E−02**
DBLP	3	**1.56E−02**	**1.56E−02**	**1.56E−02**	**1.56E−02**
Hepth	3	**1.96E−04**	3.27E−01	**1.96E−04**	1.96E−04
	6	**6.10E−05**	**6.10E−05**	**6.10E−05**	**8.36E−03**
	9	**4.88E−04**	**4.88E−04**	**4.88E−04**	2.33E−01
	12	**3.91E−03**	**3.91E−03**	**3.91E−03**	**3.91E−03**
InfectPatterns	3	**1.96E−04**	**3.27E−04**	**3.27E−04**	7.77E−01
	6	**6.10E−05**	**6.10E−05**	**6.10E−05**	**6.10E−05**
	9	**4.88E−04**	**4.88E−04**	**4.88E−04**	**4.88E−04**
	12	**3.91E−03**	**3.91E−03**	**3.91E−03**	**3.91E−03**

Statistically significant values in bold (at a significance level of 0.05) when tenClustS was the winning strategy. For brevity, WSBM, kM$_{euc}$, kM$_{cos}$ and tenClustS are denoted as SBM, kM$_e$, kM$_c$ and tCS, respectively

References

Abbe E (2017) Community detection and stochastic block models: recent developments. arXiv preprint arXiv:1703.10146

Aharony N, Pan W, Ip C, Khayal I, Pentland A (2011) Social fMRI: investigating and shaping social mechanisms in the real world. Pervasive Mob Comput 7(6):643–659

Aicher C, Jacobs AZ, Clauset A (2014) Learning latent block structure in weighted networks. J Complex Netw 3(2):221–248

Bader BW, Kolda TG (2007) Efficient MATLAB computations with sparse and factored tensors. SIAM J Sci Comput 30(1):205–231

Bader BW, Kolda TG et al (2015) Matlab tensor toolbox version 2.6. http://www.sandia.gov/~tgkolda/TensorToolbox/. Accessed 15 Sept 2016

Bebendorf M, Rjasanow S (2003) Adaptive low-rank approximation of collocation matrices. Computing 70(1):1–24

Brandes U, Lerner J (2010) Structural similarity: spectral methods for relaxed blockmodeling. J Classif 27(3):279–306

Breiger RL, Pattison PE (1978) The joint role structure of two communities' elites. Sociol Methods Res 7(2):213–226

Bro R, Kiers HA (2003) A new efficient method for determining the number of components in parafac models. J Chemom 17(5):274–286

Buehrer G, Chellapilla K (2008) A scalable pattern mining approach to web graph compression with communities. In: Proceedings of the 2008 international conference on web search and data mining. ACM, pp 95–106

Desmier E, Plantevit M, Robardet C, Boulicaut JF (2012) Cohesive co-evolution patterns in dynamic attributed graphs. In: Ganascia JG, Lenca P, Petit JM (eds) Discovery science. DS 2012. Lecture notes in computer science, vol 7569. Springer, Berlin, Heidelberg, pp 110–124

Doreian P, Batagelj V, Ferligoj A (2005) Generalized blockmodeling, vol 25. Cambridge University Press, Cambridge

Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174

Gansner ER, Koren Y, North SC (2005) Topological fisheye views for visualizing large graphs. IEEE Trans Vis Comput Graph 11(4):457–468

Henderson K, Gallagher B, Eliassi-Rad T, Tong H, Basu S, Akoglu L, Koutra D, Faloutsos C, Li L (2012) Rolx: structural role extraction and mining in large graphs. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 1231–1239

Isella L, Stehl J, Barrat A, Cattuto C, Pinton JF, den Broeck WV (2011) What's in a crowd? analysis of face-to-face behavioral networks. J Theor Biol 271(1):166–180. http://www.sociopatterns.org/datasets/infectious-sociopatterns-dynamic-contact-networks/

Kodinariya TM, Makwana PR (2013) Review on determining number of cluster in k-means clustering. Int J 1(6):90–95

Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM Rev 51(3):455–500

Kolda T, Sun J (2008) Scalable tensor decompositions for multi-aspect data mining. In: The eighth IEEE international conference on data mining. IEEE, pp 363–372

LeFevre K, Terzi E (2010) Grass: graph structure summarization. In: Proceedings of the 2010 SIAM international conference on data mining. SIAM, pp 454–465

Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, pp 177–187

Li CT, Lin SD (2009) Egocentric information abstraction for heterogeneous social networks. In: International conference on advances in social network analysis and mining, 2009, ASONAM'09. IEEE, pp 255–260

Liu Y, Dighe A, Safavi T, Koutra D (2016) A graph summarization: a survey. arXiv preprint arXiv:1612.04883

Mathioudakis M, Bonchi F, Castillo C, Gionis A, Ukkonen A (2011) Sparsification of influence networks. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 529–537

Mitchell TM (1997) Machine learning, 1st edn. McGraw-Hill Inc, New York

Navlakha S, Rastogi R, Shrivastava N (2008) Graph summarization with bounded error. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data. ACM, pp 419–432

Papalexakis EE (2016) Automatic unsupervised tensor mining with quality assessment. In: Proceedings of the 2016 SIAM international conference on data mining. SIAM, pp 711–719

Piperno A (2008) Search space contraction in canonical labeling of graphs. arXiv preprint arXiv:0804.4881

Priebe CE, Conroy JM, Marchette DJ, Park Y (2005) Scan statistics on enron graphs. Comput Math Organ Theory 11(3):229–247

Ralaivola L, Swamidass SJ, Saigo H, Baldi P (2005) Graph kernels for chemical informatics. Neural Netw 18(8):1093–1110

Riondato M, García-Soriano D, Bonchi F (2017) Graph summarization with quality guarantees. Data Min Knowl Discov 31(2):314–349

Rossi RA, Ahmed NK (2015a) The network data repository with interactive graph analytics and visualization. In: Proceedings of the twenty-ninth AAAI conference on artificial intelligence. http://networkrepository.com. Accessed 27 Feb 2017

Rossi RA, Ahmed NK (2015b) Role discovery in networks. IEEE Trans Knowl Data Eng 27(4):1112–1131

Shah N, Koutra D, Zou T, Gallagher B, Faloutsos C (2015) Timecrunch: interpretable dynamic graph summarization. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1055–1064

Shen Z, Ma KL, Eliassi-Rad T (2006) Visual analysis of large heterogeneous social networks by semantic and structural abstraction. IEEE Trans Vis Comput Graph 12(6):1427–1439

Spielman DA, Teng SH (2011) Spectral sparsification of graphs. SIAM J Comput 40(4):981–1025

Tsalouchidou I, Morales GDF, Bonchi F, Baeza-Yates R (2016) Scalable dynamic graph summarization. In: 2016 IEEE international conference on big data (big data). IEEE, pp 1032–1039