Hanke and Foraita *BMC Bioinformatics* (2017) 18:261
DOI 10.1186/s12859-017-1677-x

BMC Bioinformatics

METHODOLOGY ARTICLE Open Access

Clone temporal centrality measures for incomplete sequences of graph snapshots

Moritz Hanke* and Ronja Foraita

Abstract

Background: Different phenomena like the spread of a disease, social interactions or the biological relation between genes can be thought of as dynamic networks. These can be represented as a sequence of static graphs (so called graph snapshots). Based on this graph sequences, classical vertex centrality measures like closeness and betweenness centrality have been extended to quantify the importance of single vertices within a dynamic network. An implicit assumption for the calculation of temporal centrality measures is that the graph sequence contains all information about the network dynamics over time. This assumption is unlikely to be justified in many real world applications due to limited access to fully observed network data. Incompletely observed graph sequences lack important information about duration or existence of edges and may result in biased temporal centrality values.

Results: To account for this incompleteness, we introduce the idea of extending original temporal centrality metrics by cloning graphs of an incomplete graph sequence. Focusing on temporal betweenness centrality as an example, we show for different simulated scenarios of incomplete graph sequences that our approach improves the accuracy of detecting important vertices in dynamic networks compared to the original methods. An age-related gene expression data set from the human brain illustrates the new measures. Additional results for the temporal closeness centrality based on cloned snapshots support our findings. We further introduce a new algorithm called `REN` to calculate temporal centrality measures. Its computational effort is linear in the number of snapshots and benefits from sparse or very dense dynamic networks.

Conclusions: We suggest to use clone temporal centrality measures in incomplete graph sequences settings. Compared to approaches that do not compensate for incompleteness our approach will improve the detection rate of important vertices. The proposed REN algorithm allows to calculate (clone) temporal centrality measures even for long snapshot sequences.

Keywords: Dynamic networks, Dynamic graphs, Betweenness, Closeness, Centrality measures, Time varying networks, Shortest temporal path

Background

Many phenomena can be represented and interpreted as dynamic networks. These consist of vertices and edges that occur and vanish at different time points [1]. Global characteristics of a dynamic network's topology, e.g. its diameter, may vary over time, but also characteristics of individual vertices, such as their centralities. It is essential to take these dynamics into account when one is interested in crucial vertices and subnetworks characterizing the information flow in dynamic networks and their connectivity. The detection of such vertices or subnetworks is important for different research areas like life, social and computer science to understand empirical phenomena like the spread of a disease in a population, the connectivity within and between peer groups or cyber attacks on computer networks [2–4].

Statistical methods for static networks have been an active and fruitful field for statistical research in the last decades. In recent years the development of probabilistic models for dynamic networks as well as the development of methods for describing key properties of these networks have gained more and more attention

*Correspondence: hanke@leibniz-bips.de
Leibniz Institute for Prevention Research and Epidemiology - BIPS, Department of Biometry and Data Management, Achterstr. 30, Bremen, Germany

[5]. For this purpose, a dynamic network is often represented as a dynamic graph consisting of a vertex set V and a temporal edge set E. While some authors [5, 6] define a temporal edge as event between two vertices a and b starting at a particular time point with specific edge duration, others [7–9] define a dynamic network as a sequence of static graphs, so called snapshots, consisting of temporal edge sets E_t. The temporal order of the edge set describes the direction of the dynamics. The sequence of snapshots can either consist of static graphs of specific time points, or aggregated static graphs constructed by combining all edges present within a predefined time interval. In many scientific fields, e.g. genetic epidemiology, only static graphs of specific time points are available rather than fully observed dynamic network structures, for example because it is technologically infeasible to determine the exact starting time or duration of an edge between two vertices. Based on the representation of a snapshot sequence it is possible to extend vertex measures like closeness and betweenness centrality from static to dynamic network settings. However, it is inappropriate to apply vertex centrality measures for static settings, to quantify the importance of vertices in a dynamic network because the dynamic topology of the network will be neglected [5, 10]. This is for example the case when a dynamic network is aggregated into a static graph sequence and then 'classical' vertex centralities are calculated without taking into account the structural changes within the network over time. Calculating static centrality measures for every vertex of each snapshot and then averaging these values also neglects the the time order of the snapshots. Faisal & Milenkovic correlated static centrality measures with the time of the respective snapshot to calculate centrality values in dynamic networks [11]. However, their approach is not a *temporal* centrality measure because it does not reflect temporal paths. To address this shortcoming we use the concept of temporal paths necessary to appropriately describe the centrality of a vertex in its chronological sequence [12–14].

Tang et al. extended static centrality measures for the use in dynamic networks by accounting for shortest *temporal* paths [8]. Their approach assumes that all network information within a previously chosen window size is aggregated into one snapshot. Kim and Anderson [15] modified the representation of a sequence of graph snapshots into a single directed time graph linking each vertex with its successors in time. Based on this directed time graph the authors slightly reformulated the centrality measures of [8]. Another definition of vertex centrality was given for temporal walks [16] that allow to visit edges multiple times per time point instead of once as with shortest temporal paths. This temporal centrality measure can be interpreted as a temporal version of the static Katz centrality [17].

While the computation of dynamic network characteristics mainly assumes a fully observed dynamic network, there is a lack of approaches for incomplete graph sequences which pose two major challenges:

(a) An edge in an observed snapshot could have arisen at an earlier and unknown time point in the past and could last until an unknown time point in the future. Hence, starting time and duration of this edge are uncertain.
(b) Some edges are unobserved because they occur and vanish in the time interval between two consecutive observed snapshots. Such edges are not observed and hence also their influence on the network's dynamic is difficult to assess.

Both cases will affect temporal centrality measures and are likely to occur in real world applications, e.g. when data of gene expression networks are available only at some – maybe unequally spaced – time points [11, 18] or when rapid changes occur within the network [19]. While some authors propose metrics to quantify the overall stability of the topology of a dynamic network [20–24], the impact on centrality measures due to incomplete information was only investigated for static network settings [25, 26]. The development of temporal centrality measures accounting for incompletely observed dynamic networks is still lacking.

Our work fills this gap by introducing the problem of incomplete graph sequences and proposing an extensions of the temporal betweenness and closeness centralities of Kim & Anderson [15] by using additional snapshots in situations of incomplete graph sequences. These added snapshots are copies of observed snapshots and will be referred to as *clones* in the following. Hence we propose the *clone* temporal betweenness and closeness centrality (CTBC, CTCC). The main purpose of adding clones is to allow more moves along a graph sequence and hence to increase the number of identified temporal paths that could not have been found with the originally observed snapshot sequence. We demonstrate in simulation studies and in an application to a real dynamic gene network that our new approach provides simple improved vertex centrality estimates in situations with incomplete graph sequences. We further considered the computational aspect of our new measures. The time complexity for calculating centrality measures in dynamic graphs depends on the number of vertices and edges as well as on the number of snapshots. Especially, the calculation of temporal centrality measures based on (shortest) temporal paths can be challenging because, unlike static graphs, for dynamic graphs it does not hold that every subpath of a shortest temporal path is again a shortest path. Hence, the search for the shortest temporal path has to visit all relevant subsequences of graphs, i.e. starting from every snapshot up to the last snapshot. Otherwise the full dynamics of the network will not be considered appropriately in the calculated centrality values [15, 27]. To address this time demanding requirement, we propose a novel and easy to implement algorithm called REN (Reversed Evolution Network). Its time complexity is linear in the number of graph snapshots for a fixed number of vertices and edges. This property allows to search for shortest temporal paths in long graph sequences or in a graph sequence that has been augmented by clones. In addition, our simulations suggest that the overall running time of REN benefits from dense and sparse dynamic networks.

Methods

Let us assume a finite time interval in which a dynamic network has been observed, starting at t_{start} and ending at t_{end}, where without loss of generality $t_{start} = 0$ and $t_{end} = T$. A dynamic network is represented as a dynamic graph $G^D_{0,T} = (V, E_{0,T})$, where we assume a finite set V of $|V|$ vertices and an edge set $E_{0,T}$ that can change in the time interval $[0, T]$. While we will focus on edge sets $E_{0,T}$ consisting of temporal *undirected* edges $\{a, b\}_{i,j} \in E_{0,T}$ with $a, b \in V$ that are present in the time interval $[i, j]$ with $0 \leq i < j \leq T$, it is straightforward to extend our approach to temporal *directed* edges.

In the following we will present the basic notations to introduce incomplete graph sequences. We will then derive a modified version of the temporal betweenness centrality as an example for our approach using cloned snapshots.

Graph sequences and shortest temporal paths

To characterize structural properties of a dynamic network a dynamic graph $G^D_{0,T}$ is commonly discretized into a time ordered sequence of static graphs $\mathcal{G} = G_1, G_2 \ldots, G_S$ with corresponding edge sets E_k for $k \in \{1, 2, \ldots, S\}$, such that $G_k = (V, E_k)$. Each edge set E_k of a snapshot k consists of all edges that are present in a time window w_k of size $w \leq (t_{end} - t_{start}) = T$. Thus, the number of snapshots is given by $S = T/w$.

Sequences of graph snapshots can be represented as directed time graphs (DTG) [15, 21]. Figure 1 shows a graph sequence and its adequate DTG. Each snapshot G_k in Fig. 1a has a corresponding column d_k of *directed* edges (Fig. 1b). Hence, every vertex $a \in V$ of $\mathcal{G}$ occurs $S + 1$ times in a DTG, indicated by $a_0, a_1, \ldots, a_S$. The columns d_k of a DTG contain the (undirected) edges of the original snapshot representation plus edges from each vertex to itself at the next time point (horizontal edges). The latter edges represent *halts* in a snapshot; all other edges are called *hops*.

It is possible to formulate an *edge sequence* connecting vertices along the DTG, as indicated by the red dashed edges in Fig. 1b. We call such sequences *temporal paths*. They consist of a unique combination of hops and halts. The occurrence of an edge is considered by only allowing either one hop or halt per snapshot k (or likewise per column d_k). Thus, using the representation as a DTG, a

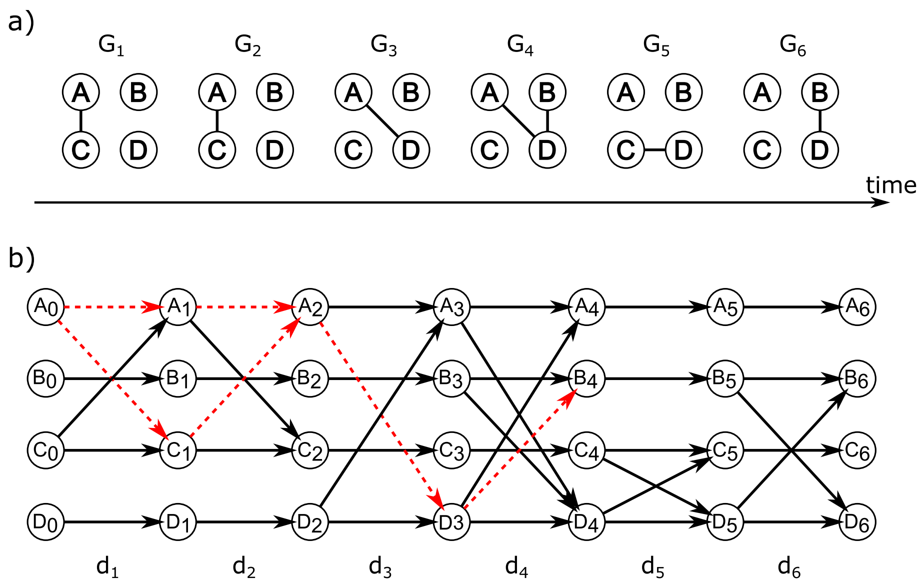

Fig. 1 Directed time graph (DTG). A graph sequence $\mathcal{G}$ of snapshots in (**a**) and its representation as a DTG in (**b**). *Horizontal edges* in (**b**) indicate halts on a vertex, *diagonal edges* represent hops. Two shortest temporal paths from vertex *A* to vertex *B* are marked by *red dashed edges*

temporal path starting at snapshot k and ending at snapshot n with $k, n \in \{1, 2, \ldots, S\}, k \leq n$ of a graph sequence $\mathcal{G} = G_1, \ldots, G_S$ is defined as an ordered sequence of vertices $p_{k,n}(a, c) = \langle a_{k-1}, \ldots, c_n \rangle$ such that $a, c \in V$. Note that $p_{k,n}(a, c)$ starts with index $k - 1$ in a DTG.

Let $\mathbf{P}_{k,n}(a, c) = \bigcup_{m=k}^{n} p_{k,m}(a, c)$, that is the set of all possible temporal paths starting from vertex a at snapshot k and ending in vertex c, at the latest, in snapshot n. Note, a temporal path from a to c can end at $m \leq n$. If a path path $p_{k,m}(a, c)$ exists, the *path length* is defined as $|p_{k,m}(a, c)| = m - k + 1$, which is the number of halts and hops needed to travel from vertex a to vertex c in the graph sequence $G_k, \ldots, G_m$. A *shortest* temporal path $\gamma_{k,m,n}(a, c)$ is then defined as the path $p_{k,m}(a, c) \in \mathbf{P}_{k,n}(a, c)$ with minimum number m, where c is reached in snapshot $m \leq n$. It's length is $|\gamma_{k,m,n}(a, c)| = m - k + 1$. The set $\Gamma_{k,m,n}(a, c) = \bigcup \gamma_{k,m,n}(a, c)$ contains all shortest temporal paths from a to c within the considered sequence $G_k, \ldots, G_n$. Consequently, all shortest temporal paths of $\Gamma_{k,m,n}(a, c)$ have the same path length $m - k + 1$.

Expanding the above notation, $\gamma_{k,m,n}(a, b_l, c) \in \Gamma_{k,m,n}(a, c)$ denotes a shortest temporal path that crosses vertex b at snapshot l. Therefore, the set $\Gamma_{k,m,n}(a, b, c) = \bigcup_{k<l<m} \gamma_{k,m,n}(a, b_l, c)$ contains all shortest paths from a to c that cross b at some snapshot l.

If a shortest temporal path $\gamma_{k,n,n}(a, b_l, c)$ contains the holds and hops of $p_{l,n}(b, c)$ we call $p_{l,n}(b, c)$ the *upper temporal subpath* of $\gamma_{k,n,n}(a, b_l, c)$. Analogously, if $\gamma_{k,n,n}(a, b_l, c)$ contains all edges of $p_{k,l}(a, b)$ we call $p_{k,l}(a, b)$ a *lower temporal subpath* of $\gamma_{k,n,n}(a, b_l, c)$. Additionally, we simply call every sequence of hops and halts of $p_{k,n}(a, c)$ starting at a snapshot $l, l > k$, and ending at a snapshot $m, m < n$, a temporal subpath of $p_{k,n}(a, c)$.

In the following we will show that every upper temporal subpath of a shortest temporal path will always be a shortest temporal path itself even if the lower temporal path is not a shortest temporal path.

Lemma 1 *Given a graph sequence $\mathcal{G} = G_k, \ldots, G_l, \ldots, G_m, \ldots, G_n$, let $\gamma_{k,n,n}(a, b_l, c)$ be a shortest temporal path from a to c that passes vertex b at snapshot l and ends at snapshot n. Then, even if the lower temporal path $p_{k,l}(a, b)$ is not a shortest temporal path, the upper temporal path $p_{l,n}(b, c)$ is a shortest temporal path, i.e. $p_{l,n}(b, c) = \gamma_{l,n,n}(b, c)$.*

Proof Assume that there exists a temporal path $p_{l,m}(b, c)$ from b to c with $|p_{l,m}(b, c)| < |\gamma_{l,n,n}(b, c)|$. Then it follows that

$$|\gamma_{k,n,n}(a, b_l, c)| = |p_{k,l}(a, b)| + |\gamma_{l,n,n}(b, c)| > |p_{k,l}(a, b)| + |p_{l,m}(b, c)| = |p_{k,m}(a, c)|$$

which is contradiction to the assumption that $\gamma_{k,n,n}(a, b_l, c)$ is the shortest temporal path from a to c over b at snapshot l. □

Note that although *all* subpaths of shortest paths are again shortest path in a *static* directed graph [28], this does not hold for a DTG. As a simple example consider a path $p_{k,n}(a, c) = \gamma_{k,n,n}(a, c) = \gamma_{k,n,n}(a, b_l, c) = \gamma_{k,n,n}(a, b_m, c)$, $l < m$, from a to c that passes vertex b at snapshots l and m. Then, $|p_{k,l}(a, b)| < |p_{k,m}(a, b)|$ and hence $p_{k,m}(a, b)$ is not a shortest path although it is a subpath of $\gamma_{k,n}(a, c)$.

While the query for (shortest) temporal paths is only meaningful in graph sequences with at least two snapshots, the *length* of a (shortest) temporal path can be one, if a and c are connected at the first snapshot of the graph sequence, that is $|p_{k,n}(a, c)| \geq |\gamma_{k,k,n}(a, c)| = 1$.

Incomplete graph sequences

If there is only limited access to S snapshots of time points $t \in [0, T]$, the observed graph sequence $\mathcal{G}$ is incomplete. In this situation it might be impossible to determine exactly when an edge occurs and how long it has existed in the network. Additionally, incomplete sequences might miss edges in total and thus can lead to unobserved edges. Figure 2 gives an example of the impact of incomplete graph sequences. Although in Fig. 2b the first snapshot G_1 at $t = 0.3$ correctly captures the occurrence of edge $\{A, C\}$, it cannot determine its duration until $t = 2$. The true edge sequence of $\{A, D\}$ followed by $\{B, D\}$ cannot be reconstructed because at the next snapshot G_2 ($t = 3.6$) both edges are aggregated into one graph. This *masks* their chronological order. Further, the second occurrence of $\{B, D\}$ in the time interval $[5, 6]$ is not detected, because the last observation of the dynamic network is G_3 at $t = 4.8$, and therefore the edge $\{B, D\}$ is missing in the observed graph sequence. The consequence is that there is no temporal path from A to B in the observed DTG (Fig. 2c).

Both, masked edge chronologies and unobserved edges affect the number of observable (shortest) temporal paths in a dynamic network.

Clone temporal betweenness centrality

In a static network, the betweenness centrality of a vertex b measures how easily b can be avoided when seeking for shortest paths to get from vertex a to c, $a \neq b \neq c \in V$. More precisely, it is the ratio between the number of shortest paths from a to c passing b and the total number of shortest paths from a to c. This idea has been extended [8, 15] to graph sequences $\mathcal{G} = G_1, \ldots, G_S$ consisting of S snapshots. Let $\sigma_{k,m,S}(a, b, c)$ denote the cardinality of the set of the shortest paths $\Gamma_{k,m,S}(a, b, c)$ and $\sigma_{k,m,S}(a, c)$ denote the cardinality of $\Gamma_{k,m,S}(a, c)$ for a graph sequence

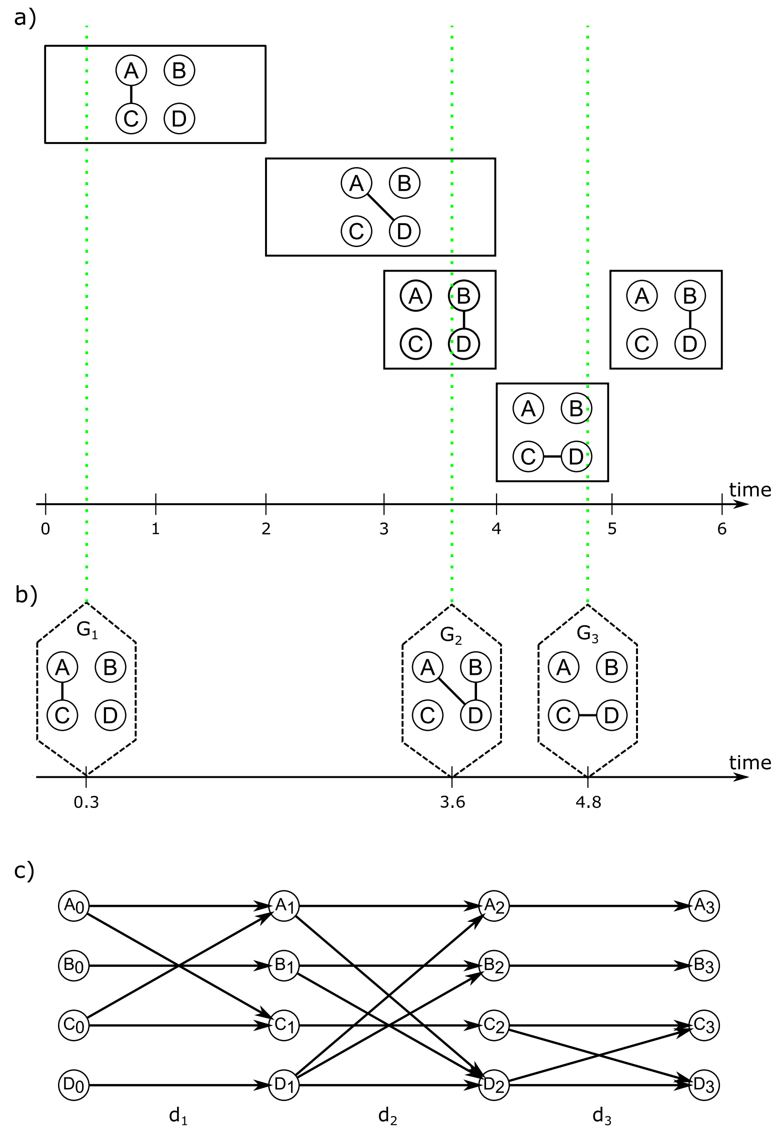

Fig. 2 Incomplete graph sequence. An incomplete observed graph sequence in (**b**) and its DTG (**c**) compared to the true but unobserved dynamic network in (**a**). *Solid boxes* in (**a**) represent time intervals of the respective edge occurrence within the true dynamics. *Dashed boxes* in (**b**) indicate snapshots at specific time points and the *green dotted lines* mark the corresponding moments in (**a**). The sequence of graph snapshots yields the incomplete graph sequence

$G_k, \ldots, G_S$. The *temporal betweenness centrality* (TBC) of vertex b is then defined as:

$$TBC_{1,S}(b) = \sum_{k=1}^{S-1} \sum_{\substack{a,c \in V \setminus b \\ \sigma_{k,m,S}(a,c)>0}} \frac{\sigma_{k,m,S}(a,b,c)}{\sigma_{k,m,S}(a,c)}. \qquad (1)$$

The second sum in Eq. (1) accounts for all shortest paths starting from vertex a and the first sum ensures that all subsequences starting at a snapshot after k, $G_l, \ldots, G_S$, $l > k$, are included in the calculation of this measure. This is necessary to adequately capture the complete dynamic behaviour in the network over time [27]. For example, consider a graph sequence with all vertices connected to each other at the first snapshot but with fewer connections at the following snapshots. Applying the TBC without summing over all later subsequences will not represent the dynamics after the first snapshots because all shortest temporal paths will be of length one due to the fully connected first snapshot. However, TBC cannot explicitly handle incomplete graph sequences and hence it will miss (shortest) temporal paths when calculating a vertex' centrality.

Consider Fig. 2 and assume that we have only observe the sequence as shown in Fig. 2b; what can then be inferred about the true underlying sequence in Fig. 2a? It is obvious that the edge $\{A, C\}$ in snapshot G_1 must have occurred before the next observed snapshot G_2. The edges $\{A, D\}$ and $\{B, D\}$ observed in snapshot G_2 on the contrary must have occurred in the dynamic network at a time point between snapshots G_1 and G_2 but we do not know the order of occurrence and thus the possible temporal paths. Our proposal is to *fill* the gap between snapshots with additional snapshots, in order to reveal additional (shortest) temporal paths that are likely to exist. These added snapshots are copies of observed snapshots and will be referred to as *clones*.

Definition 1 *Given a static graph $G_k(V, E_k)$ of snapshot k we define clones of G_k as $G_{k,j_k}(V, E_{k,j_k})$ such that $G_{k,j_k}(V, E_{k,j_k}) = G_k(V, E_k)$ for $j_k = 1, 2, \ldots, J_k$.*

Based on definition 1 and using the notation G_{k,j_k} for $G_{k,j_k}(V, E_{k,j_k})$ we can now define a cloned graph sequence.

Definition 2 *Given a original graph sequence $G_1, G_2, \ldots, G_S$ and clones G_{k,j_k} with $k = 1, 2, \ldots, S$ and $j_k = 1, 2, \ldots, J_k$ a cloned graph sequence is defined as the ordered sequence $G_{1,1}, G_{1,2}, \ldots, G_{k,j_k}, \ldots, G_{S,J_S}$.*

Augmenting the original graph sequence with clones G_{k,j_k} raises the question of how to choose the number of clones J_k per snapshot. This is generally flexible and may vary depending on the application. We propose the following three plausible approaches:

1. Adding a sufficient number of clones J_k per snapshots k such that any *static* path in $G_{k-1} \cup G_k$ not presented in G_{k-1} and G_k alone can be found as a *temporal* path. This is always possible and depends on the number of different edges between G_{k-1} and G_k.
2. Adding clones based on assumptions about the expected duration of the occurrence of edges.
3. If the number of unobserved discrete time points between G_{k-1} and G_k is known a corresponding number of clones can be added.

Figure 3 shows an example of temporal path search in a graph sequence including cloned snapshots. Given the true dynamic network depicted in Fig. 1 and the observed snapshots of Fig. 2b, we constructed the graph sequence presented in Fig. 3a and decided to clone each of its snapshot once, resulting in the graph sequence of Fig. 3b. As shown in Fig. 3c, clones can detect shortest temporal paths that are in fact a true shortest temporal paths (red dashed arrows). However, cloning compensates only for unobserved edge durations and ordering of occurrences, but it cannot detect unobserved edges and hence also no

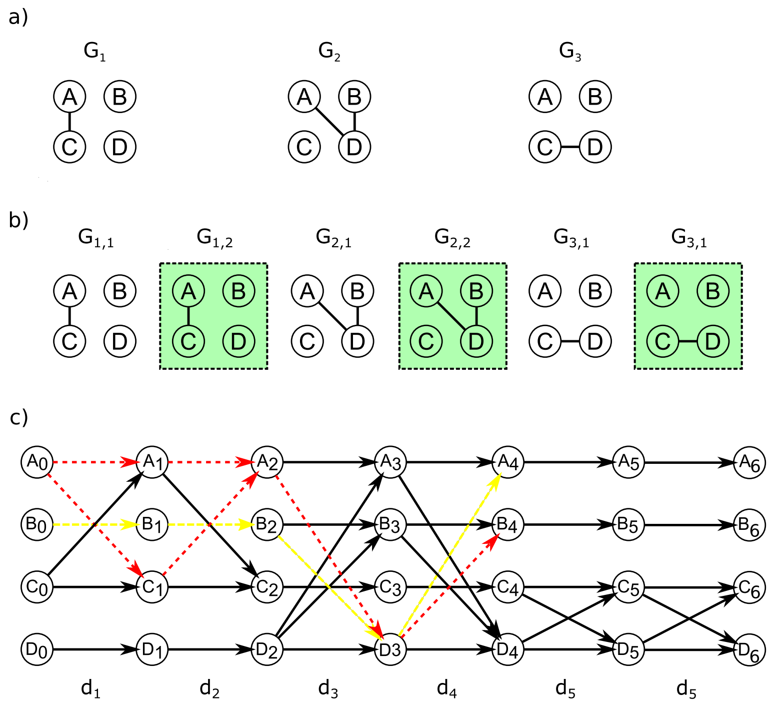

Fig. 3 Cloned graph sequence. The incomplete observed graph sequence in (**a**) is based on the incomplete graph sequence of Fig. 2. The first two observed snapshots are cloned as shown in the graph sequence in (**b**) and the respective DTG in (**c**). *Green boxes* indicate clones. Both true temporal paths from *A* to *B* (*red dashed arrows*) in the original complete graph sequence were found due to cloning (see Fig. 1). However, a spurious (shortest) temporal path was also detected that is not present in the original sequence (indicated by the *yellow dashed arrows*)

temporal paths that contain these unobserved edges. Furthermore, if cloning overestimates edge durations or the order of occurrence (as for the edge (B, D)), it might detect false shortest temporal paths (indicated by the yellow dashed arrows). We call this problem *excess of cloning* and discuss its implications in more detail in the simulation section.

Exploiting the idea of cloning snapshots, we extend the TBC of Eq. (1) to a *clone temporal betweenness centrality* (CTBC):

$$CTBC_{1,S}(b) = \sum_{k=1}^{S} \sum_{j_k=1}^{J_k} \sum_{\substack{a,c \in V \setminus b \\ \sigma^{j_k}_{k,m,S}(a,c)>0}} \frac{\sigma^{j_k}_{k,m,S}(a,b,c)}{\sigma^{j_k}_{k,m,S}(a,c)}, \qquad (2)$$

where $\sigma^{j_k}_{k,m,S}(a,b,c)$ denotes the number of shortest temporal paths from a to c passing b, starting at the j_k-th clone of snapshot k. Similarly, $\sigma^{j_k}_{k,m,S}(a,c)$ denotes the total number of shortest paths from a to b starting at the j_k-th clone of snapshot k. The CTBC successively sums the sequence of observed and cloned snapshots starting at the j_k-th clone of snapshot k until the last clone of snapshot S. CTBC is applicable for graph sequences of directed and undirected temporal networks. The idea of cloning snapshots when calculating temporal centrality measures can also easily be applied to other temporal centrality measures like the temporal closeness centrality (see Additional file 1).

`REN`: a new algorithm for finding shortest temporal paths

An appropriate algorithm is necessary to calculate the above temporal centrality measures. The summation over all subsequences in Eqs. 1 and 2 can be computationally demanding for long graph sequences because a shortest temporal path in $G_k, \ldots, G_S$ might not be a (shortest) temporal path in $G_{k+1}, \ldots, G_S$ which necessitates a new query. As a consequence, a new search for shortest temporal paths has to be started for each snapshot of the graph sequence $G_k, \ldots, G_S$. For example, there are two shortest temporal paths starting from vertex A at snapshot 1 and ending at vertex B at snapshot 4 in Fig. 1. Both paths have to pass vertex D at snapshot 3, meaning that a temporal path starting at snapshot 4 or later cannot be subpath of these shortest temporal paths.

Our `REN` algorithm tackles the problem of consecutive queries by searching for temporal paths in the reversed order of snapshots, defined as $\mathcal{G}^* = G_S, \ldots, G_1$. A *reversed temporal path* is defined as $p^*_{n,k}(c,a) = \langle c_n, \ldots, a_{k-1} \rangle = \texttt{rev}\left(p_{k,n}(a,c)\right)$, where $\texttt{rev}(\cdot)$ is the function that reverses the edge directions in a DTG and therefore the order of the vertices of a temporal path. The basic idea is then to move along all reversed temporal paths starting from a specific vertex c at snapshot S until snapshot 1 and to store each descendent vertex b of c and its lowest snapshot number k where b is connected to c by an edge or temporal path. Even if there are shortest temporal paths found before reaching the first snapshot it is crucial to move along all reversed temporal paths up to the first snapshot of the considered graph sequence. Otherwise shortest temporal paths that start at or near the first snapshot are not found.

In the following, we will prove that the computational time of `REN` is linear with respect to the number of snapshots S when searching for all shortest temporal paths in $G_k, \ldots, G_S$, $\forall\ k \in [1, S-1]$. First, we prove that a query along a particular reversed shortest temporal path finds all upper temporal subpaths that are also shortest temporal paths too.

Lemma 2 *Let $G_k, \ldots, G_n, k < n$, be a graph sequence and let $\gamma'_{k,n,n}(a,c) = p'_{k,n}(a,c)$ be a specific shortest temporal path in $\Gamma_{k,n,n}(a,c)$. Then, moving along the reversed temporal path $p^*_{n,k}(c,a) = \texttt{rev}\left(p'_{k,n}(a,c)\right)$ from vertex c to vertex a finds all $n-k$ shortest temporal paths $\gamma'_{l,n,n}(b,c)$, $k \leq l < n$ from any vertex b to vertex c that are upper temporal subpaths of $\gamma'_{k,n,n}(a,c) = \gamma'_{k,n,n}(a,b_l,c)$ and for which $b = b_l \in \gamma'_{k,n,n}(a,b_l,c)$.*

Proof A specific shortest temporal path $\gamma'_{k,n,n}(a,c) \in \Gamma_{k,n,n}(a,c)$ is characterised by a unique combination of $n-k$ hops and halts. This temporal path contains then $n-k$ upper temporal subpaths, each starting at a different snapshot $k, k+1, \ldots, n-1$. For $l = k$ it directly follows that $\gamma'_{l,n,n,}(a,c) = \gamma'_{k,n,n,}(a,c)$.
Now, let $l = k+1$ and let $b \in V \backslash c$ be a vertex on $\gamma'_{k,n,n,}(a,c)$, that is it holds $\gamma'_{k,n,n,}(a,b_l,c) = \gamma'_{k,n,n}(a,c)$. Applying Lemma 1 yields that the upper temporal subpath $p'_{l,n}(b,c)$ of $p'_{k,n}(a,c) = \gamma'_{k,n,n}(a,b_l,c)$ is also a shortest temporal path $\gamma'_{l,n,n}(b,c)$. This holds for all further $l = k+2, \ldots, n-1$, i.e. $\gamma'_{k,n,n}(a,c)$ contains $n-k$ upper temporal subpaths (including $\gamma'_{k,n,n}(a,c)$ itself) that are shortest temporal paths.
Then, it follows that $p^*_{n,k}(c,a) = \texttt{rev}\left(p'_{k,n}(a,c)\right)$ contains all reversed upper temporal subpaths $p^*_{n,l}(c,a) = \texttt{rev}\left(p'_{l,n}(a,c)\right) = \texttt{rev}\left(\gamma'_{l,n,n}(a,c)\right)$ with $k \leq l < n$. Thus, following the reversed upper temporal path $p^*_{n,k}(c,a)$ reveals all $n-k$ shortest temporal paths of $\gamma'_{k,n,n}(a,c)$. □

With Lemma 2 it is possible to show that one query for all reversed temporal paths starting at vertex c is sufficient to reveal all shortest temporal paths that end at c of a graph subsequence starting at a snapshot at or after k.

Theorem 1 *Let $\mathcal{G} = G_k, \ldots, G_n, k < n$, be a graph sequence and let $\mathbf{\Gamma}_{k,n}(\cdot, c) = \bigcup_{l=k}^{n-1} \bigcup_{m=l}^{n} \bigcup_{a \in V \setminus c} \gamma_{l,m,n}(a,c)$ be the set of all shortest temporal paths that start from any vertex at snapshot $l \geq k$ and end in vertex c at snapshot $m \leq n$. Further, let $\mathbf{p}^*_{n,k}(c, \cdot) = \bigcup_{a \in V \setminus c} p^*_{n,k}(c,a)$ be the set of all reversed temporal paths starting from vertex c at snapshot n and ending at any vertex $a \in V \setminus c$ at snapshot k. Then, every shortest temporal path $\gamma_{l,m,n}(a,c) \in \mathbf{\Gamma}_{k,n}(\cdot, c)$ is a reversed subpath of a reversed temporal path in $\mathbf{p}^*_{n,k}(c, \cdot)$ and is therefore obtained by moving along every $p^*_{n,k}(c,a) \in \mathbf{p}^*_{n,k}(c, \cdot)$.*

Proof Every shortest temporal path $\gamma_{l,m,n}(a,c) \in \mathbf{\Gamma}_{k,n}(\cdot, c)$ is a subpath of a temporal path in $\mathbf{p}_{k,n}(\cdot, c) = \bigcup_{a \in V \setminus c} p_{k,n}(a,c)$. Then, the set of all reversed temporal paths $\mathbf{p}^*_{n,k}(c, \cdot) = \texttt{rev}\left(\mathbf{p}_{k,n}(\cdot, c)\right)$ also includes the set of reversed shortest temporal paths $\mathbf{\Gamma}^*_{k,n}(\cdot, c) = \texttt{rev}\left(\mathbf{\Gamma}_{k,n}(\cdot, c)\right)$.

Lemma 2 shows for every specific shortest temporal path $\gamma'_{l,m,m}(a,c) \in \mathbf{\Gamma}_{k,n}(\cdot, c)$ that the reversed path $p^*_{m,l}(c,a) = \texttt{rev}\left(p'_{l,m}(a,c)\right) = \texttt{rev}\left(\gamma'_{l,m,m}(a,c)\right)$ contains all $m - l$ upper subpaths of $\gamma'_{l,m,m}(a,c)$ that are also shortest temporal paths. Finally, because $p^*_{m,l}(c,a)$ is a subpath of $p^*_{n,k}(c,a) \in \mathbf{p}^*_{n,k}(c, \cdot)$, it will be detected by moving along the reversed temporal paths of $\mathbf{p}^*_{n,k}(c, \cdot)$. This holds for all $a \in V$. □

Let $\mathcal{P}_{1,S}(\cdot, c) = \bigcup_{k=1}^{S} \bigcup_{m=k}^{S} \bigcup_{b \in V \setminus c} p_{k,m}(b,c)$ denote the set of all temporal paths starting from any vertex $b \in V \setminus c$ at a snapshot k and ending in vertex c not later than at snapshot S. The set of all reversed temporal paths starting in vertex c and ending in any vertex $b \neq c$ is $\mathcal{P}^*_{S,1}(c, \cdot) = \texttt{rev}\left(\mathcal{P}_{1,S}(\cdot, c)\right)$. Further, let $\mathbf{N}_k(c) \subseteq V \setminus c$ be the set of all neighbours of c, i.e. adjacent vertices of c, at snapshot k. By applying Theorem 1 to all $c \in V$, REN can be outlined as follows:

1. Reverse the order of the observed snapshot sequence as $\mathcal{G}^* = G_S, \ldots, G_1$.
2. Select a start vertex c and set $\mathcal{P}^*_{S,1}(c, \cdot) = \emptyset$.
3. For snapshot $k = S$: Find all adjacent vertices $b \in \mathbf{N}_S(c)$. Each edge between c and b forms a reversed temporal path $p^*_{S,S}(c,b) = \langle c_S, b_{S-1} \rangle$ and is stored in the set $\mathcal{P}^*_{S,1}(c, \cdot)$.
4. For snapshots $k = S - 1, \ldots, 1$:
   (a) List all adjacent vertices $b \in \mathbf{N}_k(c)$. Each edge between c and $b \in \mathbf{N}_k(c)$ forms a reversed temporal path $p^*_{k,k}(c,b) = \langle c_k, b_{k-1} \rangle$ and is stored in the set $\mathcal{P}^*_{S,1}(c, \cdot)$. Set $p_{k,k}(b,c) = \langle b_{k-1}, c_k \rangle = \gamma_{k,k,S}(b,c)$.
   (b) List all vertices $a \in V \setminus \{\mathbf{N}_k(c) \cup c\}$ that are adjacent to any vertex b for which $p^*_{m,k+1}(c,b) \in \mathcal{P}^*_{S,1}(c, \cdot)$.
   Join the reversed temporal paths $p^*_{k+1,k}(b,a)$ and $p^*_{m,k+1}(c,b)$ at vertex b to obtain the reversed temporal path $p^*_{m,k}(c,a)$ and store it in $\mathcal{P}^*_{S,1}(c, \cdot)$.
   Set $\gamma_{k,m,S}(a,c) = p_{k,m_{\min}}(a,c)$ for $m_{\min} = \arg\min_{m: k < m \leq S} |p^*_{m,k}(c,a)|$.
5. Repeat steps 2 up to 4 for all other $c \in V$.

Figure 4b gives an example of how REN finds all shortest temporal paths starting at vertex A and ending at vertex B in at least one relevant subsequence of graphs. This is achieved by applying one query for shortest temporal paths only once over the reversed snapshot order for each vertex. The graph sequence is represented as a DTG (Fig. 4a) and without loss of generality we focus on the shortest temporal paths starting from A_0 and ending at B_6 and show that the algorithm finds all shortest temporal paths for which B is a destination vertex by one linear query. The blue dashed edges indicate steps on a reversed temporal path which are potential edges of a shortest temporal path. Red dashed edges indicate shortest temporal paths whenever it was detected by following a reversed temporal path. We start the query from vertex B_6 at snapshot $S = 6$ and follow all its reversed temporal paths up to snapshot 1 (cf. images in Fig. 4b I to VI). In column d_6 in image I, vertex B has two adjacent vertices, forming the hob $\langle B_6, D_5 \rangle$ and the halt $\langle B_6, B_5 \rangle$. Focusing on D_5, there is one hop $\langle D_5, C_4 \rangle$ and one halt $\langle D_5, D_4 \rangle$ in d_5 from snapshot 5 to 4. The first reversed temporal path that connects B and A is detected in d_4 via $\langle D_4, A_3 \rangle$ and according to Lemma 1 it reveals the shortest temporal path $\gamma_{4,6,6}(A,B) = \langle A_3, D_4, D_5, B_6 \rangle$. The paths $p^*_{4,4}(B,D) = \langle B_4, D_3 \rangle$ in d_4 and $p^*_{3,3}(D,A) = \langle D_3, A_2 \rangle$ in d_3 add up to the shortest temporal path $\gamma_{3,4,6}(A,B) = \langle A_2, D_3, B_4 \rangle$ in Fig. 4b IV. Hence, we can infer that any temporal paths from A to B starting before snapshot 4 and ending after snapshot 4, are shortest temporal paths. The detection of $\gamma_{3,4,6}(A,B)$ has the following implications: further shortest paths starting at a snapshot $l < 4$ must end in B_4 or before. This means also that stored edges beyond snapshot 4 like $\langle C_4, D_5 \rangle$ or $\langle D_4, D_5 \rangle$ cannot be any longer part of further shortest temporal paths. Since $\langle D_4, D_5 \rangle$ is part of $\gamma_{4,6,6}(A,B)$, we concluded that $\gamma_{4,6,6}(A,B)$ is not part of a shortest temporal path starting at a snapshot $l < 4$. Figure 4b V and VI complete the query for shortest temporal paths via reversed temporal paths.

Usually, the time complexity of algorithms for calculating temporal centrality measures based on shortest temporal paths are dominated by the number of snapshots.

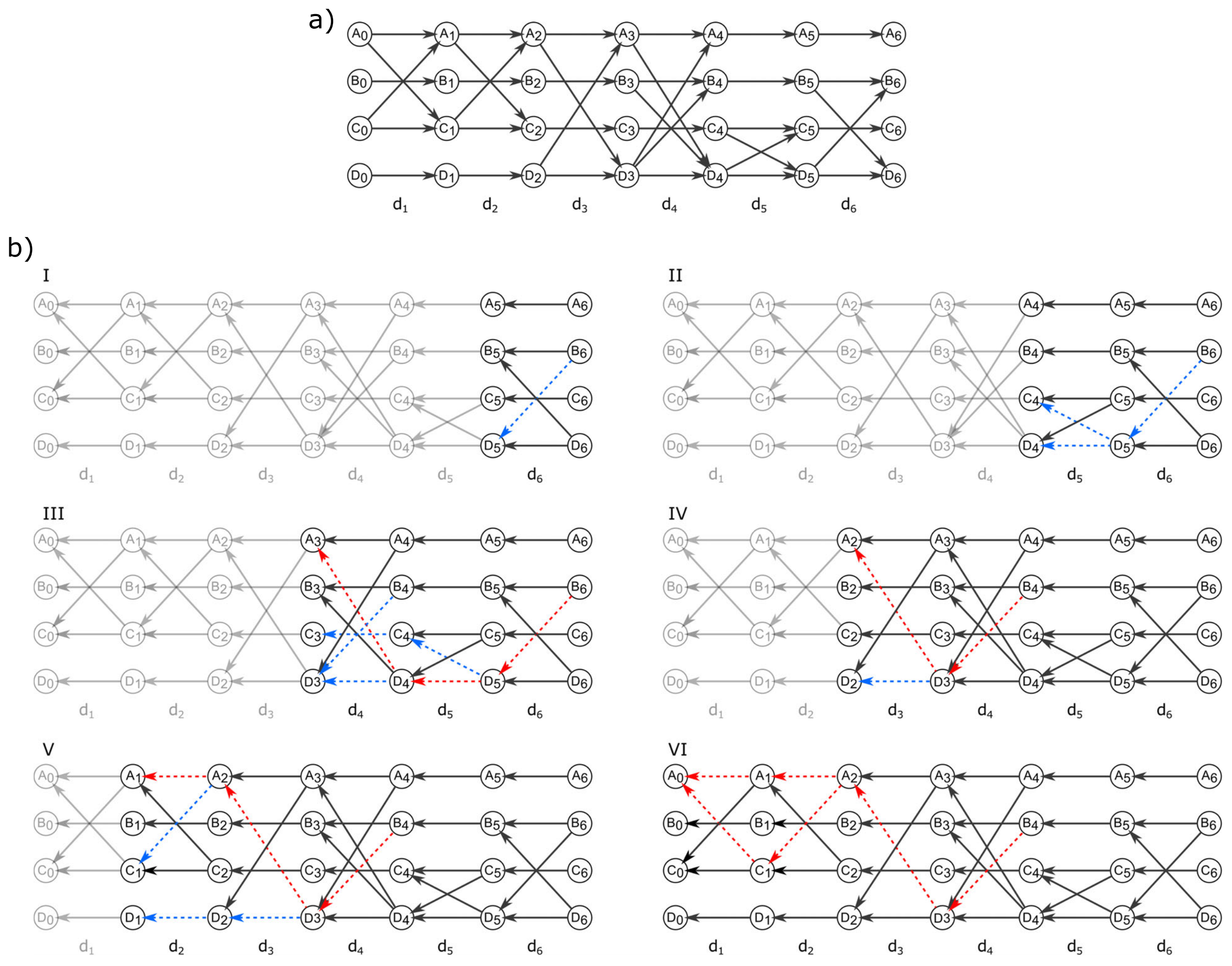

Fig. 4 Example for the algorithm procedure of REN. A DTG representation of a graph sequence $\mathcal{G} = G_1, \ldots, G_6$ in (**a**). The six reversed DTGs in (**b**) demonstrate the stepwise query for shortest temporal paths $\gamma_{1,l,6}(A, B)$ from A to B in a successively reversed graph sequence. *Blue dashed edges* indicate hops and halts that potentially are parts of shortest temporal paths. *Red dashed edges* indicate detected shortest temporal paths starting at a specific snapshot l

The authors of the original version of TBC indicate that the time complexity of their algorithm is cubic in the number of snapshots [15]. For long graph sequences they therefore propose to use larger window sizes w to discretize the dynamic graph $G^D_{0,T}$ into a reduced number of snapshots. This obviously results in a loss of information affecting the accuracy of the temporal centrality measures. REN improves on this limitation because it requires for each vertex only one search over $\mathcal{G}^*$. Thus its time complexity is linear in the number of snapshots. Hence, the calculation of centrality measures like TBC becomes feasible in settings with long graph sequences but also for graph sequences with additional snapshots like the proposed CTBC. For example, TBC and CTBC have an overall running time of $\mathcal{O}(S \cdot |V|^3)$ using REN. REN's running time benefits from *sparse* and *dense* graphs (see Fig. 5). While the first is obvious due to the small number of temporal paths, the latter can be explained by step 4 of our algorithm. In a dense graph sequence, the number of edges is close to the maximum number of edges for every snapshot, that is most vertices b will be $b \in \mathbf{N}_k(c)$. Thus, the expensive search of step 4(b) can be omitted for these vertices.

If the observed graph sequence is represented as edge list for each snapshot, the space complexity of our algorithm is $\mathcal{O}(S \cdot |E_{(1,S)}| + S \cdot |V|^2)$, where $|E_{(1,S)}| = \sum_{k=1}^{S} |E_k|$ denotes the total number of edges in the dynamic network. The second term denotes the space needed to save all temporal paths of the graph sequence. Note, in the worst case scenario, i.e. when each snapshot contains a saturated graph, space complexity will be $\mathcal{O}(2 \cdot S \cdot |V|^2)$.

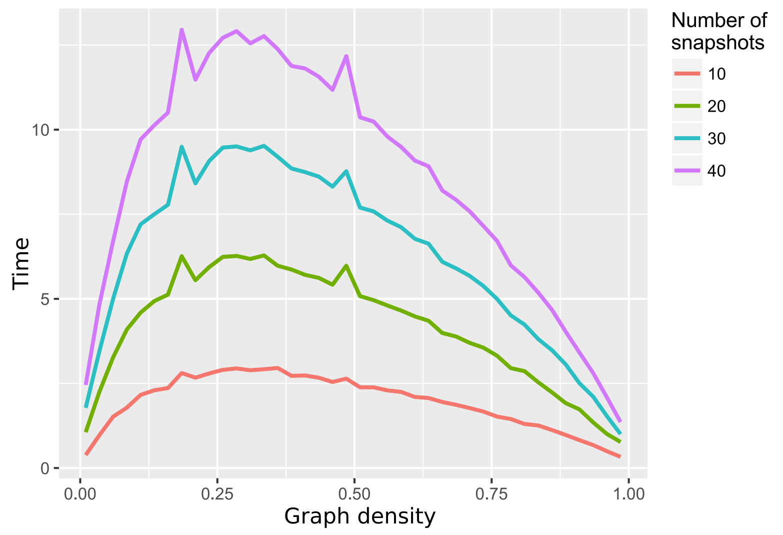

Fig. 5 Computing time of REN. Average time (in seconds) needed to calculate the CTBC for 200 vertices using the `REN` algorithm. *Colours* indicate graph sequences with different number of snapshots. Each snapshot is based on a random Erdős-Rényi model with graph densities ranging from 0.03 up to 0.96. Calculations were done on a single core of a 2.53GHz Intel hexacore processor using 8GB memory

Results

Simulation study

We define a *group infection network* (GIN) to compare the performance of the CTBC to the TBC in an incomplete graph sequence setting. A GIN contains $M \in \mathbb{N}$ subgraphs $G(V^{(m)}, E^{(m)}), m = 1, \ldots, M$. GINs are either undirected or directed, but neither multiple edges between vertices nor loops (i.e. $\{a, a\}$) are allowed. The probability p of an edge is the sum of a baseline probability τ and the probability $\tau_+ = D^{(m)}(a)/|E^{(m)}|$, where $D^{(m)}(a)$ denotes the degree of a node $a \in V^{(m)}$ (i.e. the number of its incident edges) and $|E^{(m)}|$ denotes the total number of edges in subgraph m. Thus, τ_+ reflects a rich-get-richer principle.

We used the representation of a graph sequence consisting of $k = 1, \ldots, S$ snapshots to simulate a GIN as a dynamic network. The initial GIN contains no edges. At snapshot $k = 1$ a first vertex is randomly chosen and edges connecting it with any other vertices generated independently with probability p. At snapshots $k \geq 2$ all vertices having one or more incident edges are allowed to connect with other vertices of the same subgraph with probability p. After $\kappa \cdot m < S$ snapshots, $\kappa \in \mathbb{N}$, a connected vertex is randomly chosen as bridge vertex b. At the next snapshot, only the bridge vertex builds an edge with a vertex from the next subgraph $m + 1$, meaning that only b has neighbours in $V^{(m)}$ and $V^{(m+1)}$. This process is repeated until $k = S$. Edges within a GIN remain for $\lambda \in \mathbb{N}$ snapshots and will then vanish. The dynamic of a GIN depends on λ, where small values of λ lead to rapid changes in the network structure whereas high values of λ yield slow changes in the dynamic structure. A dynamic GIN $\mathcal{G}(V, M, S, \tau, \tau_+, \kappa, \lambda)$ is thus defined by seven parameters.

We generated GINs containing 10 subgraphs, each consisting of 5,10,20,40 or 80 vertices, given an overall network size of $|V| \in [50, 100, 200, 400, 800]$, respectively. The GIN parameters were set to $\tau = 0.0125$, $\kappa = 8$ and $\lambda = 1, 2, \ldots, 10$, i.e. edge durations ranged from 1% to 10% of the total number of snapshots. We simulated 500 undirected GINs for each combination of parameters and, based on the *complete* graph sequence of 100 snapshots, we calculated the TBC from Eq. 1 for each vertex. Vertices were ranked according to their TBC values to make them comparable across graph sequences with different number of snapshots. Ranks of vertices with the same centrality value were averaged. The ranks of the *true* TBC is our reference in the following comparison.

Of each simulated complete graph sequence the *incomplete* graph sequences were generated by randomly drawing $\alpha = 10\%, 20\%, \ldots, 50\%$ snapshot, i.e. containing 10, 20, 30, 40 and 50 snapshots. TBC and CTBC (cf. Eq. 2) were estimated for each vertex and ranks were assigned according to their respective centrality values. To calculate CTBC, we set the number of clones equal to the number of unobserved snapshots between two observed snapshots, following our third proposed approach regarding the question how to choose the number of clones. This implies a tendency to overestimate the edge duration. As a consequence false temporal paths might be included (see next section).

For every simulation run, Spearman's rank correlation coefficient ρ between the ranks based on the true TBC values and the TBC respectively CTBC values of the incomplete graph sequence were computed. A high positive ρ indicates that the centrality measure relying on incomplete information ranks the vertices similar to the true ranks. In addition, the detection rate was assessed, which is the proportion of how often the most important vertex (rank 1) in the incomplete graph sequences matches the true most important vertex of the complete graph sequence in all simulation runs.

The Box plots in Fig. 6 show the results of the rank correlation ρ for different edge durations λ, observation rate α and network sizes $|V|$. In all incompleteness scenarios, CTBC outperforms TBC for all λ except for the combination ($\alpha = 10\%$, $\lambda = 1$). This could be due to an excess of

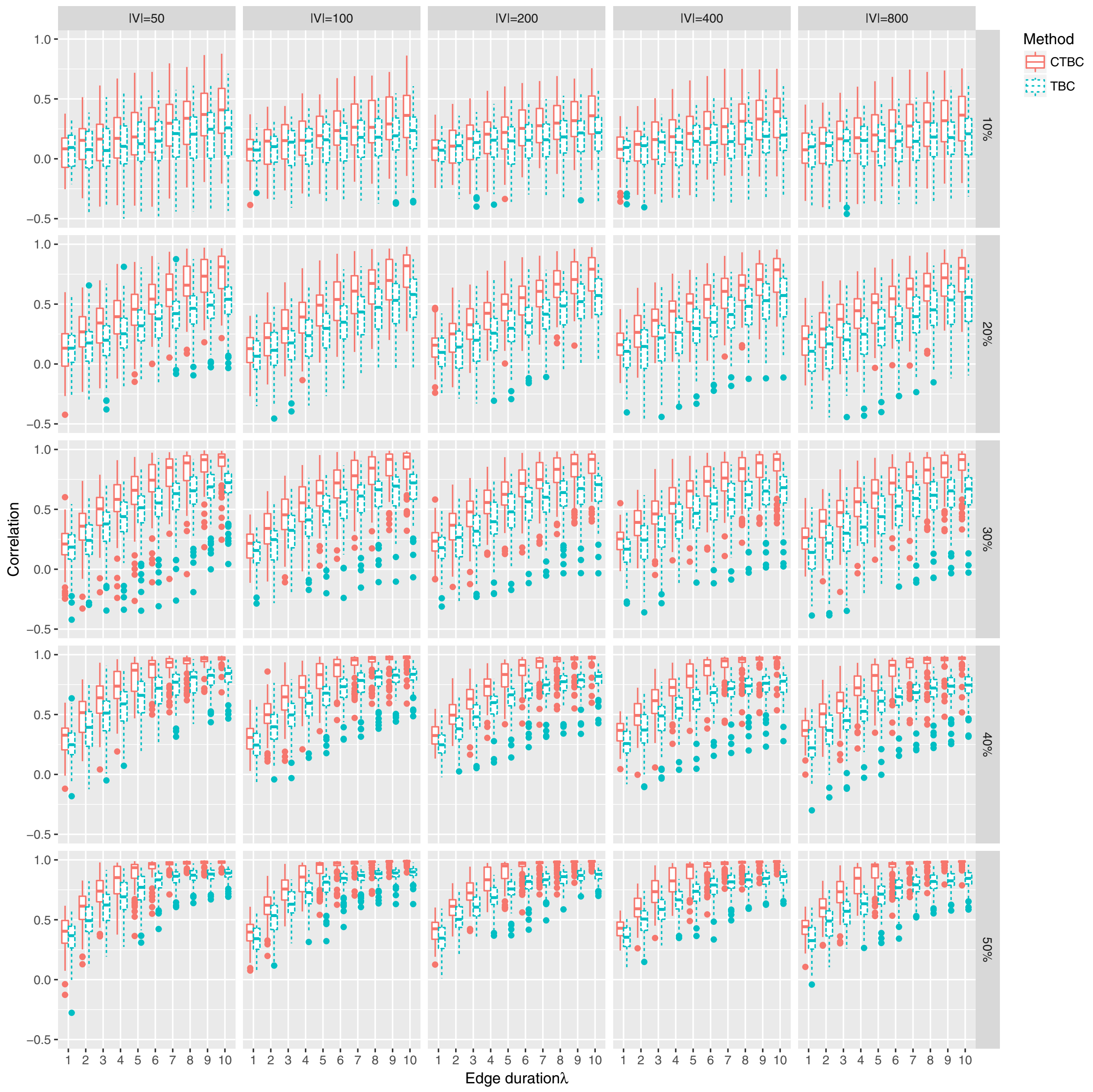

Fig. 6 Spearman's rank correlation coefficient ρ for TBC and CTBC in an undirected GIN scenario. Box plots of ρ for TBC (*dotted*) and CTBC (*solid*) based on different combinations of number of vertices ($|V| \in [50, 100, 200, 400, 800]$), different proportions of randomly observed snapshots (0.2, 0.3, 0.4, 0.5) of the original graph sequence consisting of 100 snapshots and different edge durations ($\lambda \in [1, 2, \ldots, 10]$). The results are based on 500 simulation runs for every combination

cloning. As expected, longer edge durations ($\lambda > 1$) considerably improve the performance of CTBC compared to TBC. In addition, the results indicate that the improvement is independent of the network size $|V|$ (columns of Fig. 6). Interestingly, CTBC was strongly correlated ($\rho \approx 1$) with the true TBC for longer edge durations in settings where at least 40% snapshots were observed, while TBC reached a plateau at a lower correlation. In general, if only $\alpha = 10\%$ of all snapshots were observed, both methods were weakly correlated with the true TBC, even in situations with an edge duration of $\lambda = 10$ indicating that long edge durations cannot compensate for missing edge observations.

Figure 7 shows the detection rate for the most important vertex. While TBC and CTBC had poor detection rates in settings of low observation rates ($\alpha = 10\%$), the detection

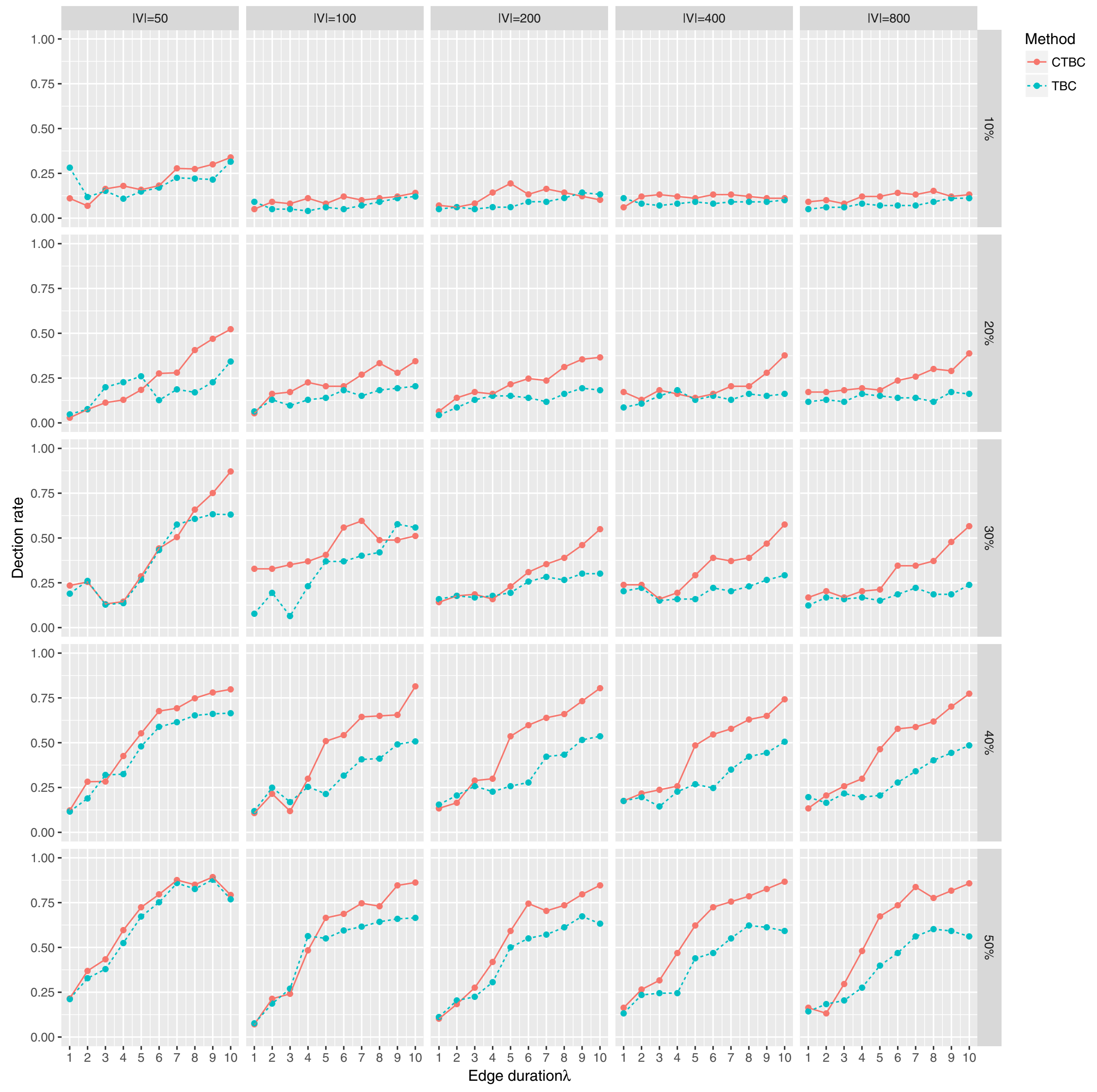

Fig. 7 Detection rate of the most important vertex based on TBC and CTBC in an undirected GIN scenario. Detection rate for TBC (*dotted*) and CTBC (*solid*) based on different combinations of number of vertices ($|V| \in [50, 100, 200, 400, 800]$), different proportions of randomly observed snapshots (0.2, 0.3, 0.4, 0.5) of the original graph sequence consisting of 100 snapshots and different edge durations ($\lambda \in [1, 2, \ldots, 10]$). The results are based on 500 simulation runs for every combination

rate of CTBC tended to be better in settings with larger observation rates, especially in combination with longer edge durations.

The simulation results for the temporal closeness centrality support our proposal of cloning snapshots, even if the benefit was smaller than for the temporal betweenness centrality, especially regarding the detection rate of the most important vertex (see Fig. 8 and Fig. 9).

Excess of cloning

As mentioned before, an excess of cloning can introduce false (shortest) temporal paths which lead to biased centrality values. In a further simulation study, we evaluated this bias by generating a GIN with the given parameters $|V| = 200$, $M = 10$, $\tau = 0.0125$, $\kappa = 8$, $S = 50$ and $\lambda = 1, 2, 3$. Incomplete graph sequences were sampled assuming an observation rate of $\alpha = 25\%, 50\%, 100\%$. That means, for example in the scenario $\alpha = 100\%$ all true snapshots were observed and for each snapshot a specified number of clones were wrongly introduced. As before, true ranks were based on the TBC values for the original graph sequence. For the calculation of CTBC, we fixed the number of clones to $n_c = 0, \ldots, 8$.

Figure 10 shows clearly the expected problem of an excess of cloning in scenario $\alpha = 100\%$ (first row). As expected, the original TBC and no cloning ($n_c = 0$) are perfectly correlated ($\rho = 1.0$), but the correlation of CTBC decreases with each additional clone. Note, that for $n_c = 1$ the length of the graph sequence is already doubled. However, the effect of an excess of cloning is less bad for longer edge durations λ.

The scenarios with lower observation rates show that the correlation values of CTBC are comparable to the values of TBC in settings with shorter edge durations or even larger for longer edge durations – despite the excess of cloning. Most important, although the performance of CTBC decreases with additional number of clones, it outperforms TBC even for large n_c.

Application to real dynamic networks

We used a real age-related dynamic network to investigate the performance of CTBC compared to TBC in a real world application. The dynamic network was created from a microarray human brain gene expression data set [18] that consists of 173 samples obtained from 55 individuals between 20 and 99 years of age. The reader may wish to refer to [11] for more details on the generation of this age-specific protein-protein-interaction network. From the original dynamic network, we selected only genes belonging to the KEGG metabolic pathways (hsa:01100) [29, 30] and their adjacent genes outside this pathway. This dynamic subnetwork contained 1,128 genes (vertices) and 31,643 temporal edges between 1,275 different vertex pairs which were connected by an edge at least in 1 out of 37 time points. Overall, the subnetwork contained 506 permanent edges that were present at all 37 snapshots, but also 1,931 temporal edges that existed only for one snapshot. Disregarding the permanent edges, the subnetwork showed a right skewed distribution of short to long edge durations.

To verify that the subnetwork kept the dynamic behavior of the whole network, we compared both regarding their dynamic edge density, that is the ratio between the observed number of edges at time t and the total number of possible edges at that time point. The dynamic edge density was similar for both networks the original network at all time points.

We used all observed 37 time points to calculate the *true* TBC of the dynamic subnetwork and ranked the vertices according to their TBC value. Then we selected every fourth snapshot to build an incomplete graph sequence with nine snapshots. The incomplete graph sequence contained 23% of the original 31,643 temporal edges that were present in 80% of the original 1,275 vertex pairs. Vertices were ranked according to their TBC and CTBC value estimated in the incomplete graph sequence. CTBC was calculated ten times where the number of clones n_c between snapshots was increased from one to ten.

The performance of TBC and CTBC was measured by the absolute rank difference (ARD) that compares the estimated ranks of the incomplete graph sequence to the true ranks of the complete graph sequence. Results are summarized in Table 1. It can be seen that all versions of $\text{CTBC}^{(n_c)}, n_c = 1, \ldots, 10$, outperformed TBC regarding the median, the first and third quartile as well as the interquartile range of the ARD. Further, the ARD of all CTBC versions showed smaller variability than of TBC. The median of CTBC has its minimum for seven or more clones, while the first and third quartiles are lowest for $\text{CTBC}^{(4)}$. The similarity of CTBC versions with six or more clones per snapshot and their coincident improvement of the ARD compared to TBC suggests that CTBC is robust against false positive edges introduced by cloning. We further calculated Spearman's rank correlation coefficient ρ between the true ranks and the estimated ranks by TBC and CTBC. Albeit all methods achieved a high positive correlation with the true ranks ($\rho \geq 0.89$), CTBC had higher correlation values than TBC in all versions.

Since incomplete graph sequences might completely miss some edges, the centrality values of vertices being incident to missing edges can be heavily biased. It is obvious that due to the information loss of these edges even cloning cannot decrease the ARD. In the real data example, this is reflected by very high absolute rank differences in all versions of CTBC and TBC, marked as outliers in the box plots (see Fig. 11).

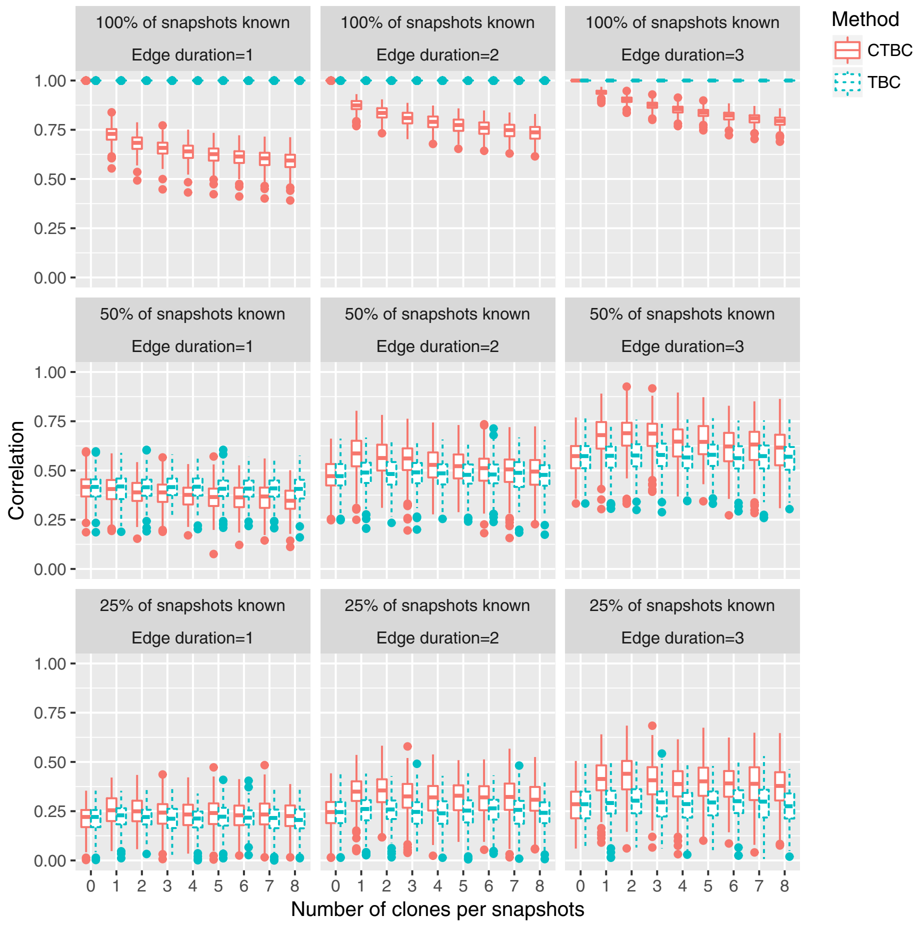

Fig. 8 Spearman's rank correlation coefficient ρ for TCC and CTCC in an undirected GIN scenario. Box plots of ρ for TCC (*dotted*) and CTCC (*solid*) based on different combinations of number of vertices ($|V| \in [200, 400, 800]$), different proportions of randomly observed snapshots (0.2, 0.3, 0.4, 0.5) of the original graph sequence consisting of 100 snapshots and different edge durations ($\lambda \in [1, 2, \ldots, 10]$). The results are based on 500 simulation runs for every combination

Discussion and conclusion

To the best of our knowledge this is the first work that introduced the problem of incomplete graph sequences when calculating temporal centrality measures. Our extension of existing temporal centrality measures addresses this problem by adding 'clones' of observed snapshots as extra snapshots into the graph sequence. The idea was motivated by real world dynamic networks, where edges occur for shorter and longer time durations rather than only during the specific observed snapshot. Furthermore, incomplete graph sequences are the rule rather than the exception in experimental and observational studies, where typically only a few snapshots of the total graph sequence can be obtained due to ethical, technical or financial reasons with varying time length between snapshots.

Since the clone temporal centralities augment the original graph sequence by adding snapshots, we needed an algorithm that can handle large graph sequences in reasonable time. With our new algorithm `REN` (Reversed Evolution Network) (shortest) temporal paths can be detected efficiently along a successively by one snapshot reduced graph sequence. The time complexity of the algorithm is linear in the number of snapshots and hence it allows the calculation of temporal centrality measures even in settings with long graph sequences.

Using the clone temporal betweenness centrality (CTBC) as an example for clone temporal centralities, our simulation studies demonstrate a superiority of CTBC relative to the original temporal betweenness centrality (TBC) [15] with respect to Spearman's ρ and the detection rate of the most important network vertex. We also applied CTBC and TBC to a data set of an age-related gene expression network of the human brain, consisting of edges with shorter and longer durations. The analysis confirmed the better performance of CTBC compared to TBC. Both, the results from the simulation study and the real data example showed that the cloned temporal

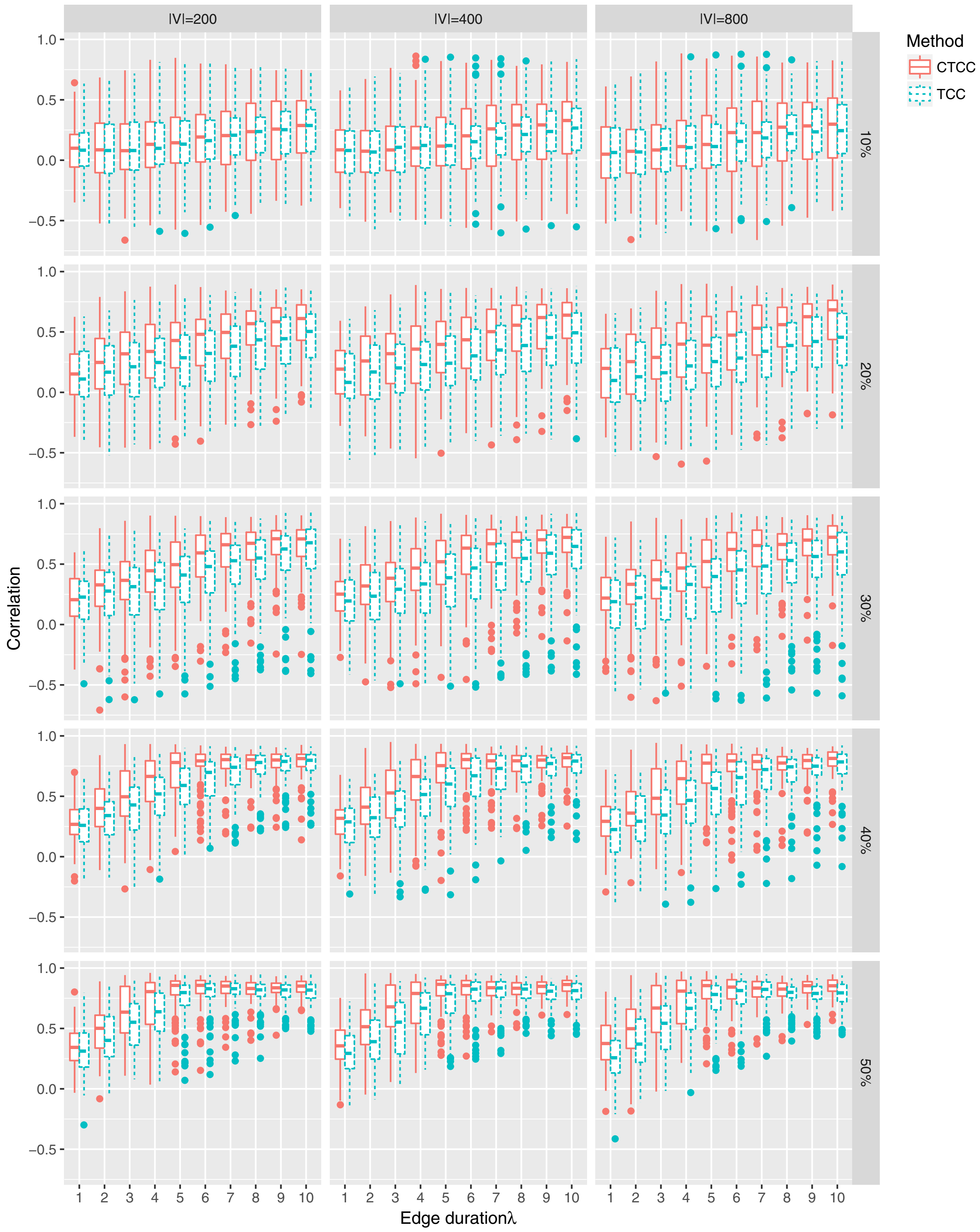

Fig. 9 Detection rate of the most important vertex based on TCC and CTCC in an undirected GIN scenario. Detection rate for TCC (*dotted*) and CTCC (*solid*) based on different combinations of number of vertices ($|V| \in [200, 400, 800]$), different proportions of randomly observed snapshots (0.2, 0.3, 0.4, 0.5) of the original graph sequence consisting of 100 snapshots and different edge durations ($\lambda \in [1, 2, \ldots, 10]$). The results are based on 500 simulation runs for every combination

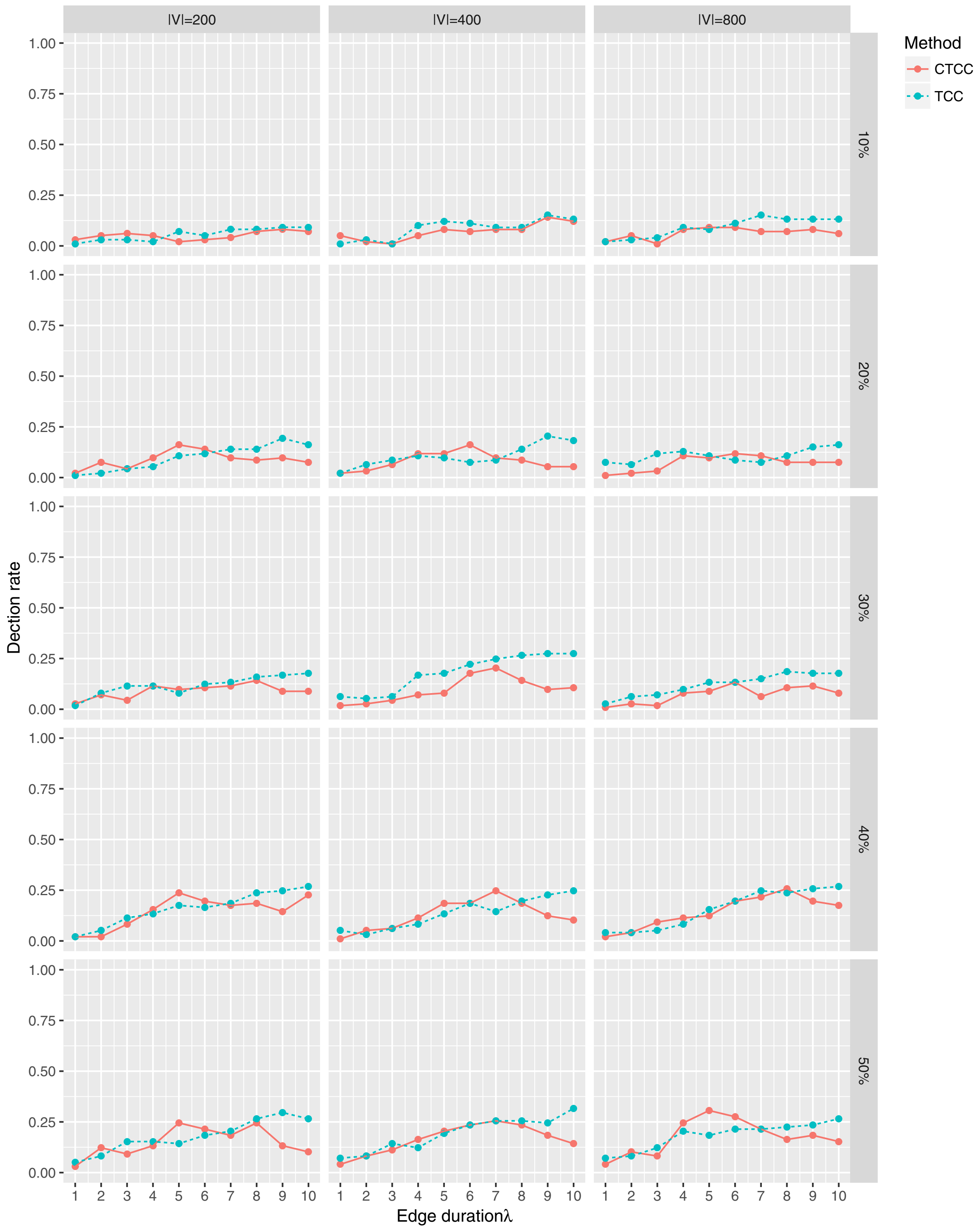

Fig. 10 Impact of an excess of cloning on Spearman's rank correlation coefficient ρ

Table 1 TBC and CTBC performance regarding absolute rank differences to true ranks and Spearman's ρ

Method	1st Qt.	Median	3rd Qt.	ρ
TBC	27.5	81.5	90.0	0.89
CTBC$^{(1)}$	24.0	49.5	65.0	**0.93**
CTBC$^{(2)}$	19.0	45.5	55.0	**0.93**
CTBC$^{(3)}$	16.5	36.5	50.0	**0.93**
CTBC$^{(4)}$	**15.0**	35.0	**47.0**	**0.93**
CTBC$^{(5)}$	17.0	35.0	48.0	0.92
CTBC$^{(6)}$	16.0	34.5	59.0	0.92
CTBC$^{(7)}$	17.0	**34.0**	61.0	0.92
CTBC$^{(8)}$	16.0	**34.0**	63.0	0.92
CTBC$^{(9)}$	18.0	**34.0**	68.0	0.92
CTBC$^{(10)}$	18.0	**34.0**	70.0	0.92

CTBC^{n_c} was calculated using n_c clones per snapshots. Bold numbers indicate the minimum value

centralities are affected by an excess of cloning, since the true edge durations will tend to be overestimated, which again can result in the detection of false temporal paths. Except in data scenarios with short edge durations, cloning still provides better results even if too many clones were introduced in the observed snapshot sequence. There are three intuitive explanations why our approach outperforms the original approach even under an excess of cloning:

1. Not all wrongly introduced temporal paths due to cloning are *shortest* temporal paths and hence will not alter the cloned temporal centrality measures that are based on shortest temporal paths.
2. The original approach does not only miss true shortest temporal paths, it also *detects* false shortest temporal paths. This is due to the definition of a shortest temporal path: it is the temporal paths with the smallest number of hops and halts of all temporal paths between two vertices. For example, assume that there exist only two temporal paths, starting at a specific snapshot. Further, let one of them be a shortest temporal path. If only the longer temporal path can be found - due to the incomplete graph sequence - it will be falsely declared as a shortest temporal path.
3. If a shortest temporal path is missed, some of its subpaths as well as paths including this shortest temporal path will be missed too. Cloning snapshots raises the chance of finding at least some of those temporal paths.

However, while cloning snapshots is easy to implement, it cannot compensate for unobserved edges, resulting in inaccurate centrality values. Moreover, our method does not rely on probabilistic models describing the evolution of a dynamic network. Hence, we plan to investigate whether using probabilistic models for dynamic networks or exploiting a priori knowledge about the network topology can improve the estimation of temporal centrality measures.

Based on our results, we recommend using our clone temporal centrality measures in settings of incomplete

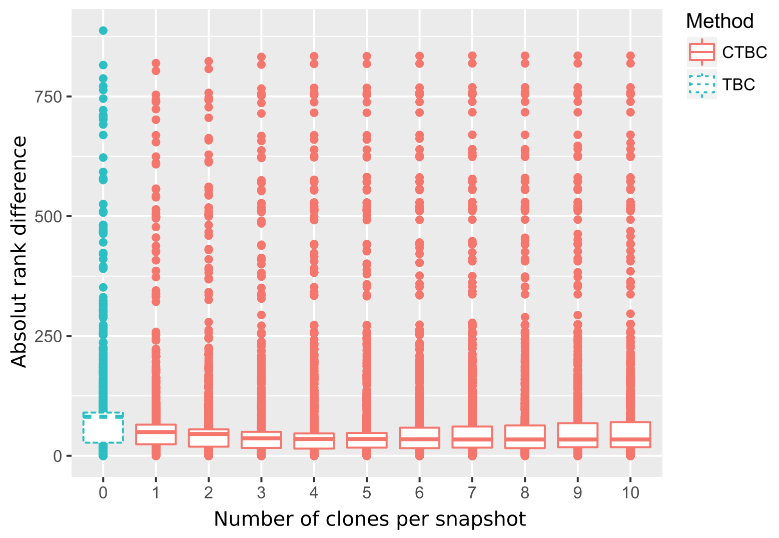

Fig. 11 Results of the age-related dynamic brain network. Box plots of the absolute rank difference for the age-related dynamic brain network. The incomplete graph sequence with 9 snapshots was built on every 4th snapshot from the original graph sequence that consisted of 37 snapshots in total. The network included 1128 vertices. CTBC was calculated with different number of clones between snapshots. Very high absolute rank differences were caused by unobserved rare edges, that were crucial for the connectivity of (groups of) vertices in the dynamic network

graph sequences instead of the original temporal centrality measures. Additionally, using REN will improve computational speed in settings of long graph sequences. The R-code of our methods is available upon request from the authors and will be made available on CRAN.

Additional file

Additional file 1: Clone temporal closeness centrality (CTCC). Definition of the clone temporal closeness centrality. (PDF 96 kb)

Acknowledgements
The authors want to thank the reviewer Benjamin Blonder and the second anonymous reviewer for their valuable comments as well as Tijana Milenkovic and Fazle Elahi Faisal for providing the age-related gene network data. Special thanks to Iris Pigeot and Vanessa Didelez for their proof-reading and valuable suggestions on an earlier draft.

Funding
The publication of this article was funded by the Open Access Fund of the Leibniz Association. The funding body played no role in the design or conclusions of this study.

Availability of data and materials
The protein-protein dataset supporting the conclusions of this article is available in the repository of Tijana Milenkovic, http://www3.nd.edu/~cone/dynetage/dynamicnetwork.html.

Authors' contributions
MH developed the CTBC/CTCC method and the REN algorithm, formulated the mathematical proofs, designed the simulation study, performed the real data analysis and drafted the manuscript. RF participated in the development of the methodology, assisted by formulating the proofs, assisted with the design of the simulation study and real data analysis and helped draft the manuscript. Both MH and RF have read and approve of the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable. Although the results contained in this manuscript were generated through the analysis of data collected from human subjects, only previously collected, publicly available and de-identified data sources were be used.

Received: 25 November 2016 Accepted: 4 May 2017
Published online: 16 May 2017

References
1. Holme P. Modern temporal network theory: a colloquium. Eur Phys J B. 2015;88(9):1–30.
2. Volz E, Meyers LA. Susceptible–infected–recovered epidemics in dynamic contact networks. Proc R Soc London B: Biol Sci. 2007;274(1628):2925–34.
3. Wölfer R, Faber NS, Hewstone M. Social network analysis in the science of groups: cross-sectional and longitudinal applications for studying intra- and intergroup behavior. Group Dyn: Theory, Res Pract. 2015;19(1):45–61.
4. Gao C, Liu J, Zhong N. Network immunization and virus propagation in email networks: experimental evaluation and analysis. Knowl Inform Syst. 2010;27(2):253–79.
5. Holme P, Saramäki J. Temporal networks. Phys Rep. 2012;519(3):97–125.
6. Hulovatyy Y, Chen H, Milenković T. Exploring the structure and function of temporal networks with dynamic graphlets. Bioinformatics. 2015;31(12):171–80.
7. Nicosia V, Tang J, Mascolo C, Musolesi M, Russo G, Latora V. In: Holme P, Saramäki J, editors. Graph Metrics for Temporal Networks. Berlin: Springer; 2013. pp. 15–40.
8. Tang J, Musolesi M, Mascolo C, Latora V, Nicosia V. Analysing information flows and key mediators through temporal centrality metrics. In: Proceedings of the 3rd Workshop on Social Network Systems. SNS '10. New York: ACM; 2010. p. 3–136.
9. Kostakos V. Temporal graphs. Phys A: Stat Mech Appl. 2009;388(6): 1007–23.
10. Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU. Complex networks: Structure and dynamics. Phys Rep. 2006;424(4–5):175–308.
11. Faisal FE, Milenković T. Dynamic networks reveal key players in aging. Bioinformatics. 2014;30(12):1721–9.
12. Tang J, Scellato S, Musolesi M, Mascolo C, Latora V. Small-world behavior in time-varying graphs. Phys Rev E. 2010;81:055101.
13. Grindrod P, Higham DJ, Parsons MC, Estrada E. Communicability across evolving networks. Phys Rev E. 2011;83:046120.
14. Pan RK, Saramäki J. Path lengths, correlations, and centrality in temporal networks. Phys Rev E. 2011;84:016105.
15. Kim H, Anderson R. Temporal node centrality in complex networks. Phys Rev E. 2012;85:026107.
16. Alsayed A, Higham DJ. Betweenness in time dependent networks. Chaos, Solitons Fractals. 2015;72:35–48.
17. Katz L. A new status index derived from sociometric analysis. Psychometrika. 1953;18(1):39–43.
18. Berchtold NC, Cribbs DH, Coleman PD, Rogers J, Head E, Kim R, Beach T, Miller C, Troncoso J, Trojanowski JQ, Zielke HR, Cotman CW. Gene expression changes in the course of normal brain aging are sexually dimorphic. Proc Nat Acad Sci. 2008;105(40):15605–10.
19. Blonder B, Wey TW, Dornhaus A, James R, Sih A. Temporal dynamics and network analysis. Methods Ecol Evolu. 2012;3(6):958–72.
20. Liang Q, Modiano E. Survivability in time-varying networks. In: 35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10–14, 2016; 2016. p. 1–9.
21. Li F, Chen S, Huang M, Yin Z, Zhang C, Wang Y. Reliable topology design in time-evolving delay-tolerant networks with unreliable links. IEEE Trans Mobile Comput. 2015;14(6):1301–14.
22. Scellato S, Leontiadis I, Mascolo C, Basu P, Zafer M. Evaluating temporal robustness of mobile networks. IEEE Trans Mobile Comput. 2013;12(1): 105–17.
23. Kempe D, Kleinberg J, Kumar A. Connectivity and inference problems for temporal networks. J Comput Syst Sci. 2002;64(4):820–42.
24. Berman KA. Vulnerability of scheduled networks and a generalization of menger's theorem. Networks. 1996;28(3):125–34.
25. Costenbader E, Valente TW. The stability of centrality measures when networks are sampled. Soc Netw. 2003;25(4):283–307.
26. Borgatti SP, Carley KM, Krackhardt D. On the robustness of centrality measures under conditions of imperfect data. Soc Netw. 2006;28(2): 124–36.
27. Magnien C, Tarissan F. Time evolution of the importance of nodes in dynamic networks. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015. ASONAM '15. New York: ACM; 2015. p. 1200–1207.
28. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. Cambridge: The MIT Press; 2009.
29. Kanehisa M, Sato Y, Kawashima M, Furumichi M, Tanabe M. KEGG as a reference resource for gene and protein annotation. Nucleic Acids Res. 2016;44(D1):457–62.
30. Kanehisa M, Goto S. KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Res. 2000;28(1):27–30.