Temporal Walk Centrality: Ranking Nodes in Evolving Networks

Lutz Oettershagen  
petra.mutzel@cs.uni-bonn.de  
University of Bonn  
Bonn, Germany

Petra Mutzel  
petra.mutzel@cs.uni-bonn.de  
University of Bonn  
Bonn, Germany

Nils M. Kriege  
nils.kriege@univie.ac.at  
University of Vienna  
Vienna, Austria

ABSTRACT
We propose the Temporal Walk Centrality, which quantifies the importance of a node by measuring its ability to obtain and distribute information in a temporal network. In contrast to the widely-used betweenness centrality, we assume that information does not necessarily spread on shortest paths but on temporal random walks that satisfy the time constraints of the network. We show that temporal walk centrality can identify nodes playing central roles in dissemination processes that might not be detected by related betweenness concepts and other common static and temporal centrality measures. We propose exact and approximation algorithms with different running times depending on the properties of the temporal network and parameters of our new centrality measure. A technical contribution is a general approach to lift existing algebraic methods for counting walks in static networks to temporal networks. Our experiments on real-world temporal networks show the efficiency and accuracy of our algorithms. Finally, we demonstrate that the rankings by temporal walk centrality often differ significantly from those of other state-of-the-art temporal centralities.

CCS CONCEPTS
• Information systems → Social networks; • Theory of computation → Graph algorithms analysis.

KEYWORDS
Temporal Network, Centrality, Node Ranking, Temporal Walk

1 INTRODUCTION
Measuring the centrality of nodes in a network is a cornerstone of network analysis. The goal is to determine the importance of nodes in the network and find the most central ones. Various concepts of centrality have been proposed [33, 48], and their informative value must be assessed based on a research question. A prime example is the PageRank algorithm [40] for ranking web pages in search engine results by performing random walks among them. Random walks also form the basis of the classical Katz centrality [18], which measures node importance in terms of the number of random walks starting (or arriving) at a node, down-weighted by their length. Thereby, the Katz centrality measures the ability of a node to send out or receive information. A different widely-used notion of centrality is betweenness. Freeman [9] defines the betweenness of a node as the fraction of shortest paths between pairs of nodes that pass through it. Therefore, betweenness determines the importance of a node by its ability to pass on information. However, considering only the shortest paths can be too restrictive since information or diseases do not necessarily spread along shortest paths. Therefore, a betweenness centrality based on random walks has been considered [32], which takes all walks that visit a node into account. These centrality measures, as well as many more, are primarily designed for static networks. However, real-world networks are often dynamic, and their topology changes over time [16]. Recently, the study of centrality in temporal networks in which temporal edges only exist at specific points in time has gained increasing attention [2, 5, 37, 38, 46]. Important examples of such temporal networks include communication and web-based social networks such as email correspondences or human contacts [16].

Our work: We introduce a new centrality measure for temporal networks called Temporal Walk Centrality, which assesses the importance of a node by its ability to obtain and distribute information. Our new centrality measure generalizes the temporal Katz centrality as well as the degree centrality [2, 12]. Like the static random walk betweenness, the temporal walk centrality counts random walks passing through a node. However, the temporal nature of sequential events modeled by temporal networks implies causality by the forward flow of time, which needs to be respected in temporal network analysis. For example, if there is a contact between individual A and B at time \( t_1 \) and B and C at time \( t_2 > t_1 \), information may pass from A to C via B but not vice versa. To take this into account, we use the concept of temporal walks that respect the flow of time in contrast to static walks. Temporal walks only allow following an edge at any node if the edge exists at a time point not earlier than the arrival time at the node. Analogously to [5], we distinguish between strict and non-strict temporal walks. We assume that a global transition time is required to traverse edges, which is allowed to be zero. In this case, non-strict temporal walks can include consecutive edges with equal time stamps and may contain cycles. If the transition time is non-zero, each edge can be contained at most once in a strict temporal walk. While some previously proposed temporal walk-based techniques only support strict temporal walks [2, 46], others are designed to take non-strict temporal walks of unbounded length into account [12]. The temporal walk centrality is general and supports both models. We propose corresponding algorithms with different properties.
Contributions: We make the following contributions:

1. We introduce the temporal walk centrality, which captures the ability of nodes in temporal networks to obtain and distribute information. We demonstrate that nodes with high temporal walk centrality are key players in the dissemination of information.

2. We present efficient algorithms for computing the temporal walk centrality with non-strict and strict temporal walks. For the first case, we propose to derive a static directed line graph from which temporal walks can be counted by general algebraic techniques. This yields an exact and iterative approximate algorithm with running time bounded by \( O(km^2) \), where \( m \) is the number of temporal edges and \( k \) the number of iterations. For strict temporal walks, we introduce a highly efficient and scalable streaming-based algorithm with running time in \( O(m \cdot \tau_{\max}) \), where \( \tau_{\max} \) is the maximal number of arrival or starting times at any node.

3. In our evaluation, we show that our approximation is efficient and achieves high-quality solutions. Furthermore, our streaming algorithm is fast even for temporal graphs with hundreds of millions of edges. Finally, our experiments show that the temporal walk centrality differs from other state-of-the-art temporal centrality measures recommending its application in scenarios where information spreads along random walks.

Due to the space restrictions, all proofs can be found in Appendix B.

Relevance to Web Research: Identifying and ranking nodes of web-based social networks and communication networks according to their importance in the dissemination of information are critical. Relevance to Web Research: Identifying and ranking nodes of web-based social networks and communication networks according to their importance in the dissemination of information are critical.

2 RELATED WORK

There are excellent introductions to temporal graphs that include surveys on different temporal centrality measures, see, e.g., [16, 25, 47]. Further overviews of centralities and their applications are provided, e.g., in [8, 10, 24, 44, 45, 48, 56]. In a recent work, Buß et al. [5] discuss several variants of temporal betweenness based on shortest paths and study their theoretical complexity and practical hardness. They consider different criteria of path lengths, such as arrival time and total travel time. They show that the running time for computing temporal betweenness using temporal shortest paths is in \( O(n^3 \cdot T^2) \) with \( n \) the number of nodes and \( T \) the total number of time steps in the temporal graph. This complexity holds for strict and non-strict temporal paths. In [52], the authors extend Brandes’ algorithm [3] for distributed computation of betweenness centrality in temporal graphs. They introduce shortest-fastest paths as a combination of the conventional distance and shortest duration. The authors of [2, 12] adapt the walk-based Katz centrality [18] to temporal graphs. Rozenstein and Gionis [46] incorporate the temporal character in the definition of the PageRank and consider a walk-based perspective. Hence, they obtain a temporal PageRank by replacing walks with temporal walks. Several papers compare temporal distance metrics as well as temporal versions of centrality measures to their static counterparts, e.g., Nicosa et al. [35], Kim and Anderson [22], Tang et al. [50, 51], and show that temporal approaches have advantages over static approaches on the aggregated graphs. In [7] and [37], the authors introduce top-k algorithms for temporal closeness variants. The authors of [13] introduce a closeness variant based on bounded random-walks. Related to the considered node importance is the concept of influence in social networks, which has been studied extensively, see, e.g., [19, 58] and references therein. Here, one is interested in a subset of the nodes that, when activated (e.g., convinced to adopt a product), have the strongest effect on the network according to some diffusion model. Finding such a set is typically NP-hard but can often be approximated with guarantees [19]. Recent, dynamic graph algorithms and approaches for temporal networks have been proposed [58].

3 PRELIMINARIES

An undirected (static) graph \( G = (V, E) \) consists of a finite set of nodes \( V \) and a finite set \( E \subseteq \{(u, v) | u \neq v \} \) of undirected edges. In a directed (static) graph \( G = ((u, v) \in V \times V | u \neq v) \). We use \( V(G) \) to denote the set of nodes of \( G \). The out-degree \( d^+(v) \) and in-degree \( d^-(v) \) of a node \( v \) in a directed graph is the number of outgoing edges \((v, \cdot)\) and incoming edges \((\cdot, v)\) in \( E \), respectively. A walk in a graph \( G \) is an alternating sequence of nodes and edges connecting consecutive nodes. For notational convenience we sometimes omit edges. The length of a walk is the number of edges it contains.

A temporal graph \( G = (V, E, \delta) \) consists of a finite set of nodes \( V \), a finite set \( E \) of directed temporal edges \( e = (u, v, t) \) with \( u, v \) in \( V \), \( u \neq v \), and availability time (or time stamp) \( t \in \mathbb{N} \). The parameter \( \delta \in \mathbb{N} \) is a global transition delay of the edges that defines how long it takes to transmit information via an edge. The starting time of an edge \( e = (u, v, t) \) at node \( u \) is \( t \), and the arrival time at node \( v \) is \( t + \delta \). The number of edges is not necessarily polynomially bounded by the number of nodes because each pair of nodes can be connected at several points in time. For \( v \in V \) let \( T(v) \) be the set of availability times of edges incident to \( v \). We define \( T(G) = \{t, t + \delta | (u, v, t) \in E\} \). Furthermore, let \( \tau_{\max} (\tau_{\max}^2) \) denote the number of maximum of distinct arrival (starting) times at any of the nodes. The in- and out-degree of a node in a temporal graph is the total numbers of incoming and outgoing edges over all time steps. We only consider directed temporal graphs and model undirectedness using a forward- and a backward-directed edge with equal time stamps for each undirected edge.

A temporal walk \( \omega \) in a temporal graph \( G \) is an alternating sequence \((v_1, e_1, \ldots, e_\ell, v_\ell+1)\) of nodes and temporal edges connecting consecutive nodes, where \( e_i = (v_i, v_{i+1}, t_i) \in E \) and \( t_i + \delta \leq t_{i+1} \) for all \( i \in \{1, \ldots, \ell\} \). We may omit nodes for notational convenience. Depending on \( \delta \), we distinguish strict and non-strict temporal walks, where for \( \delta = 0 \), the temporal walks are non-strict. We denote the length of a temporal walk \( \omega \) by \( |\omega| = \ell \). For \( \delta > 0 \), the length of a (strict) temporal walk is bounded by \( T(G) \), where for \( \delta = 0 \), there is no general upper bound on the length of non-strict temporal walks. It is common to restrict temporal walks to a time interval \([a, b]\) with \( a, b \in \mathbb{N} \) and \( a \leq b \). This case is covered by running our algorithms.
on the temporal subgraph containing only the edges \((u, v, t) \in E\) for which \(a \leq t \) and \( t + \delta \leq b \). Moreover, our definitions and algorithms can be easily extended to the case of individual transition times \(\delta_e\) for each edge \(e\) instead of the global parameter \(\delta\). Table 6 in Appendix A summarizes our notation.

4 TEMPORAL WALK CENTRALITY

The intuition of our new centrality measure is that nodes are regarded as important if they are involved in the process of passing information. The time stamps of the temporal edges imply causality and direct the information flow in the network. Therefore, we measure the contribution of a node by means of temporal walks respecting such aspects. We define the centrality of a node \(v\) as the number of temporal walks passing through \(v\), where the temporal walks are weighted depending on their length and temporal structure. We formalize this concept before proposing our new centrality measure, and define a weight function for temporal walks similar to [2] and [34].

Definition 4.1 (Temporal walk weight). Let \(G = (V, E, \delta)\) be a temporal graph, and \(\omega = (e_1, \ldots, e_t)\) a temporal walk in \(G\). We define the weight of a temporal walk \(\omega\) as

\[
\tau_{\Phi}(\omega) = \prod_{i=1}^{t-1} \Phi(t_i + \delta, t_{i+1}),
\]

where the function \(\Phi: N \times N \rightarrow \mathbb{R}\) is a time depended weight function. We define \(\tau_{\Phi}(\omega') = 1\) for walks \(\omega'\) of length zero and one.

We discuss concrete examples of the function \(\Phi\) later in this section. First, we define the total weight of incoming and outgoing temporal walks at each node \(v \in V\).

Definition 4.2. Let \(G = (V, E, \delta)\) be a temporal graph, and let \(W_{in}(v, t)\) and \(W_{out}(v, t)\) be the sets of incoming and outgoing temporal walks, resp., at node \(v\) and time \(t\). We define

\[
W_{in}(v, t) = \sum_{\omega \in W_{in}(v, t)} \tau_{\Phi}(\omega) \quad \text{and} \quad W_{out}(v, t) = \sum_{\omega \in W_{out}(v, t)} \tau_{\Phi}(\omega).
\]

The temporal walk weight functions \(\tau_{\Phi}\) of incoming and outgoing walks allow to weight incoming and outgoing walks independently. Using \(W_{in}\) and \(W_{out}\), we now define the temporal walk centrality.

Definition 4.3 (Temporal Walk Centrality). Let \(G = (V, E)\) be a temporal graph. We call

\[
C(v) = \sum_{t_1, t_2 \in T(G), t_1 \leq t_2} (W_{in}(v, t_1) \cdot W_{out}(v, t_2) \cdot \Phi_m(t_1, t_2))
\]

the temporal walk centrality of node \(v \in V\).

The time-dependent weight function \(\Phi_m\) is, similarly to \(\Phi_{in}\) and \(\Phi_{out}\), used to weight the time between obtaining and distributing information at node \(v\). Using these functions, we can weight temporal walks depending on different structural and temporal properties. We propose the following weight functions:

1. **Weighting based on length**: By setting \(\Phi(t_1, t_2) = \alpha\), with \(0 < \alpha < 1\), for all \(t_1, t_2 \in \mathbb{N}\), we obtain an exponential decay in the length of the walk, i.e., \(\tau_\Phi(\omega) = \alpha^{|\omega|}\). In this case, we set \(\Phi_m(t_1, t_2) = 1\). Hence, long walks are down-weighted compared to short walks controlled by the parameter \(\alpha\). In the following, we denote this variant with \(\Phi_\alpha\).

2. **Weighting based on waiting time**: The value of information decreases with time, and we might want to weight the ability to quickly distribute new information high. In this case, we define \(\Phi(t_1, t_2) = \frac{t_1}{t_1 + \delta} \cdot \frac{t_2}{t_2 + \delta}\). The weight decreases with increasing waiting time at every node and remains stable at nodes, where information is passed through without delay. In the following, we denote this variant with \(\Phi_t\).

In both examples, we set \(\Phi_{in}(t_1, t_2) = \Phi_{out}(t_1, t_2) = \Phi(t_1, t_2)\). Notice, that our definition is general and supports further weight functions such as a combination of (1) and (2), where we set \(\Phi(t_1, t_2) = \frac{t_1}{t_1 + \delta} \cdot \frac{t_2}{t_2 + \delta}\) for all \(t_1, t_2 \in \mathbb{N}\). Another example would be to use different values for \(\alpha\) for incoming and outgoing walks. Finally, we can achieve an exponential decay in the total waiting time or duration by choosing a suitable time-dependent exponential function for \(\Phi\).

To see how the TemporalWalkCentrality generalizes the temporal Katz and degree centrality, consider the following. If we fix \(W_{in}(v, t) = 1\), and use the weighting function \(\Phi(\omega, v) = \alpha\) it follows that \(C(v)\) equals the temporal Katz centrality. If we, additionally, only consider walks of length one, \(C(v)\) equals the outdegree centrality. Note that a walk length restriction can be added straightforwardly.

4.1 Comparison to Other Centrality Measures

We compare our new temporal walk centrality to state-of-the-art centrality measures for temporal and static graphs using an example graph and a subgraph of a real-world communication network.

**First example**: Consider the temporal graph \(G = (V, E, \delta)\) with \(\delta = 1\) and availability times shown in Figure 1. Nodes \(a, f, \) and \(g\) cannot pass any information and are marked in gray. Node \(a\) does not have any incoming edges. Thus, it cannot pass any information. Similarly, node \(g\) does not have any outgoing edges. For node \(f\) the only outgoing edge has availability time two. However, information can reach \(f\) only via edge \((e,f,5)\) at time 6 at the earliest. Hence, node \(f\) cannot pass on any information in a dissemination process.

Table 1 shows a comparison of the resulting rankings of the nodes of the temporal graph shown in Figure 1. The rankings are computed with different temporal and static centrality measures. Temporal betweenness is the shortest paths variant from [5]. It counts the number of temporal shortest paths that visit a node. The temporal PageRank [46] is a temporal version of the static PageRank centrality [4]. Temporal Katz centrality is defined in [2]. The nodes are ranked according to the number of incoming temporal walks weighted by a time depended exponential decay function. The temporal closeness centrality is a harmonic closeness variant using the shortest duration as distance function [37, 38]. Our temporal walk centrality is computed with \(\tau_{\Phi_{in}}(\omega) = \tau_{\Phi_{out}}(\omega) = 1\) for all walks \(\omega\) in \(G\), and we set \(\Phi_m(t_1, t') = 1\) for all \(t, t' \in \mathbb{N}\).

We observe that only the temporal walk centrality can identify the nodes \(a, f, \) and \(g\) as nodes that have no capability of passing information. Notice that temporal betweenness assigns the nodes to only three different ranks because it only considers the temporal shortest paths. Therefore, it does not reveal the difference between, e.g., node \(d\) and nodes \(a, f, \) or \(g\), although \(d\) may play an essential role in a dissemination process while the other cannot. Similarly, the static betweenness assigns all nodes to even only two ranks. The reason is that the computation of static shortest paths ignores
5 COMPUTATION OF THE TEMPORAL WALK CENTRALITY

Computing the walk centrality of a node \( v \in V \) involves counting the weighted in- and outgoing walks at \( v \) over time. In Definition 4.3, \( W_{in} \) can be interpreted as a matrix that contains the weighted sum of the walks that arrive at node \( v \) at time \( t \). Analogously, we have the matrix \( W_{out} \) for the outgoing walks. In the following, we first describe several methods for computing these matrices and then how to calculate the walk centrality from them.

Table 2 gives an overview of the different algorithms, their running time, and properties. Notice that our algorithms perform on par or favorable compared to related state-of-the-art algorithms.

### Table 1: Node rankings for the temporal graph shown in Figure 1 obtained by different centrality measures.

| Centrality                              | Node Ranking |
|-----------------------------------------|--------------|
| Temporal Walk                           | 1: e 2: c 3: d 4: b 5: a/f/g |
| Temporal Betweenness [5]                | 1: e 2: c 3: a/b/d/f/g |
| Temporal PageRank [46]                  | 1: e 2: c 3: a/b/d/f/g |
| Temporal Katz [2]                       | 1: g 2: f 3: e 4: c 5: d 6: b 7: a |
| Temporal Closeness [37]                 | 1: a 2: c 3: b 4: e 5: d 6: f 7: g |
| In-Degree                               | 1: deg 2: b/d/f 3: a |
| Out-Degree                              | 1: acd 2: b/d/f/g 3: g |
| Static Betweenness [9]                  | 1: ce 2: a/b/d/g |
| Static Harmonic Closeness [27]          | 1: a 2: c 3: b 4: de 5: f 6: g |
| Static Random Walk Betweenness [32]     | 1: ce 2: d 3: b/f 4: a/g |

### Table 2: Overview of algorithms for computing the temporal walk centrality and their properties. Here, \( y < 2.373 \) is the exponent of matrix multiplication, \( k \) the number of fixed-point iterations, \( e = |E(DL(G))| \leq |E|^2 \) the number of edges in the directed line graph, and \( t_{max} \) the largest cardinality of availability or arrival times at a node.

| Method      | Sec. Running Time | Space     | Non-strict Exact |
|-------------|-------------------|-----------|------------------|
| DlgMA       | \( O(|E|^2) \)   | ✓         | ✓                |
| APPROX      | \( O(k(|E| + e)) \) | ✓         | ✓                |
| STREAM      | \( O(|E| \\cdot t_{max}) \) | ✓         | ✓                |

The algorithm proposed in [32] for the static random walk betweenness has a running time in \( O((|E|+|V|) \cdot |V|^2) \) and space complexity in \( O(|V|^2) \). For a static graph \( G = (V,E) \), furthermore, the algorithm proposed in [5] for computing temporal betweenness using temporal shortest paths has a running time in \( O(|V|^3 \cdot T^2) \) with \( T \) the total number of time steps in a temporal graph \( G = (V,E,\delta) \), and a space complexity of \( O(|V|^2 \cdot T + |E|) \).

For the analysis of our algorithms for computing the temporal walk centrality, we assume \( |E| \geq \frac{1}{2} |V|^2 \), which holds unless isolated nodes exist. Since an isolated node \( v \) is not involved in non-trivial walks and has \( C(v) = 0 \), we can safely delete all isolated nodes in a preprocessing step.

5.1 Directed Line Graph Expansion

Counting (weighted) walks in static networks can conveniently be realized in terms of basic linear algebra operations. Polynomial-time computable closed-form expressions are well-known supporting walks of unbounded length when long walks are sufficiently down-weighted to guarantee convergence [18, 33]. We lift the algebraic methods for walk counting to counting temporal walks by means of the directed line graph expansion. Variants of directed line graph expansions have been previously used for survivability and reliability analysis [21, 26], and for graph kernels [36]. These variants support only strict temporal walks. In contrast, we allow temporal walks that can traverse the same edge multiple times when the transition time is zero leading to a potentially infinite number of temporal walks. Moreover, our definition uses directed graphs and we do not add additional start and sink vertices as in [21, 26].
Temporal Walk Centrality: Ranking Nodes in Evolving Networks

WWW ’22, April 25–29, 2022, Virtual Event, Lyon, France

Figure 2: Induced subgraph of the Enron email network consisting of 38 nodes and 541 temporal edges. The nodes are colored according to their centrality value with darker color meaning higher centrality.

Figure 3: Directed line graph representation of the temporal graph \( G \) shown in Figure 1.

Definition 5.1 (Directed line graph expansion). Given a temporal graph \( G = (V, E, \delta) \), the directed line graph expansion \( DL(G) = (V', E') \) is the directed graph, where every temporal edge \((u, v, t)\) in \( E \) is represented by a node \( n_{uv}^t \) and there is an edge from \( n_{uv}^t \) to \( n_{xy}^t \) if \( v = x \) and \( t + \delta \leq s \).

Figure 3 illustrates the concept of the directed line graph expansion. Clearly the number of nodes of \( DL(G) \) is \(|E|\). The number of edges is maximal when at each node of the temporal graph every incoming edge can be combined with all outgoing edges, which is, for example, the case when the transition time \( \delta \) is zero and all edges have the same time stamp. Then the number of edges in the directed line graph expansion of \( G = (V, E, \delta) \) is \(|E(DL(G))| = \sum_{v \in V(G)} d^-(v) \cdot d^+(v) = O(|E|^2)\). This corresponds to the number of edges in the directed line graph of the underlying static graph with parallel edges [14].

The walks in the directed line graph are closely related to the temporal walks in the original temporal graph. We will use this relation for algebraic weighted walk counting and establish the correspondence formally.

Lemma 5.2. Let \( G \) be a temporal graph and \( \ell \geq 1 \). Moreover, let \( W_\ell(G) \) be the walks of length \( \ell \) in the graph \( G \) and \( W_\ell(DL(G)) \) the temporal walks in the temporal graph \( G \). There is a bijection \( \Gamma: W_{\ell-1}(DL(G)) \rightarrow W_\ell(G) \) given by

\[
\left(n_{i_0}^{t_0}, n_{i_1}^{t_1}, \ldots, n_{i_{\ell-1}}^{t_{\ell-1}}\right) \mapsto \left(u_{i_0}, e_1, u_{i_2}, \ldots, e_{\ell}, v_{i+1}\right)
\]

with \( e_i = (u_{i}, v_{i+1}, t_i) \) for \( i \in \{1, \ldots, \ell\} \).

We identify the temporal walks starting at a specific node and time in a temporal graph with walks starting at several different nodes in its directed line graph expansion.

Corollary 5.3. The temporal walks of length \( \ell \geq 1 \) in a temporal graph \( G \) starting (ending) at the node \( v \) at time \( t \) are in one-to-one correspondence with the walks of length \( \ell - 1 \) in \( DL(G) \) starting (ending) at the nodes \( X_{out}(v, t) \) \( (X_{in}(v, t)) \), where

\[
X_{out}(v, t) = \{ n_{uv}^\tau \in V(DL(G)) \mid \tau = u \land t = s \}
\]

\[
X_{in}(v, t) = \{ n_{uv}^\tau \in V(DL(G)) \mid v = w \land t = s + \delta \}.
\]

We endow the directed line graph expansion with edge weights such that the weight of a walk corresponds to the temporal walk weight of its temporal walk according to Definition 4.1. In a static graph with edge weights \( w: E \rightarrow \mathbb{R} \), the weight of a walk \( \omega = (e_1, e_2, \ldots, e_\ell) \) is \( w(\omega) = \prod_{i=1}^\ell w(e_i) \). We annotate an edge \( e = (n_{uv}, n_{uw}) \) in the directed line graph by \( w_\ell(e) = \Phi(t + \delta, s) \).

Lemma 5.4. Let \( \omega \) be a walk in the directed line graph expansion \( DL(G) \) of the temporal graph \( G \), then we have \( w_\ell(\omega) = \tau_\ell(\Gamma(\omega)) \).

The combination of these results allows to compute the temporal walk centrality. To this end, we count the in- and outgoing walks in the directed line graph expansion weighted by \( \Phi_{in} \) and \( \Phi_{out} \), respectively, and apply Corollary 5.3 to derive the corresponding values for the temporal graph. More precisely, let \( W_{out}(v, \ell) \) denote the sum of weighted walks of length \( \ell \) in the static graph starting at the node \( v \). Then, we obtain

\[
W_{out}(v, \ell) = \sum_{x \in X_{out}(v, \ell)} W_{out}(x, \ell).
\]

The value \( W_{in} \) can be obtained similarly by using the set \( X_{in}(v, \ell) \) and incoming weighted walk counts \( W_{in}(x) \). Counting weighted walks in (static) graphs can be realized by means of matrix methods. However, when the transition time \( \delta \) is non-zero, the directed line graph is acyclic, since an edge can only be traversed at most once by a walk. Efficient algorithms for this case are discussed in Section 5.2 and Appendix C.

5.1.1 Computation by matrix inversion. Let \( A \) be the weighted adjacency matrix of a graph \( G \) with weights \( w \), where \( a_{uv} = w(u, v) \) if \((u, v) \in E\) and 0 otherwise. It is well known that the entry \( a_{uv}^\ell \) of \( A^\ell \) is the sum of weighted walks of length \( \ell \) from node \( u \) to node \( v \). Hence, we have \( W_{out}(x, \ell) = [A^\ell]_{1x} \) and \( W_{out}(x, \ell) = [\Sigma_{\ell=0}^{\infty} A^\ell]_{1x} \). The sum of matrix powers is known as Neumann series and the identity \( \Sigma_{\ell=0}^{\infty} A^\ell = (I - A)^{-1} \) holds if the sum converges. This is guaranteed when \( \rho(A) < 1 \), where \( \rho \) denotes the spectral radius, i.e., the largest absolute value of an eigenvalue. In
We use two passes over the edge stream. In a forward pass, we all edges (operating on the temporal graph in construction of the directed line graph representation by directly can count the weighted walks in linear time by traversing it in In the case of the approximate method in this case. We verify this hypothesis O(|E|+|E|2) transposing the weighted adjacency matrix in a preprocessing step. The weighted temporal walks in a temporal graph Lemma 5.6. 

In practice, the directed line graph expansion is often sparse in chronological order. We prove that Theorem 5.7. Let \( G = (V, E, \delta) \) with \( \delta > 0 \). Algorithm 2 computes \( W_{in} \) correctly, and has a running time in \( O(|E| \cdot \tau_{max}) \) and a space complexity in \( O(|V| \cdot \tau_{max}) \), with \( \tau_{max} \) being the maximal size of the set of availability times of edges arriving at a node over all nodes. Counting the matrix \( W_{out} \) for the outgoing walks can be done symmetrically to Algorithm 2 with equal time and space complexity, where we replace \( \tau_{max} \) by \( \tau_{max}^{-} \), i.e., the maximal number of distinct availability times of edges leaving a node over all nodes. Hence, the total running time for both directions is in \( O(|E| \cdot \max(\tau_{max}, \tau_{max}^{+})) \) and the space complexity \( O(|V| \cdot \max(\tau_{max}^{-}, \tau_{max}^{+})) \).

5.3 Computing the Temporal Walk Centrality from the Matrices

After obtaining the matrices \( W_{in} \) and \( W_{out} \), we compute the walk centrality for all nodes. At each node \( v \in V \), for all pairs of arrival and starting times \( t_a \) and \( t_s \) with \( t_a \leq t_s \) the function \( \Phi_m(t_a, t_s) \) must be evaluated. Hence, for each \( v \in V \) we have to evaluate \( \Phi_m \) at most \( \tau^{-}(v) \cdot \tau^{+}(v) \) times. Under the assumption that we can evaluate \( \Phi_m \) in constant time, the total running time is in \( O(|V| \cdot \tau_{max}^{-} \cdot \tau_{max}^{+}) \). Alternatively, we can observe that the running time is at most quadratic in the number of temporal edges. In case that \( \Phi_m(t_a, t_s) = 1 \) for all \( t_a, t_s \in \mathbb{N} \), we can compute the centrality of all nodes in time linear in the number of temporal edges. Algorithm 3 iterates over all time points \( T(v) \) in increasing order and iteratively sums up the number of incoming walks (line 4f). We multiply the total incoming weight with the current outgoing weight and add the result to the centrality value of \( v \).

Theorem 5.8. For \( \Phi_m(t_1, t_2) = 1 \) and \( t_1, t_2 \in \mathbb{N} \), Algorithm 3 computes the walk centrality from \( W_{in} \) and \( W_{out} \) in \( O(|E|) \) time.

5.2 Streaming Algorithm

In the case of \( \delta > 0 \) the directed line graph is acyclic and we can count the weighted walks in linear time by traversing it in a bottom-up fashion, cf. Appendix C. However, we can avoid the construction of the directed line graph representation by directly operating on the temporal graph in edge stream representation, where the edges are given in chronological order (ties are broken arbitrarily). The edge stream representation has been successfully applied in earlier works for temporal paths computation \[31, 57\]. We use two passes over the edge stream. In a forward pass, we compute the weights of incoming walks at each node and in a backward pass of outgoing walks. Algorithm 2 processes the edges in chronological order to compute the matrix \( W_{in} \). Hence, when processing a temporal edge \((u, v, t)\) the incoming walk counts for the node \( u \) at all arrival times before \( t \) are correctly computed since all edges \((u, t')\) with \( t' + \delta \leq t \) have already been processed.
Table 3: Statistics of the data sets with $G = DL(G)$. The type is either undirected ($\alpha$) or directed ($\delta$).

| Data set | $|V(G)|$ | $|E(G)|$ | $|\Gamma(G)|$ | $\max \gamma$ | $\max \delta$ | $|V(G)|$ | $|E(G)|$ |
|----------|--------|--------|--------|---------|---------|--------|--------|
| Hospital | 75 | 33,324 | 9,833 | 2,922 | 2,922 | 64,684 | 62,554,794 |
| HTMConf | u | 113 | 1,162,508 | 1 | 1,162,508 | 40,807,360 | 40,807,360 |
| Highschool | u | 1,894 | 1,894 | 1,894 | 1,894 | 377,916 | 349,933,330 |
| College | d | 1,899 | 58,935 | 58,935 | 58,935 | 58,935 | 40,807,360 |
| Infectious | u | 10,972 | 415,912 | 94,912 | 94,912 | 831,824 | 56,569,697 |
| Facebook | d | 637,831 | 917,605 | 321,004 | 321,004 | 827,005 | 8,651,691 |
| Enron | d | 87,101 | 1,134,461 | 213,607 | 603,533 | 1,134,046 | 361,825,773 |
| AskUbuntu | d | 159,416 | 964,457 | 217,079 | 837 | 1,134,046 | 2,967,386 |
| Digg | d | 279,640 | 1,751,652 | 6,665 | 1,238 | 1,751,652 | 94,858,234 |
| Epinion | d | 575,760 | 13,669,320 | 1,601 | 1,601 | 13,669,281 | 94,635,962 |
| WikiTalk | d | 1,420,367 | 4,418,565 | 2,846,620 | 4,173,783 | 4,418,932 | 1,696,871,574 |
| Youtube | d | 5,225,585 | 9,575,374 | 201 | 101 | 9,575,374 | 4,419,951,091 |
| Delicious | d | 4,512,699 | 219,532,884 | 1,583 | 1,583 | 219,532,884 | 83,533,926,266 |

6 EXPERIMENTS

We discuss the following research questions:

Q1. Efficiency and Scalability: How do our algorithms for computing the temporal walk centrality differ in terms of running time in practice? Do they scale to large networks?

Q2. Accuracy of APPROX: How is the accuracy of APPROX compared to the exact results?

Q3. Effect of the Parameters: How do the choices of the parameters affect the temporal walk centrality?

Q4. Node Rankings: How do the rankings by temporal walk centrality compare to other temporal centrality measures?

Data Sets: We use fourteen real-world temporal graph data sets:

1. Hospital contains the contacts between hospital patients and medical personal [53].
2. HTMConf is a contact network of visitors of a conference [17].
3. Highschool is a contact network of students over seven days [28].
4. College is based on an online social network used by students [39, 41].
5. Infectious represents face-to-face contacts of visitors of an exhibition [17].
6. Facebook is a subset of the activity of a Facebook community [54].
7. Enron is an email network between employees of a company [23].
8. AskUbuntu is a network of interactions on the stack exchange website Ask Ubuntu [42].
9. Digg is a social network in which nodes represent persons and edges friendships. The time stamps indicate when friendships were formed [15].
10. Epinion is based on a network of the product rating website Epinions [43].
11. WikiTalk is a social network based on the user pages of the Wikipedia website, where nodes represent users and edges messages on the user page [49].
12. Wikipedia is based on Wikipedia pages and hyperlinks between them [29].
13. Youtube is a social network on a video platform [30].
14. Delicious is based on a network of a bookmark website [55]. Table 3 shows the properties and statistics of the data sets. The transition time $\Delta$ is one for all data sets.

6.1 Algorithms and Experimental Protocol

We implemented the following algorithms in C++ using the GNU CC Compiler 10.3.0 and the Eigen library for matrix operations.

- **Stream** is the implementation of Algorithm 2.
- **DLGMA** uses the directed line graph expansion (DGL) and matrix inversion (Section 5.1).
- **APPROX** is the DLG-based approximation (Algorithm 1). All experiments ran on a computer cluster. Each experiment had an exclusive node with an Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz and 192 GB of RAM. The time limit was set to two hours. The source code is available at https://gitlab.com/tgpublic/twc.

6.2 Results and Discussion

Q1. Efficiency and Scalability: First, we evaluated the running time of our algorithms. For Stream, we use both $\Phi_{\alpha}(t_1, t_2) = \alpha$ and $\Phi_{\delta}(t_1, t_2) = \frac{1}{1 + e^{-\alpha t_1 - \alpha t_2}}$ to compute two variants of the temporal walk centrality. For the other algorithms, we use $\Phi_{\alpha}$. In all cases, we set $\alpha = 0.001$. We set $\Phi_{\alpha}(t_1, t_2) = 1$ in case of $\Phi_{\alpha}$, and $\Phi_{\alpha}(t_1, t_2) = \Phi_{\alpha}(t_1, t_2)$ otherwise. Table 4 shows the results. For Stream, the running time is lowest for all data sets. It is at least five times faster than the approximation APPROX, and several orders of magnitude faster than DLGMA. In the case of $\Phi_{\alpha}$, compared to $\Phi_{\alpha}$, the running times increase. The increase is less than 10% for half of the data sets, and on average only 17.3%. For WikiTalk, the increase is the most with 78.9%. The reason is the large values of $\tau_{\max}$ and $r_{\max}$ (see Table 3). Youtube has small values for $\tau_{\max}$ and $r_{\max}$, and hence, the computations of Stream for both variants of $\Phi$ are much faster compared to WikiTalk, even though Youtube has more nodes and edges. DLGMA could only compute the result for the HTMConf data set in the given time limit of two hours. The reason is that the input matrices are large and the computed inverse matrices are not always sparse. However, the approximation algorithm APPROX computed the centrality values efficiently. The very large DLGs for WikiTalk, Wikipedia, Youtube, and Delicious (see Table 3) lead to out-of-memory errors during the computation of DLGMA. Even though the DLG is often much smaller than the theoretical maximal size, the sizes of the DLGs can be a bottleneck of the DLG-based algorithms. For example, for the Enron data set APPROX uses 59.7

Table 4: Running times in seconds (Out-of-time of use, Oom-out of memory).

| Data set | Stream with various $\Phi$ | DLGMA | APPROX for various $\epsilon$ |
|----------|---------------------------|-------|-----------------------------|
| $\Phi_{\alpha}$ | $\Phi_{\delta}$ | $\epsilon$ | 0.1 | 0.001 | 0.00001 |
| Hospital | 2.20E-08 | 1.38E-10 | 3.89E-12 |
| HTMConf | 5.06E-08 | 1.06E-09 | 1.96E-12 |
| Highschool | 1.64E-09 | 9.31E-12 | 4.63E-14 |
| College | 4.00E-08 | 7.86E-11 | 3.87E-12 |
| Infectious | 1.11E-09 | 2.71E-12 | 1.25E-13 |
| Enron | 3.29E-12 | 9.08E-15 | 9.08E-15 |
| AskUbuntu | 5.42E-10 | 5.71E-12 | 5.50E-14 |
| Digg | 3.29E-12 | 5.71E-12 | 1.10E-15 |
| Epinion | 2.89E-16 | 2.89E-16 | 1.22E-16 |

Table 5: Mean relative errors of APPROX for various $\epsilon$.

| Data set | Approve with various $\epsilon$ |
|----------|-------------------------------|
| $\epsilon$ | $0.1$ | 0.001 | 0.00001 |
| Hospital | 1.29E-09 | 1.33E-09 | 1.33E-09 |
| HTMConf | 1.22E-09 | 1.22E-09 | 1.22E-09 |
| Highschool | 1.21E-09 | 1.21E-09 | 1.21E-09 |
| College | 1.23E-09 | 1.23E-09 | 1.23E-09 |
| Infectious | 1.24E-09 | 1.24E-09 | 1.24E-09 |
| Enron | 1.25E-09 | 1.25E-09 | 1.25E-09 |
| AskUbuntu | 1.26E-09 | 1.26E-09 | 1.26E-09 |
| Digg | 1.27E-09 | 1.27E-09 | 1.27E-09 |
| Epinion | 1.28E-09 | 1.28E-09 | 1.28E-09 |
Figure 4: Kendall rank correlation between the rankings computed using different variants of the temporal walk centrality and other temporal centrality measures.

GB memory, were STREAM only needs 0.58 GB (see also Table 7 in the appendix). In the case that \( \delta > 0 \) and we only have to consider strict temporal walks, STREAM shows very scalability. Even for the large data sets Wikipedia and Delicious with around 40 and 220 million edges, respectively, the computations are time- and space-efficient. Table 7 in Appendix D.1 shows the memory usage.

**Q2. Accuracy of APPROX:** We evaluated the accuracy of our approximation algorithm APPROX for \( \alpha = 0.001 \). Table 5 shows the mean relative errors for \( \varepsilon \in \{0.1, 0.001, 0.00001\} \) compared to the exact results computed with STREAM for all data sets for which the DLG could be computed. Let \( W = \{v \in V \mid C(v) \neq 0\} \) and \( \hat{C}(v) \) be the approximated value for \( v \in V \), we report
\[
\frac{1}{|W|} \sum_{v \in W} \frac{|C(v) - \hat{C}(v)|}{C(v)}.
\]

For all \( \varepsilon \), the errors are insignificant and very low. The error decreases for smaller \( \varepsilon \) as expected. A smaller value of \( \alpha \) leads to fast convergence. In conclusion, these results show that APPROX is highly accurate while being efficient (see Q1).

**Q3. Effect of the Parameters:** We computed the node rankings using temporal walk centrality for \( \Phi_\delta, \alpha \in \{0.1, 0.01, 0.001, 0.00001\} \), and for \( \Phi_t \). Furthermore, we set the transition time \( \delta = 0 \) and computed the temporal walk centrality with \( \Phi_\alpha \) for \( \alpha = 0.001 \). We measured the pairwise Kendall rank correlations (Kendall’s \( \tau \) coefficient) between the rankings [20]. The Kendall rank correlation coefficient is commonly used for determining the relationship between centrality measures [11]. The correlation coefficient takes on values between one and minus one, where values close to one indicate similar rankings, close to zero no correlation, and close to minus one a strong negative correlation. Figure 4 shows correlation matrices for HTMLConf, Facebook, Enron, and AskUbuntu. We observed similar results for the other data sets. TWC denotes the different rankings computed with the variations of the temporal walk centrality. The correlation between the rankings using temporal walk centrality with \( \Phi_\alpha \) is often strong for different \( \alpha \). The influence of \( \alpha \) seems to be limited for larger data sets when we consider the complete rankings of all nodes, because the majority of nodes obtain similar positions. In case that we only consider a fraction of the nodes with the highest centralities, the impact of \( \alpha \) is stronger (see Figure 5 in Appendix D) because the ratio of differently ranked nodes increases. Using a zero transition time of \( \delta = 0 \) did not lead to different rankings compared to \( \delta = 1 \). The correlation between the rankings using \( \Phi_t \) and \( \Phi_\alpha \) is often weaker than any of the correlations between the other TWC rankings. Hence, using the waiting time-based weighting can lead to temporal walk centrality rankings different from using distance-based weighting.

**Q4. Node Rankings:** We used the temporal PageRank (\( \alpha = 0.99, \beta = 0.5 \)), temporal closeness, temporal Katz centrality (constant weighting with \( \beta = 0.01 \), and temporal betweenness to compute the node rankings for the HTMLConf, Facebook, Enron, and AskUbuntu data sets. Due to the high running times of the temporal betweenness, the results are only computed for the HTMLConf data set. We report the results in the correlation matrices shown in Figure 4. The correlations between the variants of the temporal walk centrality and the other temporal centrality measures are between -0.004 and 0.68, with the lowest values for temporal betweenness. Hence, the rankings of the other temporal centrality measures have only a weak association with the rankings obtained using the temporal walk centrality. This is expected, as the other centrality measures are not designed for ranking vertices according to their importance in information spreading.

**7 CONCLUSION AND FUTURE WORK**

We introduced the temporal walk centrality for temporal networks, which captures the intuition of important nodes capable of obtaining and distributing information efficiently. We illustrated how the temporal walk centrality can identify nodes that are crucial for the dissemination of information. We theoretically and experimentally showed that temporal walk centrality can be computed efficiently and with high accuracy in the case of our approximation. Moreover, our streaming algorithm scales to very large temporal networks. In future work, more general weight functions for temporal walks could be studied, which not only depend on two points in time but, e.g., also on the number and availability times of incident edges of a node in the walk. Suitable weight functions can allow a probabilistic interpretation of the temporal walk centrality.

**ACKNOWLEDGMENTS**

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy – EXC-2047/1 – 390685813. Nils Kriege has been supported by the Vienna Science and Technology Fund (WWTF) through project VRG19-009.
Temporal Walk Centrality: Ranking Nodes in Evolving Networks
WWW ’22, April 25–29, 2022, Virtual Event, Lyon, France

REFERENCES

[1] Josh Alman and Virginia Vassilevska Williams. 2021. A Refined Laser Method and Faster Matrix Multiplication. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA, Daniel Marx (Ed.). SIAM, 532–539.

[2] Férec Bérés, Robert Faloutsos, Anna Omal, and Andras A. Benczur. 2018. Temporal walk based centrality metric for graph streams. Applied Network Science 3, 1 (2018), 32:1–32:26.

[3] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. The Journal of Mathematical Sociology 25, 2 (2001), 163–177.

[4] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems 30, 1-7 (1998), 107–117.

[5] Sebastian Bull, Hendrik Molter, Rolf Niedermeier, and Maciej Bymar. 2020. Algorithmic Aspects of Temporal Betweenness. In KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, ACM, 2084–2092.

[6] Don Coppersmith and Shmuel Winograd. 1987. Matrix multiplication via arithmetic progressions. In Proceedings of the nineteenth annual ACM symposium on Theory of computing (STOC ’87). ACM, New York, NY, USA, 1–6.

[7] Pierlugi Crescenzi, Clémence Magnien, and Andrea Marino. 2020. Finding top-k nodes for temporal closeness in large temporal graphs. Algorithms 13, 9 (2020), 285.

[8] Kousik Das, Sovan Samanta, and Madhumangal Pal. 2018. Study on centrality measures in social networks: A survey. Social Network Analysis and Mining 8, 1 (2018), 13.

[9] Linton C. Freeman. 1977. A Set of Measures of Centrality Based on Betweenness. Sociometry 40 (1977), 35–41.

[10] Sergio Gómez. 2019. Centrality in networks: finding the most important nodes. In Business and Consumer Analytics: New Ideas. Springer, 401–433.

[11] Felipe Grando, Diego Noble, and Luis C. Lamb. 2016. An analysis of centrality measures for complex and social networks. In 2016 IEEE Global Communications Conference (GLOBECOM). IEEE, 1–6.

[12] Peter Grindrod, Mark C. Parsons, Desmond J. Higham, and Ernesto Estrada. 2011. Communicability across evolving networks. Physical Review E 83, 4 (2011), 046120.

[13] Shahzad Haddadan, Cristina Menghini, Matteo Ronidato, and Eli Upfal. 2021. RepPBulk: Reducing Polarized Bubble Radius with Link Insertions. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 139–147.

[14] Frank Harary and Robert Z. Norman. 1960. Some properties of line digraphs. Rendiconti del Circolo Matematico di Palermo 2, 9 (May 1960), 161–168.

[15] Todd Hogg and Kristina Lerman. 2012. Social dynamics of digg. EPJ Data Science 1, 1 (2012), 5.

[16] Peter Holme. 2015. Modern temporal network theory: a colloquium. The European Physical Journal B 88, 9 (2015), 234.

[17] Lorenzo Isella, Juliette Stuhlf, Alain Barrat, Ciro Cattuto, and Clémence Magnien. 2018. Stream graphs and link streams for the modeling of interactions over time. Business & Information Systems Engineering 217–226.

[18] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence in a social network. In Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms, SODA, Dániel Marx (Ed.). SIAM, 522–539.

[19] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage. 2008. Analyzing social bookmarking systems: A del.icio. us cookbook. In Proceedings of ECAI 2008 Mining Social Data Workshop.

[20] Philippa Vanhems, Alain Barrat, Ciro Cattuto, Jean-François Pinton, Nagham Elbarchi, Corinne Régis, Byrung K. Kim, Brigitte Comte, and Nicolas Voirin. 2019. Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. PloS one 8, 9 (2013), e75970.

[21] Bimal Viswanath, Alan Mislove, Meiyoung Cha, and Krishna P. Gummadi. 2009. On the evolution of user interaction in facebook. In Proceedings of the 2nd ACM workshop on Online social networks. 37–42.
A NOTATION

Table 6 shows our commonly used symbols and their definitions.

Table 6: Commonly used notations

| Symbol | Definition |
|--------|------------|
| \( G = (V, E, \delta) \) | Temporal graph \( G \) with nodes \( V \) and temp. edges \( \delta \) |
| \( \tau \) | Temporal \( (u, v, t) \)-edge at time \( t \) |
| \( T(v) \) | Set of availability times of edges incident to node \( v \) |
| \( T(G) \) | Set of availability and arrival times in \( G \) |
| \( G = (V, E, w) \) | Directed graph with edge weights \( w : E \rightarrow \mathbb{R} \) |
| \( \Phi : N \times N \rightarrow \mathbb{R} \) | Time depended weighting function |
| \( W_{in}(v, t), W_{out}(v, t) \) | Temporal walks ending, starting at node \( v \) and time \( t \) |
| \( \Omega \) | Total weight of temp. walks in \( W_{in}(v, t), W_{out}(v, t) \) |
| \( \tau^*(v) \) | \# of distinct arrival, starting times at \( v \in V \) |
| \( \tau_{max}, \tau_{max}^* \) | Max. \# of distinct arrival, starting times over \( v \in V \) |
| \( \epsilon \) | Error-tolerance parameter for approximation |

B OMITTED PROOFS

Proof of Lemma 5.2. Let \( \omega \) be a walk in \( DL(G) \). It directly follows from Definition 5.1, that \( \Gamma(\omega) \) is a walk in a graph that satisfies the time constraints. Vice versa, every temporal walk \( \omega \) in \( G \) corresponds to a sequence of edges, whose nodes are adjacent in \( DL(G) \). \( \square \)

Proof of Lemma 5.4. Let \( \omega = (n_1, n_2, n_3, \ldots, n_i \tau_t) \). Application of edge weights yields \( w_{\Phi}(\omega) = \prod_{i=1}^{\tau_t} \Phi(n_i + \delta, n_{i+1}) = \tau_{\Phi}(\Gamma(\omega)) \). \( \square \)

Proof of Theorem 5.7. Let \( \omega \) be a walk of length \( 1 \leq \ell \leq k \) arriving at vertex \( v \). For \( \ell = 1 \), the walk is a single edge \( e_1 = ((u, v, t)) \). Algorithm 2 iterates over all edges, and therefore, also over \( e_1 \). Therefore, \( W_{in}(u, t + \delta) \) will be initialized with one, and the walk is counted. For \( \ell > 1 \), we now have to check that we count only temporal walks. However, because the algorithm processes the edges in chronological order and due to line 6, the edge can only extend walks that arrive not later than \( u \) than \( t \). Consider a walk \( \omega = ((u, w, t), (w, v, t')) \) of length \( \ell = 2 \) arriving at \( v \). The walk \((u, w, t')\) arriving at vertex \( w \) was counted before because \( t' + \delta < t, \) hence \( W_{in}(w, t + \delta) > 0 \). Next, the walks from \( w \) are added to the walks at \( v \), weighted by \( \Phi_{in}(v, t') \). Consider the walk \( \omega = (e_1, \ldots, e_{\ell} = (w, v, t') \). By our assumption, it follows that the path \( \phi_{\ell - 1} = (e_1, \ldots, e_{\ell - 1} = (w, v, t_{\ell - 1})) \) of length \( \ell - 1 \) arrived at time \( t_{\ell - 1} + \delta \) at \( v \) and was counted correctly. The algorithm adds the walks of length \( \ell - 1 \) arriving at \( w \) to the number of walks of length \( \ell \) at vertex \( v \). \( \square \)

For the running time, Algorithm 2 iterates over all \( |E| \) edges in chronological order, and for each \( e \in E \), it iterates over all entries of \( W_{in} \) for which \( W_{in}(e, u, t) > 0 \), i.e., at most \( \tau_{max} \) rows. Therefore, the running time is in \( O(|E| \cdot \tau_{max}) \), using a minimal perfect hash function for indexing the arrival times. For each vertex \( v \in V \) and arrival time \( t_a \), we store the sum of the weighted walks arriving at \( v \) at time \( t_a \).

Proof of Theorem 5.8. Algorithm 3 iterates over all nodes and over all time stamps \( t \in T(v) \). The sum of the time stamps over all nodes as well as the number of nodes itself are bounded by \( |E| \). \( \square \)

C COMPUTATION IN ACYCLIC DLG

For the special case of acyclic graphs we can sum the weights of incoming and outgoing walks for all nodes in linear total time. The weight of walks starting at a node is the sum of the weighted walks starting at its outgoing neighbors. Algorithm 4 implements weighted walk counting in a bottom-up fashion starting at the sinks and propagating weighted walk counts level-wise upwards.

**Lemma C.1.** The weighted temporal walks in a temporal graph \( G = (V, E, \delta) \) with \( \delta > 0 \) are counted exactly by Algorithm 4 in time and space \( \mathcal{O}(|E| + e) \), where \( e = |E| \cdot DL(G) \). \( \square \)

We can count the incoming weighted walks for every node analogously, either by reversing all edges, or by considering the outgoing neighbors starting from the sources.

Algorithm 4: Counting weighted outgoing walks in directed acyclic graphs.

**Input:** Directed weighted acyclic graph \( G = (V, E, w) \).
**Output:** Weighted walk counts \( W_{out}(v) \) for all \( v \in V \).

1. **for all** \( v \in V \) **do** \( W_{out}(v) \leftarrow 1 \) \( \triangleright \) Length 0 walks
2. \( S^0 \leftarrow \{ v \in V \mid d^-(v) = 0 \} \) \( \triangleright \) Find sinks
3. \( i \leftarrow 0 \)
4. **repeat**
   5. **for all** \( v \in S^i \) **do**
   6. **for all** \( u \in N^+(v) \) **do**
   7. \( W_{out}(u) \leftarrow W_{out}(u) + w(u, v) \cdot W_{out}(v) \)
   8. \( S^{i+1} \leftarrow S^{i+1} \cup \{ u \} \) \( \triangleright \) Avoid duplicates
   9. \( i \leftarrow i + 1 \)
5. **until** \( S^i \neq \emptyset \)

D ADDITIONAL RESULTS

This section provides additional experimental results.

D.1 MEMORY USAGE

Table 7 shows the total memory usage of Stream and Approx. For Stream, we report the results for \( \Phi_{in} \) with \( \alpha = 0.001 \), and for Approx with \( \epsilon = 0.001 \). Note the much higher memory usage of Approx due to the computation of the DLG.

D.2 Top-k Node Rankings

Figure 5 shows the correlation matrices that are based on the top-k rankings computed with the variants of Twc, temporal PageRank, temporal Katz, temporal betweenness, and temporal closeness centralities. The temporal betweenness is due to the long running times
Figure 5: Kendall rank correlation between the rankings computed using different variants of the temporal walk centrality and other temporal centrality measures using only the top-$k$, with $k = \lceil |V| \cdot p \rceil$ nodes with the highest centrality values with $p = 0.1$ for HTMLConf and $p = 0.001$ for the other data sets.

**Table 7:** Memory usage in GB (Out–of–memory).

| Data set       | STREAM | APPROX |
|----------------|--------|--------|
|                | $\alpha = 0.001$ | $\epsilon = 0.001$ |
| Hospital       | 0.034  | 10.152 |
| HTMLConf       | 0.023  | 2.295  |
| Highschool     | 0.183  | 56.035 |
| College        | 0.033  | 0.737  |
| Infectious     | 0.422  | 10.156 |
| Facebook       | 0.409  | 1.866  |
| Enron-Rm       | 0.582  | 59.676 |
| AskUbuntu      | 0.168  | 0.169  |
| Diag           | 0.883  | 17.129 |
| Epinion        | 6.718  | 23.331 |
| WikiTalk       | 2.221  | Oom    |
| Wikipedia      | 20.330 | Oom    |
| Youtube        | 5.126  | Oom    |
| Delicious      | 106.999 | Oom    |

only computed for the HTMLConf data set. The value $k$ is chosen to be 10% of the number of the vertices of the small HTMLConf data set, and 0.1% for the other, larger data set. We chose a higher percentage for HTMLConf due to the small number of vertices of the data set. The impact of $\alpha$ in case of the top-$k$ rankings is higher, because compared to the complete rankings the ratio of nodes that change their position in the ranking is higher.