Temporal Walk Centrality: Ranking Nodes in Evolving Networks

Lutz Oettershagen
lutz.oettershagen@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

ACM Reference Format:
Lutz Oettershagen, Petra Mutzel, and Nils M. Kriege. 2022. Temporal Walk Centrality: Ranking Nodes in Evolving Networks. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512210

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.

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

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512210

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 τ_{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 and challenging tasks, especially if the considered networks are non-static and of temporal nature. Our work provides a new centrality measure that can rank nodes according to their importance in information spreading. Possible applications are the surveillance of spreading fake news or infectious diseases in temporal networks.

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. Rozenshtein 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., Nicosia 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]. Recently, 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 $E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$ of undirected edges. In a *directed* (static) graph $E \subseteq \{(u, v) \in V \times V \mid u \neq v\}$. 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 (static) *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* $\mathcal{G} = (V, \mathcal{E}, \delta)$ consists of a finite set of nodes V, a finite set $\mathcal{E}$ 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 $T(\mathcal{G}) = \{t, t + \delta \mid (u, v, t) \in \mathcal{E}\}$. Furthermore, let τ^-_{max} (τ^+_{max}) denote the maximum number 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* ω in a temporal graph $\mathcal{G}$ is an alternating sequence $(v_1, e_1, \ldots, e_\ell, v_{\ell+1})$ of nodes and temporal edges connecting consecutive nodes, where $e_i = (v_i, v_{i+1}, t_i) \in \mathcal{E}$ and $t_i + \delta \leq t_{i+1}$ for all $i \in \{1, \ldots, \ell\}$. We may omit nodes for notational convenience. Depending on δ, 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 ω by $|\omega| = \ell$. For $\delta > 0$, the length of a (strict) temporal walk is bounded by $T(\mathcal{G})$, 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 $(u, v, t) \in \mathcal{E}$ 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 δ_e for each edge e instead of the global parameter δ. 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 $\mathcal{G} = (V, \mathcal{E}, \delta)$ be a temporal graph, and $\omega = (e_1, \ldots, e_\ell)$ a temporal walk in $\mathcal{G}$. We define the weight of a temporal walk ω as

$$\tau_\Phi(\omega) = \prod_{i=1}^{\ell-1} \Phi(t_i + \delta, t_{i+1}), \tag{1}$$

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

We discuss concrete examples of the function Φ 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 $\mathcal{G} = (V, \mathcal{E}, \delta)$ be a temporal graph, and let $\mathcal{W}_{in}(v, t)$ and $\mathcal{W}_{out}(v, t)$ be the sets of incoming and outgoing temporal walks, resp., at node v and time t. We define

$$\boldsymbol{W}_{in}(v,t) = \sum_{\omega \in \mathcal{W}_{in}(v,t)} \tau_{\Phi_{in}}(\omega) \quad \text{and} \quad \boldsymbol{W}_{out}(v,t) = \sum_{\omega \in \mathcal{W}_{out}(v,t)} \tau_{\Phi_{out}}(\omega).$$

The temporal walk weight functions $\tau_{\Phi_{in}}$ and $\tau_{\Phi_{out}}$ allow to weight incoming and outgoing walks independently. Using $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$, we now define the temporal walk centrality.

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

$$C(v) = \sum_{t_1, t_2 \in T(\mathcal{G}), t_1 \leq t_2} (\boldsymbol{W}_{in}(v, t_1) \cdot \boldsymbol{W}_{out}(v, t_2) \cdot \Phi_m(t_1, t_2))$$

the *temporal walk centrality* of node $v \in V$.

The time-depended weight function Φ_m is, similarly to Φ_{in} and Φ_{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., $\tau_\Phi(\omega) = \alpha^{|\omega|-1}$. 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 α. In the following, we denote this variant with Φ_α.
(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 $\Phi(t_1, t_2) = \frac{1}{1+t_2-t_1}$. 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 Φ_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 $\Phi(t_1, t_2) = \frac{\alpha}{1+t_2-t_1}$ for all $t_1, t_2 \in \mathbb{N}$. Another example would be to use different values for α 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 Φ.

To see how the temporal walk centrality generalizes the temporal Katz and degree centrality, consider the following. If we fix $\boldsymbol{W}_{in}(v, t) = 1$, and use the weighting function $\Phi(a, b) = \alpha$ 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 $\mathcal{G} = (V, \mathcal{E}, \delta)$ 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 ω in $\mathcal{G}$, and we set $\Phi_m(t, t') = 1$ 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

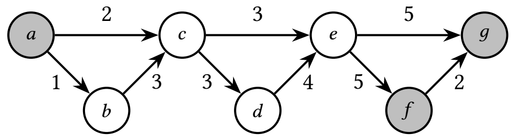


Figure 1: Temporal graph $\mathcal{G}$ with the availability times shown at the edges and $\delta = 1$. Nodes that cannot pass on information are marked in gray.

the temporal restrictions. We can see similar problems for the other centrality measures, e.g., temporal Katz and temporal closeness rank nodes g and a, respectively, highest. Similarly, the degree, temporal PageRank, static closeness, and static random walk betweenness fail to rank the nodes according to our idea of important nodes that can distribute information efficiently. In conclusion, only temporal walk centrality can distinguish the nodes that are important in dissemination processes. It ranks the nodes according to the intuition that nodes that can be reached easily and can reach other nodes are important.

Enron email network: We consider an induced subgraph of the *Enron* email network [23]. The network represents the email communication in a company, where nodes are employees, and temporal edges represent emails. The subgraph is an ego-network with radius one of the employee represented by node zero. Figure 2 shows the network with different centrality measures, where a darker node color means higher centrality. Figure 2a illustrates the temporal walk centrality with Φ_α and $\alpha = 1$. We compare it with temporal betweenness centrality (Figure 2b) and the static random walk betweenness (Figure 2c). The former assigns high centrality values only to nodes that are part of many shortest temporal paths. Therefore, only a few nodes get a relatively high centrality value, namely the nodes 3, 4, and 34. However, nodes that have a high capability to pass information, e.g., node 21, are assigned a low centrality value. On the other hand, in Figure 2c we see that the static random walk centrality, which does not respect the temporal properties of the network, assigns high values to many nodes. The reason is that considering static walks instead of temporal walks leads to over-counting node visits by ignoring the restricted temporal reachability. Hence, many nodes in Figure 2c obtained a similar centrality value resulting in less information about the importance of the nodes.

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, $\boldsymbol{W}_{in}$ 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 $\boldsymbol{W}_{out}$ 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:** afg
Temporal Betweenness [5]	**1:** c **2:** e **3:** $abdfg$
Temporal PageRank [46]	**1:** ce **2:** a **3:** fg **4:** bd
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:** ceg **2:** bdf **3:** a
Out-Degree	**1:** acd **2:** bdf **3:** g
Static Betweenness [9]	**1:** ce **2:** $abdfg$
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:** bf **4:** ag

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

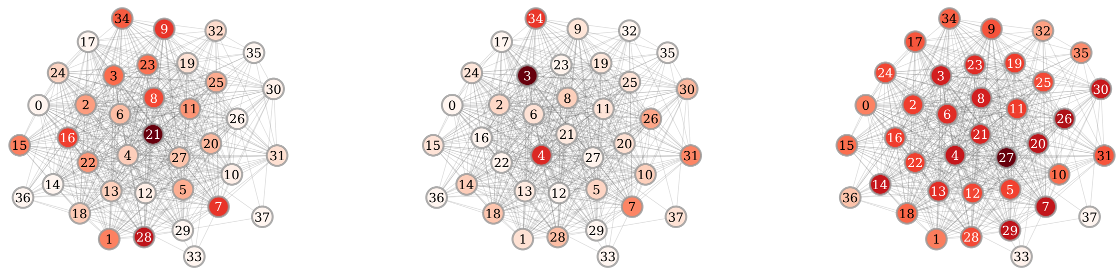
Method	Sec.	Running Time	Space	Non-strict	Exact
DlgMa	5.1.1	$O(\lvert\mathcal{E}\rvert^\gamma)$	$O(\lvert\mathcal{E}\rvert^2)$	✓	✓
Approx	5.1.2	$O(k(\lvert\mathcal{E}\rvert + e))$	$O(\lvert\mathcal{E}\rvert + e)$	✓	✗
Stream	5.2	$O(\lvert\mathcal{E}\rvert \cdot \tau_{max})$	$O(\lvert V\rvert \cdot \tau_{max})$	✗	✓

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 $\mathcal{G} = (V, \mathcal{E}, \delta)$, and a space complexity of $O(|V| \cdot T + |\mathcal{E}|)$.

For the analysis of our algorithms for computing the temporal walk centrality, we assume $|\mathcal{E}| \geq 1/2|V|$, 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].



(a) Temporal walk centrality. **(b) Temporal betweenness.** **(c) Static walk betweenness.**

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.

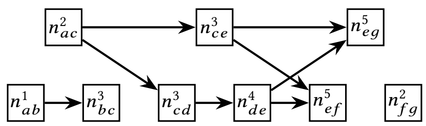


Figure 3: Directed line graph representation of the temporal graph $\mathcal{G}$ shown in Figure 1.

Definition 5.1 (Directed line graph expansion). Given a temporal graph $\mathcal{G} = (V, \mathcal{E}, \delta)$, the *directed line graph expansion* $DL(\mathcal{G}) = (V', E')$ is the directed graph, where every temporal edge (u, v, t) in $\mathcal{E}$ is represented by a node n^t_{uv}, and there is an edge from n^t_{uv} to n^s_{xy} 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 $DL(\mathcal{G})$ is $|\mathcal{E}|$. 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 δ is zero and all edges have the same time stamp. Then the number of edges in the directed line graph expansion of $\mathcal{G} = (V, \mathcal{E}, \delta)$ is $|E(DL(\mathcal{G}))| = \sum_{v \in V(\mathcal{G})} d^-(v) \cdot d^+(v) = O(|\mathcal{E}|^2)$. 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 $\mathcal{G}$ be a temporal graph and $\ell \geq 1$. Moreover, let $\mathcal{W}_\ell(G)$ be the walks of length ℓ in the graph G and $\boldsymbol{\mathcal{W}}_\ell(\mathcal{G})$ the temporal walks in the temporal graph $\mathcal{G}$. There is a bijection $\Gamma\colon \mathcal{W}_{\ell-1}(DL(\mathcal{G})) \to \boldsymbol{\mathcal{W}}_\ell(\mathcal{G})$ given by*

$$\left(n^{t_1}_{v_1 v_2}, n^{t_2}_{v_2 v_3}, \ldots, n^{t_\ell}_{v_\ell v_{\ell+1}}\right) \mapsto (v_1, e_1, v_2, \ldots, e_\ell, v_{\ell+1})$$

with $e_i = (v_i, v_{i+1}, t_i)$ 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 $\mathcal{G}$ starting (ending) at the node v at time t are in one-to-one correspondence with the walks of length $\ell - 1$ in $DL(\mathcal{G})$ starting (ending) at the nodes $X_{out}(v, t)$ ($X_{in}(v, t)$), where*

$$X_{out}(v, t) = \{n^s_{uw} \in V(DL(\mathcal{G})) \mid v = u \wedge t = s\}$$
$$X_{in}(v, t) = \{n^s_{uw} \in V(DL(\mathcal{G})) \mid v = w \wedge t = s + \delta\}.$$

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 $w\colon E \to \mathbb{R}$, 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 $e = (n^t_{uv}, n^s_{vw})$ in the directed line graph by $w_\Phi(e) = \Phi(t + \delta, s)$.

Lemma 5.4. *Let ω be a walk in the directed line graph expansion $DL(\mathcal{G})$ of the temporal graph $\mathcal{G}$, then we have $w_\Phi(\omega) = \tau_\Phi(\Gamma(\omega))$.*

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 Φ_{in} and Φ_{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 ℓ in the static graph starting at the node v. Then, we obtain

$$\boldsymbol{W}_{out}(v, t) = \sum_{x \in X_{out}(v,t)} W_{out}(x), \quad W_{out}(x) := \sum_{\ell=0}^{\infty} W_{out}(x, \ell).$$

The value $\boldsymbol{W}_{in}$ can be obtained similarly by using the set $X_{in}(v, t)$ 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 δ 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 Section C.

5.1.1 Computation by matrix inversion. Let $\boldsymbol{A}$ 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 $a^{(\ell)}_{uv}$ of $\boldsymbol{A}^\ell$ is the sum of weighted walks of length ℓ from node u to node v. Hence, we have $W_{out}(x, \ell) = \left[\boldsymbol{A}^\ell \mathbf{1}\right]_x$ and $W_{out}(x) = \left[\left(\sum_{\ell=0}^{\infty} \boldsymbol{A}^\ell\right) \mathbf{1}\right]_x$. The sum of matrix powers is know as Neumann series and the identity $\sum_{\ell=0}^{\infty} \boldsymbol{A}^\ell = (\boldsymbol{I} - \boldsymbol{A})^{-1}$ holds if the sum converges. This is guaranteed when $\rho(\boldsymbol{A}) < 1$, where ρ denotes the *spectral radius*, i.e., the largest absolute value of an eigenvalue. In

Algorithm 1: Approximate counting weighted outgoing walks in directed graphs.

Input: Directed weighted graph G, error-tolerance ε.
Output: Weighted walk counts $W_{out}(v) = [\boldsymbol{r}]_v$.

Initialize $\boldsymbol{A}$ from G and verify $\rho(\boldsymbol{A}) < 1$
$\boldsymbol{v} \leftarrow \mathbf{1}$; $\boldsymbol{r} \leftarrow \boldsymbol{v}$ ▷ *Length* 0 *walks*
repeat
  $\boldsymbol{v} \leftarrow \boldsymbol{A}\boldsymbol{v}$
  $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{v}$
until $\|\boldsymbol{v}\|_1 < \varepsilon$

this case all weighted walks without length bound can be counted by inversion of an $n \times n$ matrix, where $n = |\mathcal{E}|$. Exact matrix inversion is in time $O(n^\gamma)$, where $\gamma < 2.373$ is theoretically possible [1] by improving the seminal work of Coppersmith and Winograd [6].

Lemma 5.5. *The weighted temporal walks in a temporal graph $\mathcal{G} = (V, \mathcal{E}, \delta)$ can be counted exactly in $O(|\mathcal{E}|^\gamma)$ time with space $O(|\mathcal{E}|^2)$, where $\gamma < 2.373$ is the exponent of matrix multiplication.*

In practice, roughly cubic running time is expected. For large graphs this is prohibitive even when the directed line graph is sparse as the inverse of a sparse matrix not necessarily is sparse.

5.1.2 Approximation by fixed-point iteration. We propose an approximation algorithm based on iterated matrix-vector multiplication which benefits from algorithms and datastructures for sparse matrices. Algorithm 1 approximates $W_{out}(x)$ and can analogously be applied to compute $W_{in}(x)$ by reversing all edges, or simply transposing the weighted adjacency matrix in a preprocessing step.

Lemma 5.6. *The weighted temporal walk counts of a temporal graph $\mathcal{G} = (V, \mathcal{E}, \delta)$ can be approximated in time $O(k(|\mathcal{E}| + e))$ and space $O(|\mathcal{E}| + e)$, where $e = |E(DL(\mathcal{G}))|$ and k is the number of iterations required to meet the error tolerance.*

In practice, the directed line graph expansion is often sparse with $|E(DL(\mathbf{G}))| \ll |\mathcal{E}|^2$ and we expect significant advantages of the approximate method in this case. We verify this hypothesis experimentally in Section 6.

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. Section 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 $\boldsymbol{W}_{in}$. 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 $(\cdot, u, t')$ with $t' + \delta \leq t$ have already been processed.

Algorithm 2: Streaming algorithms for incoming walks.

Input: Temporal graph $\mathcal{G} = (V, \mathcal{E}, \delta)$, function Φ_{in}
Output: Matrix $\boldsymbol{W}_{in}$

forall $(u, v, t) \in \mathcal{E}$ **do** ▷ *in chronological order*
  if $\boldsymbol{W}_{in}(v, t + \delta)$ *not initialised* **then**
    $\boldsymbol{W}_{in}(v, t + \delta) = 0$
  $\boldsymbol{W}_{in}(v, t + \delta)$ += 1
  forall t' *with* $\boldsymbol{W}_{in}(u, t') > 0$ **do**
    if $t \geq t'$ **then**
      $\boldsymbol{W}_{in}(v, t + \delta)$ += $\boldsymbol{W}_{in}(u, t') \cdot \Phi_{in}(t', t)$

Theorem 5.7. *Let $\mathcal{G} = (V, \mathcal{E}, \delta)$ with $\delta > 0$. Algorithm 2 computes $\boldsymbol{W}_{in}$ correctly, and has a running time in $O(|\mathcal{E}| \cdot \tau^-_{max})$ and a space complexity in $O(|V| \cdot \tau^-_{max})$, with τ^-_{max} being the maximal size of the set of availability times of edges arriving at a node over all nodes.*

Counting the matrix $\boldsymbol{W}_{out}$ for the outgoing walks can be done symmetrically to Algorithm 2 with equal time and space complexity, where we replace τ^-_{max} by τ^+_{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 $O(|\mathcal{E}| \cdot \max\{\tau^-_{max}, \tau^+_{max}\})$ 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 $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$, 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 Φ_m at most $\tau^-(v) \cdot \tau^+(v)$ times. Under the assumption that we can evaluate Φ_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 $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$ in $O(|\mathcal{E}|)$ time.*

Algorithm 3: Computing temporal walk centrality for $\Phi_m(t_a, t_s) = 1$.

Input: $\boldsymbol{W}_{in}$ and $\boldsymbol{W}_{out}$
Output: Walk centrality $C(v)$ for all $v \in V$

1 **forall** $v \in V$ **do**
2   $C(v) \leftarrow 0$, $in_{sum} \leftarrow 0$
3   **forall** $t \in T(v)$ *in increasing order* **do**
4   in_{sum} += $\boldsymbol{W}_{in}(v, t)$
5     $C(v)$ += $\boldsymbol{W}_{out}(v, t) \cdot in_{sum}$

Table 3: Statistics of the data sets with $G = DL(\mathcal{G})$. The type is either undirected (u) or directed (d).

Data set	Properties							
	Type	$\lvert V(\mathcal{G})\rvert$	$\lvert E(\mathcal{G})\rvert$	$\lvert T(\mathcal{G})\rvert$	τ^-_{max}	τ^+_{max}	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$
Hospital	u	75	32 424	9 453	2 902	2 902	64 848	62 314 564
HTMLConf	u	113	20 818	5 246	1 390	1 390	41 636	13 535 789
Highschool	u	1 894	188 508	7 375	3 500	3 500	377 016	342 993 330
College	d	1 899	59 835	58 911	1 539	1 539	59 835	4 039 885
Infectious	u	10 972	415 912	76 943	393	393	831 824	56 569 497
Facebook	d	63 731	817 035	333 924	455	455	817 035	8 051 691
Enron	d	87 101	1 134 046	213 167	6 033	5 353	1 134 046	361 625 773
AskUbuntu	d	159 316	964 437	257 079	837	2 358	257 305	2 967 386
Digg	d	279 630	1 731 652	6 865	1 328	330	1 731 652	94 858 234
Epinion	d	755 760	13 668 320	501	181	498	13 668 281	94 633 962
WikiTalk	d	1 420 367	4 641 928	3 442 682	16 213	1 092 785	4 641 928	1 696 871 574
Wikipedia	d	1 870 709	39 953 145	2 198	1 931	489	39 953 145	2 763 845 227
Youtube	d	3 223 585	9 375 374	203	191	203	9 375 374	4 410 951 091
Delicious	d	4 512 099	219 532 884	1 583	1 583	1 317	219 532 884	83 533 929 266

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) *HTMLConf* 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 δ 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 (DLG) 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.

Table 4: Running times in seconds (Oot–out of time, Oom–out of memory).

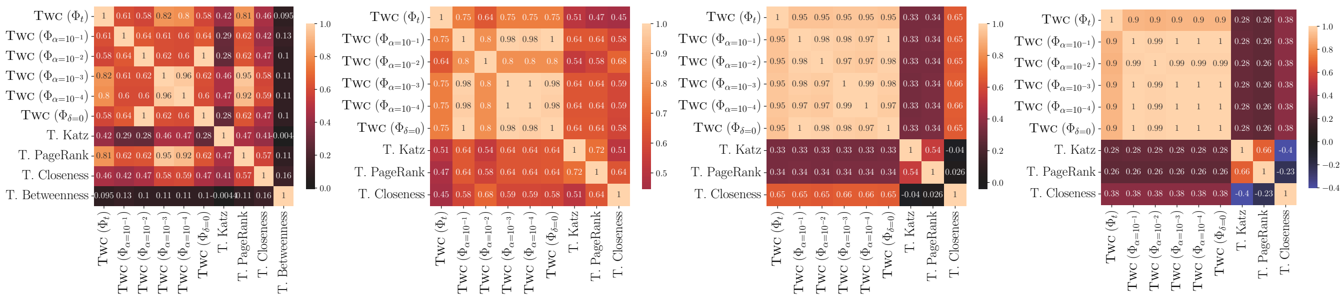
Data set	Stream with various Φ		DlgMa	Approx for various ε		
	Φ_α	Φ_t		0.1	0.001	0.00001
Hospital	2.50	3.04	Oot	18.32	18.69	19.00
HTMLConf	0.56	0.58	5 213.17	3.35	3.46	3.53
Highschool	20.98	23.24	Oot	117.95	123.67	128.44
College	0.17	0.21	Oot	1.05	1.06	1.07
Infectious	1.29	1.33	Oot	12.75	13.06	13.22
Facebook	0.57	0.64	Oot	4.23	4.30	4.30
Enron	11.31	15.09	Oot	124.13	129.33	131.94
AskUbuntu	0.21	0.28	Oot	1.06	1.07	1.08
Digg	1.70	1.81	Oot	41.95	42.48	42.90
Epinion	2.62	2.71	Oot	41.31	41.38	41.83
WikiTalk	126.42	226.25	Oom	Oom	Oom	Oom
Wikipedia	190.11	200.81	Oom	Oom	Oom	Oom
Youtube	9.63	9.91	Oom	Oom	Oom	Oom
Delicious	1 851.68	1 928.51	Oom	Oom	Oom	Oom

Table 5: Mean relative errors of Approx for various ε.

Data set	Approx with various ε		
	0.1	0.001	0.00001
Hospital	2.20E-08	1.38E-10	3.89E-12
HTMLConf	6.04E-08	1.08E-09	1.69E-12
Highschool	1.64E-09	9.13E-12	4.63E-14
College	4.00E-08	8.78E-11	3.87E-12
Infectious	1.11E-09	2.71E-12	1.26E-13
Facebook	3.29E-12	9.08E-15	9.08E-15
Enron	3.44E-09	7.13E-12	9.35E-14
AskUbuntu	5.42E-10	5.71E-12	5.50E-14
Digg	5.20E-12	7.30E-14	1.10E-15
Epinion	2.89E-16	2.89E-16	1.22E-16

6.2 Results and Discussion

Q1. Efficiency and Scalability: First, we evaluated the running times of our algorithms. For Stream, we use both $\Phi_\alpha(t_1, t_2) = \alpha$ and $\Phi_t(t_1, t_2) = \frac{1}{1+t_2-t_1}$ to compute two variants of the temporal walk centrality. For the other algorithms, we use Φ_α. In all cases, we set $\alpha = 0.001$. We set $\Phi_m(t_1, t_2) = 1$ in case of Φ_α, and $\Phi_m(t_1, t_2) = \Phi_t(t_1, t_2)$ 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 Φ_t, compared to Φ_α, 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 are the large values of the maximal number of starting or arrival times τ^-_{max} and τ^+_{max} (see Table 3). *Youtube* has small values for τ^-_{max} and τ^+_{max}, and hence, the computations of Stream for both variants of Φ are much faster compared to *WikiTalk*, even though *Youtube* has more nodes and edges. DlgMa could only compute the result for the *HTMLConf* 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 bottle-neck of the DLG-based algorithms. For example, for the *Enron* data set Approx uses 59.7



(a) ***HTMLConf*** (b) ***Facebook*** (c) ***Enron*** (d) ***AskUbuntu***

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 good 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 ε, the errors are insignificant and very low. The error decreases for smaller ε as expected. A smaller value of α 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 Φ_α, $\alpha \in \{0.1, 0.01, 0.001, 0.0001\}$, and for Φ_t. Furthermore, we set the transition time $\delta = 0$ and computed the temporal walk centrality with Φ_α for $\alpha = 0.001$. We measured the pairwise Kendall rank correlations (Kendall's τ 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 Φ_α is often strong for different α. The influence of α 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 α 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 Φ_t and Φ_α 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 betweeness 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 betweeness. 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.

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*, Dániel Marx (Ed.). SIAM, 522–539.

[2] Ferenc Béres, Róbert Pálovics, Anna Oláh, and András A. Benczúr. 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 Buß, Hendrik Molter, Rolf Niedermeier, and Maciej Rymar. 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] Pierluigi Crescenzi, Clémence Magnien, and Andrea Marino. 2020. Finding top-k nodes for temporal closeness in large temporal graphs. *Algorithms* 13, 9 (2020), 211.

[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] Shahrzad Haddadan, Cristina Menghini, Matteo Riondato, and Eli Upfal. 2021. RePBubLik: 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* 9, 2 (May 1960), 161–168.

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

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

[17] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. 2011. What's in a crowd? Analysis of face-to-face behavioral networks. *Journal of Theoretical Biology* 271, 1 (2011), 166–180.

[18] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[19] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Lise Getoor, Ted E. Senator, Pedro M. Domingos, and Christos Faloutsos (Eds.). ACM, 137–146.

[20] Maurice G. Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[21] Gaurav Khanna, Sanjay K. Chaturvedi, and Sieteng Soh. 2020. Two-terminal reliability analysis for time-evolving and predictable delay-tolerant networks. *Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering)* 13, 2 (2020), 236–250.

[22] Hyoungshick Kim and Ross Anderson. 2012. Temporal node centrality in complex networks. *Physical Review E* 85, 2 (2012), 026107.

[23] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*. Springer, 217–226.

[24] Andrea Landherr, Bettina Friedl, and Julia Heidemann. 2010. A critical review of centrality measures in social networks. *Business & Information Systems Engineering* 2, 6 (2010), 371–385.

[25] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. 2018. Stream graphs and link streams for the modeling of interactions over time. *Soc. Netw. Anal. Min.* 8, 1 (2018), 61:1–61:29.

[26] Qingkai Liang and Eytan Modiano. 2016. Survivability in time-varying networks. *IEEE Transactions on Mobile Computing* 16, 9 (2016), 2668–2681.

[27] Massimo Marchiori and Vito Latora. 2000. Harmony in the small-world. *Physica A: Statistical Mechanics and its Applications* 285, 3-4 (2000), 539–546.

[28] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. 2015. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PloS one* 10, 9 (2015), e0136497.

[29] Alan E. Mislove. 2009. *Online social networks: measurement, analysis, and applications to distributed information systems.* Ph. D. Dissertation. Rice University.

[30] Alan E. Mislove, Massimiliano Marcon, P. Krishna Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC*. ACM, 29–42.

[31] Petra Mutzel and Lutz Oettershagen. 2019. On the enumeration of bicriteria temporal paths. In *International Conference on Theory and Applications of Models of Computation*. Springer, 518–535.

[32] Mark E. J. Newman. 2005. A measure of betweenness centrality based on random walks. *Soc. Networks* 27, 1 (2005), 39–54.

[33] Mark E. J. Newman. 2010. *Networks: An Introduction.* Oxford University Press.

[34] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*. 969–976.

[35] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. 2013. Graph metrics for temporal networks. In *Temporal Networks*. Springer, 15–40.

[36] Lutz Oettershagen, Nils M. Kriege, Christopher Morris, and Petra Mutzel. 2020. Temporal Graph Kernels for Classifying Dissemination Processes. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 496–504.

[37] Lutz Oettershagen and Petra Mutzel. 2020. Efficient Top-k Temporal Closeness Calculation in Temporal Networks. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 402–411.

[38] Lutz Oettershagen and Petra Mutzel. 2022. Computing top-k temporal closeness in temporal networks. *Knowledge and Information Systems* (2022), 1–29.

[39] Tore Opsahl and Pietro Panzarasa. 2009. Clustering in weighted networks. *Social networks* 31, 2 (2009), 155–163.

[40] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

[41] Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.

[42] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 601–610.

[43] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. 2003. Trust management for the semantic web. In *The Semantic Web-ISWC 2003*. Springer, 351–368.

[44] Francisco Aparecido Rodrigues. 2019. Network centrality: An introduction. In *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*. Springer, 177–196.

[45] José Ricardo Furlan Ronqui and Gonzalo Travieso. 2015. Analyzing complex networks through correlations in centrality measurements. *Journal of Statistical Mechanics: Theory and Experiment* 2015, 5 (2015), P05030.

[46] Polina Rozenshtein and Aristides Gionis. 2016. Temporal PageRank. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016*, Vol. 9852. Springer, 674–689.

[47] Nicola Santoro, Walter Quattrociocchi, Paola Flocchini, Arnaud Casteigts, and Frederic Amblard. 2011. Time-varying graphs and social network analysis: Temporal indicators and metrics. *arXiv preprint arXiv:1102.0629* (2011).

[48] Akrati Saxena and Sudarshan Iyengar. 2020. Centrality Measures in Complex Networks: A Survey. *CoRR* abs/2011.07190 (2020). arXiv:2011.07190 https://arxiv.org/abs/2011.07190

[49] Jun Sun, Jérôme Kunegis, and Steffen Staab. 2016. Predicting User Roles in Social Networks Using Transfer Learning with Feature Transformation. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops*. IEEE Computer Society, 128–135.

[50] John Tang, Ilias Leontiadis, Salvatore Scellato, Vincenzo Nicosia, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. 2013. Applications of temporal graph metrics to real-world networks. In *Temporal Networks*. Springer, 135–159.

[51] John Tang, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Vincenzo Nicosia. 2010. Analysing information flows and key mediators through temporal centrality metrics. In *Proc. 3rd Workshop on Social Network Systems*. 1–6.

[52] Ioanna Tsalouchidou, Ricardo Baeza-Yates, Francesco Bonchi, Kewen Liao, and Timos Sellis. 2019. Temporal betweenness centrality in dynamic graphs. *International Journal of Data Science and Analytics* (2019), 1–16.

[53] Philippe Vanhems, Alain Barrat, Ciro Cattuto, Jean-François Pinton, Nagham Khanafer, Corinne Régis, Byeul-a Kim, Brigitte Comte, and Nicolas Voirin. 2013. Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. *PloS one* 8, 9 (2013), e73970.

[54] Bimal Viswanath, Alan Mislove, Meeyoung 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.

[55] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage. 2008. Analyzing social bookmarking systems: A del. icio. us cookbook. In *Proceedings of the ECAI 2008 Mining Social Data Workshop*. 26–30.

[56] Scott White and Padhraic Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.* 266–275.

[57] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. 2014. Path problems in temporal graphs. *Proc. VLDB Endowment* 7, 9 (2014), 721–732.

[58] Xudong Wu, Luoyi Fu, Zixin Zhang, Huan Long, Jingfan Meng, Xinbing Wang, and Guihai Chen. 2020. Evolving Influence Maximization in Evolving Networks. *ACM Trans. Internet Technol.* 20, 4, Article 40 (Oct. 2020).

A NOTATION

Table 6 shows our commonly used symbols and their definitions.

Table 6: Commonly used notations

Symbol	Definition
$\mathcal{G} = (V, \mathcal{E}, \delta)$	Temporal graph $\mathcal{G}$ with nodes V and temp. edges $\mathcal{E}$
δ	Global transition time
$e = (u, v, t)$	Temporal (u, v)-edge at time t
$T(v)$	Set of availability times of edges incident to node v
$T(\mathcal{G})$	Set of availability and arrival times in $\mathcal{G}$
$G = (V, E, w)$	Directed graph with edge weights $w: E \to \mathbb{R}$
$\Phi : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$	Time depended weighting function
$\mathcal{W}_{in}(v, t), \mathcal{W}_{out}(v, t)$	Temporal walks ending, starting at node v and time t
$\boldsymbol{W}_{in}(v, t), \boldsymbol{W}_{out}(v, t)$	Total weight of temp. walks in $\mathcal{W}_{in}(v, t)$, $\mathcal{W}_{out}(v, t)$
$\tau^-(v), \tau^+(v)$	# of distinct arrival, starting times at $v \in V$
$\tau^-_{max}, \tau^+_{max}$	Max. # of distinct arrival, starting times over $v \in V$
ε	Error-tolerance parameter for approximation

B OMITTED PROOFS

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

Proof of Lemma 5.4. Let $\omega = \left(n^{t_1}_{v_1 v_2}, n^{t_2}_{v_2 v_3}, \ldots, n^{t_\ell}_{v_\ell v_{\ell+1}}\right)$. Application of edge weights yields $w_\Phi(\omega) = \prod_{i=1}^{\ell-1} \Phi(t_i + \delta, t_{i+1}) = \tau_\Phi(\Gamma(\omega))$. □

Proof of Theorem 5.7. Let ω_ℓ be a walk of length $1 \leq \ell \leq k$ arriving at vertex v. For $\ell = 1$, the walk consists of a single edge $e_1 = (u, v, t)$. Algorithm 2 iterates over all edges, and therefore, also over e_1. Therefore, $\boldsymbol{W}_{in}(v, t + \delta)$ 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 at u than t. Consider a walk $\omega_2 = ((u, w, t), (w, v, t'))$ of length $\ell = 2$ arriving at v. The walk $((u, w, t))$ arriving at vertex w was counted before because $t + \delta \leq t'$, hence $\boldsymbol{W}_{in}(w, t + \delta) > 0$. Next, the walks from w are added to the walks at v, weighted by $\Phi_{in}(t, t')$. Consider the walk $\omega_\ell = (e_1, \ldots, e_\ell = (v_\ell, v_{\ell+1}, t_\ell))$. By our assumption, it follows that the path $\omega_{\ell-1} = (e_1, \ldots, e_{\ell-1} = (v_{\ell-1}, v_\ell, t_{\ell-1}))$ of length $\ell - 1$ arrived at time $t_{\ell-1} + \delta$ at v_ℓ and was counted correctly. The algorithm adds the walks of length $\ell - 1$ arriving at v_ℓ to the number of walks of length ℓ at vertex $v_{\ell+1}$ weighted by $\Phi_{in}(t_{\ell-1}, t_\ell)$.

For the running time, Algorithm 2 iterates over all $|\mathcal{E}|$ edges in chronological order, and for each $e \in \mathcal{E}$, it iterates over all entries of $\boldsymbol{W}_{in}$ for which $\boldsymbol{W}_{in}(e.u, t) > 0$, i.e., at most τ^-_{max} rows. Therefore, the running time is in $O(|\mathcal{E}| \cdot \tau^-_{max})$, 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 $|\mathcal{E}|$. □

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 $\mathcal{G} = (V, \mathcal{E}, \delta)$ with $\delta > 0$ are counted exactly by Algorithm 4 in time and space $O(|\mathcal{E}| + e)$, where $e = |E(DL(\mathcal{G}))|$.*

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 $v \in V$.

forall $v \in V$ **do** $W_{out}(v) \leftarrow 1$ ▷ *Length 0 walks*
$S^0 \leftarrow \{v \in V \mid d^+(v) = 0\}$ ▷ *Find sinks*
$i \leftarrow 0$
repeat
  forall $v \in S^i$ **do**
    forall $u \in N^-(v)$ **do**
      $W_{out}(u) \leftarrow W_{out}(u) + w(u, v) \cdot W_{out}(v)$
      $S^{i+1} \leftarrow S^{i+1} \cup \{u\}$ ▷ *Avoid duplicates*
  $i \leftarrow i + 1$
until $S^i \neq \emptyset$

D ADDITIONAL RESULTS

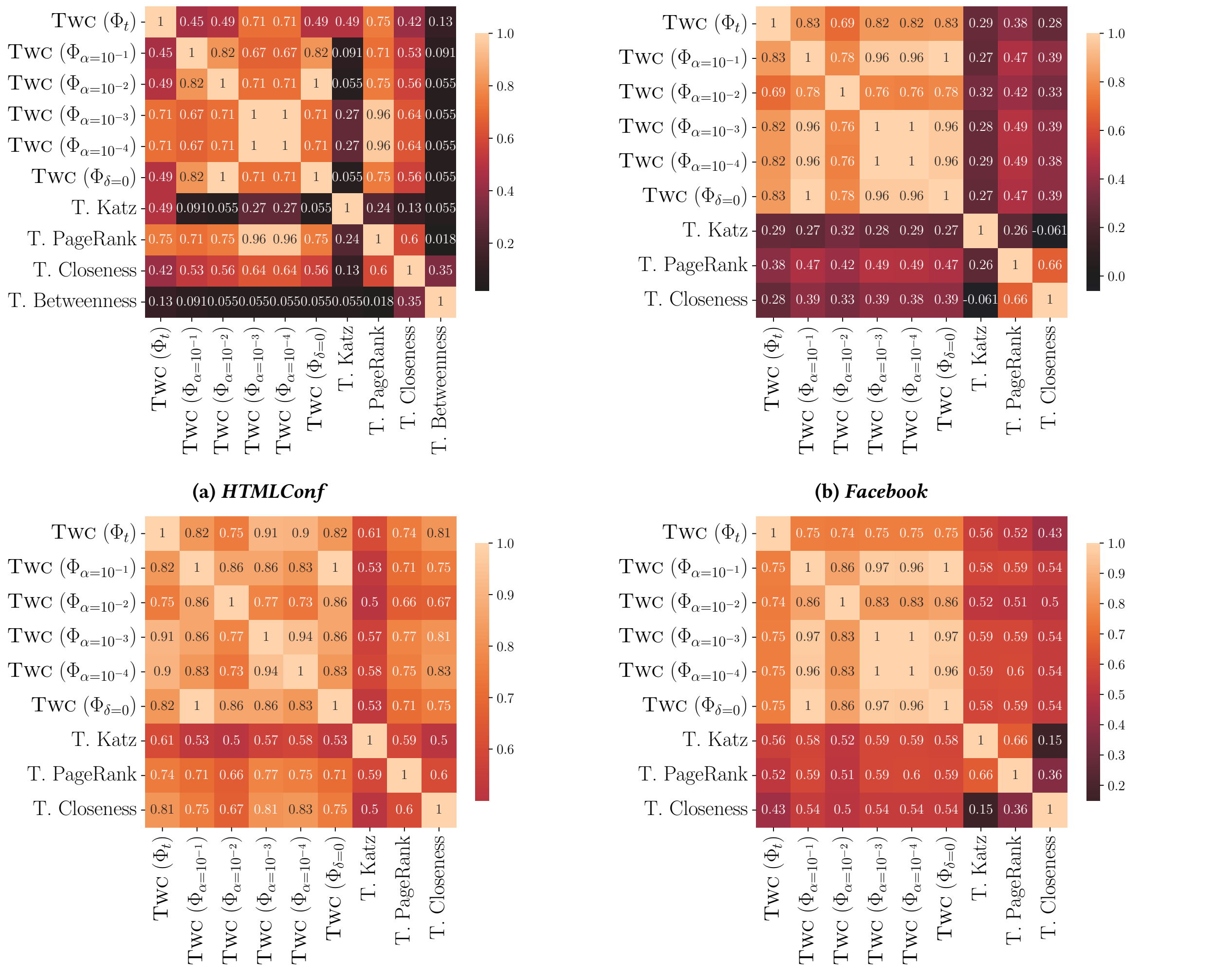
This section provides additional experimental results.

D.1 Memory Usage

Table 7 shows the total memory usage of Stream and Approx. For Stream, we report the results for Φ_α 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



(a) ***HTMLConf***

(b) ***Facebook***

(c) ***Enron***

(d) ***AskUbuntu***

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 (Oom–out of memory).

	Stream	Approx
Data set	$\alpha = 0.001$	$\varepsilon = 0.001$
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
Digg	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 α 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.