Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 27, no. 3, pp. 173–194 (2023)
DOI: 10.7155/jgaa.00619

Towards Classifying the Polynomial-Time Solvability of Temporal Betweenness Centrality

Maciej Rymar[1] *Hendrik Molter*[1,2] *André Nichterlein*[1] *Rolf Niedermeier*[1]

[1]TU Berlin, Algorithmics and Computational Complexity, Berlin, Germany
[2]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Submitted:	August 2021	Reviewed:	July 2022	Revised:	October 2022
Accepted:	February 2023	Final:	March 2023	Published:	May 2023
Article type:	Regular paper		Communicated by:	U. Brandes	

Abstract. In static graphs, the betweenness centrality of a graph vertex measures how many times this vertex is part of a shortest path between any two graph vertices. Betweenness centrality is efficiently computable and it is a fundamental tool in network science. Continuing and extending previous work, we study the efficient computability of betweenness centrality in *temporal* graphs (graphs with fixed vertex set but time-varying edge sets). Unlike in the static case, there are numerous natural notions of being a "shortest" temporal path (walk). Depending on which notion is used, it was already observed that the problem is #P-hard in some cases while polynomial-time solvable in others. In this conceptual work, we contribute towards classifying what a "shortest path (walk) concept" has to fulfill in order to gain polynomial-time computability of temporal betweenness centrality.

Keywords: temporal graphs, temporal paths and walks, network science, network centrality measures, counting complexity.

1 Introduction

Network science is a central pillar of data science. It relies on spotting and analyzing important network (graph) properties. Betweenness centrality, introduced by Freeman [16] and made a practical tool of high relevance by Brandes [9], is a key instrument in this area, in particular in the context of social network analysis.

MR partially supported by the DFG, project MATE (NI 369/17). HM supported by the DFG, project MATE (NI 369/17), and by the ISF, grants No. 1456/18 and No. 1070/20, and the ERC, grant number 949707. Main part of this work was done while HM was affiliated with TU Berlin. An extended abstract of this work appears in the proceedings of the *47th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2021)* [26]. Here, we provide full details and proofs.

E-mail addresses: m.rymar@tu-berlin.de (Maciej Rymar) molterh@post.bgu.ac.il (Hendrik Molter) andre.nichterlein@tu-berlin.de (André Nichterlein)

Informally, the *betweenness centrality* value of a graph vertex v correlates to the probability that v is visited by a randomly chosen shortest path. With the advent of investigating dynamically changing network structures and, thus, the growing interest in temporal graphs, studying concepts of temporal betweenness centrality and their algorithmic complexity became very popular over the recent years [1, 3, 11, 13, 14, 19, 20, 22, 23, 28, 29, 30, 32].

The temporal graphs we are considering have a fixed vertex set and edge set(s) that change over discrete time steps. A temporal path in such a graph has to respect time, that is, the path has to traverse edges at non-decreasing time steps. The study of *temporal* betweenness centrality is significantly richer than in classical, static graphs since in temporal graphs the term "shortest path" may have numerous different but natural interpretations. Indeed, the shortest transfer from a start vertex to a target vertex may even be a walk (and not just a path). There is intensive research on shortest-path and shortest-walk computations in temporal graphs [8, 10, 12, 33]. We refrain from going into the details here but refer to our predecessor work [11] for a more extensive discussion. What is important to note, however, is that the complexity of temporal betweenness centrality computation, which essentially boils down to a counting problem, crucially depends on the concept used. More specifically, the complexity may vary from polynomial-time solvable (with different polynomial degrees) to very hard (#P-hard). To systematically investigate this issue and to develop a better understanding of when one has to expect such a huge jump in computational complexity is the main motivation behind our work.

The by far closest reference point for our work is a previous paper from our group [11]. It also surveys the literature roughly till the year 2020. While this previous work also explored the practical side (algorithm engineering and experiments), the focus of this work is purely theoretical. After 2020, Simard et al. [28] studied a continuous-time scenario and betweenness based on shortest fastest paths, while we here focus on *discrete* time. Enright et al. [13] study the parameterized complexity and approximability of temporal betweenness variants that are #P-hard to compute. Our work has a significantly stronger conceptual objective than Buß et al.[11] had, so our classification results comprise the results there. Our contributions are based on coining the concept of *prefix-compatible* temporal walks. These walks can be counted in polynomial time and thus the corresponding temporal betweenness centrality value can be computed in polynomial time. To this end, we provide simple (still tunable) polynomial-time algorithms that apply to a whole class of temporal betweenness centrality problems. Moreover, we indicate that slightly relaxing from prefix-compatibility typically already yields #P-hardness.

2 Preliminaries

The fundamental concept we use in this work is the one of *temporal graphs*. A *directed temporal graph* $\mathcal{G}$ is a triple $(V, \mathcal{E}, T)$ such that V is a set of vertices, $\mathcal{E} \subseteq \{(u, v, t) \mid u, v \in V, u \neq v, t \in [T]\}$ is a set of time arcs, and $T \in \mathbb{N}$, where $[T] \coloneqq \{1, \ldots, T\}$ is a set of time steps; see Figure 1 for an illustration.

Throughout this work, let $n := |V|$ and $M := |\mathcal{E}|$. We call $V \times [T]$ the set of (possible) *vertex appearances*. We consider directed temporal graphs as temporal paths and walks are implicitly directed because of the ascending time labels. We call a time arc $e = (v, w, t)$ also the *transition* from v to w at time step t. We call v the starting point and w the endpoint of the transition. Using this, we can now define temporal walks and temporal paths; see Figure 1 for an illustration.

Definition 1 (Temporal Walk) *A* temporal walk W *is an ordered sequence* $(e_1, \ldots, e_k) \in \mathcal{E}^k$ *of transitions such that for each* $i \in [k-1]$ *the endpoint of* e_i *is the starting point of* e_{i+1} *and*

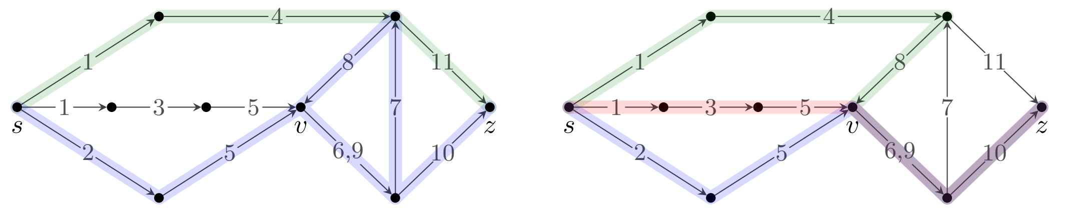


Figure 1: Our running example for a temporal graph $\mathcal{G}$ with 9 vertices and 13 time-arcs (the outgoing arc from vertex v denotes two time-arcs at time steps 6 and 9). The number(s) on the arcs denote the time steps. *Left:* Highlighted are the unique shortest s-z-path (top path in green) in $\mathcal{G}$ and a fastest s-z-walk (bottom walk in blue, v and its successor appear twice in the walk). *Right:* Three foremost s-z-paths in $\mathcal{G}$ are highlighted ($\mathcal{G}$ has two more foremost s-z-walks; for visibility not highlighted).

$t_i \leq t_{i+1}$, where t_i and t_{i+1} are the time labels of transitions e_i and e_{i+1}, respectively. A temporal walk is strict *if $t(e_i) < t(e_{i+1})$ for each $i \in \{1, \dots, k-1\}$. The length of W is* $\text{length}(W) := k$.

Let $W = (e_1, \dots, e_k)$ be a temporal walk. We call W a *temporal s-z-walk* if $e_1 = (s, v, t)$ and $e_k = (w, z, t')$ for some v, w and some t, t', and we call W a *temporal s-(z, t')-walk* if $e_1 = (s, v, t)$ and $e_k = (w, z, t')$ for some v, w and some t. We say that W *ends* in z at time t'.

A temporal walk may visit the same vertex more than once. In contrast to that, a temporal *path* visits each vertex at most once. This is analogous to static graphs. In contrast to the static setting, there are several canonical notions of "optimal" temporal walks. The three most important ones [10] are

- *shortest* temporal walks, which are temporal walks between two vertices that use a minimum number of time arcs,
- *fastest* temporal walks, which are temporal walks between two vertices with a minimum difference between the time steps of the first and the last transition used by the walk, and
- *foremost* temporal walks, which are temporal walks between two vertices with a minimum time step on their last transition, that is, they have an earliest arrival time.

See Figure 1 for an illustration. The corresponding optimal temporal paths are defined analogously. We remark that in the case of "shortest", every shortest temporal walk is in fact a temporal path, similarly to the static case.[1] However, for "fastest" and "foremost" temporal walks this is generally not the case.

For readability, we use the notation $v \xrightarrow{t} w$ instead of the triple (v, w, t). If a temporal walk W contains the transition $v \xrightarrow{t} w$, then we say that W *visits* (or *goes through*) vertex appearance (w, t) (in Figure 1 (left) the blue walk visits $(v, 5)$ and $(v, 8)$). For any $0 \leq i < j \leq \ell$, we write (if $i = 0$, then we define the following for $t_0 = 0$):

[1]In fact, all optimal temporal path concepts (we are aware of) where path counting and computing the betweenness centrality can be done in polynomial time have this property, ensuring that optimal walks are indeed paths.

$$\begin{aligned} P\left[(v_i,t_i),(v_j,t_j)\right] &\coloneqq v_i \overset{t_{i+1}}{\to} v_{i+1} \overset{t_{i+2}}{\to} \dots \overset{t_j}{\to} v_j, \\ P\left[\bullet,(v_j,t_j)\right] &\coloneqq v_0 \overset{t_1}{\to} v_1 \overset{t_2}{\to} \dots \overset{t_j}{\to} v_j, \\ P\left[(v_i,t_i),\bullet\right] &\coloneqq v_i \overset{t_{i+1}}{\to} v_{i+1} \overset{t_{i+2}}{\to} \dots \overset{t_\ell}{\to} v_\ell. \end{aligned}$$

We use an analogous notation for temporal walks. Note that for a non-strict temporal walk W the above notation may not be well-defined since the same vertex appearance (v,t) may appear more than once in W. In this case, we define $W\left[(v_i,t_i),(v_j,t_j)\right]$ to be the subwalk of W from the first appearance of $u \overset{t_i}{\to} v_i$ for any u to the last appearance of $u \overset{t_j}{\to} v_j$ for any u. The cases of $P\left[\bullet,(v_j,t_j)\right]$ and $P\left[(v_i,t_i),\bullet\right]$ are handled analogously.

Furthermore, we use $\circ$ to denote *concatenations* of temporal walks, that is, let W,W' be two temporal walks such that W ends in v and W' starts in v. Let t be the time label on the last transition of W and t' the label on the first transition of W'. Then, if $t' \geq t$, we denote with $W \circ W'$ the concatenation of W and W'.

Given a temporal graph $\mathcal{G}$, we denote by walks$(\mathcal{G})$ the set of all temporal walks in $\mathcal{G}$. Subsequently, we will need to consider the successors (or dually, the predecessors) of each vertex appearance on temporal walks in some $\mathcal{W} \subseteq$ walks$(\mathcal{G})$.

Definition 2 (Direct predecessor set, direct successor set) *Let $\mathcal{G}=(V,\mathcal{E},T)$ be a temporal graph and $\mathcal{W} \subseteq$ walks$(\mathcal{G})$ be a subset of its temporal walks. Fix a source vertex $s \in V$. Let $\mathcal{W}_s \subseteq \mathcal{W}$ be the set of temporal walks in $\mathcal{W}$ that start in s. Now let $(w,t') \in V \times [T]$ be any vertex appearance. Then $\mathrm{Pre}_s^{\mathcal{W}}(w,t')$ is the set of all direct predecessors of (w,t') on temporal walks in $\mathcal{W}_s$:*

$$\begin{aligned} \mathrm{Pre}_s^{\mathcal{W}}(w,t') \coloneqq &\left\{(v,t) \in V \times [T] \mid \exists W \in \mathcal{W}_s \colon u \overset{t}{\to} v \overset{t'}{\to} w \in W\right\} \\ &\cup \left\{(s,0) \mid \exists W \in \mathcal{W}_s \colon s \overset{t'}{\to} w \in W\right\}. \end{aligned}$$

The set $\mathrm{Succ}_s^{\mathcal{W}}(v,t)$ of successors of a vertex appearance (v,t) is the inverse of the predecessor relation:

$$\mathrm{Succ}_s^{\mathcal{W}}(v,t) \coloneqq \left\{(w,t') \mid (v,t) \in \mathrm{Pre}_s^{\mathcal{W}}(w,t')\right\}.$$

Clearly, the direct predecessor sets induce a relation over vertex appearances. We use this to define the following directed graph. We remark that this graph is similar to the so-called static expansion [21, 33, 34] that is tailored to a specific source vertex, see Figure 2 for an example.

Definition 3 (Predecessor graph) *Let $\mathcal{G}=(V,\mathcal{E},T)$ be a temporal graph and $\mathcal{W} \subseteq$ walks$(\mathcal{G})$ be a subset of its temporal walks. Fix a source vertex $s \in V$. Then $G_s^{\mathrm{Pre}} \coloneqq (U,A)$ is the* predecessor graph (of s, with respect to $\mathcal{W}$)*, where*

$$\begin{aligned} U &\coloneqq \{(s,0)\} \cup \{(w,t') \mid \mathrm{Pre}_s^{\mathcal{W}}(w,t') \neq \emptyset\}, \text{ and} \\ A &\coloneqq \{((v,t),(w,t')) \mid (v,t) \in \mathrm{Pre}_s^{\mathcal{W}}(w,t')\}. \end{aligned}$$

We next introduce some notation and definitions for temporal walk counting and temporal betweenness.

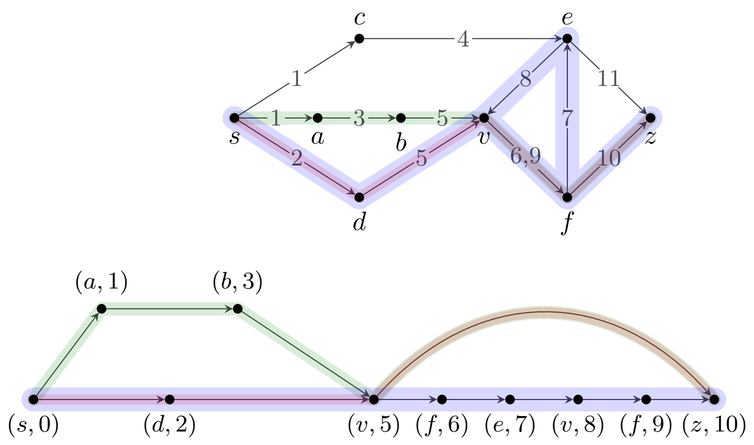


Figure 2: An illustration of the predecessor graph. *Top:* A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$ with a set $\mathcal{W}$ of three highlighted temporal walks. *Bottom:* The predecessor graph G_s^{Pre} of s with respect to $\mathcal{W}$. The horizontal position of each vertex corresponds to the time of the corresponding vertex appearance and the walks corresponding to $\mathcal{W}$ are highlighted.

Definition 4 *Let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph and $\mathcal{W} \subseteq walks(\mathcal{G})$ be a subset of its temporal walks. Let $s, v, z \in V$ and $t \in [T]$. Then,*

- *$\sigma_{sz}^{\mathcal{W}}$ is the number of temporal s-z-walks in $\mathcal{W}$ that start in s and end in z.*
- *$\sigma_{s(z,t)}^{\mathcal{W}}$ is the number of temporal s-z-walks in $\mathcal{W}$ that start in s and end in z at time t.*
- *$\sigma_{sz}^{\mathcal{W}}(v)$ is the number of temporal s-z-walks in $\mathcal{W}$ that go through the vertex v. Furthermore, $\sigma_{sz}^{\mathcal{W}}(z) = \sigma_{sz}^{\mathcal{W}}(s) = \sigma_{sz}^{\mathcal{W}}$ and $\sigma_{ss}^{\mathcal{W}}(s) = \sigma_{ss}^{\mathcal{W}}$;*
- *$\sigma_{sz}^{\mathcal{W}}(v, t)$ is the number of temporal s-z-walks in $\mathcal{W}$ that go through the vertex appearance (v, t). Furthermore, $\sigma_{sz}^{\mathcal{W}}(s, 0) = \sigma_{sz}^{\mathcal{W}}(s) = \sigma_{sz}^{\mathcal{W}}$ and $\sigma_{sz}^{\mathcal{W}}(s, t') = 0$ for all $t' \in [T]$.*

Based on this definition, we can introduce the notions of dependency of vertices on other vertices, similarly to how it was done by Brandes [9] in the static case. We remark that the notions for the temporal setting introduced in the following are very similar to the ones used by Buß et al. [11]. We present them again here for completeness and since we adapt them for general sets of temporal walks.

Definition 5 (Pair dependency, cumulative dependency) *Let $\mathcal{G}$ be a temporal graph and $\mathcal{W} \subseteq walks(\mathcal{G})$ be a subset of its temporal walks. Then,*

$$\delta_{sz}^{\mathcal{W}}(v) := \begin{cases} 0, & \text{if } \sigma_{sz}^{\mathcal{W}} = 0, \\ \sigma_{sz}^{\mathcal{W}}(v)/\sigma_{sz}^{\mathcal{W}}, & \text{otherwise;} \end{cases} \qquad \delta_{s\bullet}^{\mathcal{W}}(v) := \sum_{z \in V} \delta_{sz}^{\mathcal{W}}(v)$$

are the pair dependency *of s and z on v and the* cumulative dependency *of s on v, respectively.*

In other words, $\delta_{sz}^{\mathcal{W}}(v)$ is the fraction of temporal s-z-walks that go through v. Intuitively, the higher this fraction is, the more important v is to the connectivity of s and z in the graph.

Furthermore, $\delta_{s\bullet}^{\mathcal{W}}(v)$ is the cumulative dependency of s on v for all possible destinations. These notions can be used to define temporal betweenness centrality, which intuitively captures how *all* other vertices depend on v for their connectivity.

Definition 6 (Temporal betweenness centrality) *Let $\mathcal{G}$ be a temporal graph and $\mathcal{W} \subseteq$ walks$(\mathcal{G})$ be a subset of its temporal walks. Then, for any vertex $v \in V$,*

$$C_B^{\mathcal{W}}(v) := \sum_{s \neq v \neq z} \delta_{sz}^{\mathcal{W}}(v) \qquad \text{and} \qquad \hat{C}_B^{\mathcal{W}}(v) := \sum_{s,z \in V} \delta_{sz}^{\mathcal{W}}(v)$$

are the temporal betweenness centrality $C_B^{\mathcal{W}}(v)$ *of v and the* total temporal betweenness centrality $\hat{C}_B^{\mathcal{W}}(v)$ *of v (with respect to $\mathcal{W}$).*

If the set of walks $\mathcal{W}$ in question is clear from the context, then we omit the superscript $\mathcal{W}$. The central reason behind mainly using total temporal betweenness centrality instead of the standard temporal betweenness in the following is that it simplifies some of our proofs as it works well with our definition of cumulative dependency:

Observation 1 *For any vertex $v \in V$, $\hat{C}_B^{\mathcal{W}}(v) = \sum_{s \in V} \delta_{s\bullet}^{\mathcal{W}}(v)$.*

We have that $\hat{C}_B^{\mathcal{W}}(v)$ and $C_B^{\mathcal{W}}$ are tightly related:

Observation 2 (Lemma 2.8 in Buß et al.[11]) *For any vertex $v \in V$,*

$$C_B^{\mathcal{W}}(v) = \hat{C}_B^{\mathcal{W}}(v) - \sum_{w \in V} \left(\left[\sigma_{vw}^{\mathcal{W}} > 0\right]_{\mathbb{1}} + \left[\sigma_{wv}^{\mathcal{W}} > 0\right]_{\mathbb{1}} \right) + \left[\sigma_{vv}^{\mathcal{W}} > 0\right]_{\mathbb{1}}.$$

We use a straightforward generalization of Definition 5 to vertex appearances. Using this definition will be more convenient in our proofs.

Definition 7 (Temporal pair dependency, temporal cumulative dependency) *Let $\mathcal{G}$ be a temporal graph and $\mathcal{W}$ be a subset of its walks. Then,*

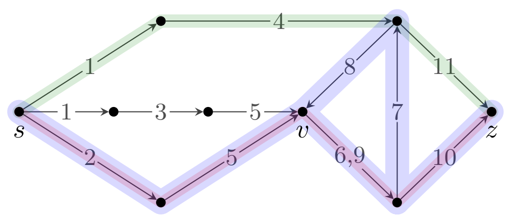
$$\delta_{sz}^{\mathcal{W}}(v,t) := \begin{cases} 0, & \text{if } \sigma_{sz}^{\mathcal{W}} = 0 \\ \sigma_{sz}^{\mathcal{W}}(v,t)/\sigma_{sz}^{\mathcal{W}}, & \text{otherwise} \end{cases} \qquad \text{and} \qquad \delta_{s\bullet}^{\mathcal{W}}(v,t) := \sum_{z \in V} \delta_{sz}^{\mathcal{W}}(v,t)$$

are the temporal pair dependency of s and z on (v,t) *and the* temporal cumulative dependency of s on (v,t), *respectively. Additionally, the special case $\delta_{sv}^{\mathcal{W}}(v,t)$ denotes the* appearance dependency of s on (v,t).

We can observe a simple relation between dependencies on vertices and dependencies on vertex appearances. In particular, we can compute dependencies on vertices using the dependencies on vertex appearances, which allows us to focus on the latter.

Observation 3 *For any vertex $v \in V$, it holds that*

$$\delta_{sz}^{\mathcal{W}}(v) = \sum_{t \in [T]} \delta_{sz}^{\mathcal{W}}(v,t) \qquad \text{and} \qquad \delta_{s\bullet}^{\mathcal{W}}(v) = \sum_{t \in [T]} \delta_{s\bullet}^{\mathcal{W}}(v,t).$$



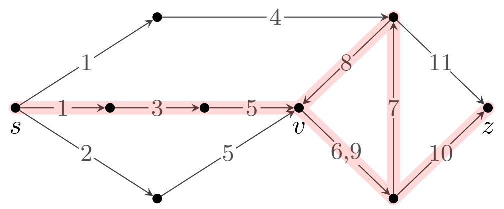


Figure 3: Examples for c-optimal walks for various c. *Left:* The top (green) s-z-path is the shortest s-z-walk, that is, $c(W)$ is the number of time arcs in the walk W. The blue walk and the purple path are the two fastest s-z-walks in the graph; here $c(W)$ is the difference of the time steps of the first and last time arc in the walk W. *Right:* Highlighted is the only *2-restless* s-z-walk, that is, the difference between the time steps of two consecutive time arcs is at most two [8, 12]. This could be encoded in c as follows: for a walk $W = (e_1, \dots, e_k)$ we have $c(W) = 1$ if $t(e_i) + 2 \geq t(e_{i+1})$ for all $i \in [k-1]$ and $c(W) = \infty$ otherwise. Notably, in general it is NP-hard to decide whether such a 2-restless *s-z-path* exists [12], but for walks even the optimization variants (shortest, fastest, ...) are polynomial-time solvable [8].

3 Prefix-compatibility

In this section, our goal is to find an easy-to-understand-and-use property for optimality concepts for temporal walks and paths that is sufficient for polynomial-time solvability of (1) counting optimal temporal walks and (2) computing the temporal betweenness with respect to that optimality concept for temporal walks. We call this property "prefix-compatibility". Intuitively, a class of temporal walks is prefix-compatible if prefixes of optimal temporal walks are also optimal ("prefix-optimality") and prefixes of optimal temporal walks can be exchanged ("prefix-exchangeability").

3.1 Definition of prefix-compatibility

To formally define optimality concepts for temporal walks, we use cost functions.

Definition 8 (Cost function) *Let $\mathcal{W}$ be the set of all temporal walks in a temporal graph $\mathcal{G}$. Then, a function of the form $c : \mathcal{W} \to \mathbb{R} \cup \{\infty\}$ is a* cost function.

We remark that for this work we assume that the cost function can be computed in constant time, which turns out to be a valid assumption for many optimality concepts. When considering cost functions that need polynomial time to be evaluated, this polynomial factor would form an extra multiplicative term in our running time results.

Let c be a cost function and let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. Fix a source $s \in V$. Then, for every vertex appearance $(v, t) \in V \times [T]$ we define $c^*_s(v, t)$ to be the minimum value of c assumed over all temporal s-(v, t)-walks. That is, we have $c^*_s(v, t) = \min_{s\text{-}(v,t)\text{-walk } W}\{c(W)\}$. If the minimum is not defined or there is no temporal s-(v, t)-walk, then $c^*_s(v, t) := \infty$. Similarly, we define the optimal c-values for the vertices $v \in V$ as $c^*_s(v) := \min_{t \in [T]}\{c^*_s(v, t)\}$. We call a temporal s-(v, t)-walk W *c-optimal* if $c(W) = c^*_s(v, t) < \infty$. Similarly, we call a temporal s-v-walk W *c-optimal* if $c(W) = c^*_s(v) < \infty$. Observe that this notion of c-optimal walks is very general and allows to capture essentially all natural walk concepts, see Figure 3 for some examples.

For a temporal graph we can now define the set of walks that is optimal with respect to some cost function c in a straightforward way:

Definition 9 (Induced set of optimal temporal walks) *Let c be a cost function and let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. For $s, z \in V$, let $\mathcal{W}_{sz}$ be the set of all temporal s-z-walks in $\mathcal{G}$. Then*

$$\mathcal{W}^{(c)} := \bigcup_{s,z \in V} \{W \in \mathcal{W}_{sz} \mid c(W) < \infty \wedge c(W) = c^*_s(z)\}$$

is the induced set of optimal temporal walks (of c)*.*

From now on, we introduce the two properties for cost functions that we need to obtain prefix-compatibility. We start with "prefix-optimality", which intuitively states that prefixes of optimal temporal walks are also optimal.

Definition 10 (Prefix-optimality) *Let c be a cost function and let $\mathcal{W}^{(c)}$ be a set of optimal temporal walks in a temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$ that is induced by c. Then c is* prefix-optimal *if for every $s, z \in V$, for every temporal s-z-walk $W \in \mathcal{W}^{(c)}$, and for every vertex appearance $(v, t) \in V \times [T]$ that is visited by W it holds that $c(W[\bullet, (v, t)]) = c^*_s(v, t)$.*

Note that we do not require the prefixes to be optimal temporal walks *to a vertex*, so the temporal walk $c(W[\bullet, (v, t)])$ is not required to be in the induced set of optimal temporal walks $\mathcal{W}^{(c)}$. If c is clear from the context and there is no danger of confusion, then we drop the superscript (c).

The second property we introduce is "prefix-exchangeability". It intuitively states that a prefix of an optimal temporal walk can be exchanged by certain other temporal walks.

Definition 11 (Prefix-exchangeability) *Let c be a cost function and let $\mathcal{W}^{(c)}$ be a set of optimal temporal walks in a temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$ that is induced by c. Then c is* prefix-exchangeable *if for every vertex appearance $(v, t) \in V \times [T]$ such that there exist $s, z \in V$ for which there is a temporal s-z-walk $W \in \mathcal{W}^{(c)}$ going through (v, t) and for every temporal s-(v, t)-walk W' with $c(W') = c^*_s(v, t)$ it holds that $W' \circ W[(v, t), \bullet] \in \mathcal{W}^{(c)}$, that is, $c(W' \circ W[(v, t), \bullet]) = c^*_s(z)$.*

In other words, if there is an optimal temporal s-z-walk W going through (v, t), then all c-optimal temporal s-(v, t)-walks can be substituted for the first part of W to get a c-optimal temporal s-z-walk.

It is convenient to combine the two main properties into one.

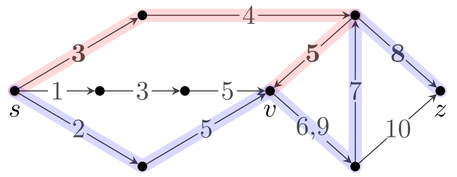
Definition 12 (Prefix-compatibility) *Let c be a cost function. Then c is* prefix-compatible *if it is both prefix-optimal and prefix-exchangeable.*

In Figure 4 we illustrate that (Δ-restless[2]) fastest paths do not satisfy prefix-compatibility.

3.2 Examples of prefix-compatible cost functions

We exemplarily show that five well-known optimality concepts for temporal paths and walks [8, 10, 33] can be expressed by prefix-compatible cost functions. The optimality concepts we consider here are *foremost temporal walks* and *shortest temporal paths*. We further consider *shortest fastest temporal paths*, which are temporal paths that are shortest among all fastest temporal paths and *shortest restless temporal walks* [8], which are the shortest temporal walks among all temporal walks where the difference of the time labels of two consecutive transitions is bounded by some constant. Lastly, we consider *strict prefix-foremost temporal paths* [33]. A temporal path is strict prefix-foremost if it is foremost and every prefix is also foremost and all transitions have increasing time labels.

[2]A Δ-*restless* temporal path can only dwell at most Δ time steps at any vertex [8, 12].



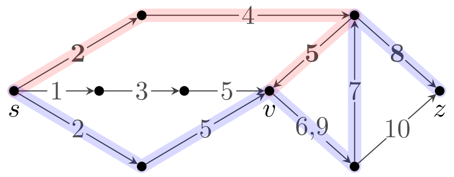


Figure 4: Counter-examples showing that (restless) fastest paths neither satisfy prefix-optimality nor prefix-exchangeability. Inhere, three time steps (indicated by bold numbers) are updated in our standard temporal graph. *Left:* The blue path (starting at time step 2) is the unique fastest 3-restless (cf. Figure 3) s-z-path with travel time $8-2=6$; it is not prefix-optimal as the red path is a faster s-$(v,5)$-path (travel time $5-2=3$ vs. $5-3=2$). *Right:* The blue path (starting at time step 2) is the fastest s-z-path with travel time $8-2=6$; it is not prefix-exchangeable as the red path is also a fastest s-$(v,5)$-path but the corresponding prefix of the blue path cannot be replaced with the red path as the resulting walk would not be a path.

Proposition 1 *The cost functions describing the following optimality concepts are prefix-compatible:*

- *foremost temporal walk,*
- *shortest temporal path,*
- *shortest fastest temporal path,*
- *shortest restless temporal walks, and*
- *strict prefix-foremost temporal path.*

Proof: For each of the concepts, let $P = v_0 \xrightarrow{t_1} v_1 \xrightarrow{t_2} \dots \xrightarrow{t_{\ell-1}} v_{\ell-1} \xrightarrow{t_\ell} v_\ell$ be a temporal path (or walk, depending on the concept).

Foremost temporal walks:

In this case, the cost function c maps temporal walks to the time label of their last transition, that is, $c(P) := t_\ell$. It is clear that this cost function is prefix-optimal, since every temporal s-(v,t)-walk, that is, a temporal walk to a vertex *appearance*, is trivially c-optimal. The cost function is also prefix-exchangeable, since every prefix of a foremost temporal walk to some vertex appearance (v,t) can be substituted by a different temporal walk to that vertex appearance and the overall temporal walk will remain foremost since the arrival time to the last vertex is not changed.

Shortest temporal paths:

In this case, the cost function c maps temporal walks to the number of transitions in the temporal walk, that is, $c(P) := \ell$. Similarly to the static case, we can observe that every c-optimal temporal walk is in fact a temporal path. We have that c is prefix-optimal, since similarly to the static case we have that every prefix of a shortest temporal path is also a shortest temporal path. The cost function c is also prefix-exchangeable, since substituting a prefix by another optimal, that is, equally long, prefix does not change the length of the temporal path. Furthermore, we still have a path since, otherwise, if a vertex is visited multiple times, then there is a shorter path that waits in the vertex that is visited multiple times, a contradiction to the assumption that the original path was shortest.

Shortest fastest temporal paths:

In this case, the cost function c maps temporal walks e. g. to the duration times n plus the number of transitions in the temporal walk, that is, $c(P) := n \cdot (t_\ell - t_1) + \ell$. Similarly to the case of shortest temporal paths, we can observe that every c-optimal temporal walk is in fact a temporal path. We have that c is prefix-optimal since a prefix not being c-optimal yields a contradiction to the assumption that the original temporal path was c-optimal: we replace the presumably non-optimal prefix by an optimal one. Since both prefixes end at the same vertex appearance, the optimal one is either faster or it has the same duration but is shorter. In both cases substituting the original prefix by the optimal one yields a "better" (according to c) temporal path (if it is not a path but a walk, we can wait at vertices that are visited multiple times). The cost function c is also prefix-exchangeable, since substituting a prefix by another optimal prefix does not change the duration or the length of the temporal path. Furthermore, we still have a path since, otherwise, if a vertex is visited multiple times, then there is a shorter path (with the same duration) that waits in the vertex that is visited multiple times, a contradiction to the assumption that the original path was shortest fastest.

Shortest restless temporal walks:

Let Δ be the upper bound on the difference of the time labels on two consecutive transitions. Similarly to shortest temporal paths, the cost function c maps temporal walks to the number of transitions in the temporal walk but under the condition that no two consecutive transitions have time labels that differ by more than Δ. If the walk contains two consecutive transitions where the corresponding time labels differ by more than Δ, then c maps this walk to infinity. Formally,

$$c(P) := \begin{cases} \ell, & \text{if for all } i \in [\ell - 1] \colon t_i - t_{i-1} \leq \Delta \\ \infty, & \text{else.} \end{cases}$$

We have that c is prefix-optimal, since a prefix not being c-optimal yields a contradiction to the assumption that the original temporal walk was c-optimal: we replace the presumably non-optimal prefix by an optimal one. Since both prefixes end at the same vertex appearance, the optimal one is shorter. Now substituting the original prefix by the optimal one yields a shorter restless temporal walk. The cost function c is also prefix-exchangeable, since substituting a prefix by another optimal, that is, equally long prefix does not change the length of the temporal walk.

Strict prefix-foremost temporal paths:

In this case, the cost function c maps to the last time label in the path if the path is indeed strict and prefix-foremost. Formally,

$$c(P) := \begin{cases} t_\ell, & \text{if } P \text{ is strict and prefix-foremost} \\ \infty, & \text{else.} \end{cases}$$

Note that the property of being prefix-foremost does not only depend on P but also on the given temporal graph. Nevertheless we have that, by definition, every prefix of a strict prefix-foremost temporal path is strict and prefix-foremost, and hence the corresponding cost function is prefix-optimal. Furthermore, we have that if we exchange a prefix of a strict prefix-foremost temporal path with another optimal prefix, then we again have the property that every prefix of the obtained path is foremost and we still have a path, since otherwise we visit a vertex multiple times at different time steps (since we are in the strict case), which is a contradiction to the original temporal path being prefix foremost. It follows that the corresponding cost function is also prefix-exchangeable.

□

We believe that many other natural optimality concepts can be shown to be prefix-compatible in a similar way to our examples.

Since we aim for a very general framework, it is not surprising that our running times for temporal betweenness computation can be improved for specific optimality concepts by tailored algorithms. As we will show in Theorems 1 and 2, we can count c-optimal temporal walks and compute the temporal betweenness centrality with respect to optimal walks for prefix-compatible cost functions c in $O(n^2MT^2)$ time. However, for example for strict prefix-foremost paths and shortest temporal paths, the corresponding temporal betweenness computation can be done in $O(nM\log M)$ and $O(n^3T^2)$ time, respectively [11]. Also the space requirements can be improved for specific optimality concepts [11]. We remark that the well-known techniques for static graphs [6, 9, 15] can mostly be transferred to the temporal setting in straightforward ways.

3.3 Necessity of prefix-optimality and prefix-exchangeability

We now briefly motivate why we need a cost function c to be both prefix-optimal and prefix-exchangeable in order to be able to count c-optimal temporal walks in polynomial time. We do this by giving examples of cost functions which have only one of the two properties and where the corresponding problem of counting c-optimal temporal walks is #P-hard. Note that this implies that the corresponding temporal betweenness computation problem is also #P-hard [11]. We remark that this does not imply that prefix-optimality and prefix-exchangeability cannot be replaced by some weaker requirements.

Proposition 2 *There exists a prefix-optimal cost function c for which counting the number of c-optimal temporal walks is #P-hard.*

Proof: Consider the set of foremost temporal paths (strict or non-strict). It is easy to see that this set is induced by the following cost function c:

$$c(W) := \begin{cases} t, & W \text{ is a (strict) temporal path with arrival time } t, \\ \infty, & \text{otherwise.} \end{cases}$$

Moreover, c is trivially prefix-optimal since all temporal paths arriving at the same vertex appearance have the same value of the criterion function. However, it is known that counting foremost temporal paths is #P-hard [1, 11]. □

As demonstrated, now we have that if we leave out prefix-exchangeability, then we obtain hardness. Next, we show that if we leave out prefix-optimality, then we also obtain hardness.

Proposition 3 *There exists a prefix-exchangeable cost function c for which counting the number of c-optimal temporal walks is #P-hard.*

Proof: We reduce the #P-hard PATHS problem [31] to the problem of counting c-optimal temporal walks for a prefix-exchangeable cost function c. In PATHS we are given a static graph $G = (V, E)$ and two vertices $s, z \in V$, and are asked to count the number of different paths from s to z in G.

Consider graph G and the two vertices $s, z \in V$. We add a special degree-one terminal vertex $z^\star$ to V and connect it to z.

Then,

$$c(W) := \begin{cases} \text{length}(W), & W \text{ is a temporal } s\text{-}v\text{-walk for a } v \in V \setminus \{z^\star\}, \\ 1, & W \text{ is a temporal } s\text{-}z^\star\text{-path}, \\ \infty, & \text{otherwise.} \end{cases}$$

Intuitively, for all $v \in V \setminus \{z^\star\}$ we consider only the shortest temporal s-v-walks as optimal (and obviously such shortest walks are also paths). This set of shortest walks is also clearly prefix-exchangeable (and prefix-optimal).

For the special vertex we proceed differently, however. We define every temporal s-$z^\star$-path as optimal, whether it is shortest or not. Note that clearly all paths to $z^\star$ need to go through z and, conversely, any temporal s-z-path can be extended to a temporal s-$z^\star$-path. Hence, the number of c-optimal temporal s-$z^\star$-paths is precisely the number of temporal s-z-paths. Also note that, in particular, any shortest temporal s-z-path can be extended to a temporal s-$z^\star$-path, so our cost function will remain prefix-exchangeable. However, prefix-optimality is now violated since we may have a non-shortest temporal s-z-path as a subpath of a c-optimal temporal s-$z^\star$-path. $\square$

4 Counting walks

In this section, complementing the hardness shown in Section 3.3, we extend classic algorithms for path and walk counting to our setting. This will provide a polynomial-time algorithm for counting optimal walks with respect to a prefix-compatible cost function.

The general idea is roughly as follows: First, compute the static predecessor graph $G_s^{\text{Pre}}(c)$ with respect to c (see Definition 3) using a slightly modified version of the classic Bellman-Ford algorithm [6, 15]. Second, count the walks in this static graph $G_s^{\text{Pre}}(c)$ with known approaches; the results correspond to the number of c-optimal walks in the temporal input graph.

We start with statements explaining the connection between $G_s^{\text{Pre}}(c)$ and the number of walks in the temporal input graph. Here, an important corner case is that there might be infinitely many c-optimal walks.

Definition 13 (Finiteness) *Let c be a cost function for a temporal graph $\mathcal{G}$. Then, c is* finite *on $\mathcal{G}$ if the induced set $\mathcal{W}^{(c)}$ of optimal temporal walks of c has finite cardinality.*

As stated next, finiteness of the cost function c coincides with the predecessor graph $G_s^{\text{Pre}}(c)$ containing directed cycles and is, thus, easy to detect.

Lemma 1 *Let c be a prefix-compatible cost function. Then c is finite if and only if the predecessor graph $G_s^{\text{Pre}}(c)$ is acyclic.*

Before proving Lemma 1, we show the following lemma.

Lemma 2 *Let c be a prefix-compatible and finite cost function. Then no c-optimal walk visits the same vertex appearance (v, t) twice.*

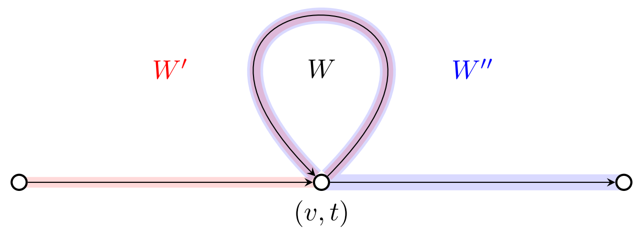


Figure 5: Illustration of the connection of the walks W, W', and W'' in the proof of Lemma 2.

Proof: Assume for the purpose of contradiction that there exists a pair of vertices $s, z \in V$ such that there exists a c-optimal walk W that visits (v, t) (at least) twice.

Let now W' be the prefix of W that ends at the second time when (v, t) appears in W and let W'' be the suffix of W that starts at the first appearance of (v, t) in W (see Figure 5).

By prefix-optimality, we have $c(W') = c_s^*(v, t)$. Hence, by prefix-exchangeability, the walk $W''' = W' \circ W''$ is also a c-optimal temporal s-z-walk. However, it visits (v, t) one more time than W does.

We can now repeat the same procedure with W''' to get a walk that visits (v, t) one more time than W''' does. By inductively carrying on the process, each time adding one more "loop" to our walk, we can see that we can visit (v, t) arbitrarily many times on a c-optimal temporal s-z-walk. This contradicts the finiteness of c. □

Using Lemma 2, we can prove the aforementioned equivalence:

Proof: [Proof of Lemma 1.] ("⇒") Assume for the purpose of contradiction that the predecessor relation is cyclic. Then there exist two vertex appearances (v, t) and (w, t) such that there exists a walk W that first visits (v, t) and then (w, t) and a walk U that does the opposite.

Let now W' be the prefix of W that ends at the first appearance of (w, t) and U' the suffix of U that starts at the first appearance of (w, t). Then, by prefix-optimality, we have $c(W') = c_s^*(w, t)$. Hence, by prefix-exchangeability, we have that $X = W' \circ U'$ is c-optimal. However, X visits the same vertex appearance (v, t) twice, contradicting Lemma 2.

("⇐") This direction is easy to see: clearly, there are only finitely many walks that do not visit the same vertex appearance more than once. Hence, if the set of c-optimal walks is infinite, there has to exist a c-optimal walk that visits some vertex appearance (v, t) twice. Therefore (v, t) is its own predecessor (transitively), creating a cycle. □

Assuming that we can efficiently compute $G_s^{\text{Pre}}(c)$, Lemma 1 allows us to detect and deal with the cases of infinitely many c-optimal walks. Moreover, if c is finite, then $G_s^{\text{Pre}}(c)$ is a DAG and, hence, counting walks is easy. Hence, we arrive at the following statement.

Lemma 3 *Let c be a prefix-compatible cost function and $G_s^{\text{Pre}}(c)$ a predecessor graph for a temporal graph $\mathcal{G}$ and a vertex s in $\mathcal{G}$. Given $G_s^{\text{Pre}}(c)$, the number of c-optimal temporal walks from s to any vertex v and any vertex appearance (v, t) in $\mathcal{G}$ can be computed in $O(|G_s^{\text{Pre}}(c)|)$ time.*

Proof: First, run Kosaraju's algorithm [2] on the graph $G_s^{\text{Pre}}(c)$ to compute in linear time the strongly connected components (SCCs) in $G_s^{\text{Pre}}(c)$ while also keeping track of their size. Now consider the DAG $G_s^{\mathrm{SCC}}(c)$ of SCCs, that is, the DAG where every node is an SCC of $G_s^{\mathrm{Pre}}(c)$ and there is an arc from one SCC A to another SCC B if there is an arc from a vertex in A to a vertex in B in $G_s^{\mathrm{Pre}}(c)$. Mark every node in $G_s^{\mathrm{SCC}}(c)$ corresponding to an SCC with more than one vertex with ∞. Then, run a BFS on $G_s^{\mathrm{SCC}}(c)$ starting in each node marked with ∞ and label all the nodes reached during the BFS with ∞. Now, for every vertex appearance (v, t) belonging to an SCC where the corresponding node in $G_s^{\mathrm{SCC}}(c)$ is marked with ∞, we set the number of temporal s-(v, t)-walks as ∞ and then remove it from $G_s^{\mathrm{Pre}}(c)$. The correctness of this step follows from the proof of Lemma 1.

Let G' be the remaining graph. Clearly, G' is a DAG. We next show that counting paths to a vertex in G' will exactly correspond to counting c-optimal temporal walks to a vertex appearance corresponding to that vertex:

We shall prove the statement above by induction on the vertices of G', taken in the topological ordering. First, $(s, 0)$ must clearly be the first vertex in that ordering. Obviously, there is only one c-optimal path to $(s, 0)$, so the computed value will be correct here.

Now, consider a vertex v_i corresponding to some appearance (v, t). By definition, all its direct predecessors v_j come before v_i in the topological ordering in the graph. Since $v_j \in \mathrm{Pre}_s^{\mathcal{W}}(v_i)$, by prefix-exchangeability, every c-optimal walk to v_j can be extended to a c-optimal walk to v_i. Conversely, by prefix-optimality, we are also not missing any c-optimal walks to (v, t). Hence, the computed number of paths to v_i will also be correct.

To compute the number of c-optimal temporal walks *to a vertex* $v \in V$, we can first find $c_s^*(v) = \min_{t \in [T]} c_s^*(v, t)$, and then compute $\sigma_{sv}^{\mathcal{W}} = \sum_{t \mid c_s^*(v,t) = c_s^*(v)} \sigma_{s(v,t)}^{\mathcal{W}}$. This path counting in a DAG is clearly doable in time linear in the size of $G_s^{\mathrm{Pre}}(c)$. □

To employ Lemma 3, we need to compute $G_s^{\mathrm{Pre}}(c)$. To this end, we run a slight variation of the classical Bellman-Ford algorithm [6, 15]. This leads to the following lemma. Recall that we assume here that c can be evaluated in $O(1)$ time.

Lemma 4 *Let c be a prefix-compatible cost function for a temporal graph $\mathcal{G}$. Let s be a vertex in $\mathcal{G}$. Then the predecessor graph $G_s^{\mathrm{Pre}}(c)$ can be computed in $O(nMT^2)$ time.*

Proof: The algorithm is a straightforward generalization of the Bellman-Ford algorithm [6, 15], adapted to our use case, see Algorithm 1.

The correctness can be easily proven by induction on the (maximal) length of an optimal walk (similar as for the classical Bellman-Ford algorithm). Again, as in the proof of Lemma 3, the key observation is that prefix-compatibility ensures that the walks can be extended step by step: By prefix-exchangeability, every c-optimal walk to a vertex appearance (v, t) can be extended to a c-optimal walk to a successor, and thus will be found during looping over all arcs (see Lines 3 and 4 in Algorithm 1). Conversely, by prefix-optimality, we are also not missing any c-optimal walks to (v, t).

As for the running time, it is easy to see that Algorithm 1 runs in $O(nMT^2)$ time. □

Applying Lemmas 3 and 4 starting from each vertex yields the following.

Theorem 1 (Walk counting) *Let c be a prefix-compatible cost function for a temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$. Then the number of c-optimal temporal walks from each vertex $s \in V$ to any vertex appearance (v, t) can be computed in $O(n^2MT^2)$ time.*

Algorithm 1 A generalized Bellman-Ford algorithm for finding the predecessor sets $\mathrm{Pre}_s^{\mathcal{W}}$ for a given source $s \in V$.

Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$, a source vertex $s \in V$, and a cost function c.
Output: Predecessor graph $G_s^{\mathrm{Pre}}(c)$ and the optimal values $c_s^*(v,t)$ for all appearances.

```
1: $\mathrm{Pre}_s^{\mathcal{W}}(v,t) \leftarrow \emptyset$; CUR-BEST$[v,t] \leftarrow \infty$ for all $v \in V, t \in [T]$        ▷ Initialization
2: for $i = 1$ to $nT$ do
3:     for $(v \xrightarrow{t'} w) \in \mathcal{E}$ do
4:         for $1 \le t \le t'$ do
5:             RELAX$((v,t),(w,t'))$
6: return $\mathrm{Pre}_s^{\mathcal{W}}$, CUR-BEST
```

Algorithm 2 Function relaxing the transition from (v,t) to (w,t'). Note that by $\langle v,t\rangle$ we mean an arbitrary temporal s-(v,t)-walk that is implicitly represented by $\mathrm{Pre}_s^{\mathcal{W}}(v,t)$. (By prefix-compatibility, all such walks are effectively interchangeable.)

```
1: function RELAX$((v,t), (w,t'))$
2:     if CUR-BEST$[v,t] = \infty$ then
3:         return
4:     if $c(\langle v,t\rangle \circ (v \xrightarrow{t'} w)) <$ CUR-BEST$[w,t']$ then
5:         $\mathrm{Pre}_s^{\mathcal{W}}(w,t') \leftarrow \emptyset$
6:         CUR-BEST$[w,t'] \leftarrow c(\langle v,t\rangle \circ (v \xrightarrow{t'} w))$
7:     if $c(\langle v,t\rangle \circ (v \xrightarrow{t'} w)) =$ CUR-BEST$[w,t']$ then
8:         $\mathrm{Pre}_s^{\mathcal{W}}(w,t') \leftarrow \mathrm{Pre}_s^{\mathcal{W}}(w,t') \cup \{(v,t)\}$
```

5 Computing temporal betweenness

In this section, we discuss how to compute temporal betweenness centrality efficiently for optimal temporal walks defined by a prefix-compatible cost function. In order to do this, we adapt the machinery of showing a recursive relation of the temporal dependencies [11] to our generalized setting. Together with the fact that we can compute the walk-counts in polynomial time (Theorem 1), we can use a Brandes-like [9] approach to compute the temporal betweenness values in polynomial time.

We prove a general dependency accumulation formula, alongside some of its implications. Note that we essentially adapt the results of **(author?)** [11] to our generalized setting, the proofs are quite similar. In the following, for convenience we use $\mathcal{W}$ (instead of $\mathcal{W}^{(c)}$) to denote a set of c-optimal walks for some arbitrary cost function c. As betweenness centrality is not well-defined for infinite cost functions, we assume here that c is finite (in the sense of Definition 13). Note that we can detect whether c is finite in polynomial time (see Section 4).

First, we define "edge dependency" which we need for the general dependency accumulation lemma.

Definition 14 (Edge dependency) $\delta_{sz}^{\mathcal{W}}(v,t,(v,w,t'))$ *denotes the fraction of temporal s-z-walks in $\mathcal{W}$ that go through the appearance (v,t) and use the temporal arc (v,w,t').*

The next lemma shows us how to compute the edge dependency.

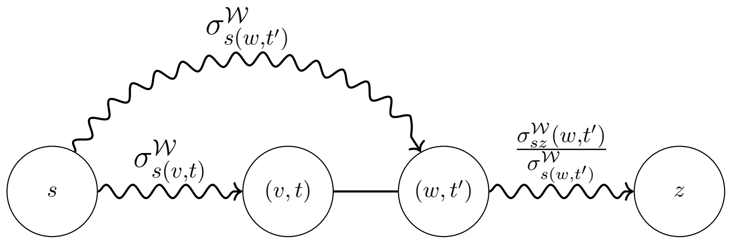

$\sigma^{\mathcal{W}}_{s(w,t')}$
$\sigma^{\mathcal{W}}_{s(v,t)}$
$\frac{\sigma^{\mathcal{W}}_{sz}(w,t')}{\sigma^{\mathcal{W}}_{s(w,t')}}$
s
(v,t)
(w,t')
z

Figure 6: Lemma 5: combining an arbitrary prefix which ends in (v,t) with an arbitrary suffix from (w,t').

Lemma 5 *Let c be a finite prefix-compatible cost function and let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. Let $\mathcal{W}$ be the set of optimal temporal walks in $\mathcal{G}$ induced by c. Fix a source $s \in V$. If $\delta^{\mathcal{W}}_{sz}(v,t,(v,w,t'))$ is positive, then*

$$\delta^{\mathcal{W}}_{sz}(v,t,(v,w,t')) = \frac{\sigma^{\mathcal{W}}_{s(v,t)}}{\sigma^{\mathcal{W}}_{s(w,t')}} \cdot \frac{\sigma^{\mathcal{W}}_{sz}(w,t')}{\sigma^{\mathcal{W}}_{sz}}.$$

Proof: Let W be an arbitrary c-optimal temporal s-z-walk that goes through (v,t) and makes the direct transition $v \xrightarrow{t'} w$. By prefix-optimality, $c(W[\bullet,(v,t)]) = c^*_s(v,t)$ and $c(W[\bullet,(w,t')]) = c^*_s(w,t')$. Now, by prefix-exchangeability, we can substitute an arbitrary c-optimal temporal s-(v,t)-walk for $W[\bullet,(v,t)]$ and still get a valid c-optimal temporal s-(w,t')-walk. Similarly, we can then substitute an arbitrary c-optimal temporal s-(w,t')-walk for $W[\bullet,(w,t')]$ and get a valid c-optimal temporal s-z-walk.

Hence, we can combine an arbitrary prefix which ends in (v,t) with an arbitrary suffix which starts at (w,t') to get a valid temporal s-z-path in $\mathcal{W}$ (see Figure 6). With all of that in mind, we can now make the following argument:

Of the $\sigma^{\mathcal{W}}_{s(w,t')}$ many walk prefixes which end in (w,t'), exactly $\sigma^{\mathcal{W}}_{s(v,t)}$ go through (v,t) and use the transition $v \xrightarrow{t'} w$. Now, there are also $\sigma^{\mathcal{W}}_{sz}(w,t')/\sigma^{\mathcal{W}}_{s(w,t')}$ many unique walk suffixes starting from the appearance (w,t') and going to the vertex z. Thus, there are $\sigma^{\mathcal{W}}_{s(v,t)} \cdot \sigma^{\mathcal{W}}_{sz}(w,t')/\sigma^{\mathcal{W}}_{s(w,t')}$ many temporal s-z-walks that go through (v,t) and use the transition $v \xrightarrow{t'} w$. Finally, we divide by the total number of temporal s-z-walks to get the expression in the statement of the lemma. □

The lemma allows to prove the following general dependency accumulation formula.

Lemma 6 (General dependency accumulation) *Let c be a finite prefix-compatible cost function and let $\mathcal{G} = (V, \mathcal{E}, T)$ be a temporal graph. Let $\mathcal{W}$ be the set of optimal temporal walks in $\mathcal{G}$ induced by c. Fix a source $s \in V$. Then,*

$$\delta^{\mathcal{W}}_{s\bullet}(v,t) = \delta^{\mathcal{W}}_{sv}(v,t) + \sum_{(w,t') \in \mathrm{Succ}^{\mathcal{W}}_s(v,t)} \frac{\sigma^{\mathcal{W}}_{s(v,t)}}{\sigma^{\mathcal{W}}_{s(w,t')}} \cdot \delta^{\mathcal{W}}_{s\bullet}(w,t').$$

Proof: We start by expanding the sum in the formula defining $\delta^{\mathcal{W}}_{s\bullet}(v,t)$, considering its summands a bit more precisely.

$$\begin{aligned}\delta_{s\bullet}^{\mathcal{W}}(v,t) &= \sum_{z\in V} \delta_{sz}^{\mathcal{W}}(v,t)\\ &= \delta_{sv}^{\mathcal{W}}(v,t) + \sum_{z\in V}\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{W}}(v,t)} \delta_{sz}^{\mathcal{W}}(v,t,(v,w,t')),\end{aligned}\tag{1}$$

where $\delta_{sz}^{\mathcal{W}}(v,t,(v,w,t'))$ is as defined in Definition 14. Since (v,t) is not a direct predecessor on any temporal s-(v,t)-walk, we need to pull it out of the sum. Conversely, for any $z \in V$, the vertex appearance (v,t) may be contained at most once in any temporal s-z-walk (by Lemma 2). Hence the inner sum of Equation (1) precisely captures $\delta_{sz}^{\mathcal{W}}(v,t)$.

We can now use Lemma 5 to simplify the summation formula:

$$\begin{aligned}&\sum_{z\in V}\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{W}}(v,t)} \delta_{sz}^{\mathcal{W}}(v,t,(v,w,t'))\\ \overset{\textit{Lemma 5}}{=}\ &\sum_{z\in V}\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{W}}(v,t)} \frac{\sigma_{s(v,t)}^{\mathcal{W}}}{\sigma_{s(w,t')}^{\mathcal{W}}}\cdot\frac{\sigma_{sz}^{\mathcal{W}}(w,t')}{\sigma_{sz}^{\mathcal{W}}}\\ =\ &\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{W}}(v,t)}\sum_{z\in V} \frac{\sigma_{s(v,t)}^{\mathcal{W}}}{\sigma_{s(w,t')}^{\mathcal{W}}}\cdot\frac{\sigma_{sz}^{\mathcal{W}}(w,t')}{\sigma_{sz}^{\mathcal{W}}}\\ =\ &\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{W}}(v,t)} \frac{\sigma_{s(v,t)}^{\mathcal{W}}}{\sigma_{s(w,t')}^{\mathcal{W}}}\sum_{z\in V}\frac{\sigma_{sz}^{\mathcal{W}}(w,t')}{\sigma_{sz}^{\mathcal{W}}}\\ \overset{\textit{Definition 7}}{=}\ &\sum_{(w,t')\in \mathrm{Succ}_s^{\mathcal{W}}(v,t)} \frac{\sigma_{s(v,t)}^{\mathcal{W}}}{\sigma_{s(w,t')}^{\mathcal{W}}}\cdot \delta_{s\bullet}^{\mathcal{W}}(w,t'),\end{aligned}$$

from which the result immediately follows. □

We now combine Lemma 6 with the results from Section 4 to show how temporal betweenness centrality can be computed for all finite prefix-compatible cost functions.

Lemma 7 *Let c be a finite prefix-compatible cost function for a temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$. Given $G_s^{\mathrm{Pre}}(c)$ for each $s \in V$, the temporal betweenness centrality of all vertices in $\mathcal{G}$ can be computed in $O(\sum_{s\in V} |G_s^{\mathrm{Pre}}(c)| + nM)$ time.*

Proof: Consider Algorithm 3. Count-Walks counts optimal temporal walks in a manner described by Lemma 3. We first prove the correctness of the algorithm before analyzing its running time. Denote with $\mathcal{W}$ the set of optimal temporal walks in $\mathcal{G}$ induced by c.

Correctness: The general idea of the algorithm is to use Lemma 6 to implicitly compute the total temporal betweenness centrality and use Observation 2 to recover the $C_B^{\mathcal{W}}$ values. As the equation in Observation 2 has the constant summand +1, in Line 3 we initialize the temporal betweenness values to 1.

The next step is to compute the cumulative dependencies $\delta_{s\bullet}^{\mathcal{W}}(v,t)$ for each source vertex s and appropriately update the temporal betweenness values. We do this in the loop starting on Line 4. We first initialize the array holding the cumulative dependencies on Line 5.

Algorithm 3 General betweenness algorithm, see Lemma 7. It uses an auxiliary function COUNT-WALKS that computes the walk counts in a manner described by Lemma 3. Note that the predecessor sets $\mathrm{Pre}_s^{\mathcal{W}}$ are encoded in the predecessor graphs $G_s^{\mathrm{Pre}}(c)$.

```
Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, T)$, the predecessor graphs $G_s^{\mathrm{Pre}}(c)$ for all $s \in V$.
Output: Betweenness $C_B^{\mathcal{W}}(v)$ of all vertices $v \in V(\mathcal{G})$.
1: $\sigma_{sv}^{\mathcal{W}}, \sigma_{s(v,t)}^{\mathcal{W}}, \delta_{sv}^{\mathcal{W}}(v,t) \leftarrow$ COUNT-WALKS($G_s^{\mathrm{Pre}}(c)$)    ▷ Lemma 3
2: for $v \in V$ do
3:     $C_B^{\mathcal{W}}[v] \leftarrow 1$    ▷ Initialize to 1 per Observation 2
4: for $s \in V$ do
5:     for $(u, v, t) \in \mathcal{E}$ do
6:         $\delta_{s\bullet}^{\mathcal{W}}[v,t] \leftarrow 0$    ▷ Reset the array
7:     for $(w, t') \in R$ in topological order determined by $\mathrm{Pre}_s^{\mathcal{W}}$ do
8:         $\delta_{s\bullet}^{\mathcal{W}}(w,t') \leftarrow \delta_{s\bullet}^{\mathcal{W}}(w,t') + \delta_{sw}^{\mathcal{W}}(w,t')$    ▷ Appearance dependency on $(w,t')$
9:         for $(v,t) \in \mathrm{Pre}_s^{\mathcal{W}}(w,t')$ do
10:            $\delta_{s\bullet}^{\mathcal{W}}[v,t] \leftarrow \delta_{s\bullet}^{\mathcal{W}}[v,t] + (\sigma_{s(v,t)}^{\mathcal{W}} / \sigma_{s(w,t')}^{\mathcal{W}}) \cdot \delta_{s\bullet}^{\mathcal{W}}[w,t']$    ▷ Sum of Lemma 6
11:            $C_B^{\mathcal{W}}[v] \leftarrow C_B^{\mathcal{W}}[v] + (\sigma_{s(v,t)}^{\mathcal{W}} / \sigma_{s(w,t')}^{\mathcal{W}}) \cdot \delta_{s\bullet}^{\mathcal{W}}[w,t']$
12:    $C_B^{\mathcal{W}}[s] \leftarrow C_B^{\mathcal{W}}[s] - |\{v \mid \exists t \in [T] : \delta_{sv}^{\mathcal{W}}(v,t) > 0\}|$    ▷ Connectivity correction
13: return $C_B^{\mathcal{W}}$
```

Finally, in the loop starting on Line 7 we compute the cumulative dependencies using the recursive formula of Lemma 6. We proceed in reverse topological order, that is, we start with vertices that have no successors (and hence for which the equation in Lemma 6 is trivial to evaluate) as our base case and then proceed backwards. This is possible as finite prefix-compatible cost functions always lead to acyclic predecessor graphs (see Lemma 1).

Finally, on Line 12 we apply a "connectivity correction." This term corresponds to the $\sum_{w \in V} \left[\sigma_{vw}^{\mathcal{W}} > 0\right]_{\mathbb{1}}$ part of the equation in Observation 2. Notice that we do not have to handle the term with $\left[\sigma_{wv}^{\mathcal{W}} > 0\right]_{\mathbb{1}}$ as on Line 11 we only add terms which correspond to the sum of the equation of Lemma 6, and never the $\delta_{sv}^{\mathcal{W}}(v,t)$ terms. (We do, however, add those terms to the $\delta_{s\bullet}^{\mathcal{W}}(v,t)$ values which then propagate into the $C_B^{\mathcal{W}}$ array, which is why we need the correction on Line 12.)

Running time: By Lemma 3, COUNT-WALKS can be computed in $O(|G_s^{\mathrm{Pre}}(c)|)$ time for each $s \in V$. We can easily see that for the loop on Line 5 we need $O(M)$ time overall.

Before the execution of the for-loop on Line 7, we may first need to compute the topological order on the predecessor graph. This can be done in time linear in the size of the graph. The sum of the sizes of the graphs for all sources is $\sum_{s \in V} |G_s^{\mathrm{Pre}}(c)|$. We also note that the for-loop on Line 7 goes over each arc of the predecessor graph exactly once, so we get an $O(\sum_{s \in V} |G_s^{\mathrm{Pre}}(c)|)$ bound for the overall execution time.

Finally, with some bookkeeping done in the loop of Line 7, the connectivity correction of Line 12 can easily be made to run in constant time.

Altogether, we get $O(\sum_{s \in V} |G_s^{\mathrm{Pre}}(c)| + nM)$ for the total running time. ☐

Using the running time bound from Lemma 4 together with Lemma 7, we immediately get our main result of this work.

Theorem 2 (General betweenness computation) *Let c be a finite prefix-compatible cost function. Then the betweenness centrality of all vertices can be computed in $O(n^2MT^2)$ time.*

Combining Proposition 1 and Theorem 2 yields the following corollary. Note that for all of the optimality concepts mentioned in the corollary Algorithm 1 can easily be implemented so that c can evaluated in amortized constant time, hence our results apply directly without any additional multiplicative factors in the running time.

Corollary 1 *The betweenness centrality of all vertices in a temporal graph can be computed in $O(n^2MT^2)$ time with respect to*

- *foremost temporal walk,*
- *shortest temporal path,*
- *shortest fastest temporal path,*
- *shortest restless temporal walks, and*
- *strict prefix-foremost temporal path.*

We remark that, while foremost temporal walk, shortest temporal path, shortest fastest temporal paths, and strict prefix-foremost temporal path were known from previous work [1, 11, 22, 28], this is a new classification for shortest restless temporal walks.

6 Conclusion

The very nature of this work is conceptual. It goes without saying that to achieve improved efficiency, exploiting specific properties of the various temporal path and walk concepts may clearly allow for further improved polynomial running times. As to future research, we wonder whether our concept of prefix-compatibility may finally lead to a full characterization of polynomial-time computable temporal betweenness centrality values. It would also be interesting to investigate whether our concept is expressible as a property of so-called graph dioids [14, 18, 23]. As to the computationally hard cases (but not only them), for high efficiency in practice, one might also explore the possibilities of efficient data reductions or approximation algorithms. This proved useful in the static graphs case, with respect to data reduction [5, 7, 24, 27] as well as with respect to approximation [4, 17, 25].

Acknowledgements

The authors wish to thank anonymous reviewers of JGAA for their valuable feedback.

In memory of Rolf Niedermeier, our colleague, friend, and mentor, who sadly passed away before this paper was published.

References

[1] A. Afrasiabi Rad, P. Flocchini, and J. Gaudet. Computation and analysis of temporal betweenness in a knowledge mobilization network. *Computational Social Networks*, 4(1):5, 2017. doi:10.1186/s40649-017-0041-7.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[3] A. Alsayed and D. J. Higham. Betweenness in time dependent networks. *Chaos, Solitons & Fractals*, 72:35–48, 2015. doi:10.1016/j.chaos.2014.12.009.

[4] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *Proceedings of the 5th International Workshop on Algorithms and Models for the Web-Graph (WAW '07)*, pages 124–137. Springer, 2007. doi:10.1007/978-3-540-77004-6_10.

[5] M. Baglioni, F. Geraci, M. Pellegrini, and E. Lastres. Fast exact computation of betweenness centrality in social networks. In *Proceedings of the 4th International Conference on Advances in Social Networks Analysis and Mining (ASONAM '12)*, pages 450–456. IEEE Computer Society, 2012.

[6] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[7] M. Bentert, A. Dittmann, L. Kellerhals, A. Nichterlein, and R. Niedermeier. An adaptive version of Brandes' algorithm for betweenness centrality. *Journal of Graph Algorithms and Applications*, 24(3):483–522, 2020. doi:10.7155/jgaa.00543.

[8] M. Bentert, A.-S. Himmel, A. Nichterlein, and R. Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Applied Network Science*, 5(1):73, 2020. doi:10.1007/s41109-020-00311-0.

[9] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001. doi:10.1080/0022250X.2001.9990249.

[10] B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003. doi:10.1142/S0129054103001728.

[11] S. Buß, H. Molter, R. Niedermeier, and M. Rymar. Algorithmic aspects of temporal betweenness. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*, pages 2084–2092. Association for Computing Machinery, 2020. doi:10.1145/3394486.3403259.

[12] A. Casteigts, A. Himmel, H. Molter, and P. Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9):2754–2802, 2021. doi:10.1007/s00453-021-00831-w.

[13] J. A. Enright, K. Meeks, and H. Molter. Counting temporal paths. *CoRR*, abs/2202.12055, 2022. URL: https://arxiv.org/abs/2202.12055.

[14] L. Falzon, E. Quintane, J. Dunn, and G. Robins. Embedding time in positions: Temporal measures of centrality for social network analysis. *Social Networks*, 54:168–178, 2018. doi:10.1016/j.socnet.2018.02.002.

[15] L. R. Ford Jr. Network flow theory. Technical report, Rand Corp Santa Monica Ca, 1956.

[16] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. doi:10.2307/3033543.

[17] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In *Proceedings of the 10th Meeting on Algorithm Engineering & Expermiments (ALENEX '08)*, pages 90–100. SIAM, 2008. doi:10.1137/1.9781611972887.9.

[18] M. Gondran and M. Minoux. *Graphs, dioids and semirings: new models and algorithms*, volume 41. Springer Science & Business Media, 2008.

[19] V. M. Gunturi, S. Shekhar, K. Joseph, and K. M. Carley. Scalable computational techniques for centrality metrics on temporally detailed social network. *Machine Learning*, 106(8):1133–1169, 2017. doi:10.1007/s10994-016-5583-7.

[20] Habiba, C. Tantipathananandh, and T. Y. Berger-Wolf. Betweenness centrality measure in dynamic networks. Technical Report 19, Department of Computer Science, University of Illinois at Chicago, Chicago, 2007. DIMACS Technical Report.

[21] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. doi:10.1006/jcss.2002.1829.

[22] H. Kim and R. Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012. doi:10.1103/PhysRevE.85.026107.

[23] N. Kontoleon, L. Falzon, and P. Pattison. Algebraic structures for dynamic networks. *Journal of Mathematical Psychology*, 57(6):310–319, 2013. doi:10.1016/j.jmp.2013.11.002.

[24] R. Puzis, Y. Elovici, P. Zilberman, S. Dolev, and U. Brandes. Topology manipulations for speeding betweenness centrality computation. *Journal of Complex Networks*, 3(1):84–112, 2015. doi:10.1109/ASONAM.2012.79.

[25] M. Riondato and E. M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2016. doi:10.1007/s10618-015-0423-0.

[26] M. Rymar, H. Molter, A. Nichterlein, and R. Niedermeier. Towards classifying the polynomial-time solvability of temporal betweenness centrality. In L. Kowalik, M. Pilipczuk, and P. Rzazewski, editors, *Graph-Theoretic Concepts in Computer Science - 47th International Workshop, WG 2021, Warsaw, Poland, June 23-25, 2021, Revised Selected Papers*, volume 12911 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2021. doi:10.1007/978-3-030-86838-3_17.

[27] A. E. Sariyüce, K. Kaya, E. Saule, and Ü. V. Çatalyürek. Graph manipulations for fast centrality computation. *ACM Transactions on Knowledge Discovery from Data*, 11(3):26:1–26:25, 2017. doi:10.1145/3022668.

[28] F. Simard, C. Magnien, and M. Latapy. Computing betweenness centrality in link streams. *CoRR*, abs/2102.06543, 2021. URL: https://arxiv.org/abs/2102.06543.

[29] J. Tang, M. Musolesi, C. Mascolo, V. Latora, and V. Nicosia. Analysing information flows and key mediators through temporal centrality metrics. In *Proceedings of the 3rd Workshop on Social Network Systems (SNS '10)*. Association for Computing Machinery, 2010. doi:10.1145/1852658.1852661.

[30] I. Tsalouchidou, R. Baeza-Yates, F. Bonchi, K. Liao, and T. Sellis. Temporal betweenness centrality in dynamic graphs. *International Journal of Data Science and Analytics*, 9(3):257–272, 2020. doi:10.1007/s41060-019-00189-x.

[31] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.

[32] M. J. Williams and M. Musolesi. Spatio-temporal networks: reachability, centrality and robustness. *Royal Society Open Science*, 3(6), 2016. doi:10.1098/rsos.160196.

[33] H. Wu, J. Cheng, Y. Ke, S. Huang, Y. Huang, and H. Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016. doi:10.1109/TKDE.2016.2594065.

[34] P. Zschoche, T. Fluschnik, H. Molter, and R. Niedermeier. The complexity of finding separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020. doi:10.1016/j.jcss.2019.07.006.