Foremost, Fastest, Shortest: Temporal Graph Realization Under Various Path Metrics

Justine Cauvi
École Normale Supérieure de Lyon, France
Inria, DI ENS, Paris, France

Nils Morawietz
LaBRI, Université de Bordeaux, Talence, France
Institute of Computer Science, Friedrich Schiller University Jena, Germany

Laurent Viennot
Inria, DI ENS, Paris, France

Abstract

In this work, we follow the current trend on temporal graph realization, where one is given a property P and the goal is to determine whether there is a temporal graph, that is, a graph where the edge set changes over time, with property P. We consider the problems where the given property P is a prescribed matrix for the duration, length, or earliest arrival time of pairwise temporal paths. This means that we are given a matrix D and ask whether there is a temporal graph such that for any ordered pair of vertices (s, t), $D_{s,t}$ equals the duration (length, or earliest arrival time, respectively) of any temporal path from s to t minimizing that specific temporal path metric. For shortest and earliest arrival temporal paths, we are the first to consider these problems as far as we know. We analyze these problems for many settings such as: strict and non-strict paths, periodic and non-periodic temporal graphs, and limited number of labels per edge (limited number of occurrences per edge over time). In contrast to all other path metrics, we show that for the earliest arrival times, we can achieve polynomial-time algorithms in periodic and non-periodic temporal graphs and for strict and and non-strict paths. However, the problem becomes NP-hard when the matrix does not contain a single integer but a set or range of possible allowed values. As we show, the problem can still be solved efficiently in this scenario, when the number of entries with more than one value is small, that is, we develop an FPT-algorithm for the number of such entries. For the setting of fastest paths, we achieve new hardness results that answers an open question by Klobas, Mertzios, Molter, and Spirakis [Theor. Comput. Sci. '25] about the parameterized complexity of the problem with respect to the vertex cover number and significantly improves over a previous hardness result for the feedback vertex set number. When considering shortest paths, we show that the periodic versions are polynomial-time solvable whereas the non-periodic versions become NP-hard.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases network design, temporal paths, foremost paths, fastest paths, shortest paths, non-strict paths, periodic temporal graphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.24

Related Version *Full Version*: https://arxiv.org/abs/2510.01702

Funding Supported by the French ANR, projects ANR-22-CE48-0001 (TEMPOGRAL) and ANR-24-CE48-4377 (GODASse).

Due to space constraints, proofs of results marked with ⋆ are (partially) deferred to the full version accessible at https://arxiv.org/abs/2510.01702.

1 Introduction

Graph realization problems have been studied since the 1960s and consists of finding a static graph that satisfies a desired property P or answering that no such graph exists. The earliest example of such a problem is the case of degree sequence realization, where one is given a non-decreasing sequence $(d_1, \ldots, d_n)$ of natural numbers, and one asks whether there is an undirected graph G with vertex set $\{1, \ldots, n\}$, such that vertex i has degree exactly d_i. This problem was introduced by Erdős and Gallai [9] and generalizations of it remain the object of active study (see, e.g., [2] when ranges are given for each degree). In another early graph realization problem by Hakimi and Yau [13], one is given an $n \times n$ distance matrix D and asks whether there is a static graph on n vertices where, for each ordered pair (i, j) of vertices, the shortest path from i to j has length exactly $D_{i,j}$. Since the introduction of these early problems, graph realization problems were considered in many variations and for many other desirable properties to realize. Recently motivated by the realization problem for distance matrices on static graphs, Klobas, Mertzios, Molter and Spirakis [16] lifted graph realization problems to the realm of temporal graphs. Here, a *temporal graph* $\mathcal{G}$ is a finite sequence $(G_1, \ldots, G_L)$ of static graphs that are all defined over the same vertex set. Temporal graphs are a valuable tool to model and analyze the behavior of real world dynamic networks [4]. In the problem introduced by Klobas et al. [16], the goal is to decide whether a desired property is fulfilled by some temporal graph. They introduced the following problem:

Fastest-path TGR:
Input: An $n \times n$ distance matrix D.
Question: Is there a temporal graph $\mathcal{G}$ with n vertices such that for any ordered pair (s, t) of vertices, the fastest temporal path from s to t has duration $D_{s,t}$?

Recall that a (strict) temporal path can start at any time step and is allowed to traverse at most one edge per time step, and its duration is the difference between the starting time and the arrival time, and that a fastest temporal path is a temporal path with minimum duration (see Section 2). More precisely, the authors considered this problem[1] where only a single label per edge is allowed and where the temporal graph is periodic, that is, where the same edges repeat every Δ time steps for some period $\Delta > 0$. They showed that the this version of the problem is NP-hard and they exhibited an FPT algorithm parameterized by the feedback edge number of the *underlying graph*, that is, the uniquely defined graph that contains an edge $\{s, t\}$ if and only if $D_{s,t} = D_{t,s} = 1$. In contrast, they showed that this version of the problem is W[1]-hard when parameterized by the feedback vertex set number of the underlying graph. Erlebach, Morawietz and Wolf [11] generalized the problem by allowing each edge to appear up to ℓ times per period and proved that this remains NP-hard even on underlying graphs that are trees for $\ell = 5$. In [19], the authors studied the problem where upper bounds on the fastest paths are given and the underlying graph is a tree. This was further considered for directed graphs by Meusel, Müller-Hannemann and Reinhardt [21]. Further recent papers on temporal graph realization include: designing a temporal graph for which the fastest path should not have duration more than α times the real distance [20], designing a temporal graph which should have a prescribed reachability relation between the vertices [10], and designing a temporal graph for which all pairs of agents can pairwise reach each other via strict temporal paths, with one label per edge, while the degree sequence of the underlying graph is prescribed [3]. It is worth mentioning that Göbel, Cerdeira and

[1] They analyzed the problem under the name Simple (Periodic) Temporal Graph Realization.

Veldman [12] considered a connectivity problem that can also be seen as a temporal graph realization problem. All these problems are motivated both from a design perspective, where we aim to design a network with a desired behavior, or from a verification perspective, where we want to verify that the behavior of our network is correct or at least plausible. From the perspective of temporal network design problems, the field is even more vibrant (see, e.g., [1, 7, 8, 14, 15, 17, 18, 23]).

Our Results. In this work, we extend the previous work on Fastest-path TGR to the non-strict case (that is, where arbitrary many edges per time step can be traversed). Furthermore, we consider the other most frequently used temporal path metrics Foremost and Shortest (formally defined in Section 2). Roughly speaking, Foremost-path TGR requires that the earliest arrival time at t, minimized over all temporal st-paths, is equal to $D_{s,t}$ for each vertex pair (s,t), whereas Shortest-path TGR requires that the number of edges, minimized over all temporal st-paths, is equal to $D_{s,t}$ for each vertex pair (s,t). Our main results are as follows:

1. In Section 3 we show that all considered versions (strict/non-strict, periodic/non-periodic) of Foremost-path TGR are polynomial-time solvable if we are allowed to put an arbitrary number of labels on each edge. This is surprising, as almost all other previously mentioned temporal graph realization problems turn out to be NP-hard (with one exception [3]). In particular, we show that all our algorithms produce a labeling with at most n^2 time labels in total, if dealing with a realizable instance. This is asymptotically tight as some realizable matrices do require $\Omega(n^2)$ time labels.
2. To show the limitations of the tractability, we also show in Section 3 that Foremost-path TGR becomes NP-hard if (i) we are only allowed to assign one label per edge or (ii) the matrix D contains more than one entry for each vertex pair, and we are to choose which of these possible values we want to realize. For the latter problem version, we present a single exponential FPT-algorithm when parameterized by the number of entries in D that have more than one possible value.
3. In Section 4 we consider Fastest-path TGR. Among other results, we answer open questions by Klobas et al. [16] and Erlebach et al. [11] about the parameterized complexity of the problem for the vertex cover number. We show that the problem is W[1]-hard when parameterized by this parameter plus the largest entry in the matrix. This result improves significantly over the previous known parameterized hardness result for the feedback vertex set number; in terms of the parameter, the construction, and the length of the proof.
4. In Section 5 we consider Shortest-path TGR and show that the problem is NP-hard for both the strict and the non-strict case, but becomes trivial when considering a periodic temporal graph.

Finally, in Section 6, we conclude with some open questions for future work.

2 Preliminaries

For natural numbers $1 \leq i \leq j$, we let $[i,j] := \{i, i+1, \dots, j\}$ and we define $[j] := [1,j]$.

(Static) graphs. An (undirected) *graph* $G = (V,E)$ is defined by its vertex set V and its edge set $E \subseteq \binom{V}{2}$. Any pair $\{u,v\} \in E$ is called an *edge* between vertices u and v. We also say that u and v are *neighbors* when $\{u,v\} \in E$. For a vertex set $S \subseteq V$, we define the *subgraph of G induced by S* as $G[S] := (S, E \cap \binom{S}{2})$. A graph $H = (V', E')$ is a *subgraph* of

G if $V' \subseteq V$ and $E' \subseteq E$. A *uv-walk* is a sequence $P = (v_0 = u, v_1, \ldots, v_k = v)$ of vertices such that $\{v_{i-1}, v_i\}$ is an edge for every $i \in [k]$. We then say that P is a walk from u to v in G. Such a walk is called a path if the vertices $v_0, \ldots, v_k$ are pairwise distinct.

Temporal graphs. A *temporal graph* is defined by a pair $\mathcal{G} = (G, \lambda)$ where $G = (V, E)$ is a graph, and $\lambda : E \to 2^{\mathbb{N}_{>0}}$ is a labeling that associates to each edge $e \in E$ the set $\lambda(e)$ of (positive) times when e appears. The graph G is called the *underlying graph* of $\mathcal{G}$ and the labeling λ is called the *time labeling* of $\mathcal{G}$. When $\mathcal{G}$ is clear from the context, we let $n = |V|$ denote the number of vertices of $\mathcal{G}$. A *time-edge* is a pair $(\{u, v\}, \tau)$ such that $\{u, v\} \in E$ and $\tau \in \lambda(\{u, v\})$. Its *appearance time* is τ. Given a time label $\tau \in \mathbb{N}_{>0}$, we define the set $E_\tau = \{e \in E : \tau \in \lambda(e)\}$ of edges appearing at time τ, and call $G_\tau = (V, E_\tau)$ the *snapshot* of $\mathcal{G}$ at time τ. The *size* of a temporal graph can be measured by its number $\sum_{e \in E} |\lambda(e)|$ of time labels. A temporal graph is said to be Δ-*periodic* if each edge appears periodically with period $\Delta \in \mathbb{N}_{>0}$, that is $\tau \in \lambda(e)$ if and only if $\tau \mod \Delta \in \lambda(e)$. Such a temporal graph is represented by its list of time-edges up to time Δ. In the following, we always assume that the vertices of a temporal graph $\mathcal{G}$ are numbered from 1 to n. We assume without loss of generality that its vertex set is $V = [n]$. We let $\mathcal{TG}$ denote the set of all such temporal graphs.

Temporal paths. A *strict* (respectively *non-strict*) *temporal uv-walk* is a walk $P = (v_0 = u, v_1, \ldots, v_k = v)$ in G with associated time labels $(\tau_1, \ldots, \tau_k)$ such that $\tau_i \in \lambda(\{v_{i-1}, v_i\})$ for each $i \in [k]$ and $\tau_1 < \cdots < \tau_k$ (respectively $\tau_1 \leq \cdots \leq \tau_k$). Equivalently, a temporal walk can be defined as the sequence of time-edges $(\{v_0, v_1\}, \tau_1), \ldots, (\{v_{k-1}, v_k\}, \tau_k)$ satisfying $\tau_1 < \cdots < \tau_k$ (respectively $\tau_1 \leq \cdots \leq \tau_k$). Moreover, P has *length* k, *departure time* τ_1, *arrival time* τ_k, and *duration* $\tau_k - \tau_1 + 1$ (number of time steps spanned). A temporal uv-walk is called a *temporal uv-path* if the vertices $v_0, \ldots, v_k$ are pairwise distinct. Note that a strict (respectively non-strict) temporal walk can always be transformed into a strict (respectively non-strict) temporal path by removing loops, and this can only reduce length, arrival time and duration. By default, we consider strict temporal paths, and simply call them temporal paths. We specify non-strict for non-strict temporal paths.

Temporal path metrics. Classically, a temporal uv-path is said to be *shortest*, *foremost*, or *fastest* if it has minimum number of edges, minimum arrival time, or minimum duration, respectively among all temporal uv-paths. These notions indeed define some kinds of metrics that we now formalize. A *distance matrix* D is any matrix of size $n \times n$ with values in $\mathbb{N} \cup \{\infty\}$ and that satisfies the following very loose notion of metric: $D_{uv} = 0$ if and only if $u = v$, for all $u, v \in [n]$. It is not assumed that D satisfies any other specific properties. In particular, D may violate the triangle inequality. A *temporal path metric* is defined as a function M that associates a distance matrix $M(\mathcal{G})$ to any temporal graph $\mathcal{G} \in \mathcal{TG}$. Consider for example, a *cost function* C that associates a positive cost in $\mathbb{N}_{>0}$ to any temporal path given as a sequence of time-edges (independently of any temporal graph). It defines a *temporal path metric* by associating to any temporal graph $\mathcal{G} \in \mathcal{TG}$ the matrix $C(\mathcal{G})$ (respectively NS-$C(\mathcal{G})$) such that $C(\mathcal{G})_{uv}$ (respectively NS-$C(\mathcal{G})_{uv}$) is the minimum cost of a strict (respectively non-strict) temporal uv-path in $\mathcal{G}$. We define $C(\mathcal{G})_{uv} := \infty$ (respectively NS-$C(\mathcal{G})_{uv} := \infty$) if no strict (respectively non-strict) temporal uv-path exists and $C(\mathcal{G})_{uv} := 0$ (respectively NS-$C(\mathcal{G})_{uv} := 0$) if $u = v$. The foremost, fastest, and shortest notions are indeed associated to the following cost functions: arrival time, duration, and length, respectively. We let Foremost, Fastest, and Shortest denote the corresponding temporal path metrics, respectively. We

also let NS-Foremost, NS-Fastest, and NS-Shortest denote the respective variants for non-strict temporal paths. For example, given a temporal graph $\mathcal{G}$, and two vertices u, v in $\mathcal{G}$, Foremost$(\mathcal{G})_{uv}$ is the earliest arrival time of a strict temporal uv-path in $\mathcal{G}$.

Temporal graph realization. Given an integer n, the temporal graph realization problem consists of finding a temporal graph with n vertices that satisfies a given property. We assume that this property can be expressed as an input sequence $I \in \{0,1\}^*$ of bits, given that the vertices of the temporal graph are $1, \dots, n$. More precisely, we define a *predicate* P as a binary relation between the set $\mathcal{TG}$ of all temporal graphs and the set $\{0,1\}^*$ of all bit sequences, that is P is a subset of $\mathcal{TG} \times \{0,1\}^*$. We then say that a temporal graph $\mathcal{G}$ *satisfies* a sequence I of bits for P if $(\mathcal{G}, I) \in P$. We equivalently say that $P(\mathcal{G}, I)$ is *satisfied*, or that $\mathcal{G}$ is a *realization* of I for P. As a simple example, the *lifetime* of a temporal graph can be tested by the predicate $P(\mathcal{G}, \Lambda) := \max(\cup_{e \in E} \lambda(e)) = \Lambda$ where $\mathcal{G} = (G, \lambda)$ and Λ encodes an integer (that we denote also by Λ with a slight abuse of notation). That is $P(\mathcal{G}, \Lambda)$ is satisfied when Λ is the last appearance time of a time-edge of $\mathcal{G}$. Given a predicate P we thus define the following (very general) problem.

P Temporal Graph Realization (P TGR):
Input: A number n and a sequence I of bits.
Question: Is there a temporal graph $\mathcal{G}$ with n vertices such that $P(\mathcal{G}, I)$ is satisfied?

This paper focuses on temporal path metric realization. More precisely, considering a temporal path metric $M \in \{\text{Foremost}, \text{Fastest}, \text{Shortest}, \text{NS-Foremost}, \text{NS-Fastest}, \text{NS-Shortest}\}$, we define the *$M$-temporal-path-metric* predicate, or *M-path* for short, as $P(\mathcal{G}, I) := M(\mathcal{G}) = D$ where the sequence I of bits encodes an $n \times n$ distance matrix D where n is the number of vertices of $\mathcal{G}$. For example, Foremost-path TGR is the following problem.

Foremost-path TGR:
Input: A number n and an $n \times n$ distance matrix D.
Question: Is there a temporal graph $\mathcal{G}$ with n vertices such that Foremost$(\mathcal{G}) = D$?

When considering an instance (n, I) of P TGR, we always let n denote the associated number. Note that, in the case of an M-path TGR instance, the input has size $\Theta(n^2)$. For brevity, given an input sequence I encoding a distance matrix D, a realization $\mathcal{G}$ of I for M-path is simply called an *M-realization* of D. If the metric M is clear from the context, we simply say that $\mathcal{G}$ is a *realization* of D or that $\mathcal{G}$ *realizes* D.

We also combine M-path predicates with additional requirements. In particular, when the input I either encodes a (static) graph G_p with n nodes, or a period Δ, we define the following additional predicates, respectively:

- Prescribed$(\mathcal{G}, G_p)$:= the underlying graph of $\mathcal{G}$ is a subgraph of G_p,
- Periodic$(\mathcal{G}, \Delta)$:= $\mathcal{G}$ is Δ-periodic.

Given two predicates P, Q, the problem "P Q TGR" asks whether there exists a temporal graph satisfying both P and Q for a given pair of inputs for P and Q. For example, Periodic Foremost-path TGR asks, given an $n \times n$ distance matrix D and a period Δ, whether there exists a Δ-periodic temporal graph that is a Foremost-realization of D. Similarly, Prescribed Foremost-path TGR supposes that, in addition to D, the input includes a static graph G_p called the *prescribed* graph, and asks whether there exists a Foremost-realization of D whose underlying graph is a subgraph of G_p. We also sometimes refer to "P Q TGR" as "Q TGR where P is required", especially if P is specified in plain text without defining a formal name for it.

3 Foremost paths

We first consider the Foremost-PATH TGR problem. Recall that, given an $n \times n$ distance matrix D, it consists of checking whether there exists a temporal graph $\mathcal{G}$ whose foremost matrix is D, i.e. Foremost$(\mathcal{G}) = D$. We also consider its non-strict variant (NS-Foremost-PATH TGR), and combining both with Prescribed and Periodic additional requirements. Similar results can be obtained for latest departure using a time-reversal argument. We often implicitly assume that we are given an $n \times n$ distance matrix D.

We first show that these main variants of the problem can be solved in polynomial time.

3.1 Polynomial time algorithms

Strict Foremost paths

In the strict setting, we can indeed state the following.

▶ **Theorem 3.1.** *FOREMOST-PATH TGR can be solved in $\mathcal{O}(n^3 \log n)$ time and $\mathcal{O}(n^2)$ space. Furthermore, if dealing with a realizable instance, a realization with at most n^2 time labels can be computed with the same complexity.*

The proof mainly comes from an algorithm given hereafter. First note that the above complexity can be expressed as $\widetilde{\mathcal{O}}(N^{3/2})$ where $N = \Theta(n^2)$ is the size of the input. We also state that the above result is asymptotically tight in terms of the number of time labels as follows.

▶ **Proposition 3.2.** *There exists a family $(D^n)_{n\in\mathbb{N}_{>0}}$ of $n \times n$ distance matrices that are* Foremost*-realizable where each realization requires $\Omega(n^2)$ time labels.*

Proof. Consider the temporal graph $\mathcal{G}^n$ with n vertices, whose underlying graph is a star rooted at 1, and where, for all $v > 1$, edge $\{1, v\}$ appears at times $nv, n(v+1)+v, n(v+2)+v, \ldots, n^2+v$. Its foremost matrix $D^n = \text{Foremost}(\mathcal{G}^n)$ then satisfies $D^n_{uv} = nu + v$ for $u > v > 1$, $D^n_{uv} = nv$ for $1 < u < v$, and $D^n_{1v} = D^n_{v1} = nv$ for $1 < v$. These temporal graphs thus define a family $(D^n)_{n\in\mathbb{N}_{>0}}$ of distance matrices that are foremost realizable and that have $\Omega(n^2)$ pairwise distinct entries. This implies that any foremost realization of such a matrix D^n must have $\Omega(n^2)$ time labels. The reason is that any foremost realization $\mathcal{G}'$ of an $n \times n$ distance matrix D containing p pairwise distinct entries must have at least p time labels. Indeed, any entry D_{uv} must correspond to some foremost temporal uv-path in $\mathcal{G}'$ whose last edge appears at time D_{uv}. ◀

▶ Remark. The above lower bound also applies to non-strict foremost realizations (using a similar proof with the same families of matrices and temporal graphs). It also holds if we restrict the problem to sparse prescribed graphs with $O(n)$ edges, or even trees, as long as the star is a possible prescribed graph.

We now present a simple algorithm for computing a Foremost-realization that leads to $\mathcal{O}(n^4)$ time complexity; this bound will be improved subsequently. It is based on the observation that any realization $\mathcal{G}$ of a distance matrix D for Foremost-path must satisfy the following compatibility property.

▶ **Definition 3.3** (Edge compatibility)**.** *Given an $n \times n$ distance matrix D, a time-edge $(\{v, w\}, \tau)$ is said to be* Foremost-edge-compatible *with D if it satisfies: $EdgeCompat(D, \{v, w\}, \tau) := \forall x \in [n], D_{xv} < \tau \implies D_{xw} \leq \tau$ and $D_{xw} < \tau \implies D_{xv} \leq \tau$.*

The intuition behind is the following. If an edge $\{v, w\}$ appears at time τ in a temporal graph $\mathcal{G}$, then any foremost temporal xv-path with arrival time less than τ can be extended by the time-edge $(\{v, w\}, \tau)$, implying that the foremost arrival time at w is at most τ. Moreover, if $\mathcal{G}$ is a realization of D, i.e. $D = \text{Foremost}(\mathcal{G})$, and $D_{xv} < \tau$, then we must have $D_{xw} \leq \tau$. By symmetry of edges, we must also have $D_{xw} < \tau \implies D_{xv} \leq \tau$. Note that, given an $n \times n$ distance matrix D and a time-edge $(\{v, w\}, \tau)$, the property $\mathit{EdgeCompat}(D, \{v, w\}, \tau)$ can easily be tested in $\mathcal{O}(n)$ time.

The algorithm behind Theorem 3.1 now consists of checking that for each entry $\tau = D_{uw}$ of D, there exists a vertex v such that $D_{uv} < \tau$ and the time-edge $(\{v, w\}, \tau)$ is Foremost-edge-compatible with D. If this is the case, such a time-edge $(\{v, w\}, \tau)$ is added to the temporal graph we are constructing (see Algorithm 1). This condition is indeed necessary as a foremost temporal uw-path in a realization of D must end with such an edge. We will show that this condition is also sufficient, and leads to a construction of a realization with at most n^2 time labels.

Algorithm 1 Foremost-path TGR.

Input: A number n and an $n \times n$ distance matrix D
Output: YES if there exists a temporal graph (G, λ) that is a Foremost-realization of D, NO otherwise

Let C be the set of all pairs (u, w) such that $u \neq w$ and $D_{uw} < \infty$
Let λ be an empty labeling, i.e. $\lambda(\{u, v\}) = \emptyset$ for all $u, v \in [n]$
Return Exists(D, C, λ)
procedure Exists(D, C, λ) ▷ Given an $n \times n$ distance matrix D, $C \subset [n] \times [n]$ and a labeling λ
    for all $(u, w) \in C$ **do**
        if $\exists v \in [n]$ such that $D_{uv} < D_{uw}$ and $\mathit{EdgeCompat}(D, \{v, w\}, D_{uw})$ **then**
            Add D_{uw} to $\lambda(\{v, w\})$
        else
            Return NO
    Return YES

The correctness of the algorithm comes with defining how a temporal graph can partially realize a matrix as follows.

▶ **Definition 3.4.** *A temporal graph $\mathcal{G}$ is said to be* Foremost-compatible *with D if $D \leq$ Foremost$(\mathcal{G})$, i.e. $D_{uv} \leq \text{Foremost}(\mathcal{G})_{uv}$ for all $u, v \in [n]$, and all time-edges of $\mathcal{G}$ are* Foremost-*edge-compatible with D.*

First note that any Foremost-realization of D must be Foremost-compatible with D.

▶ **Lemma 3.5.** *If $\mathcal{G}$ is a* Foremost-*realization of D, then $\mathcal{G}$ is* Foremost-*compatible with D.*

Proof. First, we clearly have $D \leq \text{Foremost}(\mathcal{G})$ since $\mathcal{G}$ is a Foremost-realization of D. Second, suppose for the sake of contradiction that some time-edge $(\{v, w\}, \tau)$ is not Foremost-edge-compatible with D. That is, without loss of generality, there exists a vertex x such that $D_{xv} < \tau$ and $D_{xw} > \tau$. Any foremost temporal xv-path in $\mathcal{G}$ must arrive in v at time D_{xv}. However, this temporal path can be extended with $(\{v, w\}, \tau)$, yielding a temporal xw-walk arriving at time $\tau < D_{xw}$, contradicting the fact that $\mathcal{G}$ is a Foremost-realization of D. ◀

Note also that an n-vertex empty temporal graph, i.e. without any time labels, is always Foremost-compatible with D as it does not contain any time-edges. Moreover, Foremost-compatibility is preserved by addition of a Foremost-edge-compatible time-edge as stated below.

▶ **Lemma 3.6.** *If a temporal graph $\mathcal{G} = (G, \lambda)$ is* Foremost-*compatible with D and a time-edge $(\{v, w\}, \tau)$ is* Foremost-*edge-compatible with D, then the temporal graph $\mathcal{G}'$ obtained from $\mathcal{G}$ by adding label τ to edge $\{v, w\}$ is also* Foremost-*compatible with D.*

Proof. We just need to prove $D \leq \text{Foremost}(\mathcal{G}')$. Suppose for the sake of contradiction that there exists $x \neq y$ such that $D_{xy} > \text{Foremost}(\mathcal{G}')_{xy}$, i.e., D_{xy} is greater than the arrival time of a foremost temporal xy-path P in $\mathcal{G}'$. Consider the first time-edge $(\{z, z'\}, \sigma')$ of P such that P arrives in z at time $\sigma \geq D_{xz}$ but arrives in z' before $D_{xz'}$, i.e. $\sigma' < D_{xz'}$. Such an edge must exist since P arrives in y before D_{xy} and $D_{xx} = 0$. The (strict) temporal path definition implies $\sigma < \sigma'$ which thus yields $D_{xz} < \sigma'$. As $D_{xz'} > \sigma'$, $(\{z, z'\}, \sigma')$ cannot be Foremost-edge-compatible with D. This contradicts either the Foremost-edge-compatibility of $(\{v, w\}, \tau)$ if $(\{z, z'\}, \sigma') = (\{v, w\}, \tau)$, or the Foremost-compatibility of $\mathcal{G}$ otherwise. ◀

▶ **Lemma 3.7.** *If $\mathcal{G}$ is a* Foremost-*realization of D, then for any entry $\tau = D_{uw}$ with $u \neq w$, there exists a vertex v such that $D_{uv} < \tau$ and $(\{v, w\}, \tau)$ is* Foremost-*edge-compatible with D.*

Proof. It suffices to consider the last time-edge $(\{v, w\}, \tau)$ of a foremost temporal uw-path in a Foremost-realization $\mathcal{G}$ of D. It must satisfy $\tau = D_{uw}$ since $\mathcal{G}$ is a realization of D. Since it is a strict temporal path, it arrives in v before τ, implying $D_{uv} < \tau$. Moreover, $(\{v, w\}, \tau)$ is Foremost-edge-compatible with D by Lemma 3.5. ◀

Proof of Theorem 3.1. If there exists a Foremost-realization of D, the algorithm must find a suitable time-edge $(\{v, w\}, D_{uw})$ for each pair (u, w) by Lemma 3.7. It thus returns NO, only when no such realization exists. Let K_n denote the complete graph with vertex set $[n]$. Lemma 3.6 implies that Algorithm 1 preserves the invariant that (K_n, λ) is Foremost-compatible with D. If the algorithm returns YES, the constructed temporal graph $\mathcal{G} = (K_n, \lambda)$ is thus Foremost-compatible with D, implying $D \leq \text{Foremost}(\mathcal{G})$. We now prove that we indeed must have $D = \text{Foremost}(\mathcal{G})$. Suppose for the sake of contradiction that there are pairs (u, w) satisfying $D_{uw} < \text{Foremost}(\mathcal{G})_{uw}$. Consider such a pair (u, w) such that D_{uw} is minimum. When this pair was considered, the algorithm added to $\mathcal{G}$ a time-edge $(\{v, w\}, D_{uw})$ satisfying $D_{uv} < D_{uw}$. By the choice of (u, w), we have $D_{uv} = \text{Foremost}(\mathcal{G})_{uv}$. Now, if we extend a foremost temporal uv-path in $\mathcal{G}$ with $(\{v, w\}, D_{uw})$, we obtain a temporal uw-walk arriving at time D_{uw} in contradiction with $D_{uw} < \text{Foremost}(\mathcal{G})_{uw}$. This concludes the proof of correctness of Algorithm 1.

Its time complexity is clearly $\mathcal{O}(n^4)$ as for each of the n^2 pairs (u, w), we consider at most n vertices v and the test for the Foremost-edge-compatibility of $(\{v, w\}, D_{uw})$ takes $\mathcal{O}(n)$ time. To obtain $\mathcal{O}(n^3 \log n)$, we use an interval tree data-structure (see, e.g., [5]). It can store n intervals and querying whether a value τ is in one of these intervals can be answered in $O(\log n)$ time. It uses $O(n)$ space and can be constructed in $O(n \log n)$ time. To benefit from such a data-structure, we consider all pairs (u, w) with fixed w consecutively. Before processing them, we compute for each vertex v two interval trees T_v and T'_v where T_v (respectively T'_v) contains the n intervals $[D_{xv}+1, D_{xw}-1]$ (respectively $[D_{xw}+1, D_{xv}-1]$) for $x \in [n]$, ignoring empty intervals. The Foremost-edge-compatibility of a time-edge $(\{v, w\}, \tau)$ can then be tested in $O(\log n)$ time by checking that neither T_v nor T'_v has an interval containing τ. The reason is that any vertex x violating $D_{xv} < \tau \implies D_{xw} \leq \tau$ satisfies $D_{xv} < \tau < D_{xw}$ in which case τ belongs to the interval of T_v associated to x. Analogously,

any vertex x violating $D_{xw} < \tau \implies D_{xv} \leq \tau$ is associated to an interval of T'_v that contains τ. Constructing the interval trees takes $O(n^2 \log n)$ time while processing each (u, w) pair now takes $O(n \log n)$ time as it mainly consists of n Foremost-edge-compatibility tests. The overall complexity is thus $O(n^3 \log n)$ time using $O(n^2)$ space.

Note that the algorithm adds a time label to an edge $\{v, w\}$ at most once for each vertex u, when considering either (u, w) or (u, v), depending on whether $D_{uv} < D_{uw}$ or $D_{uw} < D_{uv}$. The Foremost-realization computed by Algorithm 1 thus has at most n time labels per edge, and at most n^2 time labels in total. ◀

Non-strict foremost paths

We have a similar result for the non-strict case.

▶ **Theorem 3.8** (⋆). *NS-Foremost-path TGR can be solved in $\mathcal{O}(n^3 \log n)$ time and $\mathcal{O}(n^2)$ space. Furthermore, if dealing with a realizable instance, a realization with at most n^2 time labels can be computed with the same complexity.*

A slight modification of Algorithm 1 suffices, replacing $EdgeCompat(D, \{v, w\}, \tau)$ with:

$$NSEdgeCompat(D, \{v, w\}, \tau) := \forall x \in [n], D_{xv} \leq \tau \iff D_{xw} \leq \tau.$$

The change of $<$ for $\leq$ accounts for considering non-strict temporal paths rather than strict ones. This modification requires a different version of Lemma 3.7 with a significantly different proof where a non-strict temporal vw-path whose edges are traversed at same time τ in a realization is replaced by a single time-edge $(\{v, w\}, \tau)$ (see Lemma 3.12 in the full version).

Periodic temporal graph and prescribed graph

It is straightforward to generalize Theorem 3.1 to Periodic Foremost-path TGR and Prescribed Foremost-path TGR with appropriate definitions of edge compatibility (see Therems 3.13 and 3.14 in the full version).

Non-strict foremost paths with a prescribed graph

In this setting, a prescribed graph $G_p = ([n], E_p)$ is additionally given as input, and the realization is required to have a subgraph of G_p as underlying graph. We let $N_p(v) = \{w : \exists\{v, w\} \in E_p\}$ denote the set of neighbors of any vertex $v \in [n]$ in G_p.

The main idea is again to add time-edges $(\{v, w\}, \tau)$ that satisfy $NSEdgeCompat(D, \{v, w\}, \tau)$ and such that $\{v, w\}$ is present in the prescribed graph. But conversely to the non-strict setting considered in Theorem 3.8, it is not possible to replace an instantaneous temporal vw-path (whose edges are traversed at the same time) in a realization by a single appearance of $\{v, w\}$ as this edge might not be in the prescribed graph.

Indeed, the condition $D_{uv} < D_{uw}$ at Line 6 of Algorithm 1 now becomes problematic as the prescribed graph may impose the addition of a time-edge $(\{v, w\}, D_{uw})$ such that $D_{uv} = D_{uw}$ to fulfill an entry D_{uw}. Moreover, the order in which we can fulfill entries $D_{uw_1} = \cdots = D_{uw_p}$ in this manner may depend on the prescribed graph. A naive solution would be to let each edge $\{v, w\} \in E_p$ appear at all times $\tau \notin \{0, \infty\}$ appearing in D that satisfy $NSEdgeCompat(D, \{v, w\}, \tau)$. It would then suffice to check if the resulting temporal graph is an NS-Foremost-realization of D. However, this would result in a poor complexity and possibly $\Theta(n^4)$ time labels overall. We can still solve the problem with a better complexity and a tight number of time labels as stated below.

▶ **Theorem 3.9** (⋆). *Prescribed NS-Foremost-path TGR can be solved in $\mathcal{O}(n^2m)$ time and $\mathcal{O}(n^2)$ space, where m is the number of edges of the prescribed graph. Furthermore, if dealing with a realizable instance, a realization with at most n^2 time labels can be computed with the same complexity.*

The result is a consequence of an algorithm that works as follows: It scans the set $\{d_1, \dots, d_p\}$ of entries of D excluding 0 and ∞ (in any order). For each d_i, it checks the set C_i of all pairs (u, w) such that $D_{uw} = d_i$. If there exists a vertex $v \in N_p(w)$ such that $D_{uv} < D_{uw}$ and $NSEdgeCompat(D, \{v, w\}, d_i)$ is satisfied, it adds label D_{uw} to $\{v, w\}$, similarly to Algorithm 1. In addition, it starts a BFS like procedure to find other pairs $(u, w') \in C_i$ that can be reached at time d_i through w. See Algorithm 2 in the full version for more details.

3.2 Limits of polynomial-time algorithms for Foremost-path TGR

In this section, we show several additional requirements on instances of Foremost-path TGR for which the problem becomes NP-hard.

▶ **Theorem 3.10** (⋆). *Foremost-path TGR is NP-hard when allowing at most one label per edge.*

Foremost paths in ranges

Given a temporal path metric M, we define the following variant of M-path TGR where the input sequence encodes a matrix D of ranges. More precisely, each entry is supposed to represent a *range* $[\ell, r] = \{\ell, \dots, r\}$ of positive integers. The Ranged-M-path predicate is then defined as $P(\mathcal{G}, D) := M(\mathcal{G})_{uv} \in D_{uv}$ for all $u, v \in [n]$. For example, this leads to the following problem for $M =$ Foremost.

Ranged-Foremost-path TGR:
Input: A number n and an $n \times n$ matrix D of ranges.
Question: Is there a temporal graph $\mathcal{G}$ such that $\text{Foremost}(\mathcal{G})_{uv} \in D_{uv}$ for all $u, v \in [n]$?

An entry (u, v) is said to be *undetermined* if $D_{uv} = [\ell, r]$ with $\ell \neq r$. Note that when the number k of undetermined entries is zero, this problem is equivalent to Foremost-path TGR, for which we presented a polynomial-time algorithm. We now analyze the complexity of Ranged-Foremost-path TGR and its non-strict variant Ranged-NS-Foremost-path TGR with a focus on the parameter k.

▶ **Theorem 3.11** (⋆). *Ranged-Foremost-path TGR and Ranged-NS-Foremost-path TGR are both NP-hard. Moreover, believing the ETH, neither Ranged-Foremost-path TGR nor Ranged-NS-Foremost-path TGR can be solved in $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ time, where k is the number of undetermined entries of D.*

For the strict setting, we obtain hardness even when each range has size at most 2.

▶ **Theorem 3.12** (⋆). *Ranged-Foremost-path TGR is NP-hard even when each range has length at most two and the largest value of the matrix is 5.*

FPT algorithm for Ranged-Foremost-path TGR

We now propose a dynamic programming algorithm solving Ranged-Foremost-path TGR which has running time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. Note that this is tight in the sense that a significantly faster algorithm would contradict ETH by Theorem 3.11.

▶ **Theorem 3.13** (⋆). *Ranged-Foremost-path TGR can be solved in $\mathcal{O}(k^2 3^k n^4)$ time and $\mathcal{O}(2^k + n^2)$ space, where k is the number of undetermined entries of D.*

In the following, we let ℓ_{uw} (respectively r_{uw}) denote the lower bound (respectively upper bound) of entry (u, w), i.e., $D_{uw} = [\ell_{uw}, r_{uw}]$. Recall that an entry (u, w) is undetermined if $\ell_{uw} \neq r_{uw}$. The set of such undetermined entries is denoted by $Undet$ and its size is denoted by k.

This result relies on the fact that a realizable instance can always be realized by a temporal graph using time labels in the restricted set $\mathcal{T} = \{\ell_{uw} + j \mid u \neq w \text{ and } 0 \leq j \leq k\}$ (as proven in the full version). We then propose an algorithm that processes all times in $\mathcal{T}$ in increasing order and guesses which undetermined entries can be realized at the current time. More precisely, letting $0 < \tau_1 < \cdots < \tau_m$ denote the times in $\mathcal{T}$, we maintain, for each $i \in [m]$, a table $R[\cdot, i]$ such that $R[S, i]$ for $S \subseteq Undet$ is equal to True if and only if there exists a temporal graph $\mathcal{G}$ with time labels in $\{\tau_1, \ldots, \tau_i\}$ such that:

- for each $(u, w) \in S$, the earliest arrival time of any foremost temporal uw-path in $\mathcal{G}$ is in D_{uw} and is at most τ_i,
- for each (u, w) with $\ell_{uw} = r_{uw} = \tau \leq \tau_i$, the earliest arrival time of any foremost temporal uw-path in $\mathcal{G}$ is τ,
- for all other entries (u, w), there is no temporal uw-path in $\mathcal{G}$.

We have that $R[Undet, m] = True$ if and only if there exists a temporal graph that realizes D for Ranged-Foremost-path.

When processing time τ_i, we consider tri-partitions S, T, U of the set $Undet$ where S represents the set of undetermined entries that must be realized before time τ_i, T represents the set of undetermined entries that must be realized at time τ_i and U the set of undetermined entries that must be realized after time τ_i. Similarly to Algorithm 1, an appropriate definition of edge compatibility (with respect to D, S and T) allows to test if a temporal graph realizing $R[S, i-1] = True$ can be completed in order to set $R[S \cup T, i]$. Importantly, we do not need explicit access to such a temporal graph as we probe it through D, S, T.

▶ Remark. The algorithm proposed here can easily be generalized to a more general setting where a collection of ranges is given for each entry of D with time complexity $\mathcal{O}(k^2 3^k n^2 N)$ where $N \geq n^2$ denotes the total number of ranges in D. In particular, this provides an FPT algorithm for Ranged-Foremost-path TGR when each entry of D encodes a set of integers and at most k of them are non-singletons. Note that the hardness result of Theorem 3.12 holds in that setting, even if each set has size at most 2.

4 Fastest paths

In this section we consider temporal graph realization for fastest paths analyzed by Klobas et al. [16] and Erlebach et al. [11]. We answer an open question by both papers about the parameterized complexity with respect to the vertex cover number.

So far, Fastest-path TGR has only been considered for strict temporal paths, and if we consider a periodic temporal graph or if we consider the non-periodic version with a limited number of labels per edge [11, 16]. Our parameterized hardness result holds even for non-periodic temporal graph with arbitrary many labels per edge and without limiting the lifetime of the sought temporal graph. We then show that this hardness is preserved in the periodic case with one label per edge per period to answer the open questions.

▶ **Theorem 4.1** (⋆). *Fastest-path TGR is NP-hard and W[1]-hard when parameterized by the vertex cover number of the underlying graph plus the largest entry of D. This holds even on a family of instances for which all yes-instances are realizable with only one label per edge.*

Proof (sketch). We reduce from Multicolored Clique [6].

Multicolored Clique:
Input: An undirected graph $G = (V, E)$, an integer k, and a k-partition $(V_1 \cup \cdots \cup V_k)$ of V, such that V_i is an independent set in G for each $i \in [1, k]$.
Question: Is there a clique of size k in G?

Let $I := (G = (V_1 \cup \cdots \cup V_k, E), k)$ be an instance of Multicolored Clique where for each $1 \leq a < b \leq k$, $G[V_a \cup V_b]$ is a disjoint union of bicliques. For each $a \in [1, k]$, we call V_a a *color class*. Even under these restrictions, Multicolored Clique is NP-hard and W[1]-hard when parameterized by k [22]. Let $V := V_1 \cup \cdots \cup V_k$.

To obtain an instance D of Fastest-path TGR as follows, we first describe the underlying graph, that is, the graph G' that contains an edge $\{u, v\}$ if and only if $D_{u,v} = 1$. The graph G' is defined over the vertex set $V' := V \cup \{s, s', s'', t, t', t''\} \cup X \cup L$, where L is a vertex set of size $2k + 2$ and $X := \{x_i \mid 1 \leq i \leq k + 1\}$ (see Figure 1). We add edges between these vertices, such that $V' \setminus V$ is a vertex cover of G'. That is, there are no edges between the vertices of V in G'. We make L into a clique and adjacent to all vertices of V' besides s and t. Similarly, we make the vertices s' and s'' adjacent to all vertices of V' besides t and make the vertices t' and t'' adjacent to all vertices of V' besides s. Additionally, we make x_1 adjacent to the vertices of V_1, x_{k+1} adjacent to the vertices of V_k, and for each $i \in [2, k]$, we make x_i adjacent to the vertices of $V_{i-1} \cup V_i$. There are no edges between the vertices of X. Finally, we make s adjacent to s', s'', and x_1, and we make t adjacent to t', t'', and x_{k+1}.

This completes the underlying graph G' and thus all entries of the matrix D of value 1. Let E' denote the edges of G'. Next, we define the remaining entries. Note that each vertex of $\{s', s'', t', t''\} \cup L$ is adjacent to each other vertex of $V' \setminus \{s, t\}$. Hence, for these vertices it remains to define the entries of the table from and to the vertices s and t. We set $D_{s,q} := D_{q,s} := 2k + 3$ for each vertex $q \in \{t', t''\} \cup L$. Similarly, we set $D_{t,p} := D_{p,t} := 2k + 3$ for each vertex $p \in \{s', s''\} \cup L$. For each non-edge $\{u, v\}$ of G, we set $D_{u,v} := D_{v,u} := 2k + 3$. Finally, we set $D_{s,t} := 2k + 2$ and $D_{t,s} := 4k + 5$. All other undefined entries are set to 2. This completes the construction. Note that $X \cup L \cup \{s', s'', t', t''\}$ is a vertex cover of size $3k + 7$ and that the largest entry in D is $4k + 5$.

Intuition. The idea behind the reduction is that realizing all entries besides $D_{s,t}$ is possible, regardless of whether G contains a (multicolored) clique of size k. We will show that this is ensured by the fact that $G[V_i \cup V_j]$ with $1 \leq i < j \leq k$ is a disjoint union of bicliques. The difficulty to decide whether the matrix is realizable thus comes from the difficulty of deciding whether the entry $D_{s,t}$ can additionally be realized, which can only be done by using vertices of $X \cup V$ as intermediate vertices of the path. Let S denote the vertices of V on any path P realizing the entry $D_{s,t} = 2k + 2$. Based on the structure of the underlying graph (see Figure 1), S contains for each $i \in [1, k]$ at least one vertex of V_i. By definition of the entries between vertices of V, these vertices need to form a clique in the original graph, as only adjacent vertices u and v in G fulfill $D_{u,v} \leq 2k + 2$, which is the duration of P.

Correctness. We now show that D is realizable if and only if G admits a clique of size k. More precisely, we show that if G admits a clique of size k, then there is a realization for D with exactly one label per edge.

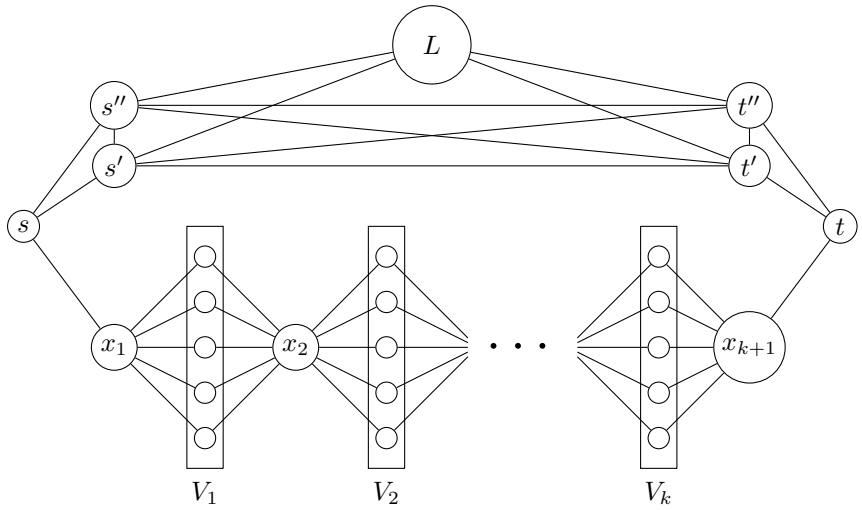

Figure 1 An illustration of the underlying graph from the reduction behind Theorem 4.1. The edges of the biclique $(L \cup \{s', s'', t', t''\}, V \cup X)$ are not depicted. Note that L is a clique of size $2k+2$.

($\Rightarrow$) Let $\lambda\colon E' \to 2^{\mathbb{N}}$ be an edge labeling, such that $\mathcal{G} := (G', \lambda)$ realizes D. We show that G has a clique of size k. Consider the entry $D_{s,t} = 2k+2$. Since $\mathcal{G}$ is a realization of D, this implies that the fastest temporal path from s to t has duration exactly $2k+2$. Let P be an arbitrary fastest temporal path from s to t in $\mathcal{G}$. Since the duration of P is $2k+2$, the duration of each (not necessarily proper) subpath of P is at most $2k+2$. Hence, for any two distinct vertices a, b of P, where a precedes b in P, the entry $D_{a,b}$ is at most $2k+2$, as D is realized by $\mathcal{G}$. This immediately implies that P does not visit any vertex of $\{s', s'', t', t''\} \cup L$, since $D_{s,t'} = D_{s,t''} = D_{s',t} = D_{s'',t} = 2k+3$ and $D_{s,\ell} = D_{\ell,t} = 2k+3$ for each vertex $\ell \in L$. That is, P only uses vertices of $V \cup \{s,t\} \cup X$. By definition, each path from s to t in $G'[V \cup \{s,t\} \cup X]$ traverses all vertices of X and one vertex of each of the color classes of V, that is, for each $i \in [1,k]$, the path contains one vertex of V_i. This in particular holds for P. Let S be the vertices of V that are visited by P. By the above, S has size at least k. Moreover, for each two distinct vertices a and b of S, $D_{a,b} = D_{b,a} \leq 2k+2$, since the subpath between a and b of P has duration at most $2k+2$ and $\mathcal{G}$ realizes D. This implies that $\{a,b\}$ is an edge of G, as otherwise, $D_{a,b} = D_{b,a}$ is defined as $2k+3$. Consequently, S is a clique of size k in G.

($\Leftarrow$) Let S be a clique of size k in G and for each $i \in [1,k]$, let v_i denote the unique vertex of $S \cap V_i$. We define a labeling $\lambda\colon E' \to \mathbb{N}$, such that $\mathcal{G} := (G', \lambda)$ realizes D. To this end, we describe several *time blocks*, that is, intervals $[\alpha, \beta]$ with $\beta \geq \alpha + 4k + 6$, such that only the described edges receive a label from this interval, and all other edges receive no label from $[\alpha - (4k+6), \beta + (4k+6)]$. The reason behind this is that labeling edges in different time blocks do not create paths of duration less than $4k+6$, which is larger than the largest entry of D. Hence, we can show that our labeling realizes D by showing that for each two vertices a and b of G' (i) there is a time block in which there is a temporal path from a to b of duration exactly $D_{a,b}$ and (ii) for each time block, there is no temporal path from a to b of duration less than $D_{a,b}$. Note that the order of time blocks does not matter. Hence, when describing the labeling, we simply describe a collection of time blocks which in total fulfill the above properties, while not explicitly defining the concrete start and end time of the time blocks. In the following, we mainly focus on realizing all entries of value at least 2 in D. Afterwards, we describe how to realize the entries of value 1.

- **Realizing all entries involving vertices of $V' \setminus (V \cup X)$ besides $D_{s,t}$.** We show that we can realize all these entries by only labeling edges that have at least one endpoint in $\{s', s'', t', t''\}$. This proof is deferred to the full version.
- **Realizing $D_{s,t}$.** Next, we define a time block $[\alpha, \beta]$ that realizes the entry $D_{s,t}$. Recall that S is a clique in G and that for each $i \in [1, k]$, v_i denotes the vertex of $S \cap V_i$. Consider the path $P := (s, x_1, v_1, \ldots, x_k, v_k, x_{k+1}, t)$ and label the edges of this path with consecutive time labels starting with α. Hence, this path has duration equal to its length, namely $D_{s,t} = 2k + 2$.

  The argument that this creates no paths that are too fast is deferred to the full version. It mainly comes from the fact that for any two vertices a and b of P that are both from V, $\{a, b\} \in E$ since S is a clique, and we have $D_{a,b} = 2$.

So far, we have realized all entries of D of value at least 2 that involve at least one vertex of $\{s, s', s'', t, t', t''\} \cup L$. In the following, we describe further time blocks to realize the entries of D of value at least 2 involving only vertices of $V \cup X$. To this end, we will only use edges between $X \cup V$ and L. Note that none of these edges has received a label in the previous time blocks, that is, the only edges incident with vertices of L that received labels so far were the edges between L and $\{s', s'', t', t''\}$. Let the vertices of L be called $\{\ell^*, \ell^{**}\} \cup \{\ell_i, \ell'_i \mid 1 \leq i \leq k\}$.

- **Realizing entries between vertices of V of value $2k + 3$ and entries between vertices of V and X.** We define a time block $[\alpha, \beta]$ as follows: For each vertex $v \in V$, we set $\lambda(\{v, \ell^*\}) := \alpha + 1$ and $\lambda(\{v, \ell^{**}\}) := \alpha + 2k + 3$. For each vertex $x \in X$, we set $\lambda(\{x, \ell^*\}) := \alpha$ and $\lambda(\{x, \ell^{**}\}) := \alpha + 2k + 4$. Finally, we set $\lambda(\{\ell^*, \ell^{**}\}) = \alpha + 2$. Note that each vertex of V has only two incident labels in this time block, namely, $\alpha + 1$ and $\alpha + 2k + 3$. Hence, no temporal path in this time block between vertices of V has duration less than $2k + 3$. Moreover, since entries involving a vertex from V and a vertex from X are of value at most 2, we guarantee that we do not create paths that are too fast in this time block. We now show that this time block realizes (i) all entries of value at least 2 between vertices of V and X and (ii) all entries between vertices of V of value $2k + 3$. For the first type, let $v \in V$ and $x \in X$ with $\{v, x\} \notin E'$. That is, $D_{x,v} = 2$. Then, there is a temporal path (x, ℓ^*, v) in this time block of duration 2. Similarly, the temporal path (v, ℓ^{**}, x) also has duration 2. Now consider the second type. For each two distinct vertices u and v of V with $D_{u,v} \neq 2$, there is the temporal path $(u, \ell^*, \ell^{**}, v)$ with labels $(\alpha + 1, \alpha + 2, \alpha + 2k + 3)$. This path has duration $2k + 3 = D_{u,v}$. This time block realizes the stated entries of D.
- **Realizing entries between vertices of X.** For each $i \in [1, k]$, we define a time block $[\alpha_i, \beta_i]$ in which we set $\lambda(\{x_i, \ell_i\}) := \alpha_i$ and $\lambda(\{\ell_i, x\}) := \alpha_i + 1$ for each $x \in X \setminus \{x_i\}$. For each $i \in [1, k]$, this realizes the entries $D_{x_i,x}$ with $x \in X \setminus \{x_i\}$. Similarly, we add a time block $[\alpha_{k+1}, \beta_{k+1}]$ in which we set $\lambda(\{x_{k+1}, \ell'_1\}) := \alpha_{k+1}$ and $\lambda(\{\ell'_1, x\}) := \alpha_{k+1} + 1$ for each $x \in X \setminus \{x_{k+1}\}$. These time blocks realize all entries of D between vertices of X.
- **Realizing entries between vertices of V of value 2.** Recall that we have to ensure that there is a path of duration 2 between the endpoints of each edge $e \in E$ in our temporal graph. To define the necessary time blocks, we will highly rely on the fact that for each $1 \leq a < b \leq k$, $G[V_a \cup V_b]$ is a vertex disjoint union of bicliques. This property will allow us to realize all edges between V_a and V_b via just two vertices of L. We can do this for several combinations of color classes via the same two vertices of L, as long as no color class occurs in more than one pair. We formalize this as follows. Let $M_1, \ldots, M_k$ be a partition of $\{(a, b) \mid 1 \leq a < b \leq k\}$, such that for each $i \in [1, k]$ and each $a \in [1, k]$, there is at most one ordered pair in M_i that contains a. That is, M_i is a matching in

the directed graph with vertex set $[1,k]$ and edge set $\{(a,b) \mid 1 \leq a < b \leq k\}$. Note that such a partition exists due to the fact that a clique on k vertices has a proper edge coloring with k colors. Let $i \in [1,k]$. We let E_{M_i} denote all edges of G between each pair of color classes in M_i, that is, $E_{M_i} := \bigcup_{(a,b)\in M_i} E(V_a, V_b)$. Since M_i is a matching and $G[V_a \cup V_b]$ is a disjoint union of bicliques for each $1 \leq a < b \leq k$, $G_i := (V, E_{M_i})$ is also a disjoint union of bicliques. That is, each connected component in G_i is a biclique. We use the vertices ℓ_i and ℓ'_i of L to realize the entries of D corresponding to the edges of E_{M_i}. For each connected component of G_i with bipartition (A,B), we add a new time block $[\alpha,\beta]$ and set $\lambda(\{v_a,\ell_i\}) := \alpha$ for each $v_a \in A$ and $\lambda(\{v_b,\ell_i\}) := \alpha+1$ for each $v_b \in B$. This realizes paths of duration 2 from each vertex of A to each vertex of B and no other temporal paths of length more than 1. Since (A,B) is a biclique, for all these vertex pairs, the entry in the matrix is also 2. In the same way, we also add a new time block $[\alpha',\beta']$ and set $\lambda(\{v_b,\ell'_i\}) := \alpha$ for each $v_b \in B$ and $\lambda(\{v_a,\ell'_i\}) := \alpha+1$ for each $v_a \in A$. This thus realizes also the entries of duration 2 from each vertex of B to each vertex of A. Since $E = \bigcup_{i\in[1,k]} E_{M_i}$, this implies that we realized the entries $D_{u,v}$ and $D_{v,u}$ of value 2 by the above time blocks for each edge $\{u,v\} \in E$.

Hence, all entries of value at least 2 in D are realized by λ. Let E'' denote the edges of E' that have not received a label yet. We add one final time block from which all edges of E'' receive the same label. This surely does not create new temporal paths of length more than 1 for which the duration is at most $4k+5$. This completes the definition of λ. Thus, also all entries of value 1 are realized. By definition of the time blocks, we showed that (G',λ) realizes the input matrix D even with just a single label per edge. ◀

Based on this reduction, we can now directly transfer the hardness result to Periodic Fastest-path TGR even when allowing at most one label per edge.

That is, we simply define the period Δ to be an integer much larger than $n^2 \cdot \max D$, which ensures that all fastest paths start and end within a window of Δ consecutive time steps (see [11]).

▶ **Theorem 4.2.** *Even when only allowed to put one label per edge and per period, Periodic Fastest-path TGR is W[1]-hard when parameterized by the vertex cover number of the underlying graph plus the largest entry of D.*

This answers an open question by Klobas et al. [16] and Erlebach et al. [11] about the parameterized complexity of the problem with respect to the vertex cover number. Furthermore, this reduction improves significantly over the known hardness result for parameter feedback vertex set number. It also shows that Periodic Fastest-path TGR can presumably not be solved in FPT time for the combined parameter of the vertex cover number plus ℓ (the number of allowed labels per edge and per period) plus the largest entry in D. Thus, in the FPT algorithm by Erlebach et al. [11] for the vertex cover number plus the period Δ (or lifetime), one cannot replace Δ by ℓ plus the largest entry of D.

A similar reduction also shows similar intractability results for Fastest-path TGR with non-strict paths. The following reduction however requires more than one label per edge.

▶ **Theorem 4.3** ($\star$). *NS-Fastest-path TGR is NP-hard and W[1]-hard when parameterized by the vertex cover number of the underlying graph plus the largest entry of D.*

5 Shortest paths

In this section, we consider the question for *shortest* temporal paths.

Shortest-path TGR:
Input: A distance matrix D of size $n \times n$.
Question: Is there a temporal graph $\mathcal{G}$ such that $\text{Shortest}(\mathcal{G}) = D$?

Note that a realization of D can only assign labels to edges $\{u, v\}$ where $D_{u,v} = D_{v,u} = 1$. Hence, $G = ([n], E)$ with $E := \{\{u, v\} \mid D_{u,v} = 1 \wedge D_{v,u} = 1\}$ is the underlying graph of every realization of D. We show the NP-hardness of both the strict and the non-strict variants.

▶ **Theorem 5.1** ($\star$). *Shortest-path TGR and NS-Shortest-path TGR are NP-hard.*

Proof (sketch). We reduce from SAT.

Let F be an instance of SAT where each variable occurs at least once positively and at least once negatively, and where no clause contains the same variable both positively and negatively.

Construction. Let X be the variable set of F and let C denote the clauses of F. To obtain an instance D of Shortest-path TGR or NS-Shortest-path TGR, we first define the underlying graph $G = (V, E)$ that contains an edge $\{u, v\}$ if and only if $D_{u,v} = D_{v,u} = 1$ (see Figure 2). The graph G contains for each variable $x \in X$ the vertices x and $\overline{x}$ which are joined by an edge. For each clause $c \in C$, we also add a vertex c, which we make adjacent to all vertices corresponding to literals that are contained in c. Additionally, we add three more vertices to G: a vertex v^* which is adjacent to all vertices of C, and two vertices $\top$ and $\bot$ that are adjacent to all vertices representing literals, that is, to the vertices of $\{x, \overline{x} \mid x \in X\}$.

Next, we describe the remaining entries of D. Let c be a clause of C. We set $D_{c,\top} := 3$ and $D_{\top,c} := D_{c,\bot} := D_{\bot,c} := 2$. For each other clause c' of C, we set $D_{c,c'} := D_{c',c} := 2$. For each positive literal x that occurs in c, we set $D_{c,\overline{x}} := D_{\overline{x},c} := 2$. Similarly, for each negative literal $\overline{x}$ that occurs in c, we set $D_{c,x} := D_{x,c} := 2$. For each variable x for which neither x nor $\overline{x}$ occurs in c, we set $D_{c,\overline{x}} := D_{\overline{x},c} := D_{c,x} := D_{x,c} := 3$. This defines all entries regarding vertices of C.

Let ℓ_1 and ℓ_2 be distinct literals, such that they are not the negation of each other. We set $D_{\ell_1,\ell_2} := D_{\ell_2,\ell_1} := 2$. For each literal ℓ, we also set $D_{\ell,v^*} := D_{v^*,\ell} := 2$.

Finally, we set $D_{v^*,\bot} := D_{\bot,v^*} := 3$, $D_{\bot,\top} := D_{\top,\bot} := 2$, $D_{v^*,\top} := 4$, and $D_{\top,v^*} := 3$. This completes the definition of D.

Note that nearly all defined entries are the exact distances between the vertices in the underlying graph G. The only exceptions are the entry $D_{v^*,\top}$ and the entry $D_{c,\top}$ for each clause $c \in C$. Hence, only for the vertex pairs $(\{v^*\} \cup C) \times \{\top\}$, one could possibly create a temporal path that has length less than the respective entry of D. Based on this property, we can prove that a labeling $\lambda\colon E \to 2^{\mathbb{N}}$ realizes D by showing the following two points:

- For distinct vertices a and b of V, there is a temporal path of length $D_{a,b}$ from a to b.
- For each $a \in \{v^*\} \cup C$, there is no temporal path of length less than $D_{a,\top}$ from a to $\top$.

Intuition. We have clause vertices that aim to reach $\top$. For each such clause vertex c_i, we want that the shortest temporal path to $\top$ has length exactly 3. By the structure of the implicit underlying graph, these paths must be of the form $(c_i, \ell, \overline{\ell}, \top)$ for some literal ℓ that occurs in clause c_i. If two clauses try to use the same variable gadget from different sides to realize their entries, that is, if $(c_i, \ell, \overline{\ell}, \top)$ and $(c_j, \overline{\ell}, \ell, \top)$ are both temporal paths in our solution graph, then in fact at least one of $(c_i, \ell, \top)$ or $(c_j, \overline{\ell}, \top)$ is also a temporal path, implying that for at least one of the clauses, the shortest temporal path has length 2,

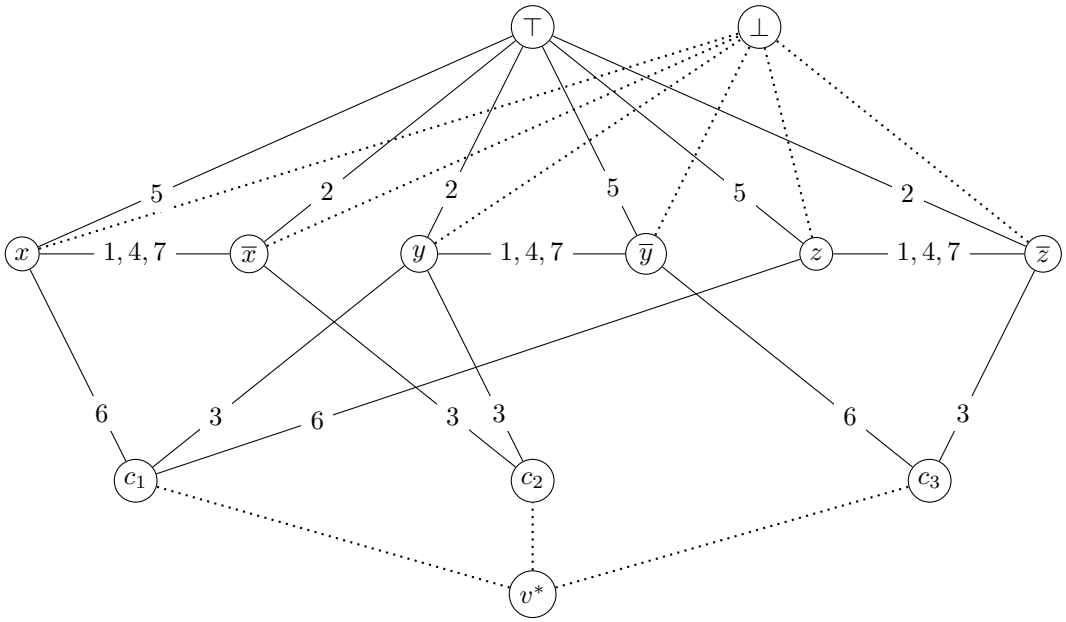

Figure 2 An example of the reduction behind Theorem 5.1 for the formula $(x \vee y \vee z) \wedge (\overline{x} \vee y) \wedge (\overline{y} \vee \overline{z})$. A labeling that realizes the matrix D is depicted, where the dashed arcs receive the label set $\{1, 2, 7, 8\}$. Moreover this labeling corresponds to a satisfying truth assignment (x = False, y = True, z = False).

which is lower than the desired length of 3. Intuitively, this means that in each solution, each variable gadget can only be used in one direction for paths between clauses and $\top$, which then encodes a satisfying truth assignment.

The correctness is deferred to the full version. ◀

Note that a realization question for shortest temporal paths in periodic temporal graphs is polynomial time solvable. Due to the periodicity, a shortest path in the underlying static graph will always be a shortest temporal path in the periodic temporal graph where each edge receives at least one label. Since the latter is mandatory, the resulting problem is answered with yes if and only if the given matrix D is the distance matrix of the underlying graph.

6 Conclusion

Our work spawns several interesting future questions.

- We showed that Foremost-path TGR is polynomial-time solvable if we are allowed to assign up to n labels per edge but becomes NP-hard when allowing only a single label per edge. What is the smallest number of labels per edge for which this problem is still polynomial-time solvable? For example, is there an efficient algorithm when we are allowed to assign only $\frac{n}{2}$ labels per edge?
- Our hardness results for Shortest-path TGR use multiple labels per edge. Does this problem become polynomial-time solvable, if we restrict the respective labeling? For example, what if we enforce that at most one label per edge is allowed or we require a proper labeling, that is, a labeling where no two adjacent edges share a label?
- Are there structural parameters for which we can solve Shortest-path TGR in FPT-time. For example, can we solve the problem efficiently if the underlying graph has bounded treewidth?

- One could consider approximation of the considered problems. For example under the measurement of fulfilling as many entries as possible, are there constant factor approximations for Shortest-path TGR or Fastest-path TGR?

References

1. Eleni C. Akrida, Leszek Gąsieniec, George B. Mertzios, and Paul G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory Comput. Syst.*, 61(3):907–944, 2017. doi:10.1007/S00224-017-9757-X.
2. Amotz Bar-Noy, Keerti Choudhary, David Peleg, and Dror Rawitz. Efficiently realizing interval sequences. *SIAM J. Discret. Math.*, 34(4):2318–2337, 2020. doi:10.1137/20M1326489.
3. Arnaud Casteigts, Michelle Döring, and Nils Morawietz. Realization of Temporally Connected Graphs Based on Degree Sequences. In *Proceedings of the 36th International Symposium on Algorithms and Computation (ISAAC)*, 2025. doi:10.48550/arXiv.2504.17743.
4. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distributed Syst.*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.
5. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*, chapter 14.3. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.
6. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
7. Argyrios Deligkas and Igor Potapov. Optimizing reachability sets in temporal graphs by delaying. *Inf. Comput.*, 285(Part):104890, 2022. doi:10.1016/J.IC.2022.104890.
8. Jessica A. Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *J. Comput. Syst. Sci.*, 119:60–77, 2021. doi:10.1016/J.JCSS.2021.01.007.
9. Paul Erdős and Tibor Gallai. Graphs with prescribed degrees of vertices. *Mat. Lapok*, 11:264–274, 1960.
10. Thomas Erlebach, Othon Michail, and Nils Morawietz. Recognizing and Realizing Temporal Reachability Graphs. In *Proceedings of the 33rd Annual European Symposium on Algorithms (ESA)*, 2025. doi:10.48550/arXiv.2503.15771.
11. Thomas Erlebach, Nils Morawietz, and Petra Wolf. Parameterized algorithms for multi-label periodic temporal graph realization. *Theoretical Computer Science*, 1051:115412, 2025. doi:10.1016/j.tcs.2025.115412.
12. F. Göbel, J. Orestes Cerdeira, and Henk Jan Veldman. Label-connected graphs and the gossip problem. *Discret. Math.*, 87(1):29–40, 1991. doi:10.1016/0012-365X(91)90068-D.
13. S. Louis Hakimi and S. S. Yau. Distance matrix of a graph and its realizability. *Quarterly of Applied Mathematics*, 22:305–317, 1965. URL: https://api.semanticscholar.org/CorpusID:118924338.
14. David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 504–513. ACM, 2000. doi:10.1145/335305.335364.
15. Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. The complexity of computing optimum labelings for temporal connectivity. *J. Comput. Syst. Sci.*, 146:103564, 2024. doi:10.1016/J.JCSS.2024.103564.
16. Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. Temporal graph realization from fastest paths. *Theoretical Computer Science*, 1056:115508, 2025. doi:10.1016/j.tcs.2025.115508.

17 George B. Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 657–668. Springer, 2013. doi:10.1007/978-3-642-39212-2_57.

18 George B. Mertzios, Othon Michail, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019. doi:10.1007/S00453-018-0478-6.

19 George B. Mertzios, Hendrik Molter, Nils Morawietz, and Paul G. Spirakis. Realizing temporal transportation trees. In *Proceedings of the 51st Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, 2025. URL: https://arxiv.org/abs/2403.18513.

20 George B. Mertzios, Hendrik Molter, Nils Morawietz, and Paul G. Spirakis. Temporal Graph Realization With Bounded Stretch. In *Proceedings of the 50th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2025. doi:10.48550/arXiv.2504.14258.

21 Julia Meusel, Matthias Müller-Hannemann, and Klaus Reinhardt. Directed temporal tree realization for periodic public transport: Easy and hard cases. In *Proceedings of the 25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS)*, 2025. doi:10.48550/arXiv.2504.07920.

22 Nils Morawietz, Carolin Rehs, and Mathias Weller. A timecop's work is harder than you think. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 71:1–71:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.71.

23 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. *J. Comput. Syst. Sci.*, 107:72–92, 2020. doi:10.1016/J.JCSS.2019.07.006.