Temporal graph realization from fastest paths

Nina Klobas
Department of Computer Science, Durham University, UK

George B. Mertzios
Department of Computer Science, Durham University, UK

Hendrik Molter
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Paul G. Spirakis
Department of Computer Science, University of Liverpool, UK

Abstract

In this paper we initiate the study of the *temporal graph realization* problem with respect to the fastest path durations among its vertices, while we focus on periodic temporal graphs. Given an $n \times n$ matrix D and a $\Delta \in \mathbb{N}$, the goal is to construct a Δ-periodic temporal graph with n vertices such that the duration of a *fastest path* from v_i to v_j is equal to $D_{i,j}$, or to decide that such a temporal graph does not exist. The variations of the problem on static graphs has been well studied and understood since the 1960's (e.g. [Erdős and Gallai, 1960], [Hakimi and Yau, 1965]).

As it turns out, the periodic temporal graph realization problem has a very different computational complexity behavior than its static (i. e., non-temporal) counterpart. First we show that the problem is NP-hard in general, but polynomial-time solvable if the so-called underlying graph is a tree. Building upon those results, we investigate its parameterized computational complexity with respect to structural parameters of the underlying static graph which measure the "tree-likeness". We prove a tight classification between such parameters that allow fixed-parameter tractability (FPT) and those which imply W[1]-hardness. We show that our problem is W[1]-hard when parameterized by the *feedback vertex number* (and therefore also any smaller parameter such as *treewidth*, *degeneracy*, and *cliquewidth*) of the underlying graph, while we show that it is in FPT when parameterized by the *feedback edge number* (and therefore also any larger parameter such as *maximum leaf number*) of the underlying graph.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Discrete mathematics

Keywords and phrases Temporal graph, periodic temporal labeling, fastest temporal path, graph realization, temporal connectivity, parameterized complexity.

Digital Object Identifier 10.4230/LIPIcs.SAND.2024.3

Related Version *Full Version*: `https://arxiv.org/abs/2302.08860`

Funding *George B. Mertzios*: Supported by the EPSRC grant EP/P020372/1.
Hendrik Molter: Supported by the ISF, grant nr. 1456/18, and by the European Union's Horizon Europe research and innovation programme under grant agreement 949707.
Paul G. Spirakis: Supported by the EPSRC grant EP/P02002X/1.

1 Introduction

The (static) *graph realization* problem with respect to a graph property $\mathcal{P}$ is to find a graph that satisfies property $\mathcal{P}$, or to decide that no such graph exists. The motivation for graph realization problems stems both from "verification" and from network design applications in engineering. In *verification* applications, given the outcomes of some experimental measurements (resp. some computations) on a network, the aim is to (re)construct an input network which complies with them. If such a reconstruction is not possible, this proves that the measurements are incorrect or implausible (resp. that the algorithm which made the computations is incorrectly implemented). One example of a graph realization (or reconstruction) problem is the recognition of probe interval graphs, in the context of the physical mapping of DNA, see [52, 53] and [36, Chapter 4]. In *network design* applications, the goal is to design network topologies having a desired property [4, 38]. Analyzing the computational complexity of the graph realization problems for various natural and fundamental graph properties $\mathcal{P}$ requires a deep understanding of these properties. Among the most studied such parameters for graph realization are constraints on the distances between vertices [7,8,10,16,17,41], on the vertex degrees [6,22,35,37,40], on the eccentricities [5,9,42,51], and on connectivity [15,29–31,34,37], among others.

In the simplest version of a (static) graph realization problem with respect to vertex distances, we are given a symmetric $n \times n$ matrix D and we are looking for an n-vertex undirected and unweighted graph G such that $D_{i,j}$ equals the distance between vertices v_i and v_j in G. This problem can be trivially solved in polynomial time in two steps [41]: First, we build the graph $G = (V, E)$ such that $v_iv_j \in E$ if and only if $D_{i,j} = 1$. Second, from this graph G we compute the matrix D_G which captures the shortest distances for all pairs of vertices. If $D_G = D$ then G is the desired graph, otherwise there is no graph having D as its distance matrix. Non-trivial variations of this problem have been extensively studied, such as for weighted graphs [41,60], as well as for cases where the realizing graph has to belong to a specific graph family [7,41]. Other variations of the problem include the cases where every entry of the input matrix D may contain a range of consecutive permissible values [7,61,63], or even an arbitrary set of acceptable values [8] for the distance between the corresponding two vertices.

In this paper we make the first attempt to understand the complexity of the graph realization problem with respect to vertex distances in the context of *temporal graphs*, i. e., of graphs whose *topology changes over time.*

▶ **Definition 1** (temporal graph [43]). *A* temporal graph *is a pair (G, λ), where $G = (V, E)$ is an underlying (static) graph and $\lambda : E \to 2^{\mathbb{N}}$ is a* time-labeling *function which assigns to every edge of G a set of discrete time-labels.*

Here, whenever $t \in \lambda(e)$, we say that the edge e is *active* or *available* at time t. In the context of temporal graphs, where the notion of vertex adjacency is time-dependent, the notions of path and distance also need to be redefined. The most natural temporal analogue of a path is that of a *temporal* (or *time-dependent*) path, which is motivated by the fact that, due to causality, entities and information in temporal graphs can "flow" only along sequences of edges whose time-labels are strictly increasing.

▶ **Definition 2** (fastest temporal path). *Let (G, λ) be a temporal graph. A* temporal path *in (G, λ) is a sequence $(e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k)$, where $P = (e_1, \ldots, e_k)$ is a path in the underlying static graph G, $t_i \in \lambda(e_i)$ for every $i = 1, \ldots, k$, and $t_1 < t_2 < \ldots < t_k$. The* duration *of this temporal path is $t_k - t_1 + 1$. A* fastest *temporal path from a vertex u to a vertex v in (G, λ) is a temporal path from u to v with the smallest duration. The* duration *of the fastest temporal path from u to v is denoted by $d(u, v)$.*

In this paper we consider *periodic* temporal graphs, i. e., temporal graphs in which the temporal availability of each edge of the underlying graph is periodic. Many natural and technological systems exhibit a periodic temporal behavior. For example, in railway networks an edge is present at a time step t if and only if a train is scheduled to run on the respective rail segment at time t [3]. Similarly, a satellite, which makes pre-determined periodic movements,

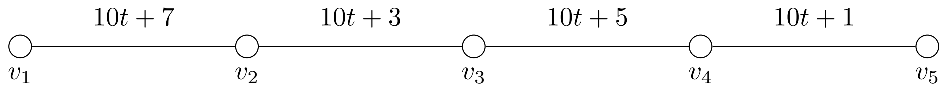


Figure 1 An example of a Δ-periodic temporal graph (G, λ, Δ), where $\Delta = 10$ and the 10-periodic labeling $\lambda : E \to \{1, 2, \ldots, 10\}$ is as follows: $\lambda(v_1v_2) = 7$, $\lambda(v_2v_3) = 3$, $\lambda(v_3v_4) = 5$, and $\lambda(v_4v_5) = 1$. Here, the fastest temporal path from v_1 to v_2 traverses the first edge v_1v_2 at time 7, second edge v_2v_3 a time 13, third edge v_3v_4 at time 15 and the last edge v_4v_5 at time 21. This results in the total duration of $21 - 7 + 1 = 15$ for the fastest temporal path from v_1 to v_5.

can establish a communication link (i. e., a temporal edge) with another satellite whenever they are sufficiently close to each other; the existence of these communication links is also periodic. In a railway (resp. satellite) network, a fastest temporal path from u to v represents the fastest railway connection between two stations (resp. the quickest communication delay between two moving satellites). Furthermore, periodicity appears also in (the otherwise quite complex) social networks which describe the dynamics of people meeting [50, 62], as every person individually follows mostly a weekly routine.

Expanding the work on periodic temporal graphs (see [13, Class 8] and [3, 25, 58, 59]), our study represents the first attempt to understand the complexity of a graph realization problem in the context of temporal graphs. Therefore, we focus in this paper on the most fundamental case, where all edges have the same period Δ (while in the more general case, each edge e in the underlying graph has a period Δ_e). As it turns out, the periodic temporal graph realization problem with respect to a given $n \times n$ matrix D of the fastest duration times has a very different computational complexity behavior than the classic graph realization problem with respect to shortest path distances in static graphs.

Formally, let $G = (V, E)$ and $\Delta \in \mathbb{N}$, and let $\lambda : E \to \{1, 2, \ldots, \Delta\}$ be an edge-labeling function that assigns to every edge of G exactly one of the labels from $\{1, \ldots, \Delta\}$. Then we denote by (G, λ, Δ) the Δ-*periodic temporal graph* (G, L), where for every edge $e \in E$ we have $L(e) = \{i\Delta + x : i \geq 0, x \in \lambda(e)\}$. In this case we call λ a Δ-*periodic labeling* of G; see Figure 1 for an illustration. When it is clear from the context, we drop Δ from the notation and we denote the (Δ-periodic) temporal graph by (G, λ). Given a duration matrix D, it is easy to observe that, similarly to the static case, if $D_{i,j} = 1$ then v_i and v_j must be connected by an edge. We call the graph defined by these edges the *underlying graph* of D.

Our contribution. We initiate the study of naturally motivated graph realization problems in the temporal setting. Our target is not to model unreliable communication, but instead to *verify* that particular measurements regarding fastest temporal paths in a periodic temporal graph are plausible (i. e., "realizable"). To this end, we introduce and investigate the following problem, capturing the setting described above:

Simple periodic Temporal Graph Realization (Simple TGR)

Input: An integer $n \times n$ matrix D, a positive integer Δ.
Question: Does there exist a graph $G = (V, E)$ with vertices $\{v_1, \ldots, v_n\}$ and a Δ-periodic labeling $\lambda : E \to \{1, 2, \ldots, \Delta\}$ such that, for every i, j, the duration of the fastest temporal path from v_i to v_j in the Δ-periodic temporal graph (G, λ, Δ) is $D_{i,j}$?

We focus on exact algorithms. We start by showing NP-hardness of the problem (Theorem 3), even if Δ is a small constant. To establish a baseline for tractability, we show that Simple TGR is polynomial-time solvable if the underlying graph is a tree (Theorem 5).

Building upon these initial results, we explore the possibilities to generalize our polynomial-time algorithm using the *distance-from-triviality* parameterization paradigm [27, 39]. That is, we investigate the parameterized computational complexity of SIMPLE TGR with respect to structural parameters of the underlying graph that measure its "tree-likeness".

We obtain the following results. We show that SIMPLE TGR is W[1]-hard when parameterized by the *feedback vertex number* of the underlying graph (Theorem 4). To this end, we first give a reduction from MULTICOLORED CLIQUE parameterized by the number of colors [26] to a variant of SIMPLE TGR where the period Δ is infinite, that is, when the labeling is non-periodic. Then we use a special gadget (the "infinity" gadget) which allows us to transfer the result to a finite period Δ. The latter construction is independent from the particular reduction we use, and can hence be treated as a reduction from the non-periodic to the periodic setting. Note that our parameterized hardness result with respect to the feedback vertex number also implies W[1]-hardness for any smaller parameter, such as *treewidth*, *degeneracy*, *cliquewidth*, *distance to chordal graphs*, and *distance to outerplanar graphs*.

We complement this hardness result by showing that SIMPLE TGR is fixed-parameter tractable (FPT) with respect to the *feedback edge number* k of the underlying graph (Theorem 6). This result also implies an FPT algorithm for any larger parameter, such as the *maximum leaf number*. A similar phenomenon of getting W[1]-hardness with respect to the feedback vertex number, while getting an FPT algorithm with respect to the feedback edge number, has been observed only in a few other temporal graph problems related to the connectivity between two vertices [14, 21, 32].

Our FPT algorithm works as follows on a high level. First we distinguish $O(k^2)$ vertices which we call "important vertices". Then, we guess the fastest temporal paths for each pair of these important vertices; as we prove, the number of choices we have for all these guesses is upper bounded by a function of k. Then we also need to make several further guesses (again using a bounded number of choices), which altogether leads us to specify a small (i. e., bounded by a function of k) number of different configurations for the fastest paths between *all pairs* of vertices. For each of these configurations, we must then make sure that the labels of our solution will not allow any other temporal path from a vertex v_i to a vertex v_j have a *strictly smaller* duration than $D_{i,j}$. This naturally leads us to build one Integer Linear Program (ILP) for each of these configurations. We manage to formulate all these ILPs by having a number of variables that is upper-bounded by a function of k. Finally we use Lenstra's Theorem [49] to solve each of these ILPs in FPT time. At the end, our initial instance is a YES-instance if and only if at least one of these ILPs is feasible.

The above results provide a fairly complete picture of the parameterized computational complexity of SIMPLE TGR with respect to structural parameters of the underlying graph which measure "tree-likeness". To obtain our results, we prove several properties of fastest temporal paths, which may be of independent interest. Due to space constraints, proofs of results marked with $\star$ are (partially) deferred to the full version on arXiv [46].

Related work. Graph realization problems on static graphs have been studied since the 1960s. We provide an overview of the literature in the introduction. To the best of our knowledge, we are the first to consider graph realization problems in the temporal setting. Very recently, Erlebach et al. [24] have built upon our results and, among others, studied the case where edges might appear more than once in each period. Many other connectivity-related problems have been studied in the temporal setting [2, 12, 18, 19, 23, 28, 33, 44, 48, 55, 57, 65], most of which are much more complex and computationally harder than their non-temporal counterparts, and some of which do not even have a non-temporal counterpart.

Several problems have been studied where the goal is to assign labels to (sets of) edges of a given static graph in order to achieve certain connectivity-related properties [1, 20, 45, 54]. The main difference to our problem setting is that in the mentioned works, the input is a graph and the sought labeling is not periodic. Furthermore, the investigated properties are temporal connectivity among all vertices [1, 45, 54], temporal connectivity among a subset of vertices [45], or reducing reachability among the vertices [20]. In all these cases, the duration of the temporal paths has not been considered.

Finally, there are many models for dynamic networks in the context of distributed computing [47]. These models have some similarity to temporal graphs, in the sense that in both cases the edges appear and disappear over time. However, there are notable differences. For example, one important assumption in the distributed setting can be that the edge changes are adversarial or random (while obeying some constraints such as connectivity), and therefore they are not necessarily known in advance [47].

Preliminaries and notation. We already introduced the most central notion and concepts. There are some additional definitions we need, to present our proofs and results which we give in the following.

An interval in $\mathbb{N}$ from a to b is denoted by $[a,b] = \{i \in \mathbb{N} : a \leq i \leq b\}$; similarly, $[a] = [1,a]$. An undirected graph $G = (V,E)$ consists of a set V of vertices and a set $E \subseteq V \times V$ of edges. For a graph G, we also denote by $V(G)$ and $E(G)$ the vertex and edge set of G, respectively. We denote an edge $e \in E$ between vertices $u, v \in V$ as a set $e = \{u,v\}$. For the sake of simplicity of the representation, an edge e is sometimes also denoted by uv. A path P in G is a subgraph of G with vertex set $V(P) = \{v_1, \ldots, v_k\}$ and edge set $E(P) = \{\{v_i, v_{i+1}\} : 1 \leq i < k\}$ (we often represent path P by the tuple $(v_1, v_2, \ldots, v_k)$).

Let $v_1, v_2, \ldots, v_n$ be the n vertices of the graph G. For simplicity of the presentation (and with a slight abuse of notation) we refer during the paper to the entry $D_{i,j}$ of the matrix D as $D_{a,b}$, where $a = v_i$ and $b = v_j$. That is, we put as indices of the matrix D the corresponding vertices of G whenever it is clear from the context.

Let $P = (u = v_1, v_2, \ldots, v_p = v)$ be a path from u to v in G. Recall that, in our paper, every edge has exactly one time label in every period of Δ consecutive time steps. Therefore, as we are only interested in the fastest duration of temporal paths, many times we refer to (P, λ, Δ) as any of the temporal paths from $u = v_1$ to $v = v_p$ along the edges of P, which starts at the edge v_1v_2 at time $\lambda(v_1v_2) + c\Delta$, for some $c \in \mathbb{N}$, and then sequentially visits the rest of the edges of P as early as possible. We denote by $d(P, \lambda, \Delta)$, or simply by $d(P, \lambda)$ when Δ is clear from the context, the duration of any of the temporal paths (P, λ, Δ); note that they all have the same duration. Many times we also refer to a path $P = (u = v_1, v_2, \ldots, v_p = v)$ from u to v in G, as a temporal path in (G, λ, Δ), where we actually mean that (P, λ, Δ) is a temporal path with P as its underlying (static) path.

We remark that a fastest path between two vertices in a temporal graph can be computed in polynomial time [11, 64]. Hence, given a Δ-periodic temporal graph (G, λ, Δ), we can compute in polynomial-time the matrix D which consists of durations of fastest temporal paths among all pairs of vertices in (G, λ, Δ).

2 Hardness results for Simple TGR

In this section we present our main computational hardness results. We first show that Simple TGR is NP-hard even for constant Δ.

▶ **Theorem 3** (⋆). *Simple TGR is NP-hard for all* $\Delta \geq 3$.

Next, we investigate the parameterized hardness of Simple TGR with respect to structural parameters of the underlying graph. We show that the problem is W[1]-hard when parameterized by the feedback vertex number of the underlying graph. The *feedback vertex number* of a graph G is the cardinality of a minimum vertex set $X \subseteq V(G)$ such that $G - X$ is a forest. The set X is called a *feedback vertex set*. Note that, in contrast to the previous result (Theorem 3), the reduction we use to obtain the following result does not produce instances with a constant Δ.

▶ **Theorem 4** (⋆). *Simple TGR is W[1]-hard when parameterized by the feedback vertex number of the underlying graph.*

Proof. We present a parameterized reduction from the W[1]-hard problem Multicolored Clique parameterized by the number of colors [26]. Here, given a k-partite graph $H = (W_1 \uplus W_2 \uplus \ldots \uplus W_k, F)$, we are asked whether H contains a clique of size k. If $w \in W_i$, then we say that w has *color* i. W.l.o.g. we assume that $|W_1| = |W_2| = \ldots = |W_k| = n$. Furthermore, for all $i \in [k]$, we assume the vertices in W_i are ordered in some arbitrary but fixed way, that is, $W_i = \{w_1^i, w_2^i, \ldots, w_n^i\}$. Let $F_{i,j}$ with $i < j$ denote the set of all edges between vertices from W_i and W_j. We assume w.l.o.g. that $|F_{i,j}| = m$ for all $i < j$ (if not we can add $k \max_{i,j} |F_{i,j}|$ vertices to each W_i and use those to add up to $\max_{i,j} |F_{i,j}|$ additional isolated edges to each $F_{i,j}$). Furthermore, for all $i < j$ we assume that the edges in $F_{i,j}$ are ordered in some arbitrary but fixed way, that is, $F_{i,j} = \{e_1^{i,j}, e_2^{i,j}, \ldots, e_m^{i,j}\}$.

We give a reduction to a variant of Simple TGR where the period Δ is infinite (that is, the sought temporal graph is not periodic and the labeling function $\lambda : E \to \mathbb{N}$ maps to the natural numbers) and we allow D to have infinity entries, meaning that the two respective vertices are not temporally connected. Note that, given the matrix D, we can easily compute the underlying graph G, as follows. Two vertices v, v' are adjacent in G if and only if $D_{v,v'} = 1$, as having an edge between v and v' is the only way that there exists a temporal path from v to v' with duration 1. For simplicity of the presentation of the reduction, we describe the underlying graph G (which directly implies the entries of D where $D_{v,v'} = 1$) and then we provide the remaining entries of D. At the end of the proof, we show how to obtain the result for a finite Δ (by introducing a so-called "infinity gadget") and a matrix D of durations of fastest paths which only has finite entries.

In the following, we give an informal description of the main ideas of the reduction. The construction uses several gadgets, where the main ones are an "edge selection gadget" and a "verification gadget".

Every *edge selection gadget* is associated with a color combination i, j in the Multicolored Clique instance, and its main purpose is to "select" an edge connecting a vertex from color i with a vertex from color j. Roughly speaking, the edge selection gadget consists of m paths, one for every edge in $F_{i,j}$ (see Figure 2 for reference). The distance matrix D will enforce that the labels on those paths effectively order them temporally, that is, in particular, the labels on one of the paths will be smaller than the labels on all other paths. The edge corresponding to this path is selected.

We have a *verification gadget* for every color i. They interact with the edge selection gadgets as follows. The verification gadget for color i is connected to all edge selection gadgets that involve color i. More specifically, this is connected to every path corresponding to an edge at a position in the path that encodes the endpoint of color i of that edge (again, see Figure 2 for reference). Intuitively, the distances in the verification gadget are only

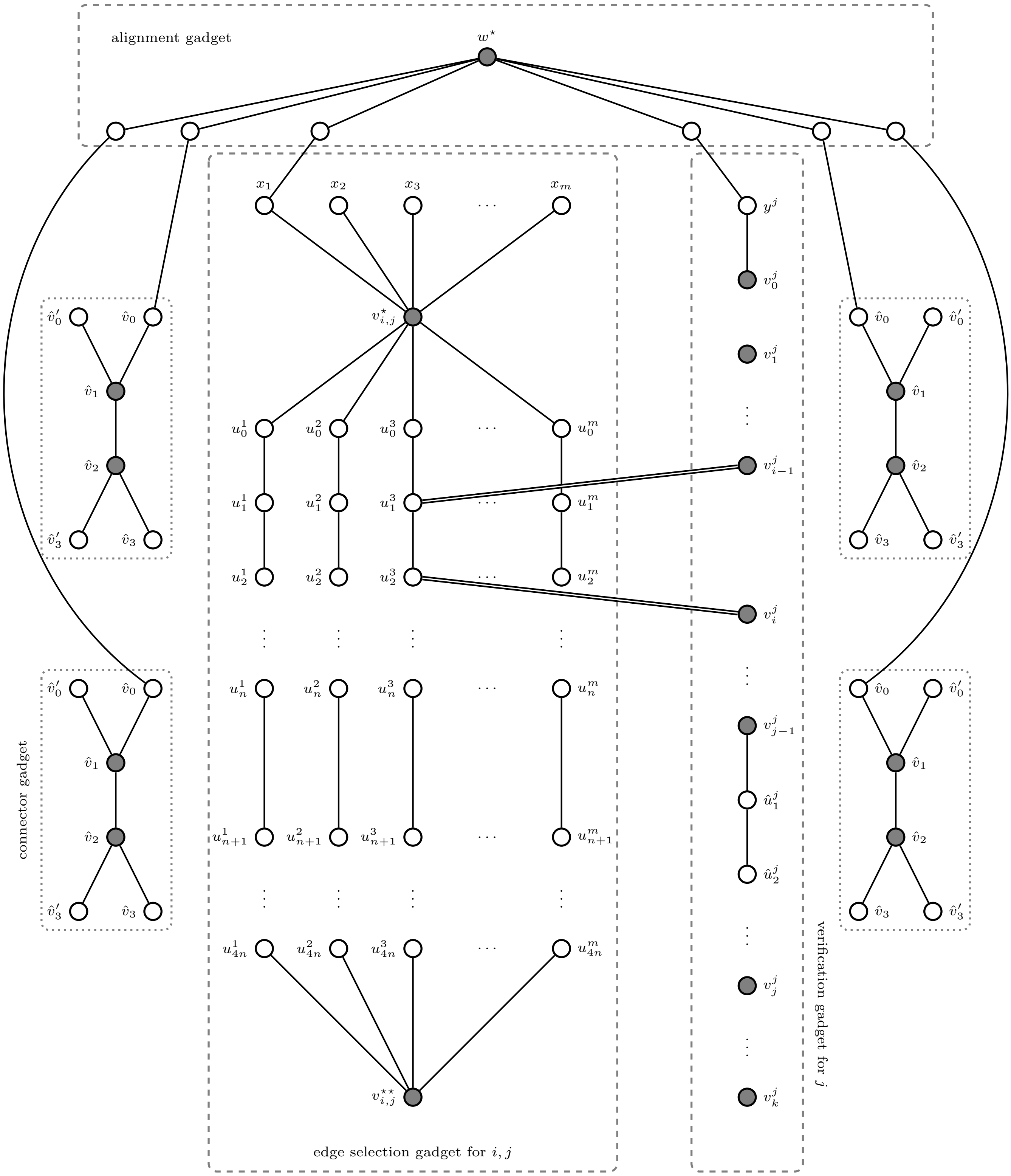


Figure 2 Illustration of part of the underlying graph G and a possible labeling. Edges incident with vertices $\hat{v}_1, \hat{v}_2$ of connector gadgets are omitted. Gray vertices form a feedback vertex set. The double line connections, between a vertex v_{i-1}^j in the verification gadget, and u_1^3 in the edge selection gadget, and, between a vertex u_2^3 in the edge selection gadget, and v_i^j in the verification gadget, consist of $5n$ vertices $a_1^{j,i,3}, a_2^{j,i,3}, \ldots, a_{5n}^{j,i,3}$ and $b_1^{j,i,3}, b_2^{j,i,3}, \ldots, b_{5n}^{j,i,3}$, respectively.

realizable if the selected edges all have the same endpoint of color i. Hence, the distances of all verification gadgets can be realized if and only if the selected edges form a clique.

Furthermore, we use an *alignment gadget* which, intuitively, ensures that the labelings of all gadgets use the same range of time labels. Finally, we use *connector gadgets* which create shortcuts between all vertex pairs that are irrelevant for the functionality of the other gadgets. This allows us to easily fill in the distance matrix with the corresponding values. We ensure that all our gadgets have a constant feedback vertex number, hence the overall feedback vertex number is quadratic in the number of colors of the Multicolored Clique instance and we get the parameterized hardness result.

In the following, for every gadget, we give a formal description of the underlying graph of this gadget (i. e., not the complete distance sub-matrix of the gadget). Due to space constraints, we defer the description of the distance matrix D and the formal proof of correctness for the reduction to [46].

Given an instance H of Multicolored Clique, we construct an instance D of Simple TGR (with infinity entries and no periods) as follows.

Edge selection gadget. We first introduce an *edge selection gadget $G_{i,j}$ for color combination i, j* with $i < j$. We start with describing the vertex set of the gadget.

- A set $X_{i,j}$ of vertices $x_1, x_2, \dots, x_m$.
- Vertex sets $U_1, U_2, \dots, U_m$ with $4n + 1$ vertices each, that is, $U_\ell = \{u^\ell_0, u^\ell_1, u^\ell_2, \dots, u^\ell_{4n}\}$ for all $\ell \in [m]$.
- Two special vertices $v^\star_{i,j}, v^{\star\star}_{i,j}$.

The gadget has the following edges.

- For all $\ell \in [m]$ we have edge $\{x_\ell, v^\star_{i,j}\}$, $\{v^\star_{i,j}, u^\ell_0\}$, and $\{u^\ell_{4n}, v^{\star\star}_{i,j}\}$.
- For all $\ell \in [m]$ and $\ell' \in [4n]$, we have edge $\{u^\ell_{\ell'-1}, u^\ell_{\ell'}\}$.

Verification gadget. For each color i, we introduce the following vertices. What we describe in the following will be used as a *verification gadget for color i*.

- We have one vertex y^i and $k + 1$ vertices v^i_ℓ for $0 \le \ell \le k$.
- For every $\ell \in [m]$ and every $j \in [k] \setminus \{i\}$ we have $5n$ vertices $a^{i,j,\ell}_1, a^{i,j,\ell}_2, \dots, a^{i,j,\ell}_{5n}$ and $5n$ vertices $b^{i,j,\ell}_1, b^{i,j,\ell}_2, \dots, b^{i,j,\ell}_{5n}$.
- We have a set $\hat{U}_i$ of $13n + 1$ vertices $\hat{u}^i_1, \hat{u}^i_2, \dots, \hat{u}^i_{13n+1}$.

We add the following edges. We add edge $\{y^i, v^i_0\}$. For every $\ell \in [m]$, every $j \in [k] \setminus \{i\}$, and every $\ell' \in [5n - 1]$ we add edge $\{a^{i,j,\ell}_{\ell'}, a^{i,j,\ell}_{\ell'+1}\}$ and we add edge $\{b^{i,j,\ell}_{\ell'}, b^{i,j,\ell}_{\ell'+1}\}$.

Let $1 \le j < i$ (skip if $i = 1$), let $e^{j,i}_\ell \in F_{j,i}$, and let $w^i_{\ell'} \in W_i$ be incident with $e^{j,i}_\ell$. Then we add edge $\{v^i_{j-1}, a^{i,j,\ell}_1\}$ and we add edge $\{a^{i,j,\ell}_{5n}, u^\ell_{\ell'-1}\}$ between $a^{i,j,\ell}_{5n}$ and the vertex $u^\ell_{\ell'-1}$ of the edge selection gadget of color combination j, i. Furthermore, we add edge $\{v^i_j, b^{i,j,\ell}_1\}$ and edge $\{b^{i,j,\ell}_{5n}, u^\ell_{\ell'}\}$ between $b^{i,j,\ell}_{5n}$ and the vertex $u^\ell_{\ell'}$ of the edge selection gadget of color combination j, i.

We add edge $\{v^i_{i-1}, \hat{u}^i_1\}$ and for all $\ell'' \in [13n]$ we add edge $\{\hat{u}^i_{\ell''}, \hat{u}^i_{\ell''+1}\}$. Furthermore, we add edge $\{\hat{u}^i_{13n+1}, v^i_i\}$.

Let $i < j \le k$ (skip if $i = k$), let $e^{i,j}_\ell \in F_{i,j}$, and let $w^i_{\ell'} \in W_i$ be incident with $e^{i,j}_\ell$. Then we add edge $\{v^i_{j-1}, a^{i,j,\ell}_1\}$ and edge $\{a^{i,j,\ell}_{5n}, u^\ell_{3n+\ell'-1}\}$ between $a^{i,j,\ell}_{5n}$ and the vertex $u^\ell_{3n+\ell'-1}$ of the edge selection gadget of color combination i, j. Furthermore, we add edge $\{v^i_j, b^{i,j,\ell}_1\}$ and edge $\{b^{i,j,\ell}_{5n}, u^\ell_{3n+\ell'}\}$ between $b^{i,j,\ell}_{5n}$ and the vertex $u^\ell_{3n+\ell'}$ of the edge selection gadget of color combination i, j.

Furthermore, we use *connector gadgets*, two for each edge selection gadget, and two for every verification gadget. They consist of six vertices $\hat{v}_0, \hat{v}'_0, \hat{v}_1, \hat{v}_2, \hat{v}_3, \hat{v}'_3$ and, intuitively, are used to connect many vertex pairs by fast paths, which will make arguing about possible

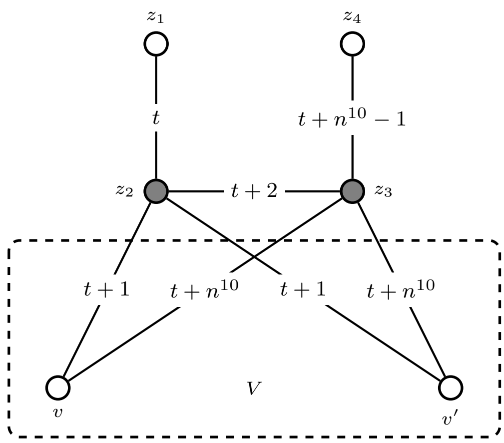


Figure 3 Illustration of the infinity gadget. Gray vertices need to be added to the feedback vertex set.

labelings in YES-instances much easier. Finally, we have an *alignment gadget*, which is a star with a center vertex $w^\star$ and a leaf for every other gadget. Intuitively, this gadget is used to relate labels of different gadgets to each other. A formal description of these two gadgets is given in [46].

This finishes the description of the underlying graph G. For an illustration see Figure 2. We can observe that the vertex set containing vertices $v^\star_{i,j}$ and $v^{\star\star}_{i,j}$ of each edge selection gadget, vertices v^i_ℓ with $0 \leq \ell \leq k$ of each verification gadget, vertices $\hat{v}_1$ and $\hat{v}_2$ of each connector gadget, and vertex $w^\star$ of the alignment gadget forms a feedback vertex set in G with size $O(k^2)$.

As mentioned before, due to space constraints, we defer the description of the distance matrix D and a formal correctness proof of the reduction to [46].

Infinity gadget. Finally, we show how to get rid of the infinity entries in D and how to allow a finite Δ. To this end, we introduce the *infinity gadget*. We add four vertices z_1, z_2, z_3, z_4 to the graph and we set $\Delta = n^{11}$. Let V denote the set of all remaining vertices. We set the following durations.

- For all $v \in V$ we set $d(z_1, v) = 2$, $d(z_2, v) = d(v, z_2) = 1$, $d(z_3, v) = d(v, z_3) = 1$, and $d(z_4, v) = 2$. Furthermore, we set $d(v, z_1) = n^{11}$ and $d(v, z_4) = n^{10} - 1$.
- We set $d(z_1, z_2) = d(z_2, z_1) = 1$, $d(z_2, z_3) = d(z_3, z_2) = 1$, and $d(z_3, z_4) = d(z_4, z_3) = 1$.
- We set $d(z_1, z_3) = 3$, $d(z_3, z_1) = n^{11} - 1$, $d(z_2, z_4) = n^{10} - 2$, and $d(z_4, z_2) = n^{11} - n^{10} + 4$.
- We set $d(z_1, z_4) = n^{10}$ and $d(z_4, z_1) = 2n^{11} - n^{10} + 2$.
- For every pair of vertices $v, v' \in V$ where previously the duration of a fastest path from v to v' was specified to be infinite, we set $d(v, v') = n^{10}$.

Now we analyse which implications we get for the labels on the newly introduced edges. Assume $\lambda(\{z_1, z_2\}) = t$, then we get the following. For all $v \in V$ we have that $d(z_1, v) = 2$ and hence we get that $\lambda(\{z_2, v\}) = t+1$. Since $d(z_1, z_4) = n^{10}$, we have that $\lambda(z_3, z_4) = t+n^{10}-1$. From this follows that for all $v \in V$, since $d(z_4, v) = 2$, that $\lambda(\{z_3, v\}) = t + n^{10}$. Finally, since $d(z_1, z_3) = 3$, we have that $\lambda(\{z_2, z_3\}) = t+2$. For an illustration see Figure 3. It is easy to check that all duration requirements between vertex pairs in $\{z_1, z_2, z_3, z_4\}$ are met and that all duration requirements between each vertex $v \in V$ and each vertex in $\{z_1, z_2, z_3, z_4\}$ are met. Furthermore, it is easy to check that the gadget increases the feedback vertex set by two (z_2 and z_3 need to be added).

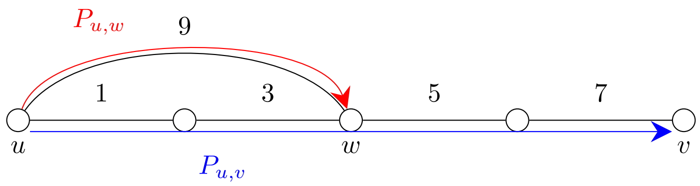


Figure 4 An example of a temporal graph (with $\Delta \geq 9$), where the fastest temporal path $P_{u,v}$ (in blue) from u to v is of duration 7, while the fastest temporal path $P_{u,w}$ (in red) from u to a vertex w, that is on a path $P_{u,v}$, is of duration 1 and is not a subpath of $P_{u,v}$.

Lastly, consider two vertices $v, v' \in V$. Note that before the addition of the infinity gadget, by construction of G we have that $d(v, v') \leq n^9 + 2$ or $d(v, v') = \infty$. Furthermore, if D is a Yes-instance, we have shown in the correctness proof of the reduction that the difference between the smallest label and the largest label is at most $n^9 + 1$. This implies that for a vertex pair $v, v' \in V$ with $d(v, v') = \infty$ we have in the periodic case with $\Delta = n^{11}$, that $d(v, v') \geq n^{11} - n^9 > n^{10}$. Which means, after adding the vertices and edges of the infinity gadget, we indeed have that $d(v, v') = n^{10}$. For all vertex pairs v, v' where in the original construction we have $d(v, v') \neq \infty$, we can also see that adding the infinity gadget and setting $\Delta = n^{11}$ does not change the duration of a fastest path from v to v', since all newly added temporal paths have duration at least n^{10}. We can conclude that the originally constructed instance D is a Yes-instance if and only if it remains a Yes-instance after adding the infinity gadget and setting $\Delta = n^{11}$. ◀

3 Algorithms for Simple TGR

In this section, to complement the discussed hardness aspects of Simple TGR, we present some algorithmic results. We start by restricting the underlying graph G of the input matrix D of Simple TGR to be a tree and get the following.

▶ **Theorem 5** ($\star$). *Simple TGR can be solved in polynomial time on trees.*

The main reason, for which Simple TGR is straightforward to solve on trees, is twofold:
- between any pair of vertices v_i and v_j in the tree T, there is a *unique* path P in T from v_i to v_j, and
- in any periodic temporal graph (T, λ, Δ) and any fastest temporal path $P = ((e_1, t_1), \ldots, (e_i, t_i), \ldots, (e_j, t_j), \ldots, (e_{\ell-1}, t_{\ell-1}))$ from v_1 to v_ℓ we have that the sub-path $P' = ((e_i, t_i), \ldots, (e_{j-1}, t_{j-1}))$ is also a fastest temporal path from v_i to v_j.

However, these two nice properties do not hold when the underlying graph is not a tree. For example, in Figure 4, the fastest temporal path from u to v is $P_{u,v}$ (depicted in blue) goes through w, however the sub-path of $P_{u,v}$ that stops at w is not the fastest temporal path from u to w. The fastest temporal path from u to w consists only of the single edge uw (with label 9 and duration 1, depicted in red).

Nevertheless, we prove that we can still solve Simple TGR efficiently if the underlying graph is similar to a tree; more specifically we show the following result, which turns out to be non-trivial.

▶ **Theorem 6** ($\star$). *Simple TGR is in FPT when parameterized by the feedback edge number of the underlying graph.*

From Theorem 4 and Theorem 6 we immediately get the following, which is the main result of the paper.

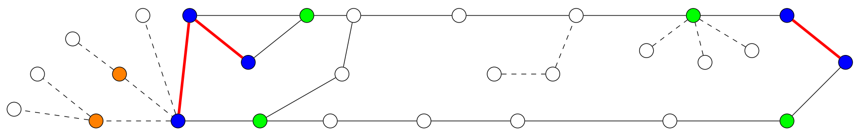


Figure 5 An example of a graph with its important vertices: U (in blue), U^* (in green) and Z^* (in orange). Corresponding feedback edges are marked with a thick red line, while dashed edges represent the edges (and vertices) "removed" from G' at the initial step.

▶ **Corollary 7.** *Simple TGR is:*

- *in FPT when parameterized by the* feedback edge number *or any larger parameter, such as the* maximum leaf number*.*
- *W[1]-hard when parameterized by the* feedback vertex number *or any smaller parameter, such as:* treewidth*,* degeneracy*,* cliquewidth*,* distance to chordal graphs*, and* distance to outerplanar graphs*.*

Before presenting the structure of our algorithm for Theorem 6, observe that, in a static graph, the number of paths between two vertices can be upper-bounded by a function $f(k)$ of the feedback edge number k of the graph [14]. Therefore, for any fixed pair of vertices u and v, we can "guess" the edges of the fastest temporal path from u to v (by guess we mean enumerate and test all possibilities). However, for an FPT algorithm with respect to k, we cannot afford to guess the edges of the fastest temporal path for each of the $O(n^2)$ pairs of vertices. To overcome this difficulty, our algorithm follows this high-level strategy:

- We identify a small number $f(k)$ of "important vertices".
- For each pair u, v of important vertices, we guess the edges of the fastest temporal path from u to v (and from v to u).
- From these guesses we can still not deduce the edges of the fastest temporal paths between many pairs of non-important vertices. However, as we prove, it suffices to guess only a small number of specific auxiliary structures (to be defined later).
- From these guesses we deduce fixed relationships between the labels of most of the edges of the graph.
- For all the edges, for which we have not deduced a label yet, we introduce a *variable*. With all these variables, we build an Integer Linear Program (ILP). Among the constraints in this ILP we have that, for each of the $O(n^2)$ pairs of vertices u, v in the graph, the duration of one specific temporal path from u to v (according to our guesses) is *equal* to the desired duration $D_{u,v}$, while the duration of each of the other temporal path from u to v is *at least* $D_{u,v}$.
- By making each of the above combinations of guesses, we essentially enumerate all possible ways that our instance of Simple TGR has a solution, and for each of these possible ways we create an ILP. That is, our instance of Simple TGR has a solution if and only if at least one of these ILPs has a feasible solution. As each ILP can be solved in FPT time with respect to k by Lenstra's Theorem [49] (the number of variables is upper bounded by a function of k), we obtain our FPT algorithm for Simple TGR with respect to k.

We now present the first part of our FPT algorithm, that is, identifying important vertices and guessing information about the fastest temporal paths. A full description of the algorithm is deferred to [46].

Important vertices. Let D be the input matrix of Simple TGR, and let G be its underlying graph, on n vertices and m edges. From the underlying graph G of D we first create a graph

G' by iteratively removing vertices of degree one from G, and denote with $Z = V(G) \setminus V(G')$, the set of removed vertices. Then we determine the set U (the *"vertices of interest"*), and the set U^* (the neighbors of the vertices of interest), as follows. Let T be a spanning tree of G', with F being the corresponding feedback edge set of G'. Let $V_1 \subseteq V(G')$ be the set of leaves in the spanning tree T, $V_2 \subseteq V(G')$ be the set of vertices of degree two in T which are incident to at least one edge in F, and let $V_3 \subseteq V(G')$ be the set of vertices of degree at least 3 in T. Then $|V_1| + |V_2| \le 2k$, since every leaf in T and every vertex in V_2 is incident to at least one edge in F, and $|V_3| \le |V_1|$ by the properties of trees. We denote with

$$U = V_1 \cup V_2 \cup V_3$$

the set of *vertices of interest*. It follows that $|U| \le 4k$. We set U^* to be the set of vertices in $V(G') \setminus U$ that are neighbors of vertices in U, i. e.,

$$U^* = \{v \in V(G') \setminus U : u \in U, v \in N(u)\}.$$

Again, using the tree structure, we get that for any $u \in U$ its neighborhood is of size $|N(u)| \in O(k)$, since every neighbor of u is the first vertex of a (unique) path to another vertex in U. It follows that $|U^*| \in O(k^2)$. From the construction of Z (i. e., by exhaustively removing vertices of degree one from G), it follows that $G[Z]$ (the graph induced in G by Z) is a forest, i. e., consists of disjoint trees. Each of these trees has a unique neighbor v in G'. Denote by T_v the tree obtained by considering such a vertex v and all the trees from $G[Z]$ that are incident to v in G. We then refer to v as the *clip vertex* of the tree T_v. In the case where v is a vertex of interest we define also the set Z_v^* of *representative vertices* of T_v, as follows. We first create an empty set C_w for every vertex w that is a neighbor of v in G'. We then iterate through every vertex r that is in the first layer of the tree T_v (i. e., vertex that is a child of the root v in the tree T_v), check the matrix D and find the vertex $w \in N_{G'}(v)$ that is on the smallest duration from r. In other words, for an $r \in N_{T_v}(v)$ we find $w \in N_{G'}(v)$ such that $D_{r,w} \le D_{r,w'}$ for all $w' \in N_{G'}(v)$. We add vertex r to C_w. In the case when there exists also another vertex $w' \in N_{G'}(v)$ for which $D_{r,w'} = D_{r,w}$, we add r also to the set $C_{w'}$. In fact, in this case $C_{w'} = C_w$. At the end we create $|N_{G'}(v)| \in O(k)$ sets C_w, whose union contains all children of v in T_v. For every two sets C_w and $C_{w'}$, where $w, w' \in N_{G'}(v)$, we have that either $C_w = C_{w'}$, or $C_w \cap C_{w'} = \emptyset$. We interpret each of these sets $\{C_w : w \in N_{G'}(v)\}$ as an *equivalence class* of the neighbors of v in the tree T_v. Now, from each equivalence class C_w we choose an arbitrary vertex $r_w \in C_w$ and put it into the set Z_v^*. We repeat the above procedure for all trees T_u with the clip vertex u from U, and define Z^* as

$$Z^* = \bigcup_{v \in U} Z_v^*. \tag{1}$$

Since $|U| \in O(k)$ and for each $u \in U$ it holds $|N_{G'}(u)| \in O(k)$, we get that $|Z^*| \in O(k^2)$. Finally, the set of *important vertices* is defined as the set $U \cup U^* \cup Z^*$. For an illustration see Figure 5.

Guesses. For every pair of important vertices $u, v \in U \cup U^* \cup Z^*$, we guess the sequence of edges in the fastest temporal path from u to v. Since $U \cup U^* \cup Z^* \in O(k^2)$ and there are $k^{O(k)}$ possibilities for a sequence of edges between a fixed vertex pair, we have $k^{O(k^5)}$ overall possible guesses. We defer further details to [46] (see guesses **G-1** to **G-6**).

With the information provided by the described guesses we are still not able to determine all fastest paths. For example consider the case depicted in Figure 6. Therefore we introduce additional guesses that provide us with sufficient information to determine all fastest paths. To do this we have to first define the following.

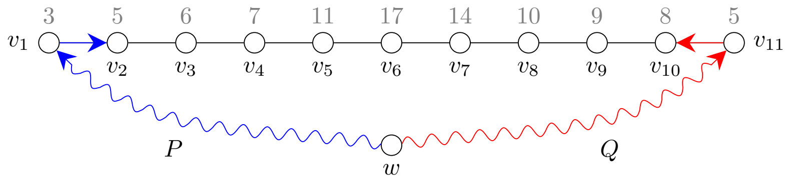


Figure 6 In the above graph vertices v_1, v_{11}, w are in U, while v_2, v_{10} are in U^*. Numbers above all v_i represent the values of the fastest temporal paths from w to each of them (i. e., the entries in the w-th row of matrix D). From the basic guesses we know the fastest temporal path P from w to v_2 (depicted in blue) and the fastest temporal path Q from w to v_{10}. From the values of durations from w to each v_i we cannot determine the fastest paths from w to all v_i. More precisely, we know that w reaches v_2, v_3, v_4, v_5 (resp. v_{10}, v_9, v_9, v_7) by first using the path P (resp. Q) and then proceeding through the vertices, but we do not know how w reaches v_6 the fastest. Therefore we have to introduce some more guesses.

▶ **Definition 8.** *Let $U \subseteq V(G')$ be a set of vertices of interest and let $u, v \in U$. A path $P = (u = v_1, v_2, \ldots, v_p = v)$ of length at least 2 in graph G', where all inner vertices are not in U, i. e., $v_i \notin U$ for all $i \in \{2, 3, \ldots, p-1\}$, is called a* segment *from u to v. We denote it as $S_{u,v}$.*

Note by Definition 8 that $S_{u,v} \neq S_{v,u}$. Observe that a temporal path in G' between two vertices of interest is either a segment, or it consists of a sequence of some segments. Furthermore, since we have at most $4k$ interesting vertices in G', we can deduce the following important result.

▶ **Corollary 9.** *There are $O(k^2)$ segments in G'.*

To describe the next guesses, we introduce the following notation. Let u, v, x be three vertices in G'. We write $u \rightsquigarrow x \to v$ to denote a temporal path from u to v that passes through x, and then goes to v (via one edge). We guess the following structures.

G-7. Inner segment guess I. Let $S_{u,v} = (u = v_1, v_2, \ldots, v_p = v)$ and $S_{w,z} = (w = z_1, z_2, \ldots, z_r = z)$ be two segments in G'. We want to guess the fastest temporal path $v_2 \to u \rightsquigarrow w \to z_2$. We repeat this procedure for all pairs of segments. Since there are $O(k^2)$ segments in G', there are $k^{O(k^5)}$ possible paths of this form.
Recall that $S_{u,v} \neq S_{v,u}$ for every $u, v \in U$. Furthermore note that we did not assume that $\{u, v\} \cap \{w, z\} = \emptyset$. Therefore, by repeatedly making the above guesses, we also guess the following fastest temporal paths: $v_2 \to u \rightsquigarrow z \to z_{r-1}$, $v_2 \to u \rightsquigarrow v \to v_{p-1}$, $v_{p-1} \to v \rightsquigarrow w \to z_2$, $v_{p-1} \to v \rightsquigarrow z \to z_{r-1}$, and $v_{p-1} \to v \rightsquigarrow u \to v_2$. For an example see Figure 7a.

G-8. Inner segment guess II. Let $S_{u,v} = (u = v_1, v_2, \ldots, v_p = v)$ be a segment in G', and let $w \in U \cup Z^*$. We want to guess the following fastest temporal paths $w \rightsquigarrow u \to v_2$, $w \rightsquigarrow v \to v_{p-1} \to \cdots \to v_2$, and $v_2 \to u \rightsquigarrow w$, $v_2 \to v_3 \to \cdots v \rightsquigarrow w$.
For fixed $S_{u,v}$ and $w \in U \cup Z^*$ we have $k^{O(k)}$ different possible such paths, therefore we make $k^{O(k^5)}$ guesses for these paths. For an example see Figure 7b.

G-9. Split vertex guess I. Let $S_{u,v} = (u = v_1, v_2, \ldots, v_p = v)$ be a segment in G', and let us fix a vertex $v_i \in S_{u,v} \setminus \{u, v\}$. In the case when $S_{u,v}$ is of length 4, the fixed vertex v_i is the middle vertex, else we fix an arbitrary vertex $v_i \in S_{u,v} \setminus \{u, v\}$. Let $S_{w,z} = (w = z_1, z_2, \ldots, z_r = z)$ be another segment in G'. We want to determine the fastest paths from v_i to all inner vertices of $S_{w,z}$. We do this by inspecting the values in matrix D from v_i to inner vertices of $S_{w,z}$. We split the analysis into two cases.

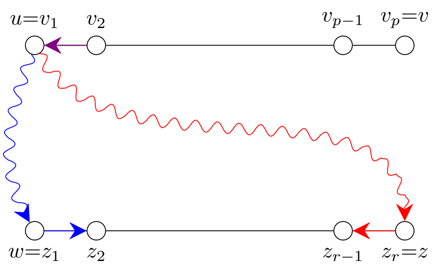
a. There is a single vertex $z_j \in S_{w,z}$ for which the duration from v_i is the biggest. More specifically, $z_j \in S_{w,z} \setminus \{w, z\}$ is the vertex with the biggest value D_{v_i,z_j}. We call this vertex a *split vertex of v_i in the segment S_{wz}*. Then it holds that $D_{v_i,z_2} < D_{v_i,z_3} < \cdots < D_{v_i,z_j}$ and $D_{v_i,z_{r-1}} < D_{v_i,z_{r-2}} < \cdots < D_{v_i,z_j}$. From this it follows that the fastest temporal paths from v_i to $z_2, z_3, \ldots, z_{j-1}$ go through w, and the fastest temporal paths from v_i to $z_{r-1}, z_{r-2}, \ldots, z_{j+1}$ go through z. We now want to guess which vertex w or z is on a fastest temporal path from v_i to z_j. Similarly, all fastest temporal paths starting at v_i have to go either through u or through v, which also gives us two extra guesses for the fastest temporal path from v_i to z_j. Therefore, all together we have 4 possibilities on how the fastest temporal path from v_i to z_j starts and ends. Besides that we want to guess also how the fastest temporal paths from v_i to z_{j-1}, z_{j+1} start and end. Note that one of these is the subpath of the fastest temporal path from v_i to z_j, and the ending part is uniquely determined for both of them, i. e., to reach z_{j-1} the fastest temporal path travels through w, and to reach z_{j+1} the fastest temporal path travels through z. Therefore we have to determine only how the path starts, namely if it travels through u or v. This introduces two extra guesses. For a fixed $S_{u,v}, v_i$ and $S_{w,z}$ we find the vertex z_j in polynomial time, or determine that z_j does not exist. We then make four guesses where we determine how the fastest temporal path from v_i to z_j passes through vertices u, v and w, z and for each of them two extra guesses to determine the fastest temporal path from v_i to z_{j-1} and from v_i to z_{j+1}. We repeat this procedure for all pairs of segments, which results in producing $k^{O(k^5)}$ new guesses. Note, $v_i \in S_{u,v}$ is fixed when calculating the split vertex for all other segments $S_{w,z}$.

b. There are two vertices $z_j, z_{j+1} \in S_{w,z}$ for which the duration from v_i is the biggest. More specifically, $z_j, z_{j+1} \in S_{w,z} \setminus \{w, z\}$ are the vertices with the biggest value $D_{v_i,z_j} = D_{v_i,z_{j+1}}$. Then it holds that $D_{v_i,z_2} < D_{v_i,z_3} < \cdots < D_{v_i,z_j} = D_{v_i,z_{j+1}} > D_{v_i,z_{j+2}} > \cdots > D_{v_i,z_{r-1}}$. From this it follows that the fastest temporal paths from v_i to $z_2, z_3, \ldots, z_j$ go through w, and the fastest temporal paths from v_i to $z_{r-1}, z_{r-2}, \ldots, z_{j+1}$ go through z. In this case we only need to guess the following two fastest temporal paths $v_i \rightsquigarrow w \rightarrow z_2$ and $v_i \rightsquigarrow z \rightarrow z_{r-1}$. Each of these paths we then uniquely extend along the segment $S_{w,z}$ up to the vertex z_j, resp. z_{j+1}, which give us fastest temporal paths from v_i to z_j and from v_i to z_{j+1}. In this case we introduce only two more guesses. We repeat this procedure for all pairs of segments. which results in creating $k^{O(k^5)}$ new guesses.

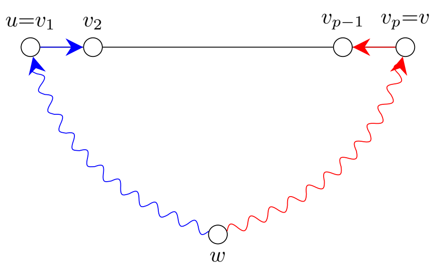
For an example see Figure 7b.

G-10. Split vertex guess II. Let $w \in U \cup Z^*$ and let $S_{u,v} = (u = v_1, v_2, \ldots, v_p = v)$. We want to guess a split vertex of w in $S_{u,v}$, and the fastest temporal path that reaches it. We again have two cases, first one where v_i is a unique vertex in $S_{u,v}$ that is furthest away from w, and the second one where v_i, v_{i+1} are two incident vertices in $S_{u,v}$, that are furthest away from w. All together we make two guesses for each pair $w, S_{u,v}$. We repeat this for all vertices in $U \cup Z^*$, and all segments, which produces $k^{O(k^5)}$ new guesses. For an example see Figure 7c. Detailed analysis follows arguing from above (as in **G-9**) and is deferred to [46].

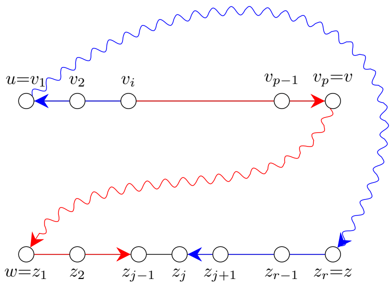
There are two more guesses **G-11** and **G-12** that are deferred to [46]. We prove in [46] that, for all guesses **G-1** to **G-12**, there are in total at most $f(k)$ possible choices, and for each one of them we create an ILP with at most $f(k)$ variables and at most $f(k) \cdot |D|^{O(1)}$ constraints. Each of these ILPs can be solved in FPT time by Lenstra's Theorem [49]. For detailed explanation and proofs of this part see [46].



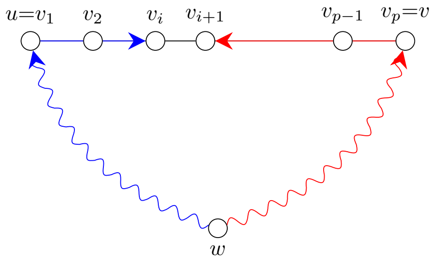
(a) Example of an Inner segment guess I (**G-7**), where we guessed the fastest temporal paths of the form $v_2 \to u \rightsquigarrow w \to z_2$ (in blue) and $v_2 \to u \rightsquigarrow z \to z_{r-1}$ (in red).



(b) Example of an Inner segment guess II (**G-8**), where we guessed the fastest temporal paths of the form $w \rightsquigarrow u \to v_2$ (in blue) and $w \rightsquigarrow v \to v_{p-1}$ (in red).



(c) Example of a Split vertex guess I (**G-9**), where, for a fixed vertex $v_i \in S_{u,v}$, we calculated its corresponding split vertex $z_j \in S_{w,z}$, and guessed the fastest paths of the form $v_i \to v_{i-1} \to \cdots \to u \rightsquigarrow z \to z_{r-1} \cdots \to z_j$ (in blue) and $v_i \to v_{i+1} \to \cdots \to v \rightsquigarrow w \to z_2 \to \cdots \to z_{j-1}$ (in red).



(d) Example of a Split vertex guess II (**G-10**), where, for a vertex of interest w, we calculated its corresponding split vertex $v_i \in S_{u,v}$, and guessed the fastest paths of the form $w \rightsquigarrow u \to v_2 \to \cdots \to v_i$ (in blue) and $w \rightsquigarrow v \to v_{p-1} \to \cdots \to v_{i+1}$ (in red).

Figure 7 Illustration of the guesses **G-7**, **G-8**, **G-9**, and **G-10**.

4 Conclusion

We believe that our work spawns several interesting future research directions and builds a base upon which further temporal graph realization problems can be investigated.

There are several structural parameters which can be considered to obtain tractability which are either larger than or incomparable to the feedback vertex number. We believe that the *vertex cover number* or the *tree depth* are promising candidates. Furthermore, we can consider combining a structural parameter such as the *treewidth* with Δ.

There are many natural variants of our problem that are well-motivated and warrant consideration. We believe that one of the most natural generalizations of our problem is to allow more than one label per edge in every Δ-period. A well-motivated variant (especially from the network design perspective) of our problem is to consider the entries of the duration matrix D as upper-bounds on the duration of fastest paths rather than exact durations. This problem variant has very recently been studied by Mertzios et al. [56].

References

1 Eleni C Akrida, Leszek Gąsieniec, George B. Mertzios, and Paul G Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61:907–944, 2017.

2 Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Christoforos Raptopoulos. The temporal explorer who returns to the base. *Journal of Computer and System Sciences*, 120:179–193, 2021.

3 Emmanuel Arrighi, Niels Grüttemeier, Nils Morawietz, Frank Sommer, and Petra Wolf. Multi-parameter analysis of finding minors and subgraphs in edge-periodic temporal graphs. In *Proceedings of the 48th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 283–297, 2023.

4 John Augustine, Keerti Choudhary, Avi Cohen, David Peleg, Sumathi Sivasubramaniam, and Suman Sourav. Distributed graph realizations. *IEEE Transactions on Parallel and Distributed Systems*, 33(6):1321–1337, 2022.

5 Amotz Bar-Noy, Keerti Choudhary, David Peleg, and Dror Rawitz. Efficiently realizing interval sequences. *SIAM Journal on Discrete Mathematics*, 34(4):2318–2337, 2020.

6 Amotz Bar-Noy, Keerti Choudhary, David Peleg, and Dror Rawitz. Graph realizations: Maximum degree in vertex neighborhoods. In *Proceedings of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 10:1–10:17, 2020.

7 Amotz Bar-Noy, David Peleg, Mor Perry, and Dror Rawitz. Composed degree-distance realizations of graphs. In *Proceedings of the 32nd International Workshop on Combinatorial Algorithms (IWOCA)*, pages 63–77, 2021.

8 Amotz Bar-Noy, David Peleg, Mor Perry, and Dror Rawitz. Graph realization of distance sets. In *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 13:1–13:14, 2022.

9 Mehdi Behzad and James E Simpson. Eccentric sequences and eccentric sets in graphs. *Discrete Mathematics*, 16(3):187–193, 1976.

10 Robert E Bixby and Donald K Wagner. An almost linear-time algorithm for graph realization. *Mathematics of Operations Research*, 13(1):99–123, 1988.

11 Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.

12 Arnaud Casteigts, Timothée Corsini, and Writika Sarkar. Invited paper: Simple, strict, proper, happy: A study of reachability in temporal graphs. In *Proceedings of the 24th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 3–18, 2022.

13 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

14 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9):2754–2802, 2021.

15 Wai-Kai Chen. On the realization of a (p, s)-digraph with prescribed degrees. *Journal of the Franklin Institute*, 281(5):406–422, 1966.

16 Fan Chung, Mark Garrett, Ronald Graham, and David Shallcross. Distance realization problems with applications to internet tomography. *Journal of Computer and System Sciences*, 63(3):432–448, 2001.

17 Joseph C. Culberson and Piotr Rudnicki. A fast algorithm for constructing trees from distance matrices. *Information Processing Letters*, 30(4):215–220, 1989.

18 Argyrios Deligkas and Igor Potapov. Optimizing reachability sets in temporal graphs by delaying. *Information and Computation*, 285:104890, 2022.

19 Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021.

20 Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021.

21 Jessica A. Enright, Kitty Meeks, and Hendrik Molter. Counting temporal paths. In *Proceedings of the 40th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 254, pages 30:1–30:19, 2023.

22 Paul Erdős and Tibor Gallai. Graphs with prescribed degrees of vertices. *Mat. Lapok*, 11:264–274, 1960.

23 Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. *Journal of Computer and System Sciences*, 119:1–18, 2021.

24 Thomas Erlebach, Nils Morawietz, and Petra Wolf. Parameterized algorithms for multi-label periodic temporal graph realization. In *Proceedings of the 3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 14:1–14:16, 2024. `doi:10.4230/LIPIcs.SAND.2024.14`.

25 Thomas Erlebach and Jakob T. Spooner. A game of cops and robbers on graphs with periodic edge-connectivity. In *Proceedings of the 46th International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*, pages 64–75, 2020.

26 Michael R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

27 Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013.

28 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020.

29 András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992.

30 András Frank. Connectivity augmentation problems in network design. *Mathematical Programming: State of the Art 1994*, 1994.

31 H. Frank and Wushow Chou. Connectivity considerations in the design of survivable networks. *IEEE Transactions on Circuit Theory*, 17(4):486–490, 1970.

32 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Delay-robust routes in temporal graphs. In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 30:1–30:15, 2022.

33 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Temporal connectivity: Coping with foreseen and unforeseen delays. In *Proceedings of the 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 17:1–17:17, 2022.

34 D.R. Fulkerson. Zero-one matrices with zero trace. *Pacific Journal of Mathematics*, 10(3):831–836, 1960.

35 Petr A. Golovach and George B. Mertzios. Graph editing to a given degree sequence. *Theoretical Computer Science*, 665:1–12, 2017.

36 Martin Charles Golumbic and Ann N. Trenk. *Tolerance Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2004.

37 Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

38 Martin Grötschel, Clyde L Monma, and Mechthild Stoer. Design of survivable networks. *Handbooks in Operations Research and Management Science*, 7:617–672, 1995.

39 Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proceedings of the 1st International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 162–173, 2004.

40 S. Louis Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962.

41 S. Louis Hakimi and Stephen S. Yau. Distance matrix of a graph and its realizability. *Quarterly of applied mathematics*, 22(4):305–317, 1965.

42 Pavol Hell and David Kirkpatrick. Linear-time certifying algorithms for near-graphical sequences. *Discrete Mathematics*, 309(18):5703–5713, 2009.

43 David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.

44 Nina Klobas, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Interference-free walks in time: Temporally disjoint paths. *Autonomous Agents and Multi-Agent Systems*, 37(1):1, 2023.

45 Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. The complexity of computing optimum labelings for temporal connectivity. In *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 62:1–62:15, 2022.

46 Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. Realizing temporal graphs from fastest travel times. *CoRR*, abs/2302.08860, 2023. URL: `https://doi.org/10.48550/arXiv.2302.08860`, `arXiv:2302.08860`.

47 Fabian Kuhn and Rotem Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, mar 2011.

48 Pascal Kunz, Hendrik Molter, and Meirav Zehavi. In which graph structures can we efficiently find temporally disjoint paths and walks? In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 180–188, 2023.

49 Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

50 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

51 Linda Lesniak. Eccentric sequences in graphs. *Periodica Mathematica Hungarica*, 6:287–293, 1975.

52 Ross M. McConnell and Jeremy P. Spinrad. Construction of probe interval models. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 866–875, 2002.

53 F.R. McMorris, Chi Wang, and Peisen Zhang. On probe interval graphs. *Discrete Applied Mathematics*, 88(1):315–324, 1998. Computational Molecular Biology DAM - CMB Series.

54 George B. Mertzios, Othon Michail, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019.

55 George B. Mertzios, Hendrik Molter, Malte Renken, Paul G. Spirakis, and Philipp Zschoche. The complexity of transitively orienting temporal graphs. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 75:1–75:18, 2021.

56 George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. Realizing temporal transportation trees. *CoRR*, abs/2403.18513, 2024. URL: `https://doi.org/10.48550/arXiv.2403.18513`, `arXiv:2403.18513`.

57 Hendrik Molter, Malte Renken, and Philipp Zschoche. Temporal reachability minimization: Delaying vs. deleting. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 76:1–76:15, 2021.

58 Nils Morawietz, Carolin Rehs, and Mathias Weller. A timecop's work is harder than you think. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 170, pages 71–1, 2020.

59 Nils Morawietz and Petra Wolf. A timecop's chase around the table. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2021.

60 A.N. Patrinos and S. Louis Hakimi. The distance matrix of a graph and its tree realization. *Quarterly of Applied Mathematics*, 30:255–269, 1972.

61 Elena Rubei. Weighted graphs with distances in given ranges. *Journal of Classification*, 33:282—-297, 2016.

62 Piotr Sapiezynski, Arkadiusz Stopczynski, Radu Gatej, and Sune Lehmann. Tracking human mobility using wifi signals. *PloS one*, 10(7):e0130824, 2015.

63 H. Tamura, M. Sengoku, S. Shinoda, and T. Abe. Realization of a network from the upper and lower bounds of the distances (or capacities) between vertices. In *Proceedings of the 1993 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2545—-2548, 1993.

64 Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016.

65 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020.