Realizing temporal transportation trees

George B. Mertzios[1⋆[0000−0001−7182−585X]],
Hendrik Molter[2⋆⋆[0000−0002−4590−798X]],
Nils Morawietz[3,4⋆⋆⋆[0000−0002−7283−4982]], and
Paul G. Spirakis[5†[0000−0001−5396−3749]]

[1] Department of Computer Science, Durham University, UK
george.mertzios@durham.ac.uk
[2] Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel molterh@post.bgu.ac.il
[3] Institute of Computer Science, Friedrich Schiller University Jena, Germany
[4] LaBRI, Université de Bordeaux, France nils.morawietz@uni-jena.de
[5] Department of Computer Science, University of Liverpool, UK
p.spirakis@liverpool.ac.uk

Abstract. In this paper, we study the complexity of the *periodic temporal graph realization* problem with respect to *upper bounds* on the fastest path durations among its vertices. This constraint with respect to upper bounds appears naturally in *transportation network design* applications where, for example, a transportation network is given, and the goal is to appropriately schedule periodic travel routes, while not exceeding some desired upper bounds on the travel times. In our work, we focus only on underlying *tree topologies*, which are fundamental in many transportation network applications.

As it turns out, the periodic upper-bounded temporal tree realization problem (TTR) has a very different computational complexity behavior than both (i) the classic graph realization problem with respect to shortest path distances in static graphs and (ii) the periodic temporal graph realization problem with *exact* given fastest travel times (which was recently introduced). First, we prove that, surprisingly, TTR is NP-hard, even for a constant period Δ and when the input tree G satisfies at least one of the following conditions: (a) G is a star, or (b) G has constant maximum degree. Second, we prove that TTR is fixed-parameter tractable (FPT) with respect to the number of leaves in the input tree G, via a novel combination of techniques for totally unimodular matrices and mixed integer linear programming.

Keywords: Temporal graph, periodic temporal labeling, fastest temporal path, graph realization, Mixed Integer Linear Programming.

⋆ Corresponding author. Supported by the EPSRC grant EP/P020372/1.
⋆⋆ Supported by the ISF, grant nr. 1470/24, by the European Union's Horizon Europe research and innovation programme under grant agreement 949707, and by the European Research Council, grant nr. 101039913 (PARAPATH).
⋆⋆⋆ Partially supported by the French ANR, project ANR-22-CE48-0001 (TEMPOGRAL)
† Supported by the EPSRC grant EP/P02002X/1.

1 Introduction

A temporal (or dynamic) graph is a graph whose topology is subject to discrete changes over time. This paradigm reflects the structure and operation of a great variety of modern networks; social networks, wired or wireless networks whose links change dynamically, transportation networks, and several physical systems are only a few examples of networks that change over time [33, 41, 44]. Inspired by the foundational work of Kempe et al. [34], we adopt here a simple model for temporal graphs, in which the vertex set remains unchanged while each edge is equipped with a set of integer time labels.

Definition 1 (temporal graph [34]). *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* 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 a path of the underlying static graph whose time labels strictly increase along the edges of the path. The *duration* of a temporal path is the number of discrete time steps needed to traverse it. Finally, a temporal path from vertex u to vertex v is *fastest* if it has the smallest duration among all temporal paths from u to v, see Figure 1 for an illustration.

The *graph realization* problem with respect to some graph property $\mathcal{P}$ is to compute a graph that satisfies $\mathcal{P}$, or to decide that no such graph exists. The main motivation for graph realization problems stems both from network design and from "verification" applications in engineering. In *network design* applications, the goal is to design network topologies having a desired property [2, 29]. On the other hand, in *verification* applications, given the outcomes of some exact experimental measurements, the aim is to (re)construct an input network that complies with them. If such a reconstruction is not possible, this proves that the measurements are incorrect or implausible (resp. that the algorithm making the computations is incorrectly implemented).

One example of a graph realization problem is the recognition of probe interval graphs, in the context of the physical mapping of DNA, see [38, 39] and [27, Chapter 4]. 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 parameters for graph realization are constraints on the distances between vertices [6, 7, 10, 15, 16, 31, 45, 46, 48], on the vertex degrees [5, 18, 26, 28, 30], on the eccentricities [4, 9, 32, 37], and on connectivity [14, 22–25, 28], among others. Although the majority of studies of graph realization problems concern *static* graphs, the *temporal* (periodic) graph realization problem with respect to given (exact) delays of the fastest temporal paths among vertices has been recently studied in [20, 35], motivated by *verification* applications where exact measurements are given.

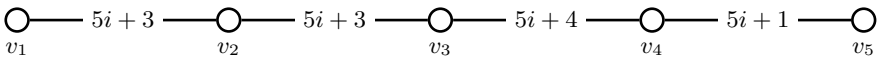


Fig. 1: Visualization of a Δ-periodic temporal graph (G, λ, Δ) with $\Delta = 5$ and the following Δ-periodic labeling $\lambda : E \to \{1, 2, \ldots, 5\}$: $\lambda(\{v_1, v_2\}) = \lambda(\{v_2, v_3\}) = 3$, $\lambda(\{v_3, v_4\}) = 4$, and $\lambda(\{v_4, v_5\}) = 1$. A fastest temporal path from v_1 to v_5 first traverses $\{v_1, v_2\}$ at time 3, then $\{v_2, v_3\}$ a time 8, then $\{v_3, v_4\}$ at time 9, and then $\{v_4, v_5\}$ at time 11, and has duration 9.

In this paper, we consider the (periodic) temporal graph realization problem with respect to the durations of the fastest temporal paths from a *network design* perspective, i. e., where only *upper bounds* on the durations of fastest temporal paths are given. One important application domain of network design problems is that of *transportation network* design where, for example, a road network is given, and the goal is to appropriately schedule periodic travel routes while not exceeding some desired upper bounds on travel times. Our work is motivated by the fact that in many transportation network applications the underlying graph has a *tree* structure [3,8,11]. For example, many airlines or railway transportation networks are even star graphs (having a big hub at the center of the network, e.g. in the capital city). The formal definition of our problem is as follows (see Section 2 for formal definitions of all used terminology).

Periodic Upper-Bounded Temporal Tree Realization (TTR)

Input: A tree $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$, an $n \times n$ matrix D of positive integers, and a positive integer Δ.

Question: Does there exist 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 *at most* $D_{i,j}$.

Many natural and technological systems exhibit a periodic temporal behavior. This is true even more on transportation networks [1], where the goal is to build periodic schedules of transportation units (e.g. trains, buses, or airplanes). The most natural constraint, while designing such periodic transportation schedules, is that the fastest travel time (i. e., the duration of the fastest temporal path) between a specific pair of vertices does not exceed a specific desired upper bound. That is, if the travel time between u and v in the resulting schedule is shorter than this upper bound, the schedule is even better (and thus, still feasible). Periodic temporal graphs have also been studied in various different contexts (see [12, Class 8] and [1, 19, 21, 42, 43]).

We focus on the most fundamental case of periodic temporal graphs, where every edge has the same period Δ and it appears exactly once within each period. As it turns out, TTR has a very different computational complexity behavior than both (i) the classic graph realization problem with respect to shortest path distances in static graphs [31] and (ii) the periodic temporal graph realization

problem with *exact* given fastest travel times [35], both of which are polynomial-time solvable on trees. We remark that recently, the case where every edge can appear multiple times in each Δ period has been studied [20]. We defer the discussion of additional related work to the full version [40].

Our contribution. Our results in this paper are given in two main directions. First, we prove that TTR is NP-hard, even for a constant period Δ and when the input tree G satisfies at least one of the following conditions: (a) G is a star (for any $\Delta \geq 3$), or (b) G has diameter at most 6 and constant pathwidth (even for $\Delta = 2$), or (c) G has maximum degree at most 8 and constant pathwidth (even for $\Delta = 2$). Note that, for $\Delta = 1$, TTR becomes trivial (as in this case fastest paths coincide with shortest paths in the underlying tree). On the one hand, our hardness results come in wide contrast to the classic (static) graph realization problem with respect to shortest path distances. Indeed, this static analogue is easily solvable in polynomial time in two steps [31]. On the other hand, the complexity of TTR is also surprisingly different than the periodic temporal graph realization problem with *exact* given fastest travel times. More specifically, when the input integer matrix D gives the exact fastest travel times, the problem becomes solvable in polynomial time on trees [35]. Note that our hardness results rule out the existence of FPT-algorithms for TTR for all reasonable parameters on trees besides the number of leaves (assuming P $\neq$ NP).

Second, we prove that TTR is fixed-parameter tractable (FPT) with respect to the number of leaves in the input tree G. That is, long chains of vertices of degree 2 do not affect the complexity of the problem. To provide our FPT algorithm, we reduce TTR to a number Mixed Integer Linear Program (MILP) instances that is upper-bounded by a function of the number of leaves in the input tree. Furthermore, the number of integer variables in each MILP instance is also upper-bounded by a function of the number of leaves in the input tree. This allows us to use a known FPT algorithm for MILP parameterized by the number of integer variables [17, 36] to solve all MILP instances in FPT-time with respect to the number of leaves of the input tree. We prove that the TTR instance is a yes-instance if and only if at least one of the MILP instances admits a feasible solution. To this end, we use a novel combination of techniques for totally unimodular matrices and mixed integer linear programming to design the MILP instances in a way that if they admit a feasible solution, then we can assume that all variables are set to integer values.

Proofs of results marked with $\star$ are (partially) deferred to the full version of the paper [40].

2 Preliminaries and Notation

An undirected graph $G = (V, E)$ consists of a set V of vertices and a set $E \subseteq \binom{V}{2}$ of edges. We denote by $V(G)$ and $E(G)$ the vertex and edge set of G, respectively.

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 $L(e) = \{\lambda(e) + i\Delta \mid i \geq 0\}$. In this case, we call λ a *Δ-periodic labeling* of G. When it is clear from the context, we drop Δ and denote the (Δ-periodic) temporal graph by (G, λ).

A *temporal (s, z)-walk* (or *temporal walk*) of length k from vertex $s = v_0$ to vertex $z = v_k$ in a Δ-periodic temporal graph (G, L) is a sequence $P = ((v_{i-1}, v_i, t_i))_{i=1}^{k}$ of triples (where, for every i, $\{v_{i-1}, v_i\}$ is an edge of G) which we call *transitions*, such that for all $i \in [k]$ we have that $t_i \in L(\{v_{i-1}, v_i\})$ and for all $i \in [k-1]$ we have that $t_i < t_{i+1}$. Moreover, we call P a *temporal (s, z)-path* (or *temporal path*) of length k if $v_i \neq v_j$ for all $i, j \in \{0, \ldots, k\}$ with $i \neq j$. Given a temporal path $P = ((v_{i-1}, v_i, t_i))_{i=1}^{k}$, we denote the set of vertices of P by $V(P) = \{v_0, v_1, \ldots, v_k\}$. A temporal (s, z)-path $P = ((v_{i-1}, v_i, t_i))_{i=1}^{k}$ is *fastest* if for all temporal (s, z)-path $P' = \left(\left(v'_{i-1}, v'_i, t'_i\right)\right)_{i=1}^{k'}$ we have that $t_k - t_0 \leq t'_{k'} - t'_0$. We say that the *duration* of P is $d(P) = t_k - t_0 + 1$. Furthermore, the concept of *travel delays* is very important for our proofs.

Definition 2 (travel delays). *Let (G, λ) be a Δ-periodic temporal graph. Let $e_1 = \{u, v\}$ and $e_2 = \{v, w\}$ be two incident edges in G with $e_1 \cap e_2 = \{v\}$. We define the* travel delay *from u to w at vertex v, denoted with $\tau_v^{u,w}$, as follows.*

$$\tau_v^{u,w} = \begin{cases} \lambda(e_2) - \lambda(e_1), & \text{if } \lambda(e_2) > \lambda(e_1), \\ \lambda(e_2) - \lambda(e_1) + \Delta, & \text{otherwise.} \end{cases}$$

From Definition 2 we immediately get the following observation.

Observation 1. *Let (G, λ) be a Δ-periodic temporal graph. Let $e_1 = \{u, v\}$ and $e_2 = \{v, w\}$ be two incident edges in G with $e_1 \cap e_2 = \{v\}$. Then we have that $\tau_v^{u,w} = \Delta - \tau_v^{w,u}$ if $\lambda(e_1) \neq \lambda(e_2)$, and $\tau_v^{u,w} = \tau_v^{w,u} = \Delta$ if $\lambda(e_1) = \lambda(e_2)$.*

Intuitively, the value of $\tau_v^{u,w}$ quantifies how long a fastest temporal path waits at vertex v when first traversing edge $\{u, v\}$ and then edge $\{v, w\}$. Formally, we have the following.

Lemma 1 ($\star$). *Let $P = ((v_{i-1}, v_i, t_i))_{i=1}^{k}$ be a fastest temporal (s, z)-path. Then we have $d(P) = 1 + \sum_{i \in [k-1]} \tau_{v_i}^{v_{i-1}, v_{i+1}}$.*

3 The Classical Complexity on Restricted Instances

In this section, we prove that TTR is NP-hard even in quite restrictive settings: when the input tree is a star and $\Delta \geq 3$, or when the input tree has constant diameter or constant maximum degree, and in addition $\Delta = 2$. We remark that the diameter can be seen as a measure of how "star-like" a tree is. In particular, on trees, the diameter upper bounds the treedepth.

Theorem 1 ($\star$). TTR *is NP-hard even if the input matrix is symmetric and*

- *$\Delta \geq 3$ and G is a star,*

– $\Delta = 2$ *and* G *has constant diameter and a constant pathwidth, or*
– $\Delta = 2$ *and* G *has constant maximum degree and a constant pathwidth.*

As we now show, the hardness reduction on stars for $\Delta \geq 3$ is tight in the sense that for $\Delta = 2$, the problem is polynomial-time solvable on stars. More generally, we show the following.

Theorem 2 ($\star$). *For $\Delta = 2$,* TTR *can be solved in $4^{\text{vc}} \cdot n^{O(1)}$ time, where* vc *denotes the vertex cover number of the input tree.*

4 Main FPT-algorithm for TTR Based on MILPs

Our hardness results from the previous section exclude FPT-algorithms for general Δ for nearly all reasonable parameters on trees like maximum degree or vertex cover number. In this section, we consider the essentially only remaining parameter on trees: the number of leaves of the tree. This parameter is larger than the maximum degree and incomparable to the vertex cover number.

Theorem 3. TTR *is fixed-parameter tractable when parameterized by the number ℓ of leaves of the input tree G.*

To show Theorem 3 we present a Turing-reduction from TTR to Mixed Integer Linear Program (MILP).

Mixed Integer Linear Program (MILP)

Input: A vector x of n variables of which some are considered integer variables, a constraint matrix $A \in \mathbb{R}^{m \times n}$, and two vectors $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

Task: Compute an assignment to the variables (if one exists) such that all integer variables are set to integer values, $Ax \leq b$, $x \geq 0$, and $c^\intercal x$ is maximized.

Given an instance (G, D, Δ) of TTR, we will produce several MILP instances. We will prove that (G, D, Δ) is a yes-instance if and only if at least one of the MILP instances admits a feasible solution. The number of produced instances is upper-bounded by a function of the number ℓ of leaves in G. Each of the produced MILP instances will have a small number of integer variables. More precisely, the number of integer variables will be upper-bounded by a function of the number ℓ of leaves in G. This will allow us to upper-bound the running time necessary to solve the MILP instances using the following known result.

Theorem 4 ([17,36]). MILP *is fixed-parameter tractable when parameterized by the number of integer variables.*

Furthermore, we build our MILP formulations in a specific way that ensures that there always exist optimal solutions where *all* variables are set to integer values. Informally, we ensure that the constraint matrix for the rational variables is totally unimodular. This allows us to use the following result.

Lemma 2 ([13]). *Let $A_{frac} \in \mathbb{R}^{m \times n_2}$ be totally unimodular. Then for any $A_{int} \in \mathbb{R}^{m \times n_1}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^{n_1+n_2}$, the MILP $\max c^\intercal x$ subject to $(A_{int}\ A_{frac})x \leq b, x \geq 0$, where $x = (x_{int}\ x_{frac})^\intercal$ with the first n_1 variables (i.e., x_{int}) being the integer variables, has an optimal solution where all variables are integer.*

The main idea for the reduction is the following. Given an instance (G, D, Δ) of TTR, we create variables for the travel delays at the vertices of G. A labeling can easily be computed from the travel delays. We use the constraints to ensure that the durations of the fastest paths respect the upper bounds given in D.

Before describing the reduction, we make a straightforward observation.

Observation 2. *Let G be a tree with ℓ leaves. Let $V_{>2}$ be the set of vertices in G with degree larger than two. We have that $|V_{>2}| \leq \ell$. Furthermore, we have for every $v \in V_{>2}$ that the degree of v is at most ℓ.*

Furthermore, we show that there always exist solutions to yes-instances of TTR, where the two edges incident with degree-2 vertices have different labels.

Lemma 3 ($\star$). *Let (G, D, Δ) be a yes-instance of* TTR*. Let V_2 be the set of vertices in G with degree two. Then there exists a solution λ for (G, D, Δ) such that for all $v \in V_2$ we have the following. Let u, w denote the two neighbors of v in G, then $\lambda(\{u, v\}) \neq \lambda(\{v, w\})$.*

From now on assume that we are given an instance (G, D, Δ) of TTR. Assume the vertices in G are ordered in an arbitrary but fixed way. Each MILP instance we create will use the same set of variables. For each vertex v in G, we create the following variables:

- If v has degree two, we create a *fractional* variable x_v. This variable will correspond to the travel delay $\tau_v^{u,w}$, where u, w are the two neighbors of v in G and u is ordered before w.
- If v has degree larger than two, then for every pair u, w of neighbors of v in G, we create an *integer* variable $y_v^{u,w}$. This variable will correspond to the travel delay $\tau_v^{u,w}$.
  Furthermore, we create an *integer* variable z_e for every edge e that is incident with v. This variable will correspond to the label $\lambda(e)$ of e.

With Observation 2 we can upper-bound the number of created integer variables.

Observation 3. *Each* MILP *instance has $O(\ell^3)$ integer variables.*

Now we consider all possibilities of how the labels of edges incident with vertices of degree larger than two can relate to each other. Formally, let v be a vertex in G that has degree larger than two and let E_v be the set of edges incident with v. By Observation 2 we have that $|E_v| \leq \ell$. Any labeling λ partitions E_v into at most $\min(\ell, \Delta)$ sets of edges with equal labels, and the values of the labels define an ordering of those sets. We call a partitioning of E_v into at most $\min(\ell, \Delta)$ sets together with the ordering for those sets a *label configuration* for v. A set of label configurations for all vertices v in G that have degree larger than two is called a *global label configuration*. By Observation 2 we get the following.

Observation 4. *There are $O(\ell^{\ell^2})$ global label configurations.*

For each global label configuration, we create an MILP instance. From now on fix a global label configuration σ. We describe how to construct an MILP instance I_σ. Consider a vertex v in G that has degree larger than two. Let u, w be a pair of neighbors of v in G. Let $e = \{u, v\}$ and $e' = \{v, w\}$. If according to the global label configuration, we have that e and e' are in different parts of E_v and the part of e is ordered before the part of e' (that is, e has a smaller label than e'), then we add the following constraint.

$$y_v^{u,w} = z_{e'} - z_e \tag{1}$$

If the part of e is ordered after the part of e' (that is, e has a larger label than e'), then we add the following constraint.

$$y_v^{u,w} = z_{e'} - z_e + \Delta \tag{2}$$

In both above cases, we additionally add the following constraint.

$$1 \leq y_v^{u,w} \leq \Delta - 1 \tag{3}$$

If according to the global label configuration, we have that e and e' are in the same part of E_v (that is, they have the same label), then we add the following constraints.

$$z_e = z_{e'} \quad (4) \qquad y_v^{u,w} = \Delta \quad (5) \qquad 1 \leq z_e \leq \Delta \quad (6) \qquad 1 \leq z_{e'} \leq \Delta \quad (7)$$

Now we consider all vertex pairs s, t in G that have distance at least two in G, that is, the (unique) path from s to t in G has at least one internal vertex. Let $P_{s,t}$ denote the path from s to t in G. Then $V_{\text{int}}^{(s,t)} = V(P_{s,t}) \setminus \{s, t\}$ is the set of internal vertices of $P_{s,t}$. We partition $V_{\text{int}}^{(s,t)}$ into two sets. Let $V_{\text{int,deg2}}^{(s,t)} \subseteq V_{\text{int}}^{(s,t)}$ be the internal vertices of $P_{s,t}$ with degree two and let $V_{\text{int,deg}>2}^{(s,t)} \subseteq V_{\text{int}}^{(s,t)}$ be the internal vertices with degree larger than two. Now we further partition the two sets of vertices.

We define $V_{\text{int,deg2,f}}^{(s,t)} \subseteq V_{\text{int,deg2}}^{(s,t)}$ such that for all $v \in V_{\text{int,deg2,f}}^{(s,t)}$, the neighbor of v that is closer to s is ordered before the neighbor of v that is closer to t. Analogously, we define $V_{\text{int,deg2,b}}^{(s,t)} \subseteq V_{\text{int,deg2}}^{(s,t)}$ such that for all $v \in V_{\text{int,deg2,b}}^{(s,t)}$, the neighbor of v that is closer to t is ordered before the neighbor of v that is closer to s.

For the vertices in $V_{\text{int,deg}>2}^{(s,t)}$ we additionally define the following. For $v \in V_{\text{int,deg}>2}^{(s,t)}$ denote by $u^{(s,t)}(v)$ the neighbor of v that is on the path from v to s, and denote by $w^{(s,t)}(v)$ the neighbor of v that is on the path from v to t.

We add the following constraint for the vertex pair s, t. Recall here that $D_{s,t}$ comes from the input matrix, and thus acts as as a non-variable in the MILP.

$$\sum_{v \in V_{\text{int,deg2,f}}^{(s,t)}} x_v + \sum_{v \in V_{\text{int,deg2,b}}^{(s,t)}} (\Delta - x_v) + \sum_{v \in V_{\text{int,deg}>2}^{(s,t)}} y_v^{u^{(s,t)}(v), w^{(s,t)}(v)} \leq D_{s,t} - 1 \tag{8}$$

Finally, for all fractional variables x_v we add the following constraint.

$$1 \leq x_v \leq \Delta - 1 \tag{9}$$

This finishes the description of the constraints. Since we only want to check whether a feasible solution exists, we arbitrarily choose the vector c for the objective function to be the all-one vector. This finishes the description of the MILP instance I_σ. Clearly, for a given global label configuration, the MILP instance can be constructed in polynomial time.

Observation 5. *Given a global label configuration σ, the* MILP *instance I_σ can be computed in polynomial time.*

In the following, we prove the correctness of the algorithm. We first show that if the given instance of TTR is a yes-instance, then at least one of the created MILP instances has a feasible solution.

Lemma 4 ($\star$). *Given a yes-instance (G, D, Δ) of* TTR*, there exists a global label configuration σ such that the* MILP *instance I_σ admits a feasible solution.*

Next, we show that if one of the produced MILP instances admits a feasible solution, then we are facing a yes-instance of TTR. To this end, we first show that the constraint matrix for the fractional variables of the produced MILP instances is totally unimodular. Then by Lemma 2 we have that if one of the MILP instances admits a feasible solution, then there also is a feasible solution for that instance where all fractional variables are set to integer values. To do this, we use a sufficient condition for matrices to be totally unimodular. Precisely, we use that every so-called *network matrix* is totally unimodular.

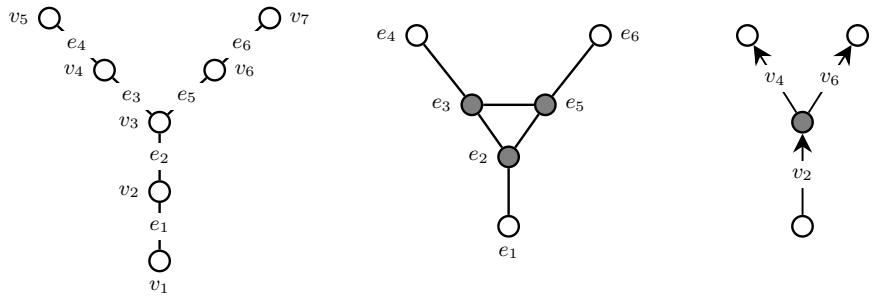
Definition 3 (network matrix). *Let $T = (V, A)$ be a directed tree, that is, a tree where each arc has an arbitrary orientation and let A' be a set of directed arcs on V, the same vertex set as T. Let M be a matrix with $|A|$ rows and $|A'|$ columns. Then M is a* network matrix *if the following holds. For all i, j let $e_i = (a, b) \in A$ and $e'_j = (s, t) \in A'$.*

- *$M_{i,j} = 1$ if arc (a, b) appears forwards in the path in T from s to t.*
- *$M_{i,j} = -1$ if arc (a, b) appears backwards in the path in T from s to t.*
- *$M_{i,j} = 0$ if arc (a, b) does not appear in the path in T from s to t.*

Lemma 5 ([47]). *Network matrices are totally unimodular.*

We show that the produced MILP instances have the following property.

Lemma 6. *Let σ be a global label configuration and let I_σ be the corresponding* MILP *instance. Let M be the constraint matrix for the fractional variables, that is, the constraint matrix obtained by treating the integer variables $y_v^{u,w}$ and z_e in all constraints of I_σ as arbitrary constants, deleting all constraints that do not involve any fractional variables, and deleting all duplicate rows. Then M is a network matrix.*



Tree G. Line graph $L(G)$. Directed tree T.

Fig. 2: Illustration of the construction of the directed tree T in Lemma 6. On the left an example tree G, where the vertices are ordered by their indices. In the middle the line graph $L(G)$ of G, where the gray vertices form a maximal clique with more than two vertices. On the right the constructed directed tree T, where the vertices of the maximal clique are merged into one vertex (gray) and the directed arcs correspond to the degree-2 vertices of G.

Proof. First, notice that Constraints (1), (2), (3), (4), (5), (6), and (7) do not involve fractional variables. Hence, from now on we only consider Constraints (8) and (9), where in Constraint (8) we treat the integer variables as constants. Let M be the resulting constraint matrix where also all duplicate rows are deleted. We show that M is a network matrix.

To this end, we define a directed tree $T = (V, A)$ (see Definition 3) as follows. We give an illustration in Figure 2. Let $L(G)$ be the line graph of the input tree G of the TTR instance. Now we exhaustively merge all maximal cliques in $L(G)$ with more than two vertices to a single vertex. Here, the new vertex is adjacent to all remaining neighbors of the originally merged vertices. Let this graph be called G'. It is easy to see that G' is a tree, and every edge in G' corresponds to a degree-2 vertex in G, and vice versa. Furthermore, we have that every vertex v in G' that has degree larger than two corresponds to a subtree G_v (which may be the degenerate tree consisting only of one vertex) in G such that all vertices in $V(G_v)$ have degree larger than two in G. Now we transform G' into T by replacing each edge of G' with an oriented arc. Consider a degree-2 vertex v of G and the corresponding edge e_v in G'. Let u and w be the two neighbors of v in G, where u is ordered before w. We distinguish four cases.

- Vertex u has degree two in G. Let e_u be the edge in G' corresponding to u. Then we orient e_v such that the resulting arc points away from the common endpoint of e_v and e_u.
- The degree of vertex u does not equal two and w has degree two in G. Let e_w be the edge in G' corresponding to w. Then we orient e_v such that the resulting arc points towards the common endpoint of e_v and e_w.

– Neither u nor w have degree two. Then we can assume that at least one of u and w has degree larger than two, otherwise G only has three vertices and we can solve the instance in constant time.
  - Vertex u has degree larger than two in G. Then let G_u denote the maximal subtree of G that contains u such that all vertices in $V(G_u)$ have degree larger than two in G. Then by construction of G', there is a vertex u' in G' corresponding to G_u that is incident with e_v. We orient e_v such that the resulting arc points away from u'.
  - Vertex u has degree one and vertex w has degree larger than two in G. Then let G_w denote the maximal subtree of G that contains w such that all vertices in $V(G_w)$ have degree larger than two in G. Then there is a vertex w' in G' corresponding to G_w that has degree larger than two and is incident with e_v. We orient e_v such that the resulting arc points towards w'.

This finishes the description of the directed tree T.

Next, we make the following observation: Each Constraint (9) produces the same coefficients for the rational variables in the constraint matrix as one of Constraint (8). To see this, note that each Constraint (9) has exactly one non-zero coefficient for one rational variable x_v, which is either 1 or -1. Variable x_v corresponds to a degree-2 vertex v in G. Let u and w be the two neighbors of v in G, where u is ordered before w. Then, since u and w have distance at least two in G, we have a Constraint (8) corresponding to the pair of vertices s, t with $s = u$ and $t = w$. This constraint has exactly one non-zero coefficient for a rational variable, which is x_v and the coefficient is 1. In the case where $s = w$ and $t = u$ we have the same situation, except that the coefficient is -1. Hence, since M has no duplicate rows, we can assume that each row of M corresponds to a Constraint (8).

Now consider a row of M corresponding to Constraint (8), which in turn corresponds to vertex pair s, t in G, where s and t have distance at least two in G. Let $P_{s,t}$ denote the (unique) path from s to t in G. Let $V^{(s,t)}_{\text{int,deg2,f}} \subseteq V(P_{s,t})$ and $V^{(s,t)}_{\text{int,deg2,b}} \subseteq V(P_{s,t})$ be defined as in the description of Constraint (8). Then we have that $V^{(s,t)}_{\text{int,deg2,f}} \cup V^{(s,t)}_{\text{int,deg2,b}}$ are all internal vertices of $P_{s,t}$ that have degree two in G. Furthermore, in Constraint (8) corresponding to vertex pair s, t we have that the fractional variable x_v has a non-zero coefficient if and only if $v \in V^{(s,t)}_{\text{int,deg2,f}} \cup V^{(s,t)}_{\text{int,deg2,b}}$. More specifically, the coefficient of x_v is 1 if $v \in V^{(s,t)}_{\text{int,deg2,f}}$ and the coefficient of x_v is -1 if $v \in V^{(s,t)}_{\text{int,deg2,b}}$.

Furthermore, by construction we have that each vertex in $V^{(s,t)}_{\text{int,deg2,f}} \cup V^{(s,t)}_{\text{int,deg2,b}}$ corresponds to an arc of T. It remains to show that there is a path P' in T that visits exactly the arcs corresponding to the vertices in $V^{(s,t)}_{\text{int,deg2,f}} \cup V^{(s,t)}_{\text{int,deg2,b}}$ such that the arcs corresponding to vertices in $V^{(s,t)}_{\text{int,deg2,f}}$ appear forwards in the path and the arcs corresponding to the vertices in $V^{(s,t)}_{\text{int,deg2,b}}$ appear backwards in the path. To this end, we order the vertices in $V^{(s,t)}_{\text{int,deg2,f}} \cup V^{(s,t)}_{\text{int,deg2,b}}$ in the order in

which they are visited by $P_{s,t}$, that is, let $V^{(s,t)}_{\text{int,deg2,f}} \cup V^{(s,t)}_{\text{int,deg2,b}} = \{v_1, v_2, \ldots, v_k\}$. Now we show by induction on k that P' exists and that its last arc corresponds to v_k. If $k = 1$ then P' only consists of the arc e_{v_1} corresponding to v_1. If $v_1 \in V^{(s,t)}_{\text{int,deg2,f}}$, then we consider P' to be the path in which e_{v_1} appears forwards. Otherwise, we consider P' to be the path in which e_{v_1} appears backwards. Now assume $k > 1$. Let P'' be the path in T that visits the arcs corresponding to vertices $\{v_1, \ldots, v_{k-1}\}$ such that for all $i \in [k-1]$ the arc corresponding to v_i appears forwards if $v_i \in V^{(s,t)}_{\text{int,deg2,f}}$ and backwards otherwise. Furthermore, we have that P'' ends with the arc corresponding to v_{k-1}. Now we show that we can append the arc corresponding to v_k to P''. We distinguish two cases.

- In the first case, v_k is visited directly after v_{k-1} by $P_{s,t}$. Then the arcs $e_{v_{k-1}}$ and e_{v_k} in T share a common endpoint which is a degree-2 vertex. It follows that we can append the arc e_{v_k} to P'' to create a new path that ends in e_{v_k}. We still need to show that e_{v_k} is appended in the correct direction. Consider the two neighbors of v_k in G. One of the neighbors is v_{k-1}. Let the second neighbor be w. Now assume that $v_k \in V^{(s,t)}_{\text{int,deg2,f}}$. This implies that v_{k-1} is ordered before w. In this case the arc e_{v_k} in T points away from the common endpoint of $e_{v_{k-1}}$ and e_{v_k}, and hence it appears forwards in the P'' appended with e_{v_k}. The case where $v_k \in V^{(s,t)}_{\text{int,deg2,b}}$ is analogous.
- In the second case, there is a non-empty set of vertices $\widehat{V}$ with degree larger than two, that are visited by $P_{s,t}$ between v_{k-1} and v_k. Let $\widehat{G}$ denote the maximal tree in G that only contains vertices that have degree larger than two in G and that contains $\widehat{V}$. By construction, there is vertex $\widehat{v}$ in T that corresponds to $\widehat{G}$ and both $e_{v_{k-1}}$ and e_{v_k} in T have $\widehat{v}$ as one of their endpoints. First, we argue that we can append e_{v_k} to P'' and thereby create a path in T that ends with e_{v_k}. To this end, notice that the vertex visited by $P_{s,t}$ before v_{k-1} cannot be part of $\widehat{G}$, otherwise G would contain a cycle. It follows that the common endpoint of $e_{v_{k-2}}$ and $e_{v_{k-1}}$ is not $\widehat{v}$ and appending e_{v_k} to P'' indeed produces a path in T that ends with e_{v_k}. It remains to show that e_{v_k} is appended in the correct direction. Consider the two neighbors u and w of v_k in G. We have that one neighbor is contained in $\widehat{G}$, assume w.l.o.g. that it is u. Then u is visited by $P_{s,t}$ before v_k. Now assume that $v_k \in V^{(s,t)}_{\text{int,deg2,f}}$. This implies that u is ordered before w. In this case the arc e_{v_k} in T points away from u, and hence it appears forwards in the path P'' appended with e_{v_k}. The case where $v_k \in V^{(s,t)}_{\text{int,deg2,b}}$ is analogous.

It follows that for every row in M, there is a path in T such that the conditions in Definition 3 are met. We can conclude that M is a network matrix. □

Now we are ready to show that if one of the produced MILP instances admits a feasible solution, then we are facing a yes-instance of TTR.

Lemma 7 (⋆). *Given an instance (G, D, Δ) of* TTR*, if there exists a global label configuration σ such that the* MILP *instance I_σ admits a feasible solution, then (G, D, Δ) is a yes-instance of* TTR*.*

Theorem 3 follows from the shown results. A formal proof is given in the full version [40].

5 Conclusion

We have initiated the investigation of the natural temporal tree realization problem TTR and shown that it is NP-hard in quite restrictive cases. On the positive side, we provided an FPT-algorithm for the number of leaves in the input tree as a parameter, essentially the only reasonable parameter on trees that is not a constant in at least one of our hardness reductions. A canonical future research direction is to investigate TTR on general graphs. For example, can our FPT-algorithm for number of leaves be transferred to general graphs with respect to the parameter max-leaf number? Further interesting parameters for general graphs include the distance to a clique (or distance to other dense graphs), maximum independent set, clique cover, or even various parameters of the input matrix D.

References

1. 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.
2. 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.
3. Jayanth R. Banavar, Francesca Colaiori, Alessandro Flammini, Amos Maritan, and Andrea Rinaldo. Topology of the fittest transportation network. *Physical Review Letters*, 84:4745–4748, 2000.
4. Amotz Bar-Noy, Keerti Choudhary, David Peleg, and Dror Rawitz. Efficiently realizing interval sequences. *SIAM Journal on Discrete Mathematics*, 34(4):2318–2337, 2020.
5. Amotz Bar-Noy, Keerti Choudhary, David Peleg, and Dror Rawitz. Graph realizations: Maximum degree in vertex neighborhoods. *Discrete Mathematics*, 346(9):113483, 2023.
6. Amotz Bar-Noy, David Peleg, Mor Perry, and Dror Rawitz. Composed degree-distance realizations of graphs. *Algorithmica*, 85(3):665–687, 2023.
7. Amotz Bar-Noy, David Peleg, Mor Perry, and Dror Rawitz. Graph realization of distance sets. *Theor. Comput. Sci.*, 1019:114810, 2024.
8. Marc Barthélemy and Alessandro Flammini. Optimal traffic networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(07):L07002, 2006.
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. Steffen Bohn and Marcelo O. Magnasco. Structure, scaling, and phase transition in the optimal transport network. *Physical Review Letters*, 98:088702, 2007.
12. 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.
13. Juhi Chaudhary, Hendrik Molter, and Meirav Zehavi. Parameterized analysis of bribery in challenge the champ tournaments. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2704–2712, 2024.
14. Wai-Kai Chen. On the realization of a (p, s)-digraph with prescribed degrees. *Journal of the Franklin Institute*, 281(5):406–422, 1966.
15. 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.
16. Joseph C. Culberson and Piotr Rudnicki. A fast algorithm for constructing trees from distance matrices. *Information Processing Letters*, 30(4):215–220, 1989.
17. Daniel Dadush, Chris Peikert, and Santosh Vempala. Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *Proceedings of the 52nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 580–589. IEEE, 2011.
18. Paul Erdös and Tibor Gallai. Graphs with prescribed degrees of vertices. *Mat. Lapok*, 11:264–274, 1960.
19. Thomas Erlebach, Nils Morawietz, Jakob T. Spooner, and Petra Wolf. A cop and robber game on edge-periodic temporal graphs. *Journal of Computer and System Sciences*, 144:103534, 2024.
20. 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 12:1–12:16, 2024.
21. 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.
22. András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992.
23. András Frank. Connectivity augmentation problems in network design. *Mathematical Programming: State of the Art 1994*, 1994.
24. H. Frank and Wushow Chou. Connectivity considerations in the design of survivable networks. *IEEE Transactions on Circuit Theory*, 17(4):486–490, 1970.
25. D.R. Fulkerson. Zero-one matrices with zero trace. *Pacific Journal of Mathematics*, 10(3):831–836, 1960.
26. Petr A. Golovach and George B. Mertzios. Graph editing to a given degree sequence. *Theoretical Computer Science*, 665:1–12, 2017.
27. Martin Charles Golumbic and Ann N. Trenk. *Tolerance Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2004.
28. 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.
29. Martin Grötschel, Clyde L Monma, and Mechthild Stoer. Design of survivable networks. *Handbooks in Operations Research and Management Science*, 7:617–672, 1995.

30. 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.
31. S. Louis Hakimi and Stephen S. Yau. Distance matrix of a graph and its realizability. *Quarterly of applied mathematics*, 22(4):305–317, 1965.
32. Pavol Hell and David Kirkpatrick. Linear-time certifying algorithms for near-graphical sequences. *Discrete Mathematics*, 309(18):5703–5713, 2009.
33. Petter Holme and Jari Saramäki. *Temporal network theory*, volume 2. Springer, 2019.
34. 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.
35. Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. Temporal graph realization from fastest paths. In *Proceedings of the 3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 16:1–16:18, 2024.
36. Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
37. Linda Lesniak. Eccentric sequences in graphs. *Periodica Mathematica Hungarica*, 6:287–293, 1975.
38. 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.
39. 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.
40. G.B. Mertzios, H. Molter, N. Morawietz, and P.G. Spirakis. Realizing temporal transportation trees. *CoRR*, abs/2403.18513, 2025. URL: `https://arxiv.org/abs/2403.18513`.
41. Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, January 2018.
42. 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 71:1–71:14, 2020.
43. 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.
44. Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal Networks*. Springer, 2013.
45. 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.
46. Elena Rubei. Weighted graphs with distances in given ranges. *Journal of Classification*, 33:282–297, 2016.
47. Alexander Schrijver. *Combinatorial optimization: Polyhedra and efficiency*, volume 24. Springer, 2003.
48. 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.