Directed Temporal Tree Realization for Periodic Public Transport: Easy and Hard Cases

Julia Meusel
Martin Luther University Halle-Wittenberg, Germany

Matthias Müller-Hannemann
Martin Luther University Halle-Wittenberg, Germany

Klaus Reinhardt
Martin Luther University Halle-Wittenberg, Germany

Abstract

We study the complexity of the *directed periodic temporal graph realization* problem. This work is motivated by the design of periodic schedules in public transport with constraints on the quality of service. Namely, we require that the fastest path between (important) pairs of vertices is upper bounded by a specified maximum duration, encoded in an upper distance matrix D. While previous work has considered the undirected version of the problem, the application in public transport schedule design requires the flexibility to assign different departure times to the two directions of an edge. A problem instance can only be feasible if all values of the distance matrix are at least shortest path distances. However, the task of realizing exact fastest path distances in a periodic temporal graph is often too restrictive. Therefore, we introduce a minimum slack parameter k that describes a lower bound on the maximum allowed waiting time on each path. We concentrate on tree topologies and provide a full characterization of the complexity landscape with respect to the period Δ and the minimum slack parameter k, showing a sharp threshold between NP-complete cases and cases which are always realizable. We also provide hardness results for the special case of period $\Delta = 2$ for general directed and undirected graphs.

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

Keywords and phrases Periodic timetabling, service quality, temporal graph, graph realization, complexity

Digital Object Identifier 10.4230/OASIcs.ATMOS.2025.3

Related Version *Full Version*: http://arxiv.org/abs/2504.07920
A 5-page version has appeared as a brief announcement in SAND 2025: https://doi.org/10.4230/LIPIcs.SAND.2025.21

1 Introduction

Designing periodic schedules for public transport is notoriously difficult. Periodic schedules are desirable for several practical and operational reasons. They are easier to memorize, and travelers can better plan their journeys if services run at regular intervals. Periodicity also enables coordinated connections between different lines at central hubs and simplifies crew and vehicle scheduling. In this paper, we consider the quality of service provided by a periodic schedule from a passenger's perspective. Specifically, we address the question of whether it is possible to design a periodic schedule for a given network that guarantees travel time bounds between important pairs of stops. We model this as a graph realization problem.

Graph realization problems are a central area of research that has been studied extensively since the 1960s for undirected [6, 9, 10] and directed graphs [1, 4, 8]. Given a set of constraints, the objective is to find a graph that satisfies them, or to decide that no such graph exists.

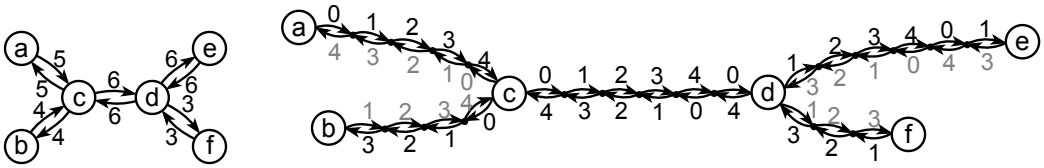
Restrictions on degrees [2, 6, 10], distances between vertices [3, 9, 22], eccentricities [14], and connectivity [5, 11] have been studied in detail. Recently, the study of realization problems on temporal graphs was started by Klobas et al. [12, 13]. Temporal graphs are graphs that have a fixed set of vertices and a set of edges that changes over time. Each edge of a static graph is assigned a set of timestamps at which it is active. In a periodic temporal graph, the set of timestamps is repeated periodically for all edges. Informally, a temporal path is a sequence of consecutive edges in the underlying static graph and corresponding increasing timestamps at which they are active. Klobas et al. and Mertzios et al. examined restrictions on the travel time between pairs of vertices in periodic temporal graphs: upper bounds as well as exact values [12, 13, 16].

This leads to several variants of the Temporal Graph Realization problem (TGR) that have been studied: In the periodic TGR, all timestamps are repeated periodically. The *simple* periodic TGR allows only one timestamp per period, unlike the *multi-label* periodic TGR. The constraints on fastest travel times can be either exact values or upper bounds. If the given underlying graph is a tree, we call the problem Temporal Tree Realization (TTR). In addition to communication networks such as social networks or satellite links, Klobas et al. mention transportation networks as examples of potential applications for the TTR [12, 13]. However, unlike the flow of information in satellite links, transportation networks typically do not carry passengers on an edge in both directions at the same time. Therefore, we examine the directed case as a natural generalization. Since most public transportation lines run in both directions, we will mostly consider bidirected graphs. In tree structures this is necessary to ensure that every vertex is reachable from every other vertex.

With an application in public transport in mind, the input graph models the infrastructure network where vertices correspond to locations of stops (or stations) and edges connect neighboring stops served by a bus, tram, train or the like. In public transport, typical values for the period Δ are 5, 10, 15 or 20 minutes in urban transport, and 30, 60 or 120 minutes in long-distance train networks. The timestamp of an edge (u, v) can be interpreted as the departure time of some vehicle at u. In a practical setting, traveling along an edge e requires $\ell_e \in \mathbb{N}$ time. An edge of length ℓ can be equivalently replaced by a simple path of ℓ edges of unit length as shown in Figure 1. While such a transformation blows up the graph size, it does not change the complexity of the problem (since hardness results are established even for unit length graphs). For simplicity, we will therefore consider only unit-length graphs.

Informally, given a directed, strongly connected graph as the underlying static graph as well as a period Δ and upper bounds on the travel time between each pair of vertices, the objective is to compute a (single) periodic timestamp for each directed edge such that a fastest temporal path between any two vertices does not exceed the given upper bound. The upper bounds between pairs of vertices can be interpreted as a guaranteed quality of service that must be met. The bounds are specified by a matrix D of integers or ∞. The latter means that we do not impose any restriction on the fastest path between the corresponding vertices. Motivated by transportation networks with fixed traffic lines, we consider only one global period instead of allowing different periods for all edges. Since waiting times are unavoidable and natural, for example, due to transfer times at transit hubs, we assume that there is a small amount of waiting time allowed on all shortest paths and introduce a minimum slack parameter k that specifies how much waiting time is at least acceptable on all shortest routes. This parameter gives us some leeway to work with, as waiting is necessary even on very small instances, such as the one shown in Figure 1.

Travelers on a public transport network typically have to change trains along the way. In practice, a minimum transfer time must be planned for each change. For simplicity, we consider only the case where all transfer times are 0. However, one could easily extend the



(a) An infrastructure network G labeled with distances (in minutes) between vertices (stops).

(b) A graph in which phantom stops have been inserted so that all edges have unit length. Here, the labels define a schedule where each value denotes a departure time with respect to a global period of $\Delta = 5$. The resulting graph with its labeling forms a periodic temporal graph.

Figure 1 Example: The infrastructure network $G = (V, E)$ with edge lengths (left) can be modelled with unit length edges by inserting phantom stops between already existing stops (right). No upper bounds are set for travel times from and to these phantom stops. Suppose we have a period of $\Delta = 5$ and the following upper bounds: $D_{a,e} = 17$, $D_{a,b} = 9$, and $D_{f,b} = 13$. Assuming, without loss of generality, that the label of the edge connecting a to the first phantom stop is assigned the timestamp 0, the given upper bounds on the travel times enforce the black edge labels. A temporal graph with edge labels as specified respects these upper bounds. The timestamps of a fastest temporal path from f to e may be, for example, $1, 2, 3, 6, 7, 8, 9, 10, 11$, giving it a duration of 11, whereas the static distance of f and e is only 9. On this fastest path, one would wait at vertex d for two timesteps before traversing the next edge with label $1 = 6 \bmod 5$ at time 6. Therefore, adding an upper bound $D_{f,e} = 9$ would make the instance infeasible.

model by imposing non-trivial minimum transfer times at each stop. More precisely, one could add constraints specifying that the difference between the arrival time at a stop and the departure time on a leaving edge must be at least a given stop-specific constant (the minimum transfer time at this stop).

Related work. Klobas et al. show that the Simple Periodic TGR with exact given fastest travel times is NP-hard even for a small constant period $\Delta \geq 3$ [12, 13, Theorem 3]. However, if the underlying static graph is a tree, the problem is solvable in polynomial time [12, 13, Theorem 27]. It is fixed-parameter tractable (FPT) with respect to the feedback edge number of the underlying graph but W[1]-hard when parameterized by the feedback vertex number of the underlying graph [12, 13, Theorem 29 and Theorem 4].

While the Simple Periodic Temporal Graph Realization problem with exact given fastest travel times is solvable in polynomial time on trees, Mertzios et al. showed that the problem given upper bounds on the fastest travel times is NP-hard even if the underlying static graph is a tree or even a star [16, Theorem 5]. This still holds for a constant period of $\Delta = 2$ and when the input tree G has a constant diameter or a constant maximum degree [16, Theorem 5]. However, it is FPT with respect to the number of leaves in the input tree G [16, Theorem 19].

While Klobas et al. and Mertzios et al. only allow one timestamp per edge, Erlebach et al. consider several timestamps per edge [7]. They examine both the periodic and the non-periodic variant with exact given fastest travel times. Among other results, they show that the Multi-Label Periodic Temporal Graph Realization problem is NP-hard, even if the underlying static graph is a star for any number $\ell \geq 5$ of labels per edge. All these models have in common that labels (periodic timestamps) are assigned to edges.

Important related problem versions assign periodic labels to vertices. The Periodic Event Scheduling Problem (PESP), introduced by Serafini and Ukovich in 1989 [21], is widely used to schedule reoccurring events in public transport. Here the input is a so-called *event-activity network* $N = (V, A)$, a period Δ, and time windows $[\ell_a, u_a]$ for each activity

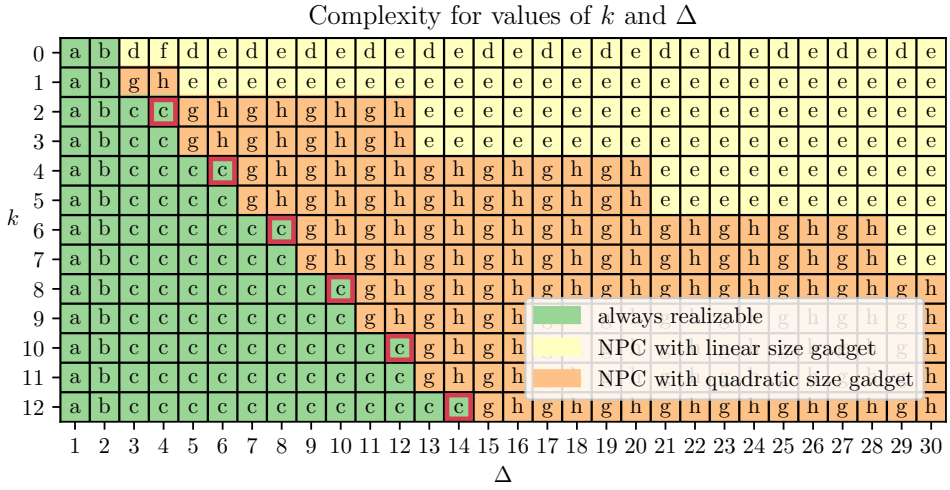


Figure 2 Complexity of the Directed Upper-Bounded Periodic Temporal Tree Realization Problem for different values of the minimum slack parameter k and period Δ (Δ-k-DiTTR). The labeling of the boxes indicates which proof was employed to achieve the respective result, where Theorem 20 is used for all NP-complete cases: (a) Theorem 6, (b) Corollary 8, (c) Theorem 7, (d) Lemma 14, (e) Lemma 15, (f) Lemma 16, (g) Lemma 17, and (h) Lemma 18. Cases in which the undirected problem version is NP-complete but all instances of the directed version are realizable are outlined in red (see Theorem 20).

$a \in A$. The set V models *events* (think of an arrival or departure of a vehicle at a stop) and the set $A \subset V \times V$ so-called *activities*. Activities model driving between neighboring stops, dwelling at a stop, transfers between different vehicles or safety constraints (minimal headways). In PESP, one seeks a periodic timestamp $\pi_v \in \{0, 1, \ldots, \Delta - 1\}$ for each event $v \in V$ such that $(\pi_w - \pi_v) \bmod \Delta \in [\ell_{(v,w)}, u_{(v,w)}]$ for all $(v, w) \in A$. The time window $[\ell_a, u_a]$ of an activity models lower and upper bounds on the difference of the timestamps between the corresponding events. In other words, the difference $u_a - \ell_a$ bounds the slack (waiting time) which can be introduced on activity a. In stark contrast to our model, these restrictions are local constraints between adjacent events, not global constraints between arbitrary events as considered in this paper. The PESP is known to be NP-complete for fixed $\Delta \geq 3$ [21, 17], but efficiently solvable for $\Delta = 2$ [19, page 87]. Deciding the feasibility of a PESP instance is NP-hard even when the treewidth is 2, the branchwidth is 2, or the carvingwidth is 3 [15].

Our contribution. In this paper, we investigate the complexity of the Directed Upper-Bounded Periodic Temporal Graph Realization Problem (DiTGR) and the Directed Upper-Bounded Periodic Temporal Tree Realization Problem (DiTTR). Our main results are as follows:

- We provide an efficiently checkable necessary and sufficient condition for feasibility in DiTTR when all pairs of vertices must be realized on shortest paths without waiting time. The basic insight is that the solvability for such instances depends on the distance between branching vertices of the given graph topology.
- We then introduce the parameter k, which specifies the minimum waiting time to be allowed on each shortest path (the *slack*), i.e., for all pairs of vertices u and v we require that the duration bound $D_{u,v}$ is at least the distance of u and v in the underlying static

graph plus the constant k. We fully characterize the complexity of the problem for bidirected tree topologies in terms of period Δ and slack parameter k. For each possible combination of parameters Δ and k, we either prove that the problem is easily solvable or hard, see Figure 2.
- We also investigate in more detail the special case of a given period $\Delta = 2$. This turns out to be NP-complete in general (for both directed and undirected graphs), but can be solved efficiently on directed bipartite graphs and graphs in which all shortest paths are unique.

Organization of the paper. In Section 2, we start with a formal problem definition. Then, in Section 3 we will first characterize feasible instances by providing a necessary and sufficient condition and then present all cases where instances with a bidirected tree topology can be easily solved. We will also discuss in more detail the special case of a given period $\Delta = 2$. In Section 4, we give our hardness results for the remaining instances with a directed bidirected tree topology. Finally, we conclude in Section 5 with a summary and suggestions for future work.

2 Formal Problem Definition

As we mainly consider the problem for directed graphs, all following definitions deal with directed temporal graphs. Both undirected edges and directed arcs are referred to as *edges*. The definitions for undirected temporal graphs are analogous. For temporal graphs and periodic temporal graphs, we follow the notation of Klobas et al. [12, 13]:

▶ **Definition 1.** *A* temporal graph *is a pair* (G, Λ)*, where* $G = (V, E)$ *is the underlying (static) graph and* $\Lambda : E \to 2^{\mathbb{N}_0}$ *is a function, that assigns a set of discrete timestamps to each edge.*

▶ **Definition 2.** *A* Δ-periodic *temporal graph is a triple* $(G = (V, E), \lambda : E \to \{0, 1, \ldots, \Delta - 1\}, \Delta)$ *which denotes the temporal graph* (G, Λ) *where* $\forall e \in E : \Lambda(e) = \{\lambda(e) + i \cdot \Delta \mid i \in \mathbb{N}_0\}$.

Informally, a temporal path is a sequence that denotes consecutive edges on a path in the underlying static graph and the times at which they are traversed. No vertex can be visited more than once. Recent literature distinguishes between the strict and non-strict version. Throughout the paper we only consider strict paths: The timestamps have to be strictly increasing. Formally, we can define a temporal path as follows:

▶ **Definition 3.** *A* temporal s-z-path *of length* ℓ *in a directed temporal graph* (G, Λ) *is a sequence* $P = (v_{i-1}, v_i, t_i)_{i=1}^{\ell}$ *for which the following holds:*
- $v_0 = s \wedge v_\ell = z$
- $\forall i, j \in \{0, \ldots, \ell\}, i \neq j : v_i \neq v_j$
- $\forall i \in \{1, \ldots, \ell\} : (v_{i-1}, v_i) \in E$
- $\forall i \in \{1, \ldots, \ell\} : t_i \in \Lambda((v_{i-1}, v_i))$
- $\forall i \in \{2, \ldots, \ell\} : t_{i-1} < t_i$

The traversal of an edge requires one time unit. The temporal s-z-path *starts* or *begins* at vertex s at time t_1, it *reaches* or *arrives* at vertex z at time $t_\ell + 1$.

▶ **Definition 4.** *The* duration $d(P)$ *of a temporal path* $P = (v_{i-1}, v_i, t_i)_{i=1}^{\ell}$ *is defined as* $d(P) = t_\ell + 1 - t_1$.

Let $\hat{d}(u, v)$ be the static *distance* of u and v in the underlying static graph. The undirected and directed problem versions can now be stated as follows.

Periodic Upper-Bounded Temporal Graph Realization (TGR)

Input: An undirected, connected graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$, an $n \times n$ matrix D of positive integers where $\forall u, v \in V : D_{u,v} \geq \hat{d}(u, v)$, and a positive integer Δ.

Question: Does there exist a Δ-periodic labeling $\lambda : E \to \{0, 1, \ldots, \Delta - 1\}$ such that, for every i, j, the duration of a fastest temporal path from v_i to v_j in the Δ-periodic temporal graph (G, λ, Δ) is *at most* $D_{i,j}$?

The restriction of TGR where the given graph is a tree is called TTR. To generalize the undirected problem version to the directed one, we restrict the considered graphs to the simplest case: we only consider directed graphs obtained by replacing each undirected edge with two antiparallel directed edges. We call graphs that are created this way bidirected.

Periodic Upper-Bounded Temporal Directed Graph Realization (DiTGR)

Input: A directed, strongly connected graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$, an $n \times n$ matrix D of positive integers where $\forall u, v \in V : D_{u,v} \geq \hat{d}(u, v)$, and a positive integer Δ.

Question: Does there exist a Δ-periodic labeling $\lambda : E \to \{0, 1, \ldots, \Delta - 1\}$ such that, for every i, j, the duration of a fastest temporal path from v_i to v_j in the Δ-periodic temporal graph (G, λ, Δ) is *at most* $D_{i,j}$?

The restriction of DiTGR to inputs of such graphs derived from trees by adding two directed edges (u, v) and (v, u) for every undirected edge $\{u, v\}$ is called DiTTR. This means that for any pair of vertices (u, v) there is exactly one path from u to v in the underlying static graph. Furthermore, we only consider instances where $\forall u, v \in V : D_{u,v} \geq \hat{d}(u, v)$, because all other instances cannot be realized anyway. The *duration* of a fastest temporal path from u to v depending on λ is denoted by $d_\lambda(u, v)$. We simply write $d(u, v)$ whenever λ is clear from the context. For brevity, we write $\lambda(u, v)$ instead of $\lambda((u, v))$. The *waiting time* at vertex v_i on a path $P = (v_{i-1}, v_i, t_i)_{i=1}^{\ell}$ is $t_{i+1} - t_i - 1$ for $1 \leq i < \ell$. The waiting time on a path P is the sum of the waiting time at all its vertices. In a bidirected tree it is equal to the difference $d(v_0, v_\ell) - \hat{d}(v_0, v_\ell)$ on a fastest path. In Figure 1b, the static distance $\hat{d}(f, e)$ between the vertices f and e is 9, whereas the duration of the fastest temporal path between them is $d(f, e) = 11$, with a waiting time of $2 = 6 - 3 - 1$ only at vertex d. For a vertex v, we denote by $\delta^+(v)$ its static outdegree, by $\delta^-(v)$ its static indegree, and by $\delta(v) = \delta^+(v) + \delta^-(v)$ its (total) static degree; $N(v)$ refers to the set of its neighbors.

For any pair of vertices (v, w) the duration of a fastest temporal path $P = (v_{i-1}, v_i, t_i)_{i=1}^{\ell}$ can be at most $d(P) \leq (\hat{d}(v, w) - 1) \cdot \Delta + 1$. This bound is achieved if $\lambda(v_{i-1}, v_i)$ is equal for all $i \in \{1, \ldots, \ell\}$. Therefore, any value of $D_{v,w}$ with $D_{v,w} \geq (\hat{d}(v, w) - 1) \cdot \Delta + 1$ is no real restriction. We write any such value simply as $D_{v,w} = \infty$ or omit it entirely.

The slack parameter k is an implicit parameter of D: D is restricted by $\forall v, w \in V : D_{v,w} \geq \hat{d}(v, w) + k$. For trees, this means that k is the minimum permitted waiting time on any path, since paths are unique. We call the versions of TTR and DiTTR where the parameters Δ and k are fixed Δ-k-TTR respectively Δ-k-DiTTR.

3 Characterization of Feasible Instances

3.1 A Necessary and Sufficient Condition for Feasibility in DiTTR

Before we look at the hardness results for the DiTTR problem, let us consider a special case: There is no waiting time allowed on any shortest path. As the underlying static graph is a tree, all paths are shortest paths and no waiting is allowed on any path. Therefore,

the given travel times are not just upper bounds but exact values. Since there is only one path between every pair of vertices, this means $\forall u, v \in V : D_{u,v} = \hat{d}(u, v)$. For undirected graphs, this special case is also covered by the result of Klobas et al. for the exact TTR: it is decidable in polynomial time whether the instance is realizable [12, 13, Theorem 27]. We provide a simple characterization of the realizability of an instance of DiTTR. To do this, we introduce so-called branching vertices. We will call a vertex with a degree of at least 6 *branching vertex*.

▶ **Observation 5.** *An instance of DiTTR with $\forall u, v \in V : D_{u,v} = \hat{d}(u, v)$ is realizable exactly when the distance of any pair of branching vertices is a multiple of $\Delta/2$.*

Proof. First we observe that a vertex with a static degree of at least 6 corresponds to a vertex with a static degree of at least 3 in the underlying undirected tree. As can easily be seen from the following argument, all branching vertices have fixed arrival and departure times, i.e. all incoming edges of a branching vertex have the same label, as well as all outgoing edges. Let v be a branching vertex and let for some neighbor w of v w.l.o.g. $\lambda(w, v) = \Delta - 1$. For all outgoing edges (v, x) of v excluding (v, w) this means $\lambda(v, x) = 0$ as no waiting is allowed. This in turn requires $\lambda(x, v) = \Delta - 1$ for the remaining incoming edges (x, v). The latter implies $\lambda(v, w) = 0$. Let y and z be two branching vertices with distance $\hat{d}(y, z)$ and let $t(y)$, $t(z)$ be the timestamps of their incoming edges. Then the timestamps of their outgoing edges have to be $(t(y) + 1) \bmod \Delta$ and $(t(z) + 1) \bmod \Delta$, respectively. Let $P = (y = v_0, v_1, t_1) \dots (v_{\ell-1}, z = v_\ell, t_\ell)$ be a fastest temporal y-z-path and its length ℓ. By Definition 4, the duration of P is $d(P) = t_\ell - t_1 + 1$. We note that for any feasible solution $d(P) = d(y, z) = \hat{d}(y, z)$. To not exceed $D_{y,z} = \hat{d}(y, z)$ the following must hold: $\exists i \in \mathbb{N}_0 : i \cdot \Delta + t(z) = t_\ell = d(P) + t_1 - 1 = \hat{d}(y, z) + (t(y) + 1) - 1$. By definition and due to symmetry, this means: $t(z) \equiv \hat{d}(y, z) + t(y) \pmod{\Delta}$ and $t(y) \equiv \hat{d}(y, z) + t(z) \pmod{\Delta}$. Therefore, the following must also apply: $0 \equiv 2 \cdot \hat{d}(y, z) \pmod{\Delta}$.

If the distance of any pair of branching vertices is a multiple of $\Delta/2$, we can start at an arbitrary branching vertex r and set its arrival time to $\Delta - 1$. Without waiting, the departure time of this vertex is 0. Then we can assign labels iteratively as enforced by already labeled adjacent edges as specified in Algorithm 1. (However, we emphasize that we have to choose the root as a branching vertex here.) As all branching vertices have a distance of a multiple of $\Delta/2$, this procedure assigns them fixed arrival and departure times with no waiting. Hence, with such a labeling there is a temporal path between every pair of vertices without waiting. If there are no branching vertices at all, we can assign increasing labels independently for both directions of the path. ◀

This necessary and sufficient condition implies immediately a linear time algorithm for these instances. This does not contradict our observation that the problem becomes NP-complete if waiting times are allowed, i.e., $D_{u,v} \geq \hat{d}(u, v)$. This is even true for the slack parameter $k = 0$, since k is a only lower bound for the allowed waiting time.

3.2 Efficiently Solvable Cases

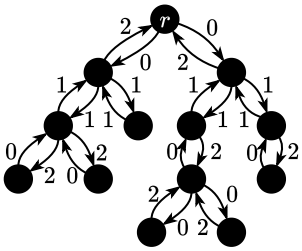
In this section we demonstrate how to realize all instances with $\Delta \leq k + 1$ for odd Δ and with $\Delta \leq k + 2$ for even Δ. Just for completeness, we start with the trivial case $\Delta = 1$.

▶ **Theorem 6.** *For $\Delta = 1$ all instances of TGR and DiTGR are feasible.*

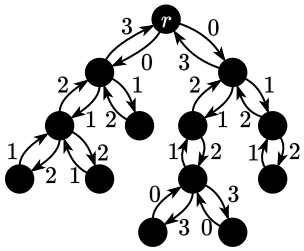
Proof. Obviously, all labels are forced to have the same value $\lambda = 0$. Since the period Δ is 1, there is no waiting time at all at any vertex. So every shortest path in the underlying static graph can be realized as a fastest temporal path with duration equal to its length, which means that all instances are realizable. ◀

Algorithm 1 Realizing instances that can always be realized

```
Choose an arbitrary vertex r as root.
For each edge e = (a, b), do:
        if e points away from the root,
                set λ(a, b) = d̂(r, a) mod Δ
        if e points to the root,
                set λ(a, b) = (Δ − d̂(r, a)) mod Δ
```



(a) Realization for $k + 1 \geq \Delta = 3$.



(b) Realization for $k + 2 \geq \Delta = 4$.

Figure 3 Example: Two realizations with waiting times at most $\Delta - 1$ respectively $\Delta - 2$ as constructed by Algorithm 1.

▶ **Theorem 7.** *All instances of* DiTTR *with* $\Delta \leq k + 1$ *for odd* Δ *and with* $\Delta \leq k + 2$ *for even* Δ *are feasible.*

Proof. All instances with $\Delta \leq k + 1$ for odd Δ and with $\Delta \leq k + 2$ for even Δ can be realized by a simple algorithm detailed in Algorithm 1. First, choose an arbitrary vertex r as the root. Then, traverse the tree and, depending on its distance from the root and whether it points to or away from the root, assign a label to each edge as specified in Algorithm 1. Two realizations computed by this algorithm can be seen in Figure 3. There are three cases for the direction of a path between two vertices:

1. The path runs entirely towards the root.
2. The path leads strictly away from the root.
3. The path first heads towards the root and then changes direction away from it.

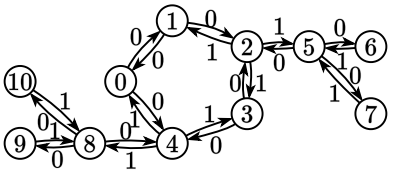
In the first two cases there is no delay at all. For the last case, it is easy to see that waiting time can only occur when changing direction and therefore only once per path. Thus, the maximum waiting time is at most $\Delta - 1$ on any path. If Δ is even, the waiting time is at most $\Delta - 2$, because by construction incoming and outgoing edges of every vertex are assigned values of different parity. Hence, there cannot be two identical labels in a row on a path. ◀

▶ **Corollary 8.** *All instances of* DiTTR *with* $\Delta = 2$ *are feasible.*

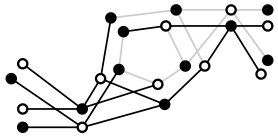
This follows from Theorem 7 with $\Delta = 2$ and $k = 0$ and can easily be generalized to directed bipartite graphs.

▶ **Corollary 9.** *All instances of* DiTGR *where the input is restricted to a directed bipartite graph* $G = (U, V, E)$ *with* $E \subseteq (U \times V) \cup (V \times U)$ *and period* $\Delta = 2$ *are feasible.*

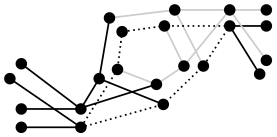
Proof. Given any directed, bipartite graph $G = (U, V, E)$ with $E \subseteq (U \times V) \cup (V \times U)$ and $\Delta = 2$, set $\lambda(e_0) = 0$ for all $e_0 \in E \cap (U \times V)$ and $\lambda(e_1) = 1$ for all $e_1 \in E \cap (V \times U)$. Since every path in G alternates between vertices in U and vertices in V, there is no waiting time. ◀



(a) A graph G with a feasible labeling for the case $D'_{0,10} = 3$.



(b) The bicolored graph G' corresponding to the labeling of G in Figure 4a with $D'_{0,10} = 3$.



(c) The graph G'' with $D''_{1,10} = 4$ is not bipartite. The dotted lines indicate an cycle of odd length.

Figure 4 Example: A graph $G = (V, E)$ containing one bidirected cycle of odd length and two attached trees. The auxiliary graphs $G' = (E, E')$ and $G'' = (E, E'')$ for two different matrices D' and D'' with $D'_{3,0} = D''_{3,0} = 2$, $D'_{6,0} = D''_{6,0} = 4$, $D'_{7,9} = D''_{7,9} = 6$, $D'_{9,1} = D''_{9,1} = 4$, $D'_{10,3} = D''_{10,3} = 3$ and $D'_{0,10} = 3$, $D''_{1,10} = 4$. Black and dotted edges are derived by the specific D' or D'', whereas gray edges correspond to potential additional constraints for general D. The graph G'_1 in Figure 4b is bicolored. The colors of the vertices correspond to the feasible labeling shown in Figure 4a. The graph G'' is not bipartite, the instance with D'' is therefore infeasible.

A graph where all shortest paths are unique is called *geodetic* [18, p. 105] or *min-unique* [20].

▶ **Theorem 10.** *Instances of DiTGR with period $\Delta = 2$ which are restricted to a directed geodetic graph $G = (V, E)$ with $D_{u,v} \in \{\hat{d}(u, v), \infty\}$ for all $u, v \in V$ can be decided in polynomial time.*

Proof. We show this by giving a reduction to 2–Vertex-Coloring as shown in Figure 4. Note that as the graph is geodetic there is by definition a unique shortest path in the underlying static graph for every pair of vertices (v, w) [18, p. 105]. Given an instance $(G = (V, E), D, \Delta)$, let $\mathtt{P} \subseteq 2^E$ be the set of all edge sets of paths in G and let $\mathtt{P_e} \subseteq \mathtt{P} \setminus \{\emptyset\}$ be the set of edge sets of all shortest paths where no waiting time is allowed. We construct an undirected auxiliary graph $G' = (E, E')$ with $E' = \{\{(x, y), (y, z)\} \mid \exists P \in \mathtt{P_e} : (x, y), (y, z) \in P\}$. This means that the labeling has to be chosen such that $\lambda(x, y) = (1 - \lambda(y, z)) \bmod \Delta$ for any $\{(x, y), (y, z)\} \in E'$ since waiting at y on the way from x to z is not allowed. As $\Delta = 2$, this is true if and only if $\lambda(x, y) \neq \lambda(y, z)$ for all $\{(x, y), (y, z)\} \in E'$. Thus, given a feasible coloring $C : E \to \{0, 1\}$ for G', we can set $\lambda(x, y) = C((x, y))$ for all $(x, y) \in E$. Therefore, the instance is feasible if and only if G' is bipartite which can be decided in polynomial time. ◀

In sharp contrast, we will show in the following section, that DiTGR is NP-complete for general graphs even for the special case that $\Delta = 2$.

4 Hardness Results

In this section, we present several hardness results. First, we extend the previously known hard cases by showing that TGR and DiTGR are hard for $\Delta = 2$. Second, we present hardness results for all remaining cases of values for k and Δ for DiTTR.

4.1 NP-completeness of DiTGR and TGR for $\Delta = 2$

▶ **Theorem 11.** *DiTGR is NP-complete even for the special case where period $\Delta = 2$, the edges E are bidirected, the constraints D are symmetric and the only odd cycle in G is a triangle.*

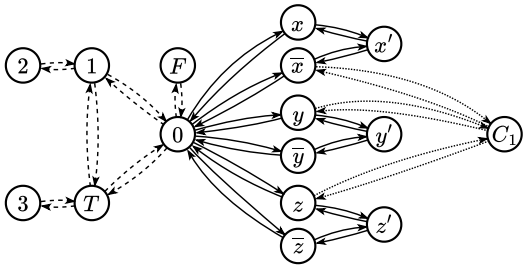


Figure 5 Construction with gadgets for the variables x, y, z and for the clause $C_1 = (\overline{x} \vee y \vee z)$. The dashed edges indicate the basic structure. The edges for the variable gadgets are solid, and those for the clauses are dotted.

Proof. We reduce 3-SAT to DiTGR as follows: Given a 3-CNF formula $\Phi = \{C_1, \ldots, C_m\}$ with a set of variables X, construct a graph $G = (V, E)$ with

$$
\begin{aligned}
V &= \{T, F, 0, 1, 2, 3\} \cup \{x, \overline{x}, x' | x \in X\} \cup \Phi \\
E &= \{(2,1), (1,2), (3,T), (T,3), (1,T), (T,1), (1,0), (0,1), (T,0), (0,T), (F,0), (0,F)\} \\
&\quad \cup \{(0,x), (x,0), (0,\overline{x}), (\overline{x},0) | x \in X\} \\
&\quad \cup \{(x,x'), (x',x), (\overline{x},x'), (x',\overline{x}) | x \in X\} \\
&\quad \cup \{(l,C), (C,l) | l \in \{x, \overline{x} | x \in X\} \wedge C \in \Phi \wedge l \in C\}
\end{aligned}
$$

as shown in Figure 5 and set the upper bounds D as follows:

$$
\begin{aligned}
&D_{2,F} = D_{F,2} = D_{3,2} = D_{2,3} = D_{3,F} = D_{F,3} = 3 && \\
&D_{T,x'} = D_{x',T} = D_{F,x'} = D_{x',F} = 3 && \text{for all } x \in X \\
&D_{x,\overline{x}} = D_{\overline{x},x} = 2 && \text{for all } x \in X \\
&D_{T,C} = D_{C,T} = 3 && \text{for all } C \in \Phi
\end{aligned}
$$

All other values of D are set to ∞. For each pair of vertices with a finite distance bound, this enforces that we have to realize a labeling without waiting on some shortest path. Let without loss of generality $\lambda(T,1) = 0$. Then the upper bounds D between the vertices 2, 3 and F enforce that $\lambda(T,1) = \lambda(1,T) = \lambda(T,0) = \lambda(0,T) = \lambda(1,0) = \lambda(0,1) = 0$ and $\lambda(3,T) = \lambda(T,3) = \lambda(2,1) = \lambda(1,2) = \lambda(F,0) = \lambda(0,F) = 1$.

For each variable $x \in X$, there exist two possible shortest paths from 0 and therefore from T respectively F to x': one over x and one over $\overline{x}$. Since the paths from T and the paths from F start with different labels (0 respectively 1), they also have to continue with different labels to avoid waiting. This means $\lambda(0,x) = 1 - \lambda(0,\overline{x}) = 1 - \lambda(x,x') = \lambda(\overline{x},x')$. The same holds for the opposite direction: $\lambda(x,0) = 1 - \lambda(\overline{x},0) = 1 - \lambda(x',x) = \lambda(x',\overline{x})$. Then $D_{x,\overline{x}} = D_{\overline{x},x} = 2$ enforces that the labels in both directions have to be the same, i.e. $\lambda(0,x) = \lambda(x,0)$: Going without waiting from x to $\overline{x}$ over 0 requires $\lambda(x,0) = 1 - \lambda(0,\overline{x}) = \lambda(0,x)$. Going over x' requires $\lambda(x,x') = 1 - \lambda(x',\overline{x}) = \lambda(x',x)$ which results in the same labeling.

For every clause $C = \{l_1, l_2, l_3\}$ there are three shortest paths from T to C: each one leads over one of the literals l_1, l_2 and l_3 in C. A fastest path can only have a duration of 3 as enforced by the upper bounds, if the edge $(0,l)$ to the literal l has the label 1. Therefore at least one of the edges $(0,l_1)$, $(0,l_2)$, $(0,l_3)$ has to have the label 1.

An assignment $a : X \mapsto \{0,1\}$ corresponds to the labeling of the edges from 0 to the variables, i.e. $\lambda(0,x) = \lambda(x,0) = a(x)$ for all $x \in X$. If $\Phi \in$ 3-SAT and a is a satisfying assignment, then we extend λ to $\lambda(l,C) = \lambda(C,l) = 1 - \lambda(0,l)$ for every clause $C \in \Phi$ which

is satified by the literal l, which leads to no waiting on the path between T and C and thus $(G, D, 2) \in$ DiTGR. If $(G, D, 2) \in$ DiTGR with the labeling λ, then the corresponding assignment a must satisfy $\Phi \in$ 3-SAT. ◀

From the proof, we can therefore conclude:

▶ **Corollary 12.** *TGR is NP-complete even for the special case that* $\Delta = 2$.

4.2 NP-complete Cases with Linear Size Gadgets

In this and the following subsection we present hardness results for all other cases, i.e. for all pairs of values (k, Δ) as shown in Figure 2. In all cases, the basic idea is to construct gadgets to enforce that for some specific edge (v_1, v_2) the timestamps for both directions have the same value, i.e. $\lambda(v_1, v_2) = \lambda(v_2, v_1)$. However, every value of λ is possible, it is not restricted by the gadget. In fact, every solution implies $\Delta - 1$ further symmetrical solutions in which all values are shifted by some value $x < \Delta$. By enforcing this for every single edge of an instance I, we can reduce the undirected TTR to DiTTR. All gadgets considered in this paper have a size polynomial in Δ and k. We distinguish between cases where we found linear-size gadgets and cases where we found quadratic-size gadgets.

▶ **Lemma 13.** *If there is a polynomial size gadget which is a realizable instance of* Δ-k-DiTTR *and which enforces* $\lambda(v_1, v_2) = \lambda(v_2, v_1)$ *for some specific pair of vertices* (v_1, v_2), *we can reduce* Δ-k-TTR *to* Δ-k-DiTTR.

Proof. Let $I = (G = (V, E), D, \Delta)$ be an instance of TTR and $(\hat{G}, \hat{D}, \Delta)$ be a gadget enforcing $\lambda(a, b) = \lambda(b, a)$. First we create a directed graph G' by replacing every undirected edge of G by two antiparallel directed edges. Then we enforce $\lambda(v, w) = \lambda(w, v)$ for every edge $e = \{u, v\} \in E$ by inserting a copy of the gadget into G', identifying (u, v) with (a, b) and (v, u) with (b, a), and setting D' consistently with D and $\hat{D}$. Since the resulting graph is a tree (G and the gadget G' are trees that are connected by one common edge) and any two copies of the gadget share at most one common vertex the construction does not produce any shortcuts.

Thus, every valid solution λ' for DiTTR with respect to G' and D' immediately implies a valid solution λ for TTR if we set $\lambda(\{u, v\}) = \lambda'((u, v)) = \lambda'((v, u))$. Conversely, we can construct a valid solution for DiTTR from a valid solution for TTR by setting $\lambda'((u, v)) = \lambda'((v, u)) = \lambda(\{u, v\})$. Since the gadget is feasible, there exists a solution where the timestamp of the edges (a, b) and (b, a) is 0. We can then set the labels in the copy of the gadget for each edge $\{u, v\}$ to a solution that is shifted modulo Δ by $\lambda(\{u, v\})$. Since Δ and k are fixed, this construction is possible in polynomial time if the time to compute the gadget is a function of only Δ and k. ◀

▶ **Lemma 14.** *We can construct a gadget* $G_{\Delta,0}$ *that enforces* $\lambda(v_1, v_2) = \lambda(v_2, v_1)$ *for some pair of vertices* (v_1, v_2) *for odd period* Δ *and minimum slack* $k = 0$.

Proof. We construct this gadget as follows:

$$V = \{1, \dots, 4 + \lfloor \frac{\Delta}{2} \rfloor\}$$

$$E = \{(1,3), (3,1), (2,3), (3,2)\}$$

$$\cup \{(i, i+1) | i \in \{3, \dots, 3 + \lfloor \frac{\Delta}{2} \rfloor\}\} \cup \{(i+1, i) | i \in \{3, \dots, 3 + \lfloor \frac{\Delta}{2} \rfloor\}\}$$

$$D_{1,2} = D_{2,1} = 2$$

$$D_{1,4+\lfloor \frac{\Delta}{2} \rfloor} = D_{4+\lfloor \frac{\Delta}{2} \rfloor,1} = D_{2,4+\lfloor \frac{\Delta}{2} \rfloor} = D_{4+\lfloor \frac{\Delta}{2} \rfloor,2} = 2 + \lfloor \frac{\Delta}{2} \rfloor$$

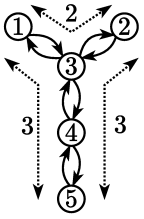


Figure 6 Gadget for $\Delta = 3$ and $k = 0$ that enforces $\lambda(4,5) = \lambda(5,4)$. The dotted lines indicate $D_{v,w} < \infty$.

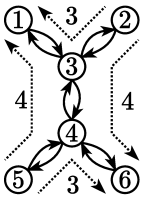


Figure 7 Gadget for $\Delta = 5$ and $k = 1$ that enforces $\lambda(3,4) = \lambda(4,3)$. The dotted lines indicate $D_{v,w} < \infty$.

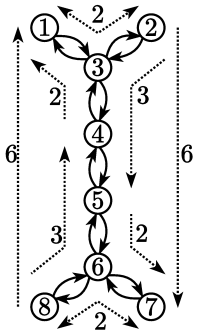


Figure 8 Gadget for $\Delta = 4$ and $k = 0$ that enforces $\lambda(4,5) = \lambda(5,4)$. The dotted lines indicate $D_{v,w} < \infty$.

All other values of D are set to ∞. As no waiting time is allowed, vertex 3 has a fixed arrival/departure time (see proof for Observation 5). Let $t(3)$ be the timestamps of its incoming edges. For the sake of convenience, let us assume that $t(3) = \Delta - 1$ and that the departure time is 0. This implies $\lambda(3,4) = 0$ and therefore $\lambda(3 + \lfloor \frac{\Delta}{2} \rfloor, 4 + \lfloor \frac{\Delta}{2} \rfloor) = \lfloor \frac{\Delta}{2} \rfloor$. Symmetrically, the following is also true: $\lambda(4,3) = \Delta - 1$ and $\lambda(4 + \lfloor \frac{\Delta}{2} \rfloor, 3 + \lfloor \frac{\Delta}{2} \rfloor) = \Delta - 1 - \lfloor \frac{\Delta}{2} \rfloor$. As Δ is odd this means $\lambda(4 + \lfloor \frac{\Delta}{2} \rfloor, 3 + \lfloor \frac{\Delta}{2} \rfloor) = \lambda(3 + \lfloor \frac{\Delta}{2} \rfloor, 4 + \lfloor \frac{\Delta}{2} \rfloor) = \lfloor \frac{\Delta}{2} \rfloor$. Therefore we can enforce $\lambda(v_1, v_2) = \lambda(v_2, v_1)$ for the vertices $v_1 = 3 + \lfloor \frac{\Delta}{2} \rfloor$ and $v_2 = 4 + \lfloor \frac{\Delta}{2} \rfloor$. ◀

For $\Delta = 3$ the gadget is shown in Figure 6. In this gadget, $\lambda(5,4) = \lambda(4 + \lfloor \frac{\Delta}{2} \rfloor, 3 + \lfloor \frac{\Delta}{2} \rfloor) = \lambda(3 + \lfloor \frac{\Delta}{2} \rfloor, 4 + \lfloor \frac{\Delta}{2} \rfloor) = \lambda(4,5)$ is enforced. This gadget has a size linear in Δ and only six values in D that are not infinity. Furthermore, D is symmetric.

▶ **Lemma 15.** *There is a gadget $G_{\Delta,k}$ that even with the limitation $D_{u,v} \geq \hat{d}(u,v) + k$ enforces $\lambda(v_1, v_2) = \lambda(v_2, v_1)$ for some pair of vertices (v_1, v_2) as long as $\Delta \geq 4 \cdot k + 1 \wedge k \geq 1$ and minimum slack k is odd. For any even values of k with $\Delta \geq 4 \cdot k + 5$ we can simply use the gadget for $k + 1$.*

Proof. The following gadget enforces $\lambda(3 + \lfloor \frac{k}{2} \rfloor, 3 + \lceil \frac{k}{2} \rceil) = \lambda(3 + \lceil \frac{k}{2} \rceil, 3 + \lfloor \frac{k}{2} \rfloor)$:

$$
\begin{aligned}
V &= \{1, \ldots, k+5\} \\
E &= \{(k+3, k+4), (k+4, k+3), (k+3, k+5), (k+5, k+3)\} \\
&\quad \cup \{(1,3), (3,1), (2,3), (3,2)\} \\
&\quad \cup \{(i, i+1) | i \in \{3, \ldots, k+2\}\} \\
&\quad \cup \{(i+1, i) | i \in \{3, \ldots, k+2\}\} \\
D_{2,1} &= D_{k+4,k+5} = k+2 \\
D_{2,k+5} &= D_{k+4,1} = 2 \cdot k + 2
\end{aligned}
$$

All other values of D are set to ∞. Let w.l.o.g. $\lambda(k+4, k+3) = 0$. We show that there is only one solution by looking at possible values of λ for the edges from and to leaves. We observe $\hat{d}(k+4, 1) = \hat{d}(2, k+5) = k+2$. As there is no waiting required but at most a waiting time of k allowed on the path from vertex $k+4$ to vertex 1, the following must hold: $\lambda(3,1) \in \{k+1, \ldots, 2 \cdot k + 1\}$ (recall $\Delta \geq 4 \cdot k + 1$). That means in turn that $\lambda(2,3)$ is in the range $\{0, 1, \ldots, 2 \cdot k\}$. The corner case $\lambda(2,3) = 0$ occurs when $\lambda(3,1) = k+1$ and with maximum waiting time. Conversely, with $\lambda(3,1) = 2 \cdot k + 1$ and with no waiting time, $\lambda(2,3)$ is at most $2 \cdot k$.

In the same way we can conclude that $\lambda(k+3,k+5) \in \{k+1,\dots,4\cdot k,(4\cdot k+1) \bmod \Delta\}$. Because $\Delta \geq 4\cdot k+1$ only the last item may be affected by the modulo operator and would be assigned the value 0 in that case. Suppose now that $\lambda(k+3,k+5) \in \{k+2,\dots,4\cdot k,(4\cdot k+1) \bmod \Delta\}$, i.e. any other possible value than $k+1$. Then $d(k+4,k+5) \geq k+2-0+1 = k+3 > D_{k+4,k+5} = k+2$ and thus, the solution cannot be valid for these values of $\lambda(k+3,k+5)$. Therefore $\lambda(k+3,k+5) = k+1$ which means that there is no waiting time at all on the paths from $k+4$ to 1 and from 2 to $k+5$ but a waiting time of k on the paths from 2 to 1 and from $k+4$ to $k+5$. Thus, $\lambda(3+\lfloor\frac{k}{2}\rfloor, 3+\lceil\frac{k}{2}\rceil) = \lceil\frac{k}{2}\rceil = \lambda(3+\lceil\frac{k}{2}\rceil, 3+\lfloor\frac{k}{2}\rfloor)$. Therefore we can enforce $\lambda(v_1,v_2) = \lambda(v_2,v_1)$ for the vertices $v_1 = 3+\lceil\frac{k}{2}\rceil$ and $v_2 = 3+\lfloor\frac{k}{2}\rfloor$. ◀

Figure 7 shows the gadget for $\Delta = 5$ and $k = 1$. This gadget also has a size linear in k and therefore also in Δ and a constant amount of values in D that are not infinity. The constraints D are not symmetric; however, setting them as such doesn't affect the proof: For all feasible solutions $\lambda(3+\lfloor\frac{k}{2}\rfloor, 3+\lceil\frac{k}{2}\rceil) = \lambda(3+\lceil\frac{k}{2}\rceil, 3+\lfloor\frac{k}{2}\rfloor)$ holds and there is still at least one feasible solution.

▶ **Lemma 16.** *There is a gadget for $\Delta = 4$ and $k = 0$ that enforces $\lambda(v_1,v_2) = \lambda(v_2,v_1)$ for some pair of vertices (v_1,v_2).*

The rough idea is that we need a break in symmetry. So we start with two copies of a known gadget (see Lemma 14) that we merge at the edges for which we can enforce equality of the labels. The resulting gadget would be infeasible, so we relax the constraints D slightly, making them asymmetric, and thereby obtain a feasible gadget of constant size with the claimed property.

Proof. The following gadget enforces $\lambda(4,5) = \lambda(5,4)$. It is shown in Figure 8. On the path from vertex 8 to vertex 1, waiting time is allowed, but only at vertex 4. Symmetrically, there can be no waiting time on the path from 2 to 6 except at vertex 5.

$$
\begin{aligned}
V &= \{1,\dots,7\} \\
E &= \{(1,3),(3,1),(2,3),(3,2)\} \cup \{(6,7),(7,6),(6,8),(8,6)\} \\
&\quad \cup \{(i,i+1) | i \in \{3,\dots,5\}\} \cup \{(i+1,i) | i \in \{3,\dots,5\}\} \\
D_{1,2} &= D_{2,1} = D_{7,8} = D_{8,7} = 2 \\
D_{5,7} &= D_{4,1} = 2 \\
D_{8,4} &= D_{2,5} = 3 \\
D_{8,1} &= D_{2,7} = 6
\end{aligned}
$$

All other values of D are set to ∞. Let w.l.o.g. $\lambda(8,6) = 1$. As no waiting is allowed at vertex 6, it has a fixed arrival/departure time of 2. This leads to $\lambda(5,6) = \lambda(7,6) = 1$, $\lambda(6,5) = \lambda(6,7) = \lambda(6,8) = 2$ and combined with $D_{8,4} = 3$ to $\lambda(5,4) = 3$. On the path from vertex 2 to vertex 7, waiting time is only allowed at vertex 5 and must not exceed one time step. Thus, $\lambda(4,5) \in \{3,0\}$ and $\lambda(3,4) \in \{2,3\}$. Furthermore the following holds: $\lambda(4,3) \in \{0,1\}$. Because there can be no waiting time at vertex 3, it has a fixed arrival/departure time and $\lambda(3,4) - \lambda(4,3) \equiv 1 \pmod{\Delta}$. Therefore only $\lambda(4,3) = 1$ and $\lambda(3,4) = 2$ is feasible. Hence, $\lambda(4,5) = 3 = \lambda(5,4)$. ◀

4.3 NP-Complete Cases with Quadratic Size Gadgets

For the remaining cases with odd period Δ, we have found a gadget of quadratic size $2\cdot(1+\Delta\cdot(k+1))$, the shape of which is reminiscent of a comb.

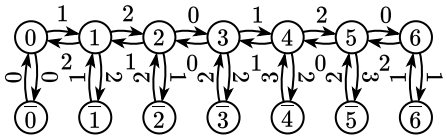


Figure 9 The gadget for $\Delta = 3$, $k = 1$ which enforces $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$, $\lambda(3,\overline{3}) = \lambda(\overline{3},3)$ and $\lambda(6,\overline{6}) = \lambda(\overline{6},6)$.

▶ **Lemma 17.** *There is a gadget G_i that even with the limitation $D_{u,v} \geq \hat{d}(u,v) + k$ enforces $\lambda(v_1,v_2) = \lambda(v_2,v_1)$ for some pair of vertices (v_1,v_2) as long as $\Delta \geq k+2 \wedge k \geq 1$ and period Δ is odd.*

The gadget for $\Delta = 3$, $k = 1$ is shown in Figure 9.

Proof. We show the result for $k = \Delta - 2$. This inherits to all smaller k since the limitation for those is weaker and thus complements to Theorem 7 for odd Δ.

The following gadget enforces $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$ as shown in Figure 10:

$$
\begin{aligned}
V &= \{0,\dots,(k+1)\Delta, \overline{0},\dots,\overline{(k+1)\Delta}\}\\
E &= \{(i-1,i),(i,i-1) \mid 1 \leq i \leq (k+1)\Delta\}\\
&\quad \cup \{(i,\overline{i}),(\overline{i},i) \mid 0 \leq i \leq (k+1)\Delta\}\\
D_{\overline{i},\overline{j}} &= \hat{d}(\overline{i},\overline{j}) + k \text{ for all } 0 \leq i,j \leq (k+1)\Delta
\end{aligned}
$$

All other values of D are set to ∞. In the following, we will call the part that consists of the vertices $\{0,\dots,(k+1)\Delta\}$ the main path. We call the direction from 0 to $(k+1)\Delta$ the *forward direction*. The reverse direction is called the *backward direction*.

We investigate the increase of the maximum waiting time in forward direction to (respectively in backward direction from) i defined by

$$
\begin{aligned}
w^+_\lambda(i) &:= \max_{j<i}\{d(\overline{j},i) - \hat{d}(\overline{j},i)\} \text{ and analogously}\\
w^-_\lambda(i) &:= \max_{j<i}\{d(i,\overline{j}) - \hat{d}(i,\overline{j})\}.
\end{aligned}
$$

We show that both w^+_λ and w^-_λ have to increase after every Δ edges on the main path. This means there is waiting time in one direction on the main path at all vertices $i \cdot \Delta$.

We have $w^+_\lambda(0) = w^-_\lambda(0) = -\infty$ since there is no $j < i$. We can easily choose λ such that $w^+_\lambda(1) = w^-_\lambda(1) = 0$. An equivalent inductive definition is

$$
\begin{aligned}
w^+_\lambda(i+1) = \max\{&w^+_\lambda(i) + (\lambda(i,i+1) - \lambda(i-1,i) - 1) \bmod \Delta,\\
&(\lambda(i,i+1) - \lambda(\overline{i},i) - 1) \bmod \Delta\} \text{ and analogously}\\
w^-_\lambda(i+1) = \max\{&w^-_\lambda(i) + (\lambda(i,i-1) - \lambda(i+1,i) - 1) \bmod \Delta,\\
&(\lambda(i,\overline{i}) - \lambda(i+1,i) - 1) \bmod \Delta\}.
\end{aligned}
$$

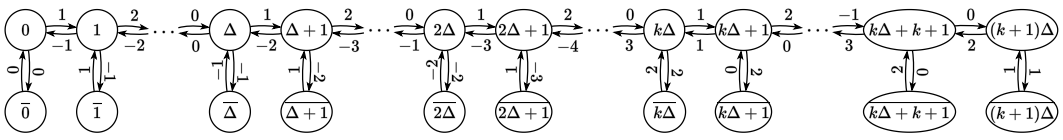


Figure 10 The gadget $(G = (V,E), D, \Delta)$ with a feasible labeling for any odd value of Δ.

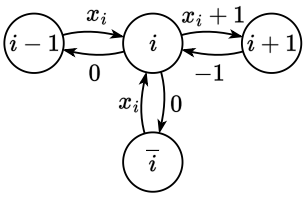


Figure 11 Let w.l.o.g. $\lambda(i, i-1) = 0$. For the case $x_i > w_\lambda^+(i)$ we get $w_\lambda^-(i+1) = w_\lambda^-(i)$.

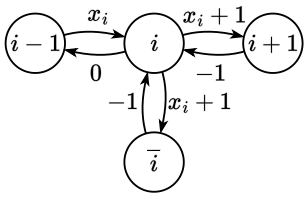


Figure 12 Let w.l.o.g. $\lambda(i, i-1) = 0$. For the case $x_i \leq w_\lambda^+(i)$ we get $w_\lambda^-(i+1) = \max\{w_\lambda^-(i), x_i + 1\}$.

For the case that there is no waiting at i on the main path in either direction, this is simplified to

$$w_\lambda^+(i+1) = \max\{w_\lambda^+(i), (\lambda(i, i+1) - \lambda(\bar{i}, i) - 1) \bmod \Delta\} \text{ and analogously}$$
$$w_\lambda^-(i+1) = \max\{w_\lambda^-(i), (\lambda(i, \bar{i}) - \lambda(i+1, i) - 1) \bmod \Delta\}.$$

We want to have each increase of the maximum waiting time on the main path as late as possible in forward direction. We first derive conditions under which $w_\lambda^-(i)$ respectively $w_\lambda^+(i)$ must increase. We then show that there is a solution in which we have the increase as late as possible and in which the waiting time just does not exceed k. At every increase, we have to wait in one direction on the main path. As the gadget is symmetrical, an earlier increase is also not possible, as that would mean waiting on the main path at a later point viewed from the other side. No increase, that means $w_\lambda^-(i+1) = w_\lambda^-(i)$, requires that there is no waiting at vertex i on the main path, i.e. $\lambda(i, i-1) \equiv \lambda(i+1, i) + 1 \pmod{\Delta}$ (see Figure 11 and Figure 12). However, there can be waiting at vertex i on the path from $i+1$ to $\bar{i}$ as long as it does not exceed the previous maximum waiting time. Note that we can wait at i on the way from $i-1$ to $\bar{i}$ at most $k - w_\lambda^+(i)$ times, otherwise we would wait on the way from some $\bar{j}$ to $\bar{i}$ more than k times. This means $\lambda(i, \bar{i}) - (\lambda(i-1, i) + 1) \leq k - w_\lambda^+(i)$. Let $x_i = (\lambda(i-1, i) - \lambda(i, i-1)) \bmod \Delta$. We will discuss two cases: $x_i > w_\lambda^+(i)$ and $x_i \leq w_\lambda^+(i)$.

The first case allows us to set $\lambda(i, \bar{i}) = (\lambda(i+1, i) + 1) \bmod \Delta = \lambda(i, i-1)$ as in Figure 11 since $(\lambda(i, i-1) - (\lambda(i-1, i) + 1)) \bmod \Delta = (-x_i - 1) \bmod \Delta \leq \Delta - 2 - w_\lambda^+(i) = k - w_\lambda^+(i)$. This means there is no waiting at vertex i on the path from $i+1$ to $\bar{i}$. We could also wastefully set $\lambda(i, \bar{i})$ to any value that agrees with both constraints (waiting from $i-1$ and waiting from $i+1$). In any case, there is no increase of the maximum waiting time in backwards direction, i.e. $w_\lambda^-(i+1) = w_\lambda^-(i)$.

In the other case, the smallest timestamp we can assign to $(i, \bar{i})$ is $(\lambda(i-1, i) + 1) \bmod \Delta$ (see Figure 12) which leads to the smallest possible waiting time to $\bar{i}$. This enforces $w_\lambda^-(i+1) > w_\lambda^-(i)$ only if the waiting time exceeds the previous maximum waiting time, i.e. $(\lambda(i, \bar{i}) - (\lambda(i+1, i) + 1)) \bmod \Delta = (\lambda(i-1, i) - \lambda(i+1, i)) \bmod \Delta = x_i + 1 \bmod \Delta > w_\lambda^-(i)$. Therefore, there is an increase with $x_i \geq w_\lambda^-(i)$.

This means that with $\lambda(i, i-1) \equiv \lambda(i+1, i) + 1 \pmod{\Delta}$ an increase of the waiting time $w_\lambda^-(i+1) > w_\lambda^-(i)$ is enforced if and only if $w_\lambda^+(i) \geq x_i \geq w_\lambda^-(i)$ and analogously $w_\lambda^+(i+1) > w_\lambda^+(i)$ is enforced if and only if $w_\lambda^-(i) \geq x_i \geq w_\lambda^+(i)$.

Starting as in Figure 10 with $\lambda(0, \bar{0}) = \lambda(\bar{0}, 0)$ and no waiting at vertex 0, we get $w_\lambda^-(i) = w_\lambda^+(i) = 0$ for $1 \leq i \leq \Delta$ (since $x_i = 2i \pmod{\Delta}$ assumes all values $< \Delta$ once), and only at vertex Δ, we get $\lambda(\Delta - 1, \Delta) = \lambda(\Delta, \Delta - 1)$ which means $x_\Delta = 0$ enforcing an increase of the waiting time to $w_\lambda^-(\Delta + 1) = w_\lambda^+(\Delta + 1) = 1$. If we would instead start with $\lambda(0, \bar{0}) \neq \lambda(\bar{0}, 0)$ then we would get $x_i = 0$ already for an $i < \Delta$ leading to an earlier increase.

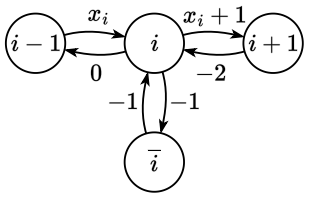


Figure 13 Let w.l.o.g. $\lambda(i, i-1) = 0$. For the case $x_i = w^-_\lambda(i) = w^+_\lambda(i)$ we choose $\lambda(i+1, i)$ such that $w^-_\lambda(i+1) = w^+_\lambda(i+1) = w^-_\lambda(i) + 1$ and in this way the value of x_{i+1} is increased by 3 instead of 2 relative to x_i. This means that for the following $\Delta - 1$ vertices, the value x_j will run through all other values modulo Δ, before it becomes $w^-_\lambda(j)$ again.

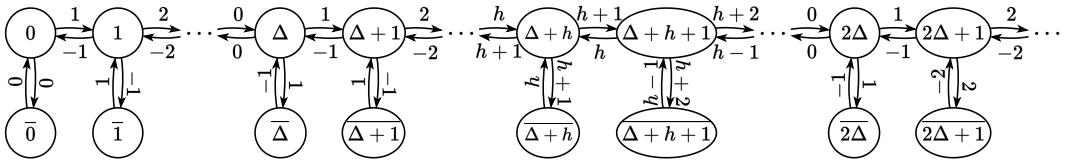


Figure 14 If we do not wait at vertex Δ in one direction on the main path, we get $w^+_\lambda(\Delta+h+1) = w^+_\lambda(\Delta+h+1) = 1$ for $h := \frac{\Delta-1}{2}$ leading to an increase $w^+_\lambda(\Delta+h+2) = w^+_\lambda(\Delta+h+2) = 2$ at vertex $\Delta+h+1$ already.

In fact, we can set $\lambda(\Delta, \Delta+1) = (\lambda(\Delta-1, \Delta)+1) \bmod \Delta$ and $\lambda(\Delta+1, \Delta) = (\lambda(\Delta, \Delta-1)-2) \bmod \Delta$ as in Figure 13. We then set $\lambda(\Delta, \overline{\Delta}) = \lambda(\overline{\Delta}, \Delta) = (\lambda(\Delta, \Delta-1)-1) \bmod \Delta$ and in this way get $w^-_\lambda(i) = w^+_\lambda(i) = 1$ for $\Delta < i \leq 2\Delta$ and only at vertex 2Δ, we get $\lambda(2\Delta-1, 2\Delta) \equiv \lambda(2\Delta, 2\Delta-1)+1 \pmod{\Delta}$ enforcing an increase of the waiting time to $w^+_\lambda(2\Delta+1) = w^+_\lambda(2\Delta+1) = 2$.

By induction on j, we can then set $\lambda(j\Delta, j\Delta+1) = (\lambda(j\Delta-1, j\Delta)+1) \bmod \Delta$ and $\lambda(j\Delta+1, j\Delta) = (\lambda(j\Delta, j\Delta-1)-2) \bmod \Delta$ as well as $\lambda(j\Delta, \overline{j\Delta}) = \lambda(\overline{j\Delta}, j\Delta) = (\lambda(j\Delta, j\Delta-1)-j) \bmod \Delta$. This way we get $w^-_\lambda(i) = w^+_\lambda(i) = j$ for $j\Delta < i \leq (j+1)\Delta$ and only at vertex $(j+1)\Delta$, we get $x_{(j+1)\Delta} = j$ enforcing an increase of the waiting time to $w^-_\lambda(j\Delta+1) = w^+_\lambda(j\Delta+1) = j$. The increase just reaches vertex $(k+1)\Delta$ at the end of the gadget with the allowed waiting time of k, where there is no further increase as it is the last vertex. This is accomplished by waiting in one of the two directions on the main path at vertices $i \cdot \Delta$ for $1 \leq i \leq k$.

However, if we do not wait in one of the two directions on the main path in one of these k cases as in Figure 13 but instead continue as in Figure 12, the increase of w^+_λ and w^-_λ would already be enforced the next time on a position $\frac{\Delta+1}{2}$ later, as shown in Figure 14. In the figure, this position would be $\Delta + \frac{\Delta+1}{2} = \Delta + h + 1$. Nevertheless, the increase would be enforced after every Δ steps from this point on, which would lead to a waiting time of $k+1$ before the end of the gadget.

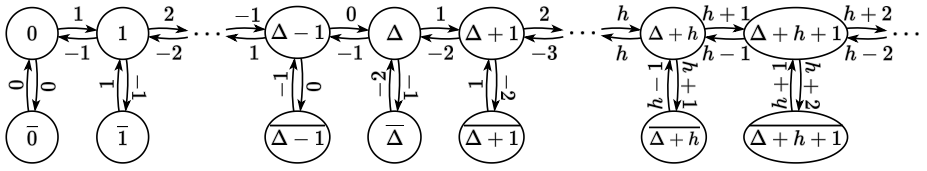


Figure 15 Waiting already at $\Delta - 1$ leads to $w^+_\lambda(\Delta) = 0$ and $w^-_\lambda(\Delta) = 1 = x_\Delta$ for $h := \frac{\Delta-1}{2}$ immediately leading to $w^-_\lambda(\Delta+1) = 1$ and even $w^+_\lambda(\Delta+1) = 2$.

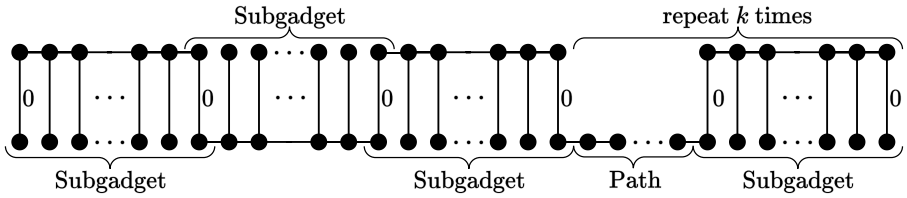
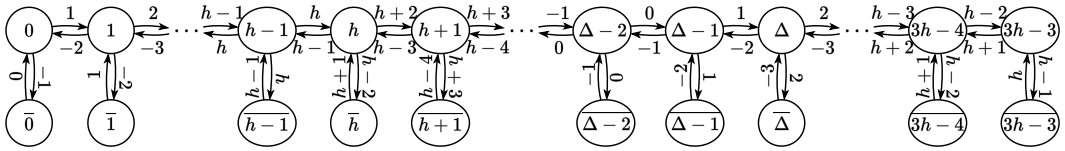
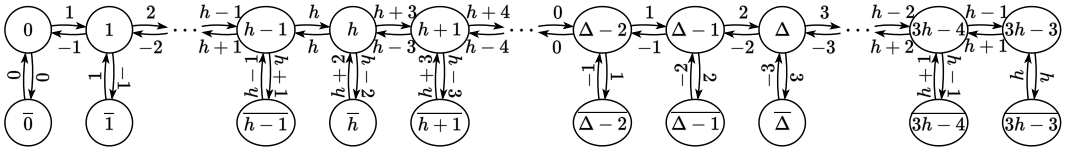


Figure 16 The gadget consists of Δ subgadgets, where the second is upside down and overlaps wit the first and the third at one edge and the others are connected by a path of length $\Delta - 2$. The waiting time between two vertices within a subgadget is limited to k.

Conversely, if we wait earlier than necessary in either direction on the main path, this would mean an increase later than possible in the symmetric case. Another way to see this is shown in Figure 15. Waiting early forces an increase at the next multiple of Δ anyway. This means that $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$ and symmetrically $\lambda((k+1)\Delta, \overline{(k+1)\Delta}) = \lambda(\overline{(k+1)\Delta}, (k+1)\Delta)$ is enforced in the gadget. ◀



(a) Realization for the gadget from Lemma 17 for even Δ starting with $\lambda(0,\overline{0}) = \lambda(\overline{0},0) + 1$.



(b) Realization for the gadget from Lemma 17 for even Δ starting with $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$.

Figure 17 Two realizations for the gadget from Lemma 17 for even Δ. It turns out that starting the labeling with $\lambda(0,\overline{0}) = \lambda(\overline{0},0) + 1$ requires waiting one time at some vertex (here h) in both directions (see Figure 17a) but that starting the labeling with $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$ requires waiting even two times at some vertex (here h) in both directions (see Figure 17b). Therefore we cannot use this gadget to enforce $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$ for even Δ.

Finally, we have constructed a similar gadget for the remaining cases with even Δ.

▶ **Lemma 18.** *There is a gadget G_i that even with the limitation $D_{u,v} \geq \hat{d}(u,v) + k$ enforces $\lambda(e_1,e_2) = \lambda(e_2,e_1)$ for some pair of vertices (e_1,e_2) as long as $\Delta \geq k+3 \wedge k \geq 1$ and period Δ is even.*

The gadget is shown in Figure 16. We show that there cannot be any waiting time in the first subgadget, since the remaining subgadgets and paths already enforce a total waiting time of k by investigating again the increase in maximum waiting time on the main path to and from some vertex i.

Proof. For even Δ the same gadget as in Lemma 17 does not enforce $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$. In fact, starting with $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$ requires more waiting than starting with $\lambda(0,\overline{0}) \neq \lambda(\overline{0},0)$ as can be seen in Figure 17. The comparison of both solutions favors a labeling that does not meet our requirements. For this reason the following construction uses a more complex

gadget as sketched in Figure 16. We show the result for $k = \Delta - 3$. This inherits to all smaller k since the limitation for those is weaker and thus complements to Theorem 7 for even Δ.

Let $h := \Delta/2$ and $\ell = 3\Delta - 3 + (2\Delta - 3)(\Delta - 3)$. The following gadget of length $\hat{d}(\overline{0}, \overline{\ell}) = \ell + 2(\Delta - 1)$ as depicted in Figure 16 enforces $\lambda(0, \overline{0}) = \lambda(\overline{0}, 0)$:

$$
\begin{aligned}
V &= \{0, \dots, \ell, \overline{0}, \dots, \overline{\ell}\} \\
E &= \{(i-1, i), (i, i-1) && \mid 1 \le i \le \Delta - 1\} \\
&\cup \{(i, \overline{i}), (\overline{i}, i) && \mid 0 \le i \le \Delta - 1\} \\
&\cup \{(\overline{i-1}, \overline{i}), (\overline{i}, \overline{i-1}) && \mid \Delta \le i \le 2\Delta - 2\} \\
&\cup \{(i, \overline{i}), (\overline{i}, i) && \mid \Delta - 1 \le i \le 2\Delta - 2\} \\
&\cup \{(i-1, i), (i, i-1) && \mid 2\Delta - 1 + j(2\Delta - 3) \le i \le 3\Delta - 3 + j(2\Delta - 3), \\
& && \quad 0 \le j \le \Delta - 3\} \\
&\cup \{(i, \overline{i}), (\overline{i}, i) && \mid 2\Delta - 2 + j(2\Delta - 3) \le i \le 3\Delta - 3 + j(2\Delta - 3), \\
& && \quad 0 \le j \le \Delta - 3 \\
&\cup \{(\overline{i-1}, \overline{i}), (\overline{i}, \overline{i-1}) && \mid 3\Delta - 2 + j(2\Delta - 3) \le i \le 4\Delta - 5 + j(2\Delta - 3), \\
& && \quad 0 \le j < \Delta - 4\} \\
D_{\overline{i},\overline{j}} &= \hat{d}(\overline{i}, \overline{j}) + k && \text{for all } 0 \le i, j \le \Delta - 1 \\
D_{i,j} &= \hat{d}(i, j) + k && \text{for all } \Delta - 1 \le i, j \le 2\Delta - 2 \\
D_{\overline{i},\overline{j}} &= \hat{d}(\overline{i}, \overline{j}) + k && \text{for all } 2\Delta - 2 + j(2\Delta - 3) \le i, j \le 3\Delta - 3 + p(2\Delta - 3), \\
& && \quad 0 \le p < \Delta - 3 \\
D_{\overline{0},\overline{\ell}} &= \hat{d}(\overline{0}, \overline{\ell}) + k
\end{aligned}
$$

All other values of D are set to ∞. For simplicity we give the vertex set as $\{0, \dots, \ell, \overline{0}, \dots, \overline{\ell}\}$ instead of explicitly removing the isolated vertices. We show that there is a labeling for the gadget in which the waiting on the main path takes place exclusively in both directions of the k connecting paths instead of on the subgadgets corresponding to the canonical gadget in Lemma 17. Furthermore any λ has to wait at least two times in any direction in the context (the path or the neighboring subgadgets) of each connecting path. This leaves no waiting time for the first subgadget, which enforces $\lambda(0, \overline{0}) = \lambda(\overline{0}, 0)$.

First we examine the waiting times for each subgadget individually before we investigate them in the context of the whole gadget. In the solution without waiting on the main paths of the subgadgets the labeling is the same for each subgadget.

▷ Claim 19. Each subgadget consisting of a comb of 2Δ vertices enforces either $\lambda(0, \overline{0}) = \lambda(\overline{0}, 0) = \lambda(\Delta - 1, \overline{\Delta - 1}) = \lambda(\overline{\Delta - 1}, \Delta - 1)$ or requires waiting on the main path at least two times in the total of both directions.

Proof. We define the waiting time w^+_λ and w^-_λ as before in Lemma 17 and make similar conclusions which differ in the following statements because $\Delta - k$ is now 3 instead of 2: We still have $\lambda(i, \overline{i}) - (\lambda(i-1, i) + 1) \le k - w^+_\lambda$ and now discuss the two cases $x_i > w^+_\lambda(i) + 1$ and $x_i \le w^+_\lambda(i) + 1$. Again, the first case allows us to set $\lambda(i, \overline{i}) = (\lambda(i+1, i) + 1) \bmod \Delta = \lambda(i, i-1)$ (see Figure 11) since $(\lambda(i, i-1) - \lambda(i-1, i) - 1) \bmod \Delta = (-x_i - 1) \bmod \Delta \le \Delta - 3 - w^+_\lambda(i) = k - w^+_\lambda(i)$ with no increase of the waiting time in backwards direction, i.e. $w^-_\lambda(i+1) = w^-_\lambda(i)$. In the other case, the smallest timestamp time we can assign $(i, \overline{i})$ is again $(\lambda(i-1, i) + 1) \bmod \Delta$. This enforces $w^-_\lambda(i+1) > w^-_\lambda(i)$ only if the waiting time

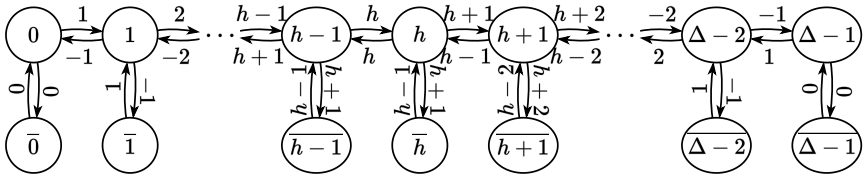


Figure 18 A subgadget with a feasible labeling without waiting on the main path. Observe that λ is symmetric on the subgadget.

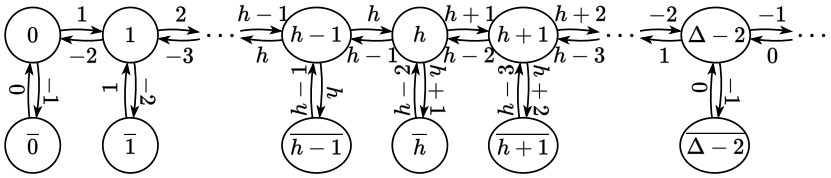


Figure 19 A labeling for the subgadget starting with x_0 odd that would lead to $w^+_\lambda(\Delta-1) = k+1$.

exceeds the previous maximum waiting time, which means $(\lambda(i,\bar{i}) - (\lambda(i+1,i)+1)) \bmod \Delta = (\lambda(i-1,i) - \lambda(i+1,i)) \bmod \Delta = (x_i+1) \bmod \Delta > w^-_\lambda(i)$ (see Figure 12). Therefore there is an increase with $x_i \geq w^-_\lambda(i)$.

That means with $\lambda(i,i-1) \equiv \lambda(i+1,i)+1 \pmod{\Delta}$ an increase of the waiting time $w^-_\lambda(i+1) > w^-_\lambda(i)$ is enforced if and only if $w^+_\lambda(i)+1 \geq x_i \geq w^-_\lambda(i)$ and analogously $w^+_\lambda(i+1) > w^+_\lambda(i)$ is enforced if and only if $w^-_\lambda(i)+1 \geq x_i \geq w^+_\lambda(i)$.

Starting with the equal labeling $\lambda(0,\bar{0}) = \lambda(\bar{0},0)$ and no waiting at vertex 0, we get $w^-_\lambda(i) = w^+_\lambda(i) = 0$ for $1 \leq i \leq h := \Delta/2$ and only here, we get $\lambda(h-1,h) = \lambda(h,h-1) = h$ enforcing an increase of the waiting time to $w^-_\lambda(h+1) = w^+_\lambda(h+1) = 1$. But here again we have $x_{h+1} = \lambda(h,h+1) - \lambda(h+1,h) = w^+_\lambda(h+1)$ enforcing an increase of the waiting time to $w^-_\lambda(h+2) = w^+_\lambda(h+2) = 2+1 = 3$. By induction on j, we have $x_{h+j} = w^+_\lambda(h+j)$ enforcing an increase of the waiting time to $w^+_\lambda(h+j+1) = w^+_\lambda(h+j+1) = 2j+1$.

This just reaches $\Delta-1$ at the end of the subgadget as shown in Figure 18 with the allowed waiting time of $k = 2h-3$. However, waiting on the main path in one of the two directions as in Figure 13 would not help since $x_{h+j} = w^+_\lambda(h+j)+1$ still enforces an increase of the waiting time.

The same holds if we start differently as can be seen in the following consideration: If we start with values where $\lambda(0,\bar{0}) - \lambda(\bar{0},0)$ is even then we would get the latest increase with $\lambda(0,\bar{0}) = \lambda(\bar{0},0)$, since $x_i = 2i \pmod{\Delta}$ assumes all even values $< \Delta$ once. Values $\lambda(0,\bar{0}) \neq \lambda(\bar{0},0)$ would mean a larger difference x_0 and would lead to $x_i = 0$ already for an $i < h$ leading to an earlier increase. Like before, waiting on the main path in one direction does not change this.

In case $\lambda(0,\bar{0}) - \lambda(\bar{0},0)$ is odd we get the latest increase with $x_0 = 1$, since $x_i = 2i+1 \pmod{\Delta}$ assumes all odd values $< \Delta$ once. A larger difference x_0 would cause the increase to happen at latest at vertex h, which requires an additional increase at vertex $h+1$ and this in turn an increase at vertex $h+2$ and so on. This is shown in Figure 19. The increase progresses like this until we reach vertex $\Delta-2$ with $x_{\Delta-2} = k$ and $w^-_\lambda(\Delta-2) = w^+_\lambda(\Delta-2) = k-1$, which also enforces an increase. We get a maximum waiting time of $k+1$ at vertex $\Delta-1$, therefore not producing a feasible labeling. Waiting once in one of the two directions in this case would mean that $\lambda(\overline{\Delta-1},\Delta-1) - \lambda(\Delta-1,\overline{\Delta-1})$ is even. Therefore, we can apply symmetry to show that waiting once still does not help. ◁

We can construct a solution by choosing λ such that there is no waiting time in any of the subgadgets and for all subgadgets the following holds: $\lambda(0,\overline{0}) = \lambda(\overline{0},0) = \lambda(\Delta-1,\overline{\Delta-1}) = \lambda(\overline{\Delta-1},\Delta-1)$. As the paths consist of $\Delta-1$ vertices and consequently $\Delta-2$ edges we have to wait one time in each direction on the path for this to be possible. As the gadget contains k paths this does not exceed the at most allowed waiting time.

If we do not wait two times on any path we prevent one of the neighboring subgadgets from starting with equal values in both directions on the first or last edge. Because of Claim 19 we would have to wait at least two times in the respective subgadget. Therefore, waiting at least two times is required in the context of each of the k paths and thus, no waiting is possible in the first subgadget. Hence, for the first subgadget, which is not contained in the context of any path, $\lambda(0,\overline{0}) = \lambda(\overline{0},0)$ holds. ◀

▶ **Theorem 20.** *For every Δ and k with $\Delta > k+2$ or $\Delta = k+2$ with Δ odd, Δ-k-DiTTR is NP-complete. Therefore the problem DiTTR is NP-complete.*

Proof. The construction in Proposition 6 of [16] to show that TTR is NP-complete holds for any $\Delta \geq 3$ and all values of k with $k+2 \leq \Delta$, because $D_{u,v} \in \{\hat{d}(u,v)+\Delta-2,\infty\}$. Note that the reduction from vertex coloring implies strong NP-completeness. This covers all cases as claimed in Figure 2. We can combine Lemma 13 and the lemmas referenced in Figure 2 to show that for every Δ and k with $\Delta > k+2$ or $\Delta = k+2$ with Δ odd, Δ-k-DiTTR is NP-complete. That also means DiTTR is NP-complete since already the special versions Δ-k-DiTTR, where k, which is an implicit parameter of D in the input, and Δ are two constants in one of the cases (d)-(h) in Figure 2, are NP-complete. This also holds if Δ is part of the input, as TTR is strongly NP-complete. ◀

Remark: The proof of Theorem 20 also implies that for even $\Delta > 2$ and $\Delta = k+2$ the undirected problem version is NP-complete, while every instance of the corresponding directed problem version is realizable, as indicated in Figure 2.

5 Conclusion and Further Research

In this paper, we have initiated the study of the directed version of the graph realization problem for periodic temporal graphs subject to pairwise upper bounds on the fastest paths. We obtained hardness results for several special cases and identified some easily solvable ones. For trees, we provided a full characterization for all periods Δ and all values of the minimum slack parameter k, giving a lower bound on the maximum allowed waiting time on each path.

For future work, many problem variants are worth further consideration. An interesting extension would be to also consider upper bounds on the slack. Instead of uniform bounds on the slack, one could also consider multiplicative bounds to reflect that more waiting is acceptable on longer paths. Or one could turn the problem into an optimization problem where one wants to minimize some measure of the deviation from the fastest paths or the desired quality of service. In some practical applications, it is useful to further restrict the labeling with additional constraints. For example, when planning train or tram timetables for single-track lines, it is necessary to ensure that a corresponding track section is only served in one direction at a time. Thus, an interesting type of constraint could be to require a solution with $\lambda(a,b) \neq \lambda(b,a)$ for all (or only certain specified) pairs of vertices $a,b \in V$. According to the proof of Theorem 11, this is NP-complete for $\Delta = 2$ for general graphs, which motivates the problem for trees but the constructions in this paper using Theorem 20 do not work with this property.

Further theoretical investigations may consider more general graph classes than just trees. Finally, it would be interesting to investigate the practical solvability of instances derived from real network topologies.

References

1 Richard P. Anstee. Properties of a class of (0,1)-matrices covering a given matrix. *Canadian Journal of Mathematics*, 34(2):438–453, 1982. doi:10.4153/CJM-1982-029-3.

2 Jamil N. Ayoub and Ivan T. Frisch. Degree realization of undirected graphs in reduced form. *Journal of the Franklin Institute*, 289(4):303–312, 1970. doi:10.1016/0016-0032(70)90273-5.

3 Amotz Bar-Noy, David Peleg, Mor Perry, and Dror Rawitz. Graph realization of distance sets. *Theoretical Computer Science*, 1019:114810, 2024. doi:10.1016/j.tcs.2024.114810.

4 Wai-Kai Chen. On the realization of a (p,s)-digraph with prescribed degrees. *Journal of the Franklin Institute*, 281(5):406–422, 1966. doi:10.1016/0016-0032(66)90301-2.

5 Jack Edmonds. Existence of k-edge connected ordinary graphs with prescribed degrees. *J. Res. Nat. Bur. Standards Sect. B*, 68:73–74, 1964.

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

7 Thomas Erlebach, Nils Morawietz, and Petra Wolf. Parameterized Algorithms for Multi-Label Periodic Temporal Graph Realization. In Arnaud Casteigts and Fabian Kuhn, editors, *3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2024)*, volume 292 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:16, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAND.2024.12.

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

9 S. Louis Hakimi and Stephen S. Yau. Distance matrix of a graph and its realizability. *Quarterly of Applied Mathematics*, 22:305–317, 1965.

10 Václav Havel. Poznámka o existenci konečných grafů. *Časopis pro pěstování matematiky*, 080(4):477–480, 1955. URL: http://eudml.org/doc/19050.

11 Daniel J. Kleitman and D. L. Wang. Decomposition of a graph realizing a degree sequence into disjoint spanning trees. *SIAM Journal on Applied Mathematics*, 30(2):206–221, 1976.

12 Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. Realizing temporal graphs from fastest travel times, 2024. doi:10.48550/arXiv.2302.08860.

13 Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. Temporal Graph Realization from Fastest Paths. In Arnaud Casteigts and Fabian Kuhn, editors, *3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2024)*, volume 292 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAND.2024.16.

14 Linda Lesniak. Eccentric sequences in graphs. *Periodica Mathematica Hungarica*, 6(4):287–293, December 1975. doi:10.1007/BF02017925.

15 Niels Lindner and Julian Reisch. An analysis of the parameterized complexity of periodic timetabling. *J. of Scheduling*, 25(2):157–176, 2022. doi:10.1007/s10951-021-00719-1.

16 George B. Mertzios, Hendrik Molter, Nils Morawietz, and Paul G. Spirakis. Realizing temporal transportation trees, April 2025. Extended abstract to appear in Proceedings of 51st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2025), LNCS, Springer. doi:10.48550/arXiv.2403.18513.

17 Michiel A. Odijk. Construction of periodic timetables, part 1: A cutting plane algorithm. Technical report, Technical Report 94-61, TU Delft, 1994.

18 Øystein Ore. *Theory of graphs*, volume XXXVIII of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 1965. Second printing.

19 Leon W. P. Peeters. *Cyclic railway timetable optimization*. PhD thesis, Erasmus Research Institute of Management, Erasmus University Rotterdam, The Netherlands, 2003.

20 Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM Journal on Computing*, 29(4):1118–1131, 2000. doi:10.1137/S0097539798339041.

21 Paolo Serafini and Walter Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989. doi:10.1137/0402049.

22 Hiroshi Tamura, Masakazu Sengoku, Shoji Shinoda, and Takeo Abe. Realization of a network from the upper and lower bounds of the distances (or capacities) between vertices. In *1993 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2545–2548. IEEE, 1993.