Minimize the Sum of Waiting Times in Periodic Temporal Trees

Julia Meusel
Martin Luther University Halle-Wittenberg, Germany

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

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

Klaus Reinhardt
Martin Luther University Halle-Wittenberg, Germany

Abstract

We introduce and analyze the problem of finding a Δ-labeling λ for an undirected tree $G = (V, E)$, such that the sum of overall waiting times of fastest paths between all vertex pairs is minimized in the Δ-periodic temporal graph (G, λ). That is, we aim to minimize $\sum_{(u,v)\in V\times V}(\text{dur}(u, v)-\text{dist}(u, v))$, where $\text{dur}(u, v)$ is the duration of a fastest temporal path from u to v and $\text{dist}(u, v)$ is the length of the shortest path between u and v in G. We show that this objective function essentially boils down to a known problem about partitioning a set of natural numbers that has applications in scheduling. From that problem we lift and adapt several upper and lower bounds for our problem. For example, we show that the problem admits an EPTAS, that is, an algorithm that can compute a $(1 + \epsilon)$–approximation to our problem in time $f(\frac{1}{\epsilon}) \cdot n^{\mathcal{O}(1)}$ for each $\epsilon > 0$. To the best of our knowledge, this is the first example of an efficient approximation algorithm for a temporal graph realization problem.

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

Keywords and phrases graph realization, fastest temporal path, periodic temporal graphs

Digital Object Identifier 10.4230/LIPIcs.SAND.2026.19

Funding *Nils Morawietz*: Supported by the French ANR, project ANR-22-CE48-0001 (TEMPOGRAL).

1 Introduction

Graph realization problems ask whether there is a graph that meets certain given conditions, such as prescribed degrees or distances [7, 9, 13]. Recent research has extended these problems to temporal graphs, also known as dynamic or time-varying graphs, where the edges are only available at specific times. In such models, the task is to assign time labels to the edges of a given static graph so that the resulting temporal graph satisfies constraints on properties such as connectivity [1, 4, 6, 12, 17, 15], degree sequences [5], reachability [3, 10, 8] or the duration of fastest paths [18, 16, 11, 19]. These graphs arise naturally in communication, social, and transportation networks, where the connectivity changes over time.

In many of these naturally occurring networks, connectivity is repeated periodically. To model this, periodic temporal graphs are particularly well-suited. Here, every edge becomes active again after a time period has passed. In the context of public transport, the vertex set V corresponds to stations, and the edge set E specifies which station pairs are directly connected. The objective is then to determine a labeling that optimizes specific criteria

related to travel times between pairs of vertices in the resulting periodic temporal graph. A path may use an edge only at times when that edge is active, and traversing an edge takes one unit of time. A fastest path between two vertices minimizes the travel time between them while only traversing active edges. We concentrate on the strict setting where the labels have to be strictly increasing whereas in the non-strict setting, multiple edges with the same label can be traversed in one timestep. We write $\mathrm{dur}(u, v)$ for the duration of such a fastest temporal path from u to v and $\mathrm{dist}(u, v)$ for the length of a shortest path, i.e., the path between these vertices traversing the fewest edges. Erlebach et al. [11] and Klobas et al. [16] investigated restrictions on the exact duration of the fastest temporal paths in periodic temporal graphs, while Mertzios et al. [19, 18] and Meusel et al. [21] focused on upper bounds for durations. Mertzios et al. also introduce the *stretch*, a multiplicative bound on the duration of fastest paths and optimize for this measure [18].

In this paper, we study a new global measure: the sum of the waiting times on a fastest path over all vertex pairs. This allows a few vertex pairs to have extremely long travel time if this means keeping the overall travel times low. In contrast to the deviation measures (i) $\max_{u,v}(\mathrm{dur}(u, v) - \mathrm{dist}(u, v))$ considered by Meusel et al. [21] and (ii) $\max_{u,v}(\mathrm{dur}(u, v)/\mathrm{dist}(u, v))$ considered by Mertzios et al. [18], we take a different approach. Specifically, we consider the deviation measure $\sum_{u,v}(\mathrm{dur}(u, v) - \mathrm{dist}(u, v))$, which we call the *total excess time*. Note that a fastest temporal path does not necessarily coincide with a shortest path. Long waiting times on an otherwise short route can mean that an alternative route with more edges is actually faster overall. The excess time consists of two components: the length of the detour compared to the shortest path, and the waiting time on the fastest path itself. On trees, this measure can also be called the *total waiting time* since there are no alternative paths that take a detour.

On general graphs, a reduction by Mertzios et al. [18] already implies that it is NP-hard to decide for $\Delta = 3$ whether a total excess time of 0 can be achieved. Hence, we will just focus on input graphs that are trees. This restricted graph class is still well motivated, as discussed by Mertzios et al. [19] and Meusel et al. [21].

Related work. Meusel et al. studied the Upper-Bounded Directed Temporal Tree Realization problem, where given a bidirected tree and a matrix D of upper bounds, the goal is to assign each edge exactly one label such that $\mathrm{dur}(u, v) \leq D_{u,v}$ for any vertex pair (u, v) in the resulting periodic temporal graph [21]. They fully characterized the complexity of Upper-Bounded Directed Temporal Tree Realization with respect to Δ and a parameter k. Here, k is a lower bound on the upper bounds in D, i.e., for the permitted waiting time on each path. In particular, they showed that there are parameter combinations for which the undirected problem is NP-hard, but the directed version becomes trivial.

Mertzios et al. [18] introduced the Stretched Temporal Graph Realization (STGR) problem, where the stretch α of an instance is the maximum of $\mathrm{dur}(u, v)/\mathrm{dist}(u, v)$ over all pairs of vertices in a periodic temporal graph. They prove that STRG is NP-hard even if $\Delta = 3$, $\alpha = 1$ and the graph has diameter 2. They further develop a local search algorithm and show that the optimization version of STGR, where the goal is to minimize the stretch, is hard to approximate within a factor of $\Delta^{1-\epsilon}$ or 2^{n^c} for each fixed $0 < \epsilon < 1$ and $c > 1$ respectively. On the other hand, it is possible to compute a solution with stretch at most $\Delta - \frac{\Delta-1}{\min(\mathrm{rad}+1,\mathrm{diam})}$ in polynomial time, where rad and diam denote the radius and diameter of the given input graph, respectively. Meusel et al. [20] extended this work by further characterizing the complexity landscape and proving the existence of (i) instances with stretch arbitrarily close to Δ and, (ii) instances with stretch arbitrarily close to 1.

Our contribution. In this work, we introduce the Temporal Waiting Time Minimization problem on trees and show how it can be expressed by a set of partitioning problems that can be solved individually. Exploiting the similarity to the known partitioning problem Sum of Square Partition, we transfer positive and negative results to Temporal Tree Waiting Time Minimization. We show that Temporal Tree Waiting Time Minimization is NP-hard and W[1]-hard when parametrized by Δ. This holds even for spider graphs, that is, trees with at most one vertex of degree at least 3. Temporal Tree Waiting Time Minimization can be solved in $n^{\mathcal{O}(\Delta)}$ time and in $3^d \cdot n^{\mathcal{O}(1)}$ time, where n is the number of vertices and d is the maximum degree of the input graph, respectively.

Finally, we show that the problem admits an efficient polynomial time approximation scheme (EPTAS) on trees, that is, an algorithm that can compute a $(1+\epsilon)$–approximation to our problem in time $f(\frac{1}{\epsilon}) \cdot n^{\mathcal{O}(1)}$ for each $\epsilon > 0$. To the best of our knowledge, this is the first example of an efficient approximation scheme for a temporal graph realization problem.

Overview. In Section 2, we start with a formal problem definition and notations. Then, in Section 3, we first present our key technical lemma implying that Temporal Tree Waiting Time Minimization can be solved locally as a series of individual partitioning problems at vertices. In Section 3.1, we present our hardness result and parameterized algorithms and in Section 3.2 we introduce our efficient approximation scheme. Finally, in Section 4, we conclude with a summary of our results and suggestions for future research.

2 Formal Problem Definition

In this section, we provide the necessary definitions and notations to define the Temporal Waiting Time Minimization problem formally.

For integers $i \leq j$, we may write $[i, j]$ for the set $\{\ell \mid i \leq \ell \leq j\}$. For a set X of integers, we may write $\sum X$ as a shorthand for $\sum_{x \in X} x$. For a graph G and a vertex $v \in V(G)$, we denote by $N_G(v)$ the *neighborhood* of v in G, that is, the set of all adjacent vertices of v.

▶ **Definition 2.1.** *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 2.2.** *A* Δ-periodic *temporal graph is a triple* $(G = (V, E), \lambda : E \to [0, \Delta - 1], \Delta)$ *which denotes the temporal graph* (G, Λ) *where* $\forall e \in E : \Lambda(e) = \{\lambda(e) + i \cdot \Delta \mid i \in \mathbb{N}_0\}$.

We call $\lambda : E \to [0, \Delta - 1]$ a Δ*-labeling*. 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 2.3.** *A* temporal s-z-path *of length* ℓ *in an undirected temporal graph* (G, Λ) *is a sequence* $P = (v_{i-1}, v_i, t_i)_{i=1}^{\ell}$ *for which the following holds:*

- $v_0 = s \wedge v_\ell = z$
- $\forall i, j \in \{0, \ldots, \ell\}, i \neq j : v_i \neq v_j$
- $\forall i \in \{1, \ldots, \ell\} : \{v_{i-1}, v_i\} \in E$
- $\forall i \in \{1, \ldots, \ell\} : t_i \in \Lambda((v_{i-1}, v_i))$
- $\forall i \in \{2, \ldots, \ell\} : t_{i-1} < t_i$

Let $\mathrm{dist}(u, v)$ be the static *distance* of u and v in the underlying static graph, that is, the length of the shortest path between these vertices. The traversal of an edge requires one time unit. The temporal s-z-path *starts* or *begins* at vertex s at time t_1, and it *reaches* or *arrives* at vertex z at time $t_\ell + 1$.

▶ **Definition 2.4.** *The* duration $\mathrm{dur}(P)$ *of a temporal path* $P = (v_{i-1}, v_i, t_i)_{i=1}^{\ell}$ *is defined as* $\mathrm{dur}(P) = t_\ell - t_1 + 1$.

A temporal u-v-path of smallest duration among all temporal u-v-paths is called a *fastest temporal u-v-path*. The duration of a fastest temporal path from u to v depending on Λ is denoted by $\mathrm{dur}_\Lambda(u, v)$. We simply write $\mathrm{dur}(u, v)$ whenever Λ is clear from the context. For brevity, we write $\lambda(u, v)$ instead of $\lambda(\{u, v\})$. Note that the symmetry $\lambda(u, v) = \lambda(v, u)$ persists. The *waiting time* of a temporal path $P = (v_{i-1}, v_i, t_i)_{i=1}^{\ell}$ at vertex v_i with $0 < i < \ell$ is defined as $\mathrm{wait}_{v_i}(P) = t_{i+1} - 1 - t_i$. Therefore, if $t_{i+1} - t_i \leq \Delta$ in a periodic temporal graph, that is, if the next edge is traversed as early as possible, the waiting time at vertex v_i is $(\lambda(v_i, v_{i+1}) - 1 - \lambda(v_{i-1}, v_i)) \bmod \Delta$. Note that this holds in particular for every fastest path. We call $\mathrm{dur}(u, v) - \mathrm{dist}(u, v)$ the *excess time*.

Formally, we analyze the following problem.

Temporal Waiting Time Minimization
Input: A graph $G = (V, E)$ and two integers Δ and k.
Question: Is there a Δ-labeling $\lambda\colon E \to [0, \Delta - 1]$, such that the sum of all pairwise excess times, i.e. the total excess time, in the resulting Δ-periodic temporal graph (G, λ) is at most k?

Restricted to trees, we call the problem Temporal Tree Waiting Time Minimization. Here, there are no alternative paths between any two vertices. This means the total excess time equals the total waiting time because no time is lost to detours.

Temporal Tree Waiting Time Minimization
Input: An undirected tree $G = (V, E)$ and two integers Δ and k.
Question: Is there a Δ-labeling $\lambda\colon E \to [0, \Delta - 1]$, such that the sum of all pairwise waiting times, i.e. the total waiting time, in the resulting Δ-periodic temporal graph (G, λ) is at most k?

Let G be a tree and let $\Delta \geq 2$ be an integer. We then define for each edge $\{v, w\}$ the set $V^w_{\{w,v\}}$ to be all the vertices that can be reached from v via w. That is, $V^w_{\{w,v\}}$ is the set of vertices that is in the same connected component as w after removing the (bridge) edge $\{v, w\}$ from the tree G. Note that for each path $P_{s,t}$ in G that has v as an internal vertex, the following applies: $s \in V^w_{\{w,v\}}$ and $t \in V^{w'}_{\{w',v\}}$ for distinct neighbors w and w' of v. For a Δ-labeling λ of G, we define the set $B^v_i := \{w \in N(v)\colon \lambda(v, w) = i\}$ for each $i \in [0, \Delta - 1]$. That is, the set of all neighbors w of v for which the edge $\{v, w\}$ receives label i under λ.

3 Our Results

We first show that on trees, the objective function, that is, the total waiting time, can be expressed as partitioning problems of numbers with the goal to minimize the sum of squares of each part of the partition. As a first step, we present the following key technical lemma, which rewrites the total waiting time objective.

▶ **Lemma 3.1.** *Let G be a tree and let λ be a Δ-labeling of G. Then, the total waiting time under λ is*

$$\sum_{v\in V}\left[\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\left(\sum_{w\in B_k^v}|V_{\{w,v\}}^w|\right)^2+\frac{\Delta-2}{2}\left(\sum_{w\in N(v)}|V_{\{w,v\}}^w|\right)^2-(\Delta-1)\sum_{w\in N(v)}|V_{\{w,v\}}^w|^2\right].$$

Proof. Trees have the benefit that there is only a single path $P_{s,t}$ from s to t for any vertex pair (s,t), and that the only path $P_{t,s}$ from (t,s) is the reverse path of $P_{s,t}$. This allows us to rewrite the formula of the total waiting time as the sum of waiting times on all internal vertices of a path, summed up over all paths of the graph. That is,

$$\sum_{u,v\in V}(\mathrm{dur}(u,v)-\mathrm{dist}(u,v))=\sum_{P_{u,v}=(u=p_0,p_1,\dots,p_{\ell-1},p_\ell=v)}\sum_{i=1}^{\ell-1}\mathrm{wait}_{p_i}(P_{u,v}),$$

where $\mathrm{wait}_{p_i}(P_{u,v})$ denotes the waiting time of path $P_{u,v}$ at vertex p_i.

We can further rewrite the total waiting time as the sum over all waiting times on each vertex. That is,

$$\sum_{\substack{P_{u,v}=\\(u=p_0,p_1,\dots,p_{\ell-1},p_\ell=v)}}\sum_{i=1}^{\ell-1}\mathrm{wait}_{p_i}(P_{u,v})=\sum_{v\in V}\left[\sum_{P_{s,t};s\in V_{\{i,v\}}^i;t\in V_{\{j,v\}}^j;i,j\in N(v);i\neq j}\mathrm{wait}_v(P_{s,t})\right].$$

In the following, let v be a fixed vertex of V. We will first rewrite the inner term $\sum_{P_{s,t};s\in V_{\{i,v\}}^i;t\in V_{\{j,v\}}^j;i,j\in N(v);i\neq j}\mathrm{wait}_v(P_{s,t})$. To this end, let $x_k:=|V_{\{k,v\}}^k|$ for each neighbor $k\in N(v)$.

The waiting time of a fastest path $P_{u,w}=(u=p_0,p_1,\dots,p_{\ell-1},p_\ell=w)$ at an intermediate vertex $v=p_i$ only depends on the labels of the incident edges of v on the path. More precisely, if $\lambda(p_{i-1},p_i)=\lambda(p_{i+1},p_i)$, then $\mathrm{wait}_{p_i}(P_{u,w})=\mathrm{wait}_{p_i}(P_{w,u})=\Delta-1$. Otherwise, $\mathrm{wait}_{p_i}(P_{u,w})+\mathrm{wait}_{p_i}(P_{w,u})=\Delta-2$. In particular, this means that the exact values of the labels are irrelevant. A shorter waiting time is obtained whenever two consecutive labels do not have the same value. With this, we can rewrite further:

$$\begin{aligned}\sum_{P_{s,t};s\in V_{\{i,v\}}^i;t\in V_{\{j,v\}}^j;i,j\in N(v);i\neq j}\mathrm{wait}_v(P_{s,t})&=\sum_{P_{s,t};s\in V_{\{i,v\}}^i;t\in V_{\{j,v\}}^j;i,j\in N(v);i\neq j}\mathrm{wait}_v(P_{i,j})\\&=\sum_{i\in N(v)}\sum_{j\in N(v)\setminus\{i\}}(x_i\cdot x_j\cdot\mathrm{wait}_v(P_{i,j})).\end{aligned}$$

Moreover, as already discussed, $\mathrm{wait}_v(P_{i,j})+\mathrm{wait}_v(P_{j,i})=2\Delta-2$ if $\lambda(i,v)=\lambda(j,v)$, that is, if there is some $k\in[0,\Delta-1]$, such that $i\in B_k^v$ and $j\in B_k^v$. Otherwise, $\mathrm{wait}_v(P_{i,j})+\mathrm{wait}_v(P_{j,i})=\Delta-2$. Hence, we get that

$$\begin{aligned}&\sum_{i\in N(v)}\sum_{j\in N(v)\setminus\{i\}}(x_i\cdot x_j\cdot\mathrm{wait}_v(P_{i,j}))\\&\qquad=\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\sum_{i\in B_k^v}\sum_{j\in B_k^v;i\neq j}(x_i\cdot x_j)+\frac{\Delta-2}{2}\sum_{i\in N(v)}\sum_{j\in N(v)\setminus\{i\}}(x_i\cdot x_j),\end{aligned}$$

as the average waiting time of $\mathrm{wait}_v(P_{i,j})$ and $\mathrm{wait}_v(P_{j,i})$ is $\frac{\Delta-2}{2}$ plus $\frac{\Delta}{2}$ if there is some $k\in[0,\Delta-1]$, such that $i\in B_k^v$ and $j\in B_k^v$.

Finally, we rewrite the last formula into the desired form:

$$\begin{aligned}
&\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\sum_{i\in B_k^v}\sum_{j\in B_k^v;i\neq j}(x_i\cdot x_j)+\frac{\Delta-2}{2}\sum_{i\in N(v)}\sum_{j\in N(v)\setminus\{i\}}(x_i\cdot x_j)\\
&=\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\left(\sum_{i\in B_k^v}\sum_{j\in B_k^v}(x_i\cdot x_j)-\sum_{i\in B_k^v}x_i^2\right)+\frac{\Delta-2}{2}\sum_{i\in N(v)}\left(\sum_{j\in N(v)}(x_i\cdot x_j)-x_i^2\right)\\
&=\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\sum_{i\in B_k^v}\sum_{j\in B_k^v}(x_i\cdot x_j)+\frac{\Delta-2}{2}\sum_{i\in N(v)}\sum_{j\in N(v)}(x_i\cdot x_j)-(\frac{\Delta}{2}+\frac{\Delta-2}{2})\sum_{i\in N(v)}x_i^2\\
&=\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\left(\sum_{i\in B_k^v}(x_i\cdot\sum_{j\in B_k^v}x_j)\right)+\frac{\Delta-2}{2}\sum_{i\in N(v)}\left(x_i\cdot\sum_{j\in N(v)}x_j\right)-(\Delta-1)\sum_{i\in N(v)}x_i^2\\
&=\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\left((\sum_{i\in B_k^v}x_i)\cdot(\sum_{j\in B_k^v}x_j)\right)+\frac{\Delta-2}{2}\left((\sum_{i\in N(v)}x_i)\cdot(\sum_{j\in N(v)}x_j)\right)-(\Delta-1)\sum_{i\in N(v)}x_i^2\\
&=\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\left(\sum_{i\in B_k^v}x_i\right)^2+\frac{\Delta-2}{2}\left(\sum_{i\in N(v)}x_i\right)^2-(\Delta-1)\sum_{i\in N(v)}x_i^2
\end{aligned}$$

As we sum this up over all vertices $v\in V$, we finally get that

$$\begin{aligned}
&\sum_{u,v}(\mathrm{dur}(u,v)-\mathrm{dist}(u,v))\\
&=\sum_{v\in V}\left[\frac{\Delta}{2}\sum_{k=0}^{\Delta-1}\left(\sum_{w\in B_k^v}|V_{\{w,v\}}^w|\right)^2+\frac{\Delta-2}{2}\left(\sum_{w\in N(v)}|V_{\{w,v\}}^w|\right)^2-(\Delta-1)\sum_{w\in N(v)}|V_{\{w,v\}}^w|^2\right].
\end{aligned}$$

◀

Computing an optimal solution for this objective function thus boils down to solving for each vertex $v\in V$ an instance of the following problem:

Waiting Time Minimization at Vertex
Input: A multiset of positive integers A of total sum N and a positive integer Δ.
Task: Find a partition of A into Δ disjoint subsets $(B_1,\ldots,B_\Delta)$, such that $\frac{(\Delta-2)}{2}N^2-(\Delta-1)\cdot\sum_{a\in A}(a)^2+\frac{\Delta}{2}\sum_{k=1}^{\Delta}(\sum_{b\in B_i}b)^2$ is minimized.

▶ **Corollary 3.2.** *An optimal solution for Temporal Tree Waiting Time Minimization can be found by solving for each vertex v an instance of Waiting Time Minimization at Vertex.*

Here, the multiset A corresponds to the vertex counts of the subtrees around v, i.e. $A=\{|V_{\{w,v\}}^w|\colon w\in N(v)\}$).

The crucial insight is that the instances can be solved independently, as the objective function for each individual vertex v only needs to consider the labels of the edges around v, and we are dealing with a tree (see Figure 1). That is, we can compute the optimal sum of waiting times in a tree by computing for each vertex v an optimal solution (given as a labeling λ_v of the edges incident with v) for the Waiting Time Minimization at Vertex-instance for v. Note that the labelings λ_v and λ_w for adjacent v and w do not necessarily assign the same label to the common edge. However, since the only important factor for

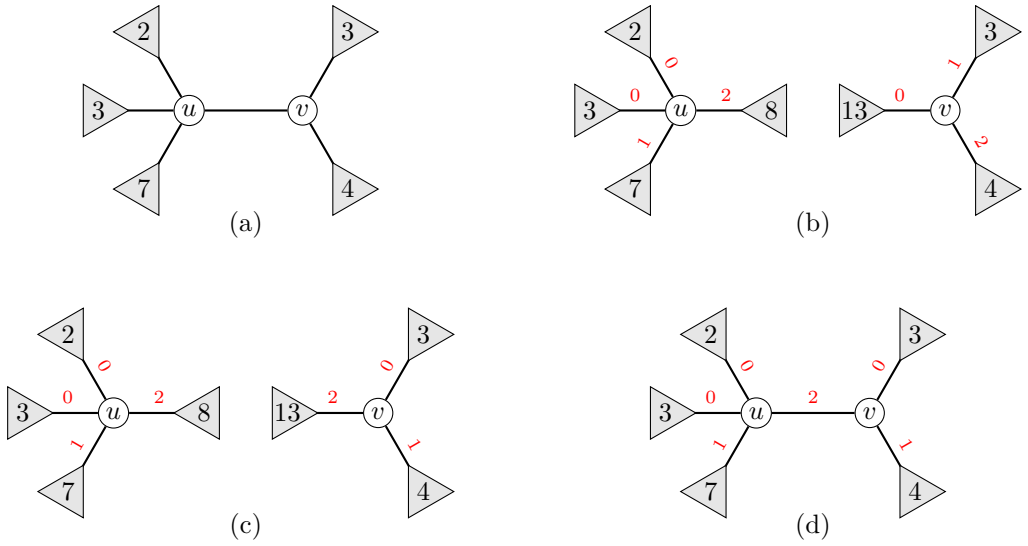

Figure 1 An illustration on the process of solving an instance of Temporal Tree Waiting Time Minimization for $\Delta = 3$ by solving for each vertex of the tree a respective instance of Waiting Time Minimization at Vertex and combining the solutions. The numbers in triangles indicate the vertex count of the respective subtree. Figure (d) shows an optimal solution for the depicted part of the tree shown in Figure (a). To obtain this labeling, we computed optimal solutions for both Waiting Time Minimization at Vertex instances shown in Figure (b) for vertices u and v. To let the two solutions agree on the label of the common edge $\{u, v\}$, we shifted all labels of the solution for v in Figure (c).

calculating the waiting time is whether consecutive labels are distinct, we can rearrange the specific values. For example, we can shift each label assigned by λ_w cyclic with respect to period Δ, until the resulting labeling λ'_w agrees with λ_v on the edge between w and v. As in a tree, each edge is a bridge, this process terminates in linear time if applied in a depth first search (DFS) fashion, hence yielding us an optimal labeling for our Temporal Tree Waiting Time Minimization instance.

Further, note that the objective function of Temporal Tree Waiting Time Minimization is minimized if and only if

$$\sum_{v \in V} \sum_{k=0}^{\Delta-1} \left(\sum_{w \in B_k^v} |V^w_{\{w,v\}}| \right)^2$$

is minimized, as all other parts of the objective function are identical for each partition of A. Hence, since the objective of Temporal Tree Waiting Time Minimization and therefore also Waiting Time Minimization at Vertex reduces to minimizing a sum of squared subset sums, we consider the following partitioning problem [2].

Sum of Square Partition
Input: A multiset of positive integers A and two positive integer k and t.
Question: Is there a partition of A into k disjoint subsets $(B_1, \ldots, B_k)$, such that $\sum_{1 \le i \le k} (\sum B_i)^2 \le t$?

▶ **Corollary 3.3.** *An optimal solution for Waiting Time Minimization at Vertex is also an optimal solution for Sum of Square Partition and vice versa.*

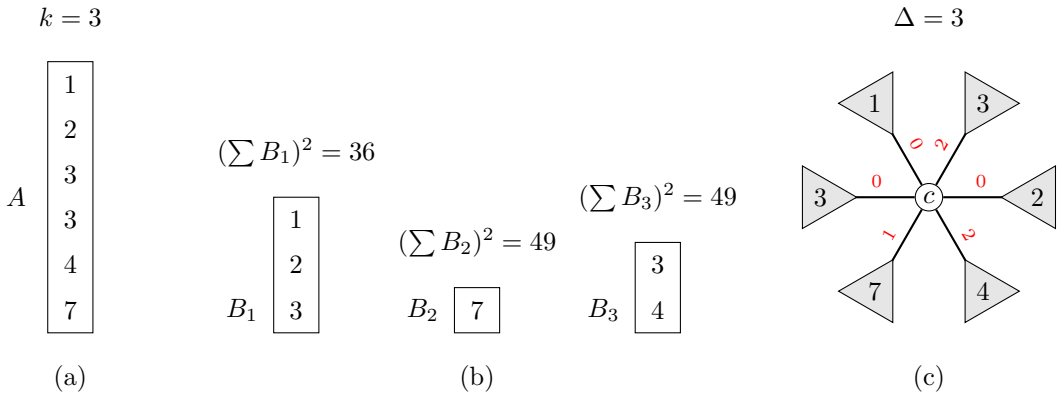

Figure 2 An illustration of the reduction from Sum of Square Partition to Waiting Time Minimization at Vertex and thus also to Temporal Tree Waiting Time Minimization with $k = \Delta = 3$. The numbers in the triangles indicate the number of vertices of that subtree, which is actually just a path with that many vertices.

3.1 Consequences for Upper and Lower Bounds

Based on this connection of the two problems, we will now transfer both positive and negative results from Sum of Square Partition to Temporal Tree Waiting Time Minimization. We start by showing that Temporal Tree Waiting Time Minimization is NP-hard and presumably does not admit an FPT algorithm for parameter Δ.

▶ **Corollary 3.4.** *Temporal Tree Waiting Time Minimization is NP-hard and W[1]-hard when parameterized by Δ even on spider graphs.*

Proof. First, note that Sum of Square Partition is NP-hard and W[1]-hard when parameterized by k even when all numbers are encoded in unary. This is due to the fact that the following variant of Bin Packing is NP-hard and W[1]-hard when parameterized by k even when all numbers are encoded in unary. This is implicitly shown in the proof of Lemma 6 in [14].

Bin Packing
Input: A multiset of positive integers A of total sum N and an integer k.
Question: Is there a partition of A into k disjoint subsets $(B_1, \ldots, B_k)$, such that $\sum_{a \in B_i} = \frac{N}{k}$?

The hardness transfers to Sum of Square Partition due to the convexity of $f(x) = x^2$, which ensures that the optimal possible value for a k–partition of A can only be achieved if each part sums up to $\frac{N}{k}$. The reduction to Waiting Time Minimization at Vertex then follows immediately (see Figure 2), as the objective functions of both problems are optimized by the same partitions (by setting $\Delta = k$). The reduction to Temporal Tree Waiting Time Minimization then simply creates a path on a vertices for each $a \in A$ and adds a single vertex c which is adjacent to one endpoint of each such path. ◀

Even though an FPT algorithm for temporal waiting time minimization parametrized by Δ is unlikely, Temporal Tree Waiting Time Minimization can still be solved in polynomial time for each constant Δ.

▶ **Corollary 3.5.** *Temporal Tree Waiting Time Minimization can be solved in $n^{\mathcal{O}(\Delta)}$ time for n being the number of vertices of the input graph.*

Proof. As we can solve an instance of Temporal Tree Waiting Time Minimization by solving an instance of Waiting Time Minimization at Vertex for each internal vertex of the tree, it suffices to show that Waiting Time Minimization at Vertex can be solved in $|A|^{\mathcal{O}(\Delta)}$ time. Essentially, this result can be obtained via dynamic programming in the same way as the folklore $n^{k+\mathcal{O}(1)}$-time algorithm for Bin Packing. Let $I := (A, \Delta)$ be an instance of Waiting Time Minimization at Vertex, let $q := |A|$, and let N be the sum of all elements of the multiset A. Note that $N \in |I|^{\mathcal{O}(1)}$, as each number in N is unary encoded. Moreover, assume without loss of generality that the elements of A are denoted by $A = \{a_1, \ldots, a_q\}$. We use a dynamic programming table D with $\Delta + 1$ dimensions, where the first dimension ranges from 0 to q and each other dimension ranges from 0 to N.

For each $i \in [0, q]$, we denote the set of the first i elements of A as $A_{\leq i} := \{a_j \mid 1 \leq j \leq i\}$. The table entry $D[i, b_1, \ldots, b_\Delta]$ stores a truth value which is 1 if and only if there is a Δ-partition $(B'_1, \ldots, B'_\Delta)$ of $A_{\leq i}$ with $|B'_\ell| = b_\ell$ for each $\ell \in [1, \Delta]$. We call B^1 the set of values for $(b_1, \ldots, b_\Delta)$ such that $D[q, b_1, \ldots, b_\Delta] = 1$. To compute (the objective value of) some optimal Δ-partition of A, it suffices to evaluate $\min_{(b_1,\ldots,b_\Delta)\in B^1} \sum_{\ell=1}^{\Delta}(b_\ell)^2$.

To initialize, we set $D[0, b_1, \ldots, b_\Delta] = \begin{cases} 1 & \forall \ell \in [1, \Delta] : b_\ell = 0 \\ 0 & \exists \ell \in [1, \Delta] : b_\ell > 0 \end{cases}$.

Then, we can set $D[i, b_1, \ldots, b_\Delta] = \begin{cases} 1 & \exists 1 \leq j \leq \Delta : D[i-1, b_1, \ldots, b_j - a_i, \ldots, b_\Delta] = 1 \\ 0 & \text{else} \end{cases}$.

Intuitively, this means that we determine which partially filled bags we can put a_i in to partition the remaining objects of $A_{\leq i-1}$ and achieve the prescribed sums.

Each entry can be computed in polynomial time and there are $(q+1) \cdot (N+1)^k$ entries in total. Hence, the whole algorithm runs in the desired running time. Note that a corresponding optimal Δ-partition can be computed in the same asymptotic running time via trace-back. ◀

Next, we show that for the maximum degree of the input graph, an FPT algorithm is possible.

▶ **Corollary 3.6.** *Temporal Tree Waiting Time Minimization can be solved in $3^d \cdot n^{\mathcal{O}(1)}$ time, where d denotes the maximum degree of the input graph.*

Proof. As the degree of a vertex is the size n of the corresponding set A of the respective Waiting Time Minimization at Vertex-instance at that vertex, it suffices to show that Waiting Time Minimization at Vertex admits an $3^n \cdot |I|^{\mathcal{O}(1)}$ time algorithm.

We again use a dynamic program. This time, the table D is indexed by an integer i ranging from 0 to Δ and a subset $S \subseteq A$. Conceptually, this corresponds to a table of size $(\Delta + 1) \cdot 2^{|A|}$, where each subset S serves as an index of the second dimension. Every entry stores the smallest value $\sum_{\ell=1}^{i}(\sum B_\ell)^2$ of any i-partition of $(B_1, \ldots, B_i)$ of the multiset S.

For $i = 0$, we set $D[i, S] = 0$ if $S = \emptyset$, and $D[i, S] = \infty$ otherwise. For $i > 0$, we set

$$D[i, S] := \min_{B_i \subseteq S} D[i-1, S \setminus B_i] + \left(\sum B_i\right)^2.$$

That is, we search for the best subset B_i of S to put as ith part and divide the remaining objects in $S \setminus B_i$ into $i - 1$ parts.

The optimal value for Waiting Time Minimization at Vertex can then be found by computing $\frac{(\Delta-2)}{2}N^2 - (\Delta - 1) \cdot \sum_{a\in A}(a)^2 + \frac{\Delta}{2}D^*$, where $D^* := D[\Delta, A]$. Note that each entry $D[i, S]$ can be computed in $2^{|S|} \cdot n^{\mathcal{O}(1)}$ time and that there are $(\Delta + 1) \cdot \binom{n}{s}$ entries for subsets S of size exactly s. Hence, the total running time to compute all entries of D and also to compute the value D^* is $3^n \cdot n^{\mathcal{O}(1)}$ time. Note that a corresponding optimal Δ-partition can be computed in the same asymptotic running time via trace-back. ◀

3.2 An Approximation Algorithm

In the remainder, we lift an EPTAS for Sum of Square Partition to Temporal Tree Waiting Time Minimization. This is not as straight-forward as the previous results, as the objective function of Waiting Time Minimization at Vertex (and Temporal Tree Waiting Time Minimization respectively) contains a possibly negative part as second summand in contrast Sum of Square Partition which only sums up over positive values.

To deal with the case, we exploit the observation that each element of value larger than the average bin load will be guaranteed to receive a bin of its own. (This observation is well-known in the context of scheduling of parallel machines [2].)

▶ **Observation 3.7** ([2]). *Let A be a multiset where N is the sum of all numbers in A, and let k be a positive integer. If $\max A > \frac{N}{k}$, then in each k-partition $(B_1, \ldots, B_k)$ of A that minimizes $\sum_{\ell=1}^{k}(\sum B_\ell)^2$, there is some $\ell \in [1, k]$ with $B_i := \{\max A\}$.*

We now present our approximation algorithm.

▶ **Theorem 3.8.** *Temporal Tree Waiting Time Minimization admits an EPTAS.*

Proof. Let $I := (A, \Delta)$ be an instance of Waiting Time Minimization at Vertex, let $\epsilon > 0$ be a constant, and let N denote the sum of all numbers in A. We describe how to obtain an algorithm that outputs a $(1+\epsilon)$-approximation for I in $f(\frac{1}{\epsilon}) \cdot N + N^{\mathcal{O}(1)}$ time. To this end, assume that the elements of A are ordered non-increasingly.

We first use Observation 3.7 to exclude the largest elements that we know for sure will go into their own bins. The index of the first remaining element will be called p. We let p never exceed $\max(1, \Delta - 2)$ and require that for each $1 \leq i < p$ the element a_i has value at least $\frac{1}{4} \cdot \sum_{\ell=i}^{n} a_i$, which is larger than $\frac{1}{\Delta - i + 1} \cdot \sum_{\ell=i}^{n} a_i$ as required by Observation 3.7, since $1 \leq i < p$ and $p \leq \max(1, \Delta - 2)$ imply that $i \leq \Delta - 3$. This allows us to work with the fixed coefficient of $\frac{1}{4}$ in the remaining calculations. Formally, we define

$$p := \max\left(\{1\} \cup \{j \mid 1 \leq j \leq \Delta - 2, \forall 1 \leq i < j \colon a_i \geq \frac{1}{4} \cdot \sum_{\ell=i}^{|A|} a_\ell\}\right).$$

Moreover, let $\Delta' := \Delta - p + 1$ and let $\mathcal{B}_{<p} := \{\{a_i\} \mid 1 \leq i < p\}$. Due to Observation 3.7, for each optimal solution $\mathcal{B}^*$ for I, $\mathcal{B}_{<p} \subseteq \mathcal{B}^*$. We consider two cases.

If $\Delta' \leq 3$, we compute an optimal solution $\mathcal{B}'$ for the instance $(\{a_i \mid p \leq i \leq |A|\}, \Delta')$ in $\mathcal{O}(N^{\Delta'}) \subseteq N^{\mathcal{O}(1)}$ time due to the algorithm behind Corollary 3.5. Then, $\mathcal{B} := \mathcal{B}' \cup \mathcal{B}_{<p}$ is an optimal solution for I, since the optimal solution for I are exactly the optimal solutions for the instance (A, Δ) of Sum of Square Partition. This clearly is a 1-approximation for I that runs in $\mathcal{O}(N^3)$ time.

Otherwise, if $\Delta' \geq 4$, then compute a $(1+\epsilon)$-approximation $\mathcal{B}$ for the instance $I' := (\{a_i \mid p \leq i \leq |A|\}, \Delta')$ of Sum of Square Partition. This can be done in $f(\frac{1}{\epsilon}) \cdot N$ time [2]. Let $\mathcal{B}^*$ be an optimal solution for I'. Recall that $\mathcal{B}_{<p} \cup \mathcal{B}^*$ is an optimal solution for I. We show that $\mathcal{B}_{<p} \cup \mathcal{B}$ is a $(1+\epsilon)$-approximation for I. That is, we show that

$$\frac{\frac{(\Delta-2)}{2}N^2 - (\Delta-1) \cdot \sum_{i=1}^{|A|}(a_i)^2 + \frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2 + \frac{\Delta}{2}\sum_{B \in \mathcal{B}}(\sum B)^2}{\frac{(\Delta-2)}{2}N^2 - (\Delta-1) \cdot \sum_{i=1}^{|A|}(a_i)^2 + \frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2 + \frac{\Delta}{2}\sum_{B \in \mathcal{B}^*}(\sum B)^2} \leq 1 + \epsilon.$$

Since $\mathcal{B}$ is a $(1+\epsilon)$-approximation for the instance I' of Sum of Square Partition, $\sum_{B \in \mathcal{B}}(\sum B)^2 \leq (1+\epsilon) \cdot \sum_{B \in \mathcal{B}^*}(\sum B)^2$. This implies that

$$\frac{\frac{(\Delta-2)}{2}N^2-(\Delta-1)\cdot\sum_{i=1}^{|A|}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2+\frac{\Delta}{2}\sum_{B\in\mathcal{B}}(\sum B)^2}{\frac{(\Delta-2)}{2}N^2-(\Delta-1)\cdot\sum_{i=1}^{|A|}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2+\frac{\Delta}{2}\sum_{B\in\mathcal{B}^*}(\sum B)^2} \tag{1}$$

$$\leq \frac{\frac{(\Delta-2)}{2}N^2-(\Delta-1)\cdot\sum_{i=1}^{|A|}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2+(1+\epsilon)\cdot\frac{\Delta}{2}\sum_{B\in\mathcal{B}^*}(\sum B)^2}{\frac{(\Delta-2)}{2}N^2-(\Delta-1)\cdot\sum_{i=1}^{|A|}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2+\frac{\Delta}{2}\sum_{B\in\mathcal{B}^*}(\sum B)^2} \tag{2}$$

$$\leq 1+\epsilon\cdot\frac{\frac{\Delta}{2}\sum_{B\in\mathcal{B}^*}(\sum B)^2}{\frac{(\Delta-2)}{2}N^2-(\Delta-1)\cdot\sum_{i=1}^{|A|}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2+\frac{\Delta}{2}\sum_{B\in\mathcal{B}^*}(\sum B)^2}. \tag{3}$$

To show that the latter part is upper-bounded by ϵ, it suffices to show that $\frac{(\Delta-2)}{2}N^2-(\Delta-1)\cdot\sum_{i=1}^{|A|}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2\geq 0$. To this end, let $N_{<p}:=\sum_{i=1}^{p-1}a_i$ and let $N':=N-N_{<p}$. With this, we get

$$\frac{\Delta-2}{2}N^2-(\Delta-1)\cdot\sum_{i=1}^{|A|}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2 \tag{4}$$

$$=\frac{\Delta-2}{2}N^2-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2-(\Delta-1)\cdot\sum_{i=1}^{p-1}(a_i)^2+\frac{\Delta}{2}\sum_{i=1}^{p-1}(a_i)^2 \tag{5}$$

$$=\frac{\Delta-2}{2}N^2-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2-\frac{\Delta-2}{2}\sum_{i=1}^{p-1}(a_i)^2 \tag{6}$$

$$=\frac{\Delta-2}{2}(N_{<p}+N')^2-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2-\frac{\Delta-2}{2}\sum_{i=1}^{p-1}(a_i)^2 \tag{7}$$

$$\geq\frac{\Delta-2}{2}((N_{<p})^2+(N')^2)-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2-\frac{\Delta-2}{2}\sum_{i=1}^{p-1}(a_i)^2 \tag{8}$$

$$=\frac{\Delta-2}{2}(N')^2-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2+\frac{\Delta-2}{2}(N_{<p})^2-\frac{\Delta-2}{2}\sum_{i=1}^{p-1}(a_i)^2 \tag{9}$$

$$=\frac{\Delta-2}{2}(N')^2-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2+\frac{\Delta-2}{2}((\sum_{i=1}^{p-1}a_i)^2-\sum_{i=1}^{p-1}(a_i)^2) \tag{10}$$

$$\geq\frac{\Delta-2}{2}(N')^2-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2. \tag{11}$$

Since $N':=\sum_{i=p}^{|A|}a_i$ and the fact that $\Delta'\geq 4$, $a_p<\frac{1}{4}\sum_{i=p}^{|A|}a_i=\frac{1}{4}N'$. This implies that $(N')^2\geq 4\cdot\sum_{i=p}^{|A|}(a_i)^2$. To see this, consider a square S with volume $(N')^2$ where the subsquares with volume $(a_p)^2,(a_{p+1})^2,\ldots,(a_{|A|})^2$ are aligned next to each other on the bottom line of S. Each such square has height at most $a_p\leq\frac{1}{4}N'$ which implies that the sum of volume of all these subsquares does not exceed $\frac{1}{4}(N')^2$.

Thus, $\frac{\Delta-2}{2}(N')^2\geq 4\cdot\frac{\Delta-2}{2}\cdot\sum_{i=p}^{|A|}(a_i)^2=2(\Delta-2)\cdot\sum_{i=p}^{|A|}(a_i)^2\geq(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2$ since $\Delta\geq\Delta'\geq 4$. Consequently, $\frac{\Delta-2}{2}(N')^2-(\Delta-1)\cdot\sum_{i=p}^{|A|}(a_i)^2\geq 0$, which implies that Equation (11) can be lower-bounded by 0 and further that Equation (3) can be upper-bounded by $1+\epsilon$. That is, $\mathcal{B}_{\leq p}\cup\mathcal{B}$ is a $(1+\epsilon)$-approximation for I that can be computed in $f(\frac{1}{e})\cdot N$ time.

Combining both cases, we can obtain a $(1+\epsilon)$-approximation for Waiting Time Minimization at Vertex in $f(\frac{1}{\epsilon})\cdot N+N^{\mathcal{O}(1)}$ time.

Similarly, we obtain the $(1+\epsilon)$-approximation for Temporal Tree Waiting Time Minimization by computing a $(1+\epsilon)$-approximation for Waiting Time Minimization at Vertex for each internal vertex of the tree and combining the respective solutions to a global labeling of the tree. ◀

4 Conclusion

In this work, we introduced the Temporal Waiting Time Minimization problem and transferred results from Sum of Square Partition. This leads to results on the NP-hardness and W[1]-hardness of Temporal Tree Waiting Time Minimization and upper bounds on the running time. Furthermore, we derived that there is an EPTAS for Temporal Tree Waiting Time Minimization. To the best of our knowledge, this is the first example of an efficient approximation algorithm for a temporal graph realization problem.

As a next step, it would be interesting to analyze the complexity and structural properties of Temporal Waiting Time Minimization on directed graphs. A direct transfer of our results to bidirected trees and a connection to Sum of Square Partition, however, seems unlikely. This is due to the fact that in an undirected tree, we could decompose the waiting times (in both directions) between u and v as the sum of waiting times at each internal vertex, which averaged over both directions, only cared on whether both incident edges of that internal vertex on the path are labeled equally. This, however, does not work in a bidirected tree, as the unique u-v-path and the unique v-u-path are in fact arc-disjoint.

Generally, it would also be interesting to analyze the parameterized complexity of Temporal Waiting Time Minimization on general graphs. As discussed before, deciding whether the total waiting time of 0 is realizable for $\Delta = 3$ is NP-hard as implicitly shown by Mertzios at al. [18]. This surely rules out approximation algorithms for the case of $\Delta = 3$ but the parameterized complexity is open and possible approximation algorithms for $\Delta \neq 3$ are not ruled out.

References

1 Eleni C Akrida, Leszek Gąsieniec, George B Mertzios, and Paul G Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017. doi:10.1007/S00224-017-9757-X.

2 Noga Alon, Yossi Azar, Gerhard J Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

3 Filippo Brunelli, Pierluigi Crescenzi, and Laurent Viennot. Maximizing reachability in a temporal graph obtained by assigning starting times to a collection of walks. *Networks*, 81(2):177–203, 2023. doi:10.1002/net.22123.

4 Daniele Carnevale, Gianlorenzo D'Angelo, and Martin Olsen. Approximating optimal labelings for temporal connectivity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):26490–26497, April 2025. doi:10.1609/aaai.v39i25.34849.

5 Arnaud Casteigts, Michelle Döring, and Nils Morawietz. Realization of Temporally Connected Graphs Based on Degree Sequences. In Ho-Lin Chen, Wing-Kai Hon, and Meng-Tsung Tsai, editors, *36th International Symposium on Algorithms and Computation (ISAAC 2025)*, volume 359 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2025.17.

6 Esteban Christiann, Eric Sanlaville, and Jason Schoeters. On Inefficiently Connecting Temporal Networks. 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 8:1–8:19, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAND.2024.8.

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

8 Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021. doi:10.1016/j.jcss.2020.08.001.

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

10 Thomas Erlebach, Othon Michail, and Nils Morawietz. Recognizing and Realizing Temporal Reachability Graphs. In Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman, editors, *33rd Annual European Symposium on Algorithms (ESA 2025)*, volume 351 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 93:1–93:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2025.93.

11 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.

12 F. Göbel, Jorge Orestes Cerdeira, and Henk Jan Veldman. Label-connected graphs and the gossip problem. *Discrete Mathematics*, 87(1):29–40, 1991. doi:10.1016/0012-365X(91)90068-D.

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

14 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013. doi:10.1016/J.JCSS.2012.04.004.

15 Nina Klobas, George B Mertzios, Hendrik Molter, and Paul G Spirakis. The complexity of computing optimum labelings for temporal connectivity. *Journal of Computer and System Sciences*, 146:103564, 2024. doi:10.1016/J.JCSS.2024.103564.

16 Nina Klobas, George B. Mertzios, Hendrik Molter, and Paul G. Spirakis. Temporal Graph Realization from Fastest Paths. 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.

17 George B. Mertzios, Othon Michail, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, April 2019. doi:10.1007/s00453-018-0478-6.

18 George B. Mertzios, Hendrik Molter, Nils Morawietz, and Paul G. Spirakis. Temporal Graph Realization with Bounded Stretch. In Paweł Gawrychowski, Filip Mazowiecki, and Michał Skrzypczak, editors, *50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025)*, volume 345 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2025.75.

19 Geroge B. Mertzios, Hendrik Molter, Nils Morawietz, and Paul G. Spirakis. Realizing temporal transportation trees. In Henning Fernau and Philipp Kindermann, editors, *Graph-Theoretic Concepts in Computer Science*, pages 390–404, Cham, 2026. Springer Nature Switzerland. doi:10.1007/978-3-032-11835-6_28.

20 Julia Meusel, Nils Morawietz, Matthias Müller-Hannemann, and Klaus Reinhardt. Brief Announcement: Revisiting the Realizability of Periodic Temporal Graphs with Bounded Stretch. In George B. Mertzios and Andréa W. Richa, editors, *5th Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2026)*, volume 373 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:6, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAND.2026.21.

21 Julia Meusel, Matthias Müller-Hannemann, and Klaus Reinhardt. Directed Temporal Tree Realization for Periodic Public Transport: Easy and Hard Cases. In Jonas Sauer and Marie Schmidt, editors, *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2025)*, volume 137 of *Open Access Series in Informatics (OASIcs)*, pages 3:1–3:22, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2025.3.