Better Late, Then? The Hardness of Choosing Delays to Meet Passenger Demands in Temporal Graphs

David C. Kutner[1] ✉ ⌂ iD
Department of Computer Science, Durham University, UK

Anouk Sommer ✉ iD
Karlsruher Institut für Technologie (KIT), Germany

─── **Abstract** ───

In train networks, carefully-chosen delays may be beneficial for certain passengers, who would otherwise miss some connection. Given a simple (directed or undirected) temporal graph and a set of passengers (each specifying a starting vertex, an ending vertex, and a desired arrival time), we ask whether it is possible to delay some of the edges of the temporal graph to realize all the passengers' demands. We call this problem DELAYBETTER (DB), and study it along with two variants: in δ-DELAYBETTER, each delay must be of at most δ; in (δ-)PATH DB, passengers also fully specify the vertices they should visit on their journey. On the positive side, we give a polynomial-time algorithm for PATH DB and δ-PATH DB, and obtain as a corollary a polynomial-time algorithm for DB and δ-DB on trees. We also provide an fpt algorithm for both problems parameterized by the size of the graph's Feedback Edge Set together with the number of passengers. On the negative side, we show NP-completeness of (1-)DB on bounded-degree temporal graphs even when the lifetime is 2, and of (10-)DB on bounded-degree planar temporal graphs of lifetime 19. Our results complement previous work studying reachability problems in temporal graphs with delaying operations. This is to our knowledge the first such problem in which the aim is to facilitate travel between specific points (as opposed to facilitating or impeding a broadcast from one or many sources).

1 Introduction

In the first half of 2024, punctuality of Deutsche Bahn's long distance trains was 62.7% [8]. Disruptions to train networks often result in passengers arriving later than planned or not at all. Whenever a train is late and the passengers of this train would miss a connecting

---

[1] Corresponding author.
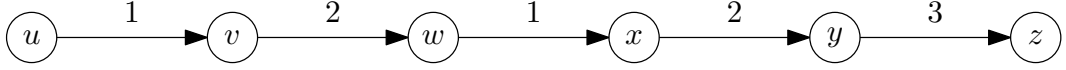
■ **Figure 1** A temporal graph on 6 vertices. Consider the case where passengers at each of u, v, and w wish to travel to each of x, y, and z respectively, arriving at or before time 4. Then delaying the edge from w to x by at least 2 is necessary for the two leftmost passengers to arrive on time, but entails that the passenger starting at w cannot arrive at z before time 5.

train, there are two choices: either, the second train departs on time, meaning that the passengers of the first train miss their connection, or the second train waits, meaning that the passengers can make the connection, at the cost of this train now also being late. The problem of deciding whether (and by how much) such services should wait is the Delay Management problem, well studied in Operations Research.

Separately, the field of temporal graph theory provides a general, rigorous mathematical framework with which to investigate the complexity due to the intrinsically dynamic properties of certain real-world networks. Briefly, a temporal graph is one whose edge set changes over time. Much work has been devoted to *modification* problems of the form "Given a temporal graph $\mathcal{G}$, apply some (delaying or other) operations to satisfy some reachability property" (see Table 1), but interestingly the problem of managing delays to ensure that specific passengers arrive at their destination on time has yet to be studied in this framework. Figure 1 shows a simple example of a temporal graph illustrating such a scenario.

The present work aims to study the practically interesting problem of Delay Management through the lens of temporal graph theory. We introduce the decision problem DELAYBETTER (or simply DB) which asks, given a temporal graph and a collection of passengers on its vertices, each with a desired destination and arrival time, whether it is feasible to delay some edges of the graph to satisfy each of the passengers. We also consider variants of the problem: PATH-DB, where passengers must be routed along specific edges prescribed in the input, δ-DB, where each edge can be delayed by at most some fixed δ, and δ-PATH-DB combines both constraints. We present (parameterized) tractability and hardness results for these problems, including on structurally restricted graph classes.

1.1   Problem setting

We denote $[i, j]$ the integer interval $\{i, i+1, \ldots, j\}$, and say a graph is *cubic* if all vertices in the graph have degree 3. We now give some definitions from temporal graph theory.

▶ **Definition 1** (Temporal graph, temporal path). *A temporal graph $\mathcal{G} = (G, \lambda)$ consists of a static graph G (also called its* footprint*, denoted $\mathcal{G}_\downarrow$) and a temporal assignment $\lambda : E(G) \to \mathbb{N}$. The* lifetime *$\tau \in \mathbb{N}$ of a temporal graph (G, λ) is the maximum time assigned to any edge by λ, and the* time-edges *of a temporal graph $\mathcal{E}(\mathcal{G})$ are $\{(u, v, \lambda(u, v))|(u, v) \in E(G)\}$. A temporal path is a path in $\mathcal{G}_\downarrow$ whose edges have strictly increasing time-labels, and the* arrival time *of a temporal path is the time-label of its last edge.*

We take this opportunity to note a more general definition of temporal graphs is often studied, where each edge may have multiple time labels (*non-simple* temporal graphs). Another variant applies a notion of temporal reachability which allows for the traversal of consecutive edges at nondecreasing (rather than strictly increasing) times (*non-strict* temporal paths). For a discussion of these different models (in the undirected setting only) we refer the interested reader to [5]. Hardness results from the simple setting generalize to

■ **Table 1** Comparison of our problems DelayBetter and Path DelayBetter/Path DB to problems in the literature. R_u^t (resp. R_S^t) denotes the set of vertices reachable from vertex u (resp. any vertex $s \in S$) by time-step t (when t is the lifetime of the temporal graph, it is omitted).

Problem	Operation	Restriction	Reachability condition	Additional inputs
ReachFast [6]	Shift $(+-)$	N/A	$\forall x \in S : R_x = V$	sources $S \subseteq V$, $\tau \in \mathbb{N}$ to be minimized
TRLP [10]	Shift $(+-)$	up to η edges, by up to δ each	$|R_x| \geq k$	designated source $x \in V, \eta, \delta, k \in \mathbb{N}$
MinReachDelay [25]	Delay	up to η time-edges by exactly δ each	$|R_S| \leq k$	sources $S \subseteq V$, $\eta, \delta, k \in \mathbb{N}$
MinReach [7]	Delay	up to η time-edges by up to δ each	$|R_S| \leq k$	sources $S \subseteq V$, $\eta, \delta, k \in \mathbb{N}$
MaxMinTaRDiS [21]	Choose time-labels	lifetime is τ	$\nexists S \subseteq V, |S| < k :$ $R_S = V$	$\tau, k \in \mathbb{N}$
$(\delta\text{-})$DelayBetter	Delay	(by up to δ per edge)	$(u,v,t) \in D$ $\Rightarrow v \in R_u^t$	$D \subseteq V \times V \times \mathbb{N}$ $(\delta \in \mathbb{N})$
$(\delta\text{-})$Path DB	Delay	(by up to δ per edge)	as above along specified path	$D \subseteq V \times V \times \mathbb{N} \times 2^E$ $(\delta \in \mathbb{N})$

the non-simple setting, and tractability for the non-simple setting may be applied to the simple setting. In the present work, we focus exclusively on strict temporal paths and simple (directed or undirected) temporal graphs.

▶ **Definition 2** (Delaying). *We say that a temporal assignment λ' is a* delaying *of an assignment λ if $\lambda'(e) \geq \lambda(e)$ for every e. If $\lambda'(e) = \lambda(e) + \delta$, we say that e is delayed by δ in λ', and that λ' is a δ-delaying of λ if every e is delayed by at most δ in λ' (hence a δ-delaying is also a $(\delta + 1)$-delaying).*

We can now introduce our protagonists:

$(\delta\text{-})$DelayBetter
Instance: Temporal graph $\mathcal{G} = (G, \lambda)$, demands $D \subseteq V(G) \times V(G) \times \mathbb{N}$ $(, \delta \in \mathbb{N})$.
Question: Does there exist a $(\delta\text{-})$delaying λ' of λ such that for each $(u, v, t) \in D$ there is a temporal path from u to v with arrival time at most t in (G, λ')?

$(\delta\text{-})$Path DelayBetter
Instance: Temporal graph $\mathcal{G} = (G, \lambda)$, demands $D \subseteq V(G) \times V(G) \times \mathbb{N} \times 2^{E(G)}$ $(, \delta \in \mathbb{N})$.
Question: Does there exist a $(\delta\text{-})$delaying λ' of λ such that for each $(u, v, t, P) \in D$ there is a temporal path from u to v in (G, λ') with arrival time at most t and footprint P?

We say a temporal graph $\mathcal{G}$ is *planar* if its footprint $\mathcal{G}_\downarrow$ is planar, and *directed* (resp. *undirected*) if $\mathcal{G}_\downarrow$ is directed (resp. undirected). We use the shorthand DB for our problems, referring to, for example, 3-DB, Path DB, or DB. For a demand d, we denote $d = (d_s, d_z, d_t)$.

Restriction of, or parameterization by, the lifetime τ is often leveraged to obtain tractability of temporal graph problems. In our case, we denote by T_{init} the *initial lifetime* (that of the temporal graph $\mathcal{G}$ before delays are applied), and by $T_{\max}$ the latest arrival time required by any single demand – that is, $\max_{d \in D} d_t$. We call $T_{\max}$ *final lifetime* because it upper-bounds the lifetime of the temporal graph (G, λ') (after delays are applied): any time-edge delayed beyond $T_{\max}$ in some feasible solution could instead be delayed to $T_{\max}$ instead (or not at all), since it will not be used by any passengers. For the same reason, we may assume without loss that T_{init} is at most $T_{\max}$.

1.2   Related work

Temporal Graphs

As we touched on earlier, modifying (or choosing) λ to optimize a notion of reachability is a well-studied problem in temporal graph theory. Broadly, problems in this paradigm may either aim to *worsen* or *improve* the input temporal graph's connectedness. Problems in the first category (including MINREACH [7] and MINREACHDELAY [25]) are typically motivated by practical cases where spread is undesired, such as epidemics. In the case of transportation networks, where connectedness is desired, the second category (which contains TRLP [10] and MAXMINTARDIS [21]) is of greater relevance. Of course, if the delays are controlled by an adversary, the opposite motivation becomes relevant to each problem: is there any strategy for the adversary to disconnect a transporation network, or facilitate disease spread? A related, but slightly different perspective on delays in temporal graphs is explored in [13] and [12], who determine how robust against unforeseen delays a given temporal graph is with and without re-routing of the passengers, respectively.

Delay Management

The Delay Management (DM) problem concerns itself with finding a good delaying strategy in a public transport network to minimize passenger inconvenience. Usually, this means minimizing the total passenger delay, but other objectives like simultaneously minimizing the number of delayed trains or the operational costs have also been studied. In the original problem, as introduced by [28], passengers stick with their initial routes (as in PATH-DB); a popular variant of the problem allows passenger re-routing (as in DB) [9]. Both settings have since been the subject of much study, spanning both theory and practice.

On the theoretical side, different models and algorithmic approaches have been introduced over the years [1, 16, 17, 31, 33]. Due to modeling differences, studies of the computational complexity of different DM problem variants [14, 15, 27] do not necessarily yield results for our problems. In addition to minimizing an aggregate function (e.g., total weighted passenger delay [14, 15]) rather than asking whether some specific set of passenger demands can be satisfied (as we do), DM problems are commonly formalized using *event-activity networks* – which are more expressive than temporal graphs. For example, the definition of DM in [27] includes *headway constraints* (where two trains cannot use the same track segment simultaneously). Several interesting practically-motivated extensions are studied in this line of work, including a setting with slack times (trains may catch up on their delay), which makes the problem hard when the rail network is a line [15], and the incorporation of rolling-stock circulation into the problem [27] – though results in such settings do not straightforwardly translate into our model. Nonetheless, some results from these works can be adapted into the our setting; for example, Theorem 6.1 in [14] could be adapted to show

that DELAYBETTER is NP-complete in the directed setting with $T_{\max} = 3$ (we strengthen this in Theorem 15). Also, all of these works consider a directed model (as is natural for rail networks), whereas our results are proven for both directed and undirected temporal graphs.

On the more practical side, there have been a number of case-studies and data-driven approaches to this problem [4, 23, 32]. In [30], a model for optimizing delays in rail and air travel combined is proposed, together with a European case study. For a more comprehensive overview of the work in delay management, we refer the reader to [3], [22], and [29]. A related area of research is the Timetabling Problem, which concerns itself with designing a timetable that is robust against delays. We refer the reader to [20] for an introduction.

1.3   Our contribution

We introduce the problems $(\delta\text{-})$DELAYBETTER and $(\delta\text{-})$PATH DB, presenting (to our knowledge for the first time) a temporal graph-theoretic approach to the well-studied Delay Management problem. On the positive side, we give a polynomial-time algorithm for $(\delta\text{-})$PATH-DB, and tractability for $(\delta\text{-})$DELAYBETTER on trees as a corollary. Later, we leverage this algorithm to obtain a fixed-parameter tractable (fpt) algorithm parameterized by the number of demands and the size of the feedback edge set of the footprint graph. On the negative side, we establish that DELAYBETTER remains NP-complete on inputs with $T_{\max} = 2$ in both the directed and undirected setting (which entails that 1-DELAYBETTER is NP-complete under the same constraint). Moreover, we show that the problem remains hard on planar (directed or undirected) temporal graphs with $T_{\max} = 19$, even when $\delta = 10$. Our results provide a first insight into the structural restrictions which do (and do not) suffice to guarantee tractability of this natural problem. Proofs of statements marked $(*)$ can be found in the full version of the paper.

1.4   Discussion and open questions

We show that $(\delta\text{-})$PATH DB is in P and that $(\delta\text{-})$DELAYBETTER is fpt parameterized by $|D| + \rho$, where ρ is the size of smallest feedback edge set of (the undirected version of) $\mathcal{G}_\downarrow$. It seems likely that the techniques used in those proofs could actually solve a broader family of problems – including, for example, the natural extension of DELAYBETTER wherein demands specify a departure time as well as an arrival time, but also possibly problems which do not specify individual demands as part of the input. Can dependence on $|D|$ be eliminated from our fpt result? If not, then what structural parameter is sufficient to yield an fpt result without requiring $|D|$ as a parameter? A more general question for future study is: what family of temporal graph modification problems admit an fpt algorithm in the size of the feedback edge set? Separately, what is complexity of the problems parameterized by fine-grained temporal parameters (e.g., vertex interval membership width [2]), or by smaller structural parameters than ρ (e.g., the feedback vertex number)? We note that for directed graphs, the size of a minimum feedback arc set (the deletion of which leaves a directed acyclic graph, or DAG) is insufficient, since we show in Theorem 17 that the problem is NP-complete restricted to (planar) DAGs.

Another question we leave open is: what is the complexity of DELAYBETTER restricted to planar inputs with $T_{\max} \in [2, 18]$? (Our proofs of Theorems 14 and 15 do not preserve planarity, and moreover reduce from a variant of NAE 3SAT, the restriction of which to planar instances is solvable in polynomial time [26].) Also stemming from our planar proof is the question of whether δ-DELAYBETTER restricted to planar graphs is computationally easy or hard for values of δ below 10. Our proof was aimed at minimizing $T_{\max}$ while retaining

planarity, so we expect that some easy adjustments to it might yield hardness for, e.g., $\delta = 9$, but we expect different techniques are necessary to deal with the case of $\delta = 1$ on planar graphs.

Yet another direction our investigation could be extended is to consider *non-simple* temporal graphs. Our hardness results extend immediately to this case, but our algorithms do not – in the non-simple setting, we do not expect our linear programming approach to work, and it is not even obvious whether our problems would be tractable restricted to trees.

Lastly, we observe that our results for directed and undirected versions of the problem are the same. This is particularly surprising because some of our results require substantially different proofs for each setting. An open question for future work is then: are there any natural restrictions on the input which entail instances are tractable in the directed case and computationally hard in the undirected case (or vice versa)?

2   Preliminary Results

We begin with some basic results, the proofs of which may help to familiarize the reader with the behavior of our problems. We first establish a useful relation between δ-DB and DELAYBETTER. Clearly, DELAYBETTER is reducible to δ-DELAYBETTER, by simply assigning a sufficiently large value to δ (e.g., the final lifetime $T_{\max}$ of the DELAYBETTER instance). Interestingly, the converse also holds:

▶ **Lemma 3.** *For any $\delta \in \mathbb{N}$, δ-DELAYBETTER is reducible in linear time to DELAYBETTER. If the input instance is planar (resp. has bounded final lifetime) then the same holds for the output.*

Proof. We require different reductions for directed and undirected graphs. In both cases, substitute a gadget in place of each edge in the original instance, and increase the lifetime of the instance. Both constructions preserve planarity, and the DELAYBETTER-instance has final lifetime at most $2T_{\max} + 2\delta + 1$ (directed) or $T_{\max} + \delta + 2$ (undirected), where $T_{\max}$ is the final lifetime of the δ-DELAYBETTER instance.

We first deal with the undirected case. We begin with a δ-DELAYBETTER instance $((G, \lambda), D, \delta)$ having the property that every time is at time 3 or later (if necessary, this can be achieved by uniformly incrementing all times in the demands and in the temporal assignment by 2). We then create, for every time-edge (u, v, t), a gadget on $3\delta + 3$ new vertices $\{uv_t, \ldots, uv_{t+\delta}, u_1, \ldots, u_\delta, v_1, \ldots, v_\delta, u', v'\}$ and $\delta + 1$ new demands $\{(uv_i, v', i+1) : i \in [t, t+\delta]\}$, as shown in Figure 2.

Now each of the $\delta + 1$ demands into v' must be routed through a different path. Because there are $\delta + 1$ possible paths in total, some demand (uv_i, v', t) must be routed through the edge (u, v) – and this entails that $\lambda'(u, v) = i$, yielding the desired result since $i \in [t, t + \delta]$ by construction.

We now give the proof for the directed case. Given an instance $(\mathcal{G} = (G, \lambda), D, \delta)$ of δ-DELAYBETTER, we produce an instance $(\mathcal{G}' = (G', \lambda'), D')$ of DELAYBETTER as follows. (For this proof, we use λ' to refer to the initial assignment of the new instance, not the delaying of λ.) We first include $\{(u, v, 2t) | (u, v, t) \in D\}$ as demands, which we call *travelers*. Next, we replace every time-edge (u, v) at time t with the gadget pictured in Figure 3, and add the demand $(u', v', 2t + 2\delta + 1)$. We call demands introduced in this step *hermits*, the edge (u', u) that hermit's *trailhead*, and the edge (u, uv) (resp. (uv, v)) a *first-half* (resp. *second-half*) edge. This concludes the construction.
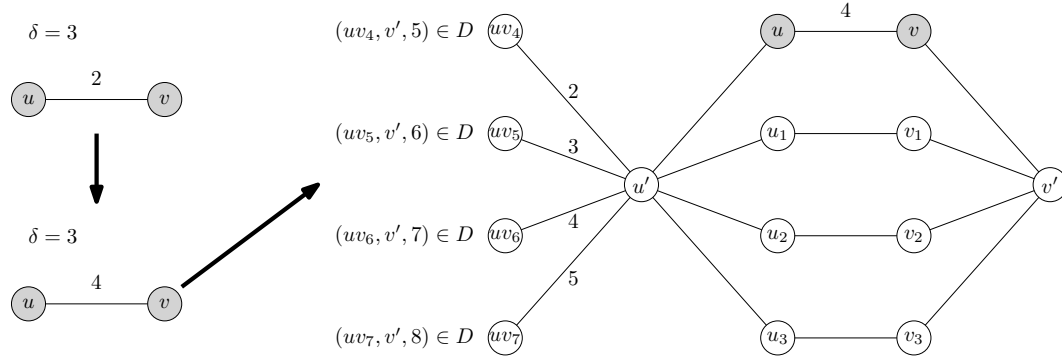
Figure 2 A sketch of our reduction for undirected temporal graphs for a time-edge $(u, v, 2)$ in an instance with $\delta = 3$. For readability, edges assigned time 1 in the output instance are unlabeled.
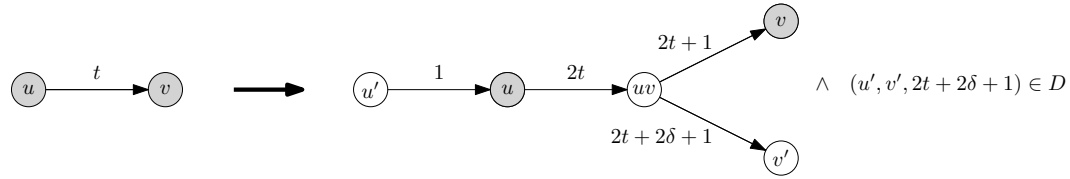
Figure 3 A sketch of our reduction for directed temporal graphs.

Clearly, if the δ-DB instance reduced from was a yes-instance, then the DELAYBETTER instance obtained is also a yes-instance: whenever some edge (u, v) is delayed by some amount x in the original instance, delay both (u, uv) and (uv, v) by $2x$. It remains to show the converse.

Let λ^* be a solution to our modified problem with a pareto-optimal time-assignment of the edges (that is, one such that there is no other solution whose time-labels are all strictly smaller or equal to those under λ^*).

$\triangleright$ **Claim 4** ($*$). Hermits leave early: if e is a hermit's trailhead, then $\lambda^*(e) = 1$.

$\triangleright$ **Claim 5**. Under λ^*, every first-half edge (resp. second-half edge) is assigned an even (resp. odd) time.

Proof. Suppose otherwise. We must deal with two cases:

We deal first with the case where the earliest edge violating this claim is a first-half at an odd time. Let (u, uv) be the earliest first-half edge assigned an odd time (say, t') under λ^*. By pareto-optimality of λ^* is must be that assigning time $t' - 1$ to the edge (u, uv) would stop some demand from being satisfied. This demand cannot be a hermit because of Claim 4 – so there must be some traveler arriving at u at time $t' - 1$, a contradiction since our premise for this case was that (u, uv) was the earliest edge violating the claim.

Suppose instead that the earliest offending edge is a second-half edge (wu, u) assigned an even time (say, $t' - 1$). We again quickly find that this can only be due to the first-half edge (w, wu) being used by a traveler at time $t' - 2$ - again reaching a contradiction, since this is a strictly earlier odd time assigned to a first-half edge. $\triangleleft$

$\triangleright$ **Claim 6** ($*$). For each time-edge (u, v, t) in the original instance, $\lambda^*(u, uv) \in [2t, 2t + 2\delta]$.

Claim 5 allows us to recover a time-labeling for our initial δ-DELAYBETTER instance by assigning to each the edge (u, v) the time $\frac{(u,uv)}{2}$ while preserving the temporal paths of travelers. Claim 6 entails that this time-labeling does not delay any edge by more than δ, and the result follows. $\blacktriangleleft$

▶ **Lemma 7** (∗). *An instance of* DELAYBETTER *or* PATH DB *(resp. δ-*DELAYBETTER *or δ-*PATH DB*) may be reduced in polynomial time to an instance of the same problem with* $T_{\max} \in \mathrm{poly}(n)$ *(resp.* $T_{\max} \in \mathrm{poly}(n + \delta)$*).*

▶ **Lemma 8** (∗). *(δ-)*DELAYBETTER *is contained in NP.*

3   Tractability Results

We begin with a small positive result which can be obtained easily from prior work.

▶ **Lemma 9.** DELAYBETTER *is solvable in polynomial time when* λ *is the constant function* **1** *and all demands in* D *have the same source.*

Proof. We use the One Source Reach Fast algorithm from [6]: They show that the time-assignment of their algorithm computes, for a given source $v \in V$ and every remaining vertex $u \in V$, the individual minimum time that v needs to reach u. If this computed minimum time is at most our demanded arrival time for all demands $(v, u, t) \in D$, then we have a YES-instance, otherwise we have a NO-instance.   ◀

We now turn to the case where passenger demands fully prescribe the path they must be routed along, establishing tractability through a linear programming argument.

▶ **Theorem 10.** PATH DELAYBETTER *and δ-*PATH DELAYBETTER *are both in P.*

Proof. Let $((G, \lambda), D)$ be an instance of PATH DELAYBETTER (or, let $((G, \lambda), D, \delta)$ be an instance of δ-PATH-DELAYBETTER – the proof differs only in a few details).

We begin by introducing some notation. Our proof is for directed and undirected inputs – we shall use uv to mean the edge (u, v), but in the undirected case $uv = vu$ whereas in the directed case these are $uv \neq vu$. For a demand $d \in D$, we denote by d_P the specified static path in G from d_s to d_z, and d_f the final edge of d_P, which is incident to d_z and must be at time d_t or earlier to satisfy the demand. We also use t_{uv} as shorthand for $\lambda(u, v)$, and t'_{uv} for $\lambda'(u, v)$. Lastly, we define the relation $(u, v) \prec (v, w)$, to be true if and only if (u, v) immediately precedes (v, w) in the path d_P for some $d \in D$.

Consider the following linear program:

$$\text{maximize} \sum_{d \in D} d_t - t'_{d_f} \text{ , subject to} \tag{1}$$

$$t_{uv} \leq t'_{uv} \text{ for each } (u, v) \in E(G) \tag{2}$$

$$t'_{uv} \leq t_{uv} + \delta \text{ for each } (u, v) \in E(G) \text{ (only for δ-PATH-DB)} \tag{3}$$

$$t'_{uv} \leq t'_{vw} + 1 \text{ for each pair of edges } uv \text{ and } vw \text{ such that } uv \prec vw \tag{4}$$

$$t'_{d_f} \leq d_t \text{ for each demand } d \tag{5}$$

This LP has $\{t'_{uv} : (u, v) \in E(G)\}$ as its set of *unknown variables*. The variables $\{t_{uv} : (u, v) \in E(G)\} \cup \{d_t : d \in D\}$ correspond to given integers fully specified by the PATH-DB instance $((G, \lambda), D)$ (and d_f likewise refers to a specific edge of G).

▷ **Claim 11** (∗).   The LP is integral. Meaning: at least one optimal solution of the LP assigns integers to all of its unknown variables. Moreover, an integral solution may be recovered from a non-integral solution in polynomial time.

Since linear programs are solvable in polynomial time [19], we may first solve the LP and then (if the solution is not already integral) apply Claim 11 to recover an integral solution. We note here that a modification of Kahn's algorithm [18] for topological sorting may be used to compute a solution to this particular LP directly and more efficiently. A detailed proof would be quite technical and incongruous with the rest of the paper, so has been omitted.

An integral solution to this LP fully specifies a delaying λ' satisfying the (δ-)PATH DB instance. Note that: λ' is indeed a (δ-)delaying of λ (due to Equations (2) and (3)); enables strict temporal paths along each path specified in D (due to Equation (4)); and that each of these paths reaches the destination vertex by the arrival time prescribed (due to Equation (5)). Conversely, it should be clear that any delaying λ' satisfying the (δ-)PATH DB instance specifies a (not necessarily optimal) solution to the LP. In fact, the LP allows us not only to *decide* (δ-)PATH DB, but more strongly to solve its optimization variant.   ◄

Since trees are characterized by any pair (u, v) being connected by a unique (static) path, we obtain the following corollary:

▶ **Corollary 12.** *(δ-)DELAYBETTER is in P when the underlying graph $\mathcal{G}_\downarrow$ is a (directed) tree.*

Next, we are able to extend this result to "tree-like" graphs, by parameterizing by the size of the instance's feedback edge set.

▶ **Theorem 13.** *On directed (reps. undirected) temporal graphs, with $|FES(\mathcal{G}_\downarrow)| = \rho$, ($\delta$-)DELAYBETTER is solvable in time $O(\rho! \cdot 2^{\rho \cdot |D|} \cdot \mathrm{poly}(n))$ (resp. $O(\rho! \cdot 3^{\rho \cdot |D|} \cdot \mathrm{poly}(n))$).*

Proof. The proofs for directed and undirected graphs differ only in small details, and those for for δ-DB and DELAYBETTER are identical (until we apply Theorem 10).

Let E' be a feedback edge set of (the undirected version of) $\mathcal{G}_\downarrow$ of size ρ. We iterate over each of the $\rho!$ possible orderings $(e_1, e_2, ...e_\rho)$ of E', and require that $t_{e_1} \leq t_{e_2} \leq ... \leq t_{e_\rho}$. (Note that if $T_{\max}$ is small and ρ is large, we may prefer to iterate over all $(T_{\max})^\rho$ assignments and obtain an ordering from those.)

In any solution, each demand $d \in D$ is satisfied by a strict temporal path from d_u to d_v using some subset of the edges of E'. In the directed case, specifying this subset (together with the ordering fixed earlier) fully specifies the path from d_u to d_v; in the undirected case, it is also necessary to specify the direction taken for each edge. The journey from one edge in the subset to the next is uniquely determined due to the fact that it can only use the edges of the spanning tree obtained by removing E' from $\mathcal{G}$.

For directed graphs, this means there are at most 2^ρ possible paths for each demand (an edge is either chosen or not), and thus $2^{\rho \cdot |D|}$ for all demands. For undirected graphs, we get 3^ρ possible paths per demand (an edge $(u, v) \in E'$ is either traversed from u to v, from v to u, or not at all), and thus $3^{\rho \cdot |D|}$ for all demands. For each ordering of E' and collection of subsets of E', there is a corresponding instance of (δ-)PATH DB.

In total, it is sufficient to solve $\rho! \cdot 2^{\rho \cdot |D|}$ such instances of (δ-)PATH DB for directed graphs, and $\rho! \cdot 3^{\rho \cdot |D|}$ instances of (δ-)PATH DB for undirected graphs. Since (δ-)PATH DB is solvable in polynomial time by Theorem 10, we obtain the desired result.   ◄

4   Hardness results

Our first two hardness results are in the restrictive setting wherein $T_{\max} = 2$ and the initial temporal assignment is the constant function **1**. In this setting, the problems DELAYBET-TER and δ-DELAYBETTER essentially ask only whether there *exists* any λ satisfying our

passenger demands; any such λ can be assumed without loss of generality to have lifetime 2, and could be obtained by delaying all time-edges by at most 1 - meaning our results hold for any $\delta \geq 1$.

▶ **Theorem 14.** *On undirected graphs, DELAYBETTER (and δ-DELAYBETTER with any $\delta \geq 1$) is NP-complete even restricted to instances where $T_{\max} = 2$, the initial temporal assignment is the constant function* **1***, and the $\mathcal{G}_{\downarrow}$ has diameter 6.*

Proof. Our reduction is from POSITIVE NOT-ALL-EQUAL EXACTLY 3SAT [11], an NP-complete problem taking as input a formula ϕ consisting of triples of variables (which appear only positively). The formula ϕ is a yes-instance if there is an assignment to the variables such that every triple contains at least one true variable and at least one false variable.

We shall construct a graph G which admits a temporal assignment $\lambda' : E(G) \to \{1, 2\}$ satisfying all our demands if and only if ϕ admits a satisfying assignment. Figure 4 may be of use to the reader in following the proof. Solid (resp. dashed) edges in bold are ones which are necessarily assigned 1 (resp. 2) in any temporal assignment λ' satisfying all demands.
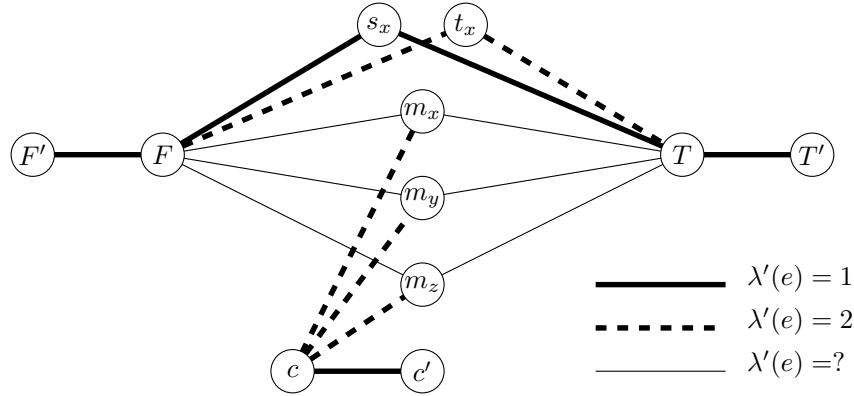
Figure 4 A sketch of our construction. The vertices s_x, t_x, m_x constitute the gadget for a variable x, and the vertices m_x, m_y, m_z, c, c' constitute the gadget for a triple $c = \mathrm{nae}(x, y, z)$.

We shall refer to the demand (u, v, t) as a t-demand from u to v. We begin with four special vertices F, F', T, T', with 1-demands from F' to F and T' to T. Then, for each variable x in ϕ, we introduce vertices s_x, t_x, m_x and edges from each of these to each of T, F. We further introduce 1-demands from s_x to each of T, F (enforcing that both edges must be assigned time 1) and 2-demands from each of T', F' to t_x (enforcing that both of $(T, t_x), (F, t_x)$ must be assigned time 2). Lastly, we introduce 2-demands from s_x to m_x and from m_x to t_x, which together with the previous constraints, guarantees that $\lambda'(m_x, T) \neq \lambda'(m_x, F)$.

Next, for each triple c in ϕ, we create vertices c and c' and a 1-demand between these, and connect the vertex c to m_x by an edge if x appears in the triple c and introduce a 2-demand from c' to m_x. We also introduce 2-demands from each of T and F to c.

The intention is that assigning $\lambda'(m_x, T) = 1$ will correspond to an assignment of **true** to x in ϕ, and assigning $\lambda'(m_x, F) = 1$ will correspond to an assignment of **false** to x in ϕ. Suppose that some λ' satisfies all demands. Then the assignment in which variable x is set to **true** if $\lambda'(m_x, T) = 1$ and **false** otherwise is a satisfying assignment of ϕ.

Suppose that ϕ has a satisfying assignment X. Consider the temporal assignment λ' in which $\lambda'(m_x, T) = 1$ and $\lambda'(m_x, F) = 2$ if x is **true** under X and $\lambda'(m_x, T) = 2$ and $\lambda'(m_x, F) = 1$ otherwise (and all other values of λ' are as specified in Figure 4). Under λ', every clause c is adjacent to some pair of vertices m_x, m_y such that $x =$ **true** under X and $y =$ **false** under X – so the 2-demand from T (resp. F) to c can be routed through m_x (resp. m_y). It is clear that λ' satisfies all other demands.   ◀

Our result for directed graphs requires a slightly different proof:

▶ **Theorem 15.** *On directed graphs, DELAYBETTER (and δ-DELAYBETTER with any $\delta \geq 1$) is NP-complete even restricted to instances where $T_{\max} = 2$ and G has no directed cycles.*

Proof. The reduction is again from POSITIVE NOT-ALL-EQUAL EXACTLY 3SAT. Given a formula ϕ, we construct a directed graph G as follows: Then, for each variable x in ϕ, we introduce six vertices $s_x, s_x^T, s_x^F, t_x, t_x^T, t_x^F$ and connect them as shown in Figure 5. Further, we introduce a vertex c identified with each triple c in ϕ, and create directed edges from c to s_x^T and from c to s_x^F.
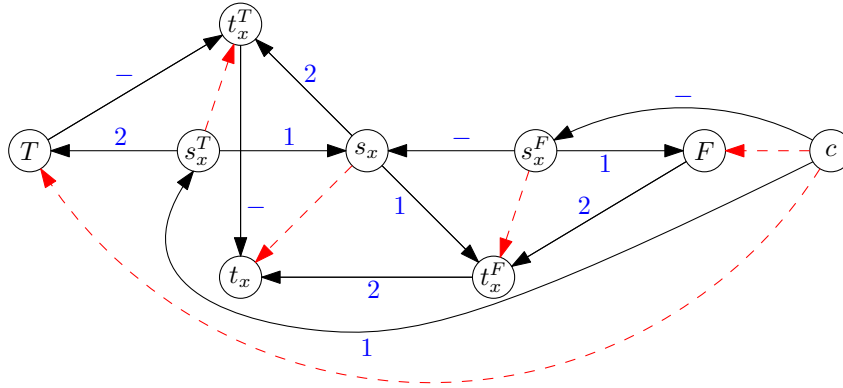
Figure 5 A sketch of our construction showing NP-completeness of DELAYBETTER for digraphs. The vertices $s_x, s_x^T, s_x^F, t_x, t_x^T, t_x^F$ constitute the gadget for a variable x, and the vertex c (together with its out-edges) constitutes the gadget for a triple $c \ni x$. Directed edges in G are solid, whereas 2-demands are shown as dashed arrows in red. The temporal assignment shown (in blue) is one corresponding to the assignment x=true ($-$ denotes an arbitrary choice).

We now specify the demands for our instance; for each variable x, we have 2-demands from s_x (resp. s_x^T, s_x^F) to t_x (resp. t_x^T, t_x^F), and for each clause c we have 2-demands from c to each of T and F. (All of our demands are 2-demands, and these are shown as red dashed arrows in Figure 5.) We let the constant function **1** be the initial temporal assignment for our directed graph, and this concludes the construction of our (δ-)DELAYBETTER instance (together with specifying $\delta = 1$, if necessary).

▷ **Claim 16.** Let λ' be any temporal assignment satisfying all demands in our construction. Then $\lambda'(s_x^T, T) = 2$ entails $\lambda'(s_x^F, F) = 1$, and $\lambda'(s_x^F, F) = 2$ entails $\lambda'(s_x^T, T) = 1$.

Proof. Suppose $\lambda'(s_x^T, T) = 2$ for some x. Since all our demands are satisfied, we have that there must be a temporal path from s_x^T to t_x^T arriving at time 2. Such a path necessarily leaves at time 1 (since the two vertices are at distance 2). Consequently, $\lambda'(s_x^T, s_x) = 1$ and $\lambda'(s_x, t_x^T) = 2$. Similarly, we now must have that the 2-demand from s_x to t_x is routed through t_x^F, entailing that $\lambda'(s_x, t_x^F) = 1$ and $\lambda'(t_x^F, t_x) = 2$. Applying the same logic a third time, the 2-demand from s_x^F to t_x^F must be routed through F, and the desired claim follows. (The other direction is symmetric.) ◁

Suppose that some λ' satisfies all demands. Consider the truth assignment in which a variable x is set to true if $\lambda'(s_x^T, T) = 2$, and false otherwise. Suppose for contradiction that under this truth assignment, some triple c is not satisfied. Then either: (a) all variables in c are true under our truth assignment, and leveraging Claim 16, the vertex c cannot reach

the vertex F by time 2; or, (b), all variables in c are `false` under our truth assignment, and there c cannot reach the vertex T by time 2. In either case, some demand is not satisfied and we derive the desired contradiction.

Now suppose that there is some truth assignment satisfying ϕ. Consider the temporal assignment λ' in which:

- If $x \in c$ and x is `true` (resp. `false`) under the assignment, then $\lambda'(c, s_x^T) = 1$ (resp. $\lambda'(c, s_x^F) = 1$), and
- If x is `true` (resp. `false`) under the truth assignment, then $\lambda'(s_x^T, T) = 2$ (resp. $\lambda'(s_x^F, F) = 2$) and temporal assignments to other directed edges in each variable gadget being chosen consistently with the proof of Claim 16 to satisfy demands within the variable gadget, as shown in Figure 5.
- All other edges are assigned times arbitrarily.

Under λ', c has a path to T (resp. F) through s_x^T (resp. s_x^F) if and only if $x \in c$ is assigned `true` (resp. `false`). It should be clear that λ' satisfies all other demands in our instance by construction, and the result follows. ◀

Having shown that the instance being a tree yields tractability in Corollary 12, we consider the case of planar graphs - a well-studied superclass of trees.

▶ **Theorem 17.** *δ-DELAYBETTER is NP-complete under any combination of the following:*
- *G is planar and has maximum degree 10.*
- *Either G is undirected, or G is a directed acyclic graph (DAG).*
- *Either $T_{\max} = 19$ and $T_{\text{init}} = 1$ (with any $\delta \geq 19$), or $T_{\max} = 19$ and $\delta = 10$.*

Proof. Our reduction is from CUBIC BIPARTITE PLANAR EDGE PRECOLORING EXTENSION (CBP-EPE). That problem asks, given an undirected graph G (which is planar, bipartite, and cubic) and a precoloring of its edges $P : E(G) \rightarrow \{R, G, B, U\}$ (indicating red, green, blue, and uncolored edges respectively) whether there is a proper edge-coloring $C : E(G) \rightarrow \{R, G, B\}$ of G such that $P(e) \in \{R, G, B\} \implies C(e) = P(e)$. Let A, B be an arbitrary bipartition of $V(G)$, and fix an arbitrary order on $V(G)$ (so we may refer to the ith neighbor of some vertex).

We shall make use of the following hardness result:

▶ **Lemma 18** (Theorem 2.3 in [24])**.** *CUBIC BIPARTITE PLANAR EDGE PRECOLORING EXTENSION is NP-complete.*

Construction

Our construction for the directed case is a specific orientation of our construction for the undirected case. Consequently, we shall describe the directed construction, which implicitly also specifies the undirected construction – but still detail explicitly, for example, that edge-gadgets can only be traversed from an A-gadget to a B-gadget (which is trivial in the directed case).

In our construction, the inclusion of a *bold time-edge* (x, y, t) essentially dictates that the edge (x, y) is assigned time t exactly in any temporal assignment satisfying all demands. To realize this constraint, we introduce a temporal path of length and duration $t - 1$ on new vertices $xy_1, \ldots, xy_{t-1}$ and x, as shown in Figure 6 and include (xy_1, y, t) in our demands. Note that in the case where $t = 1$ no new vertices are created – only the demand $(x, y, 1)$.

Figure 6 Our gadget ensuring that bold time-edges are never delayed.
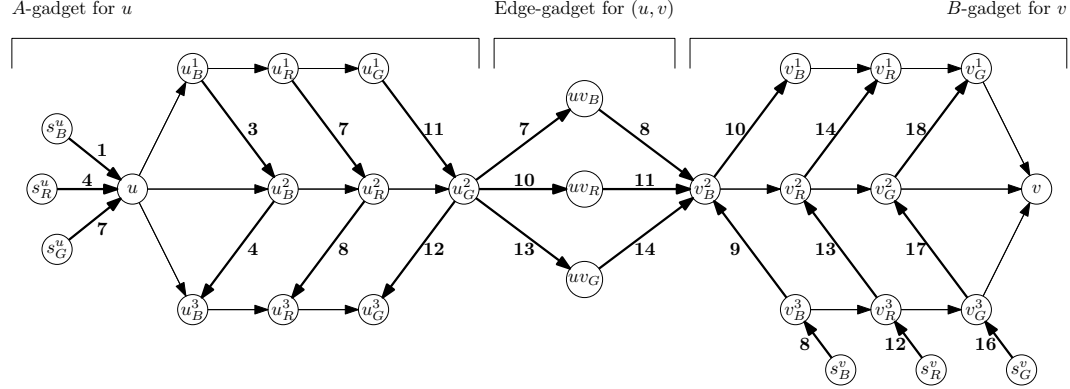
Figure 7 A sketch of our reduction from Cubic Bipartite Planar Edge Precoloring Extension to DelayBetter. Only bold time-edges are labeled.

The reader may find the diagram in Figure 7 helpful. We first describe the graph G' for our instance of DelayBetter, and then the demands D. (For now, we let the initial temporal assignment λ be 1 everywhere except for bold time-edges and their gadgets.)

For each vertex $v \in V(G)$, we create a *vertex-gadget* consisting of a copy of v and 12 other vertices $s_B^v, s_R^v, s_G^v, v_B^1, v_B^2, v_B^3, v_R^1, v_R^2, u_R^3, v_G^1, v_G^2, v_G^3$ (subscripts represent color; superscript i represents the ith neighbor of v). These vertices are connected differently depending on whether $v \in A$ or $v \in B$, as shown in Figure 7. In a vertex-gadget, we call *spoke edges* those edges which are not bold, and *blue* (resp. *red, green*) *layer* the vertices v_B^i (resp. v_R^i, v_G^i).

For each edge $(u, v) \in E(G)$ with $u \in A, v \in B$, we also create three vertices uv_B, uv_R, uv_G. Then if u is the ith neighbor of v and v is the jth neighbor of u, we introduce six bold time-edges $(u_G^i, uv_B, 7), (uv_B, v_B^j, 8), (u_G^i, uv_R, 10), (uv_R, v_B^j, 11), (u_G^i, uv_G, 13), (uv_G, v_B^j, 14)$. (In Figure 7 u is the second neighbor of v and vice versa.) If $P(u, v)$ is precolored G under P, then we delete two of uv_B, uv_R, or uv_G, as appropriate, leaving just one path from u to v. In Figure 7: u is the second neighbor of the v; v is the second neighbor of u; and $P(u, v) = U$.

We make use of three types of demands:

Bold demands as described earlier and shown in Figure 6.

Hermits demands from a vertex in a vertex-gadget to another vertex in the same gadget. For each vertex $u \in A$ we have demands $(s_B^u, u_B^3, 4), (s_R^u, u_R^3, 8)$, and $(s_G^u, u_G^3, 12)$, and for each vertex $v \in B$ we have demands $(s_B^v, v, 13), (s_R^v, v, 16)$, and $(s_G^v, v, 19)$. (We say hermits have the color of the layer their source or destination lies in.)

Travelers demands from a vertex in an A-gadget to a vertex in a B-gadget. For each edge (u, v) in the CBP-EPE instance with $u \in A$ and $v \in B$, we add a demand $(u, v, 19)$.

This concludes our construction.

Correctness

$\triangleright$ Claim 19. If the CBP-EPE instance G, P is a yes-instance, then the DelayBetter instance $(G', \lambda), D$ is a yes-instance.

Proof. We shall construct a delaying λ' of the initial temporal assignment λ satisfying all demands in D. Consider a proper edge coloring C of G which extends P.

First, we do not delay any bold time-edges – i.e., for those, $\lambda(e) = \lambda'(e)$. Note that all bold demands are immediately satisfied under any such labeling.

Let (u, v) be an edge assigned color B (resp. R, G) under C, with u being the ith neighbor of v and v being the jth neighbor of u. We assign:

- $\lambda'(u, u_B^j) = 2$ (resp. $5, 8$)
- $\lambda'(u_B^j, u_R^j) = 3$ (resp. $6, 9$)
- $\lambda'(u_R^j, u_G^j) = 4$ (resp. $7, 10$)
- (time-edges into and out of uv_B, uv_R, uv_G are all bold)
- $\lambda(v_B^i, v_R^i) = 11$ (resp. $14, 17$)
- $\lambda(v_R^i, v_G^i) = 12$ (resp. $15, 18$)
- $\lambda(v_G^i, v) = 13$ (resp. $16, 19$)

It should be clear that this labeling creates a temporal path from u to v for each edge $(u, v) \in G$ such that the traveler demands are satisfied (via uv_B, uv_R, or uv_G depending on whether the edge was colored B, R, or G under P).

We now show hermit demands are satisfied as well: because P is a proper 3-edge-coloring of a cubic graph, every vertex is incident to exactly one edge of each color.

In A-gadgets, the hermit starting at s_B^u (resp. s_R^u, s_G^u) has a temporal path to u_B^i (resp. u_R^i, u_G^i) arriving by time 2 (resp. $6, 10$) if the edge from u to its ith neighbor is assigned B (resp. R, G) under C. The hermit can then (if $i \neq 3$) use the bold time-edges to reach u_B^3 (resp. u_R^3, u_G^3).

In B-gadgets, the hermit starting at s_B^v (resp. s_R^v, s_G^v) has a temporal path to v_B^i (resp. v_R^i, v_G^i) arriving by time 10 (resp. $14, 18$) using the bold time-edges. If the edge from v to its jth neighbor is assigned B (resp. R, G) under C, then the hermit can extend this path by using the spoke edges from v_B^j (resp. v_R^j, v_G^j) into v. ◁

The remainder of the proof is devoted to showing the opposite implication; that is, if DELAYBETTER instance $(G', \lambda), D$ is a yes-instance (i.e., there exists some delaying λ' of λ satisfying all demands in D) then the CBP-EPE instance G, P is a yes-instance. For some λ', we say that the traveler from u to v is *blue* (resp. *red, green*) if that traveler is routed through a vertex uv_B (resp. uv_R, uv_G). (If several paths are possible, one may be chosen arbitrarily - though as we shall see this never happens.) No traveler has more than one color: each traveler goes through exactly one edge-gadget, from its starting A-gadget to its ending B-gadget (due to the bold time-edges enforcing the direction of the edge-gadget).

We make repeated use of the fact that, by construction, bold time-edges are never delayed. Note that if $\lambda = \mathbf{1}$ everywhere including bold gadgets, then these force their edge to be at exactly the intended time in the delaying λ'.

▷ **Claim 20.** Let $u \in A$. Then there is exactly one i such that $\lambda'(u, u_B^i) \in [2, 4]$ (resp. $[5, 7], [8, 10]$); there is at least one i such that $\lambda'(u_B^i, u_R^i) \in [6, 8]$ (resp. [9-11]); and there is at least one i such that $\lambda'(u_R^i, u_G^i) \in [10, 12]$.

Proof. The claim holds as a consequence of the hermit demands. The blue (resp. red, green) hermit must reach the blue (resp. red, green) layer using at least one (resp. two, three) spoke edge(s), arriving by time 4 (resp. 8, 12) at the latest and departing from u at time 2 (resp. 5, 8) at the earliest. ◁

▷ **Claim 21.** At most $\frac{1}{3}$ of travelers are blue and at most $\frac{2}{3}$ of travelers are red or blue.

Proof. First, suppose over a third of travelers are blue. Then the A-gadget of some vertex u has at least two travelers reaching different vertices of its green layer by time 6 (and, necessarily, different vertices of its red layer by time 5). This entails that at least two of the spoke edges between the blue and red layers in that gadget are at time 5 or less, which contradicts Claim 20. Similarly, if over two thirds of travelers are red or blue, then the A-gadget of some vertex u has at least three travelers reaching three different vertices of the green layer by time 9, entailing that the three spoke edges from the red layer to the green layer are at time 9 or earlier and again contradicting Claim 20.   ◁

The following result is obtained through similar reasoning to that for Claim 20:

▷ **Claim 22** (∗). Let $v \in B$. Then under λ', there is some i such that $v_B^i - v_R^i - v_G^i - v$ is a temporal path with departure time in $[9, 11]$ and arrival time in $[11, 13]$; there is some i such that $v_R^i - v_G^i - v$ is a temporal path with departure time in $[13, 15]$ and arrival time in $[14, 16]$; and there is some i such that $\lambda'(v_G^i, v) \in [17, 19]$.

▷ **Claim 23**. Let $v \in B$. Then at least 1 traveler arrives at the B-gadget of v at time 8; and at least 2 travelers arrive at the B-gadget of v at time 8 or time 11.

Proof. First note that all travelers arriving at the B-gadget come from some edge-gadget and consequently arrive at a time in $\{8, 11, 14\}$. Applying Claim 22, there is some i such that any traveler arriving at v_B^i strictly after time 10 would be stranded there – so the traveler arriving from the ith neighbor of v must arrive at time 8. Likewise, there is some j different from i such that any traveler arriving at v_R^j strictly after time 14 would be stranded there – so the traveler arriving from the jth neighbor of v must arrive at v_R^j by time 14 and so at v_B^j at time 8 or time 11.   ◁

The proof of the following is similar to that of Claim 21:

▷ **Claim 24** (∗). At least $\frac{1}{3}$ of travelers are blue and at least $\frac{2}{3}$ of travelers are red or blue.

For some λ', we say that the traveler from u to v is *blue* (resp. *red, green*) if the temporal path used to route that traveler goes through a vertex uv_B (resp. uv_R, uv_G).

▷ **Claim 25**. The colors of travelers in (G', λ') fully specify a proper edge-coloring of G which is consistent with the precoloring P.

Proof. First, note that the precoloring is consistent with P because precolored edges in G have edge-gadgets consisting of only one vertex, ensuring that the traveler is assigned the appropriate color.

Next, observe that Claims 21 and 24 together entail that *exactly* $\frac{1}{3}$ of travelers are blue and *exactly* $\frac{1}{3}$ of travelers are red. Moreover, the proof of those claims holds locally; exactly one of the three travelers leaving any given A-vertex is blue (resp. red), and exactly one of the three travelers arriving at any given B-vertex is blue (resp. red).   ◁

This concludes the proof that the CBP-EPE instance (G, P) is a yes-instance if the DELAYBETTER instance $(G, \lambda), D$ was a yes-instance.

We emphasize at this point that our construction preserves planarity and that in the directed case, the footprint contains no directed cycles. We recall that in the undirected case bold time-edges enforce that travelers can only go from an A-gadget to a B-gadget once. The maximum degree in the graph is 10 (due to vertices u_G^i, which are incident to 4 bold gadgets in addition to 6 normal edges). Note that the proof still holds if the initial temporal assignment λ assigns time 2 to every non-bold edge in an A-gadget and time 9 to every

non-bold edge in a B-gadget, in which case the largest delay is of 10 (delaying a time-edge from the green layer of a B-gadget to a B-vertex v to be at time 19). Consequently, our proof also shows that δ-DELAYBETTER is NP-hard for $\delta \geq 9$.

On the other hand, the proof also holds if the initial temporal assignment is instead the constant function $\mathbf{1}$: studying Figure 6 it can be seen that this would still result in bold time-edges being assigned the intended time under λ'.

We have membership of NP from Lemma 8, and the result follows.   ◀

───── **References** ─────

1   Stefan Binder, Yousef Maknoon, and Michel Bierlaire. The multi-objective railway timetable rescheduling problem. *Transportation Research Part C: Emerging Technologies*, 78:78–94, May 2017. `doi:10.1016/j.trc.2017.02.001`.

2   Benjamin Merlin Bumpus and Kitty Meeks. Edge exploration of temporal graphs. In Paola Flocchini and Lucia Moura, editors, *Combinatorial Algorithms*, Lecture Notes in Computer Science, pages 107–121, Cham, 2021. Springer. `doi:10.1007/978-3-030-79987-8_8`.

3   Valentina Cacchiani, Dennis Huisman, Martin Kidd, Leo Kroon, Paolo Toth, Lucas Veelenturf, and Joris Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37, May 2014. `doi:10.1016/j.trb.2014.01.009`.

4   S. Carosi, S. Gualandi, F. Malucelli, and E. Tresoldi. Delay Management in Public Transportation: Service Regularity Issues and Crew Re-scheduling. *Transportation Research Procedia*, 10:483–492, 2015. `doi:10.1016/j.trpro.2015.09.002`.

5   Arnaud Casteigts, Timothée Corsini, and Writika Sarkar. Simple, strict, proper, happy: A study of reachability in temporal graphs, August 2022. `doi:10.48550/arXiv.2208.01720`.

6   Argyrios Deligkas, Eduard Eiben, and George Skretas. Minimizing Reachability Times on Temporal Graphs via Shifting Labels. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 5333–5340, Macau, SAR China, August 2023. International Joint Conferences on Artificial Intelligence Organization. `doi:10.24963/ijcai.2023/592`.

7   Argyrios Deligkas and Igor Potapov. Optimizing reachability sets in temporal graphs by delaying. *Information and Computation*, 285:104890, May 2022. `doi:10.1016/j.ic.2022.104890`.

8   Punctuality | Deutsche Bahn Interim Report 2024. `https://zbir.deutschebahn.com/2024/en/interim-group-management-report-unaudited/product-quality-and-digitalization/punctuality/`. [Accessed 19-09-2024].

9   Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. Delay Management with Rerouting of Passengers. *Transportation Science*, 46(1):74–89, February 2012. `doi:10.1287/trsc.1110.0375`.

10 Jessica A. Enright, Laura Larios-Jones, Kitty Meeks, and William Pettersson. Reachability in temporal graphs under perturbation. In *SOFSEM 2025: Theory and Practice of Computer Science - 50th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2025, Bratislava, Slovak Republic, January 20-23, 2025, Proceedings, Part I*, volume 15538 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2025. `doi:10.1007/978-3-031-82670-2_19`.

11 Ivan Tadeu Ferreira Antunes Filho. *Characterizing Boolean satisfiability variants*. PhD thesis, Massachusetts Institute of Technology, 2019. URL: `https://dspace.mit.edu/handle/1721.1/124241`.

12 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Delay-Robust Routes in Temporal Graphs, January 2022. arXiv:2201.05390 [cs]. `doi:10.48550/arXiv.2201.05390`.

13 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Temporal Connectivity: Coping with Foreseen and Unforeseen Delays, January 2022. arXiv:2201.05011 [cs]. `doi:10.48550/arXiv.2201.05011`.

14 Michael Gatto, Björn Glaus, Riko Jacob, Leon Peeters, and Peter Widmayer. Railway Delay Management: Exploring Its Algorithmic Complexity. In *Algorithm Theory - SWAT 2004*, volume 3111, pages 199–211. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. Series Title: Lecture Notes in Computer Science. `doi:10.1007/978-3-540-27810-8_18`.

15 Michael Gatto, Riko Jacob, Leon Peeters, and Anita Schöbel. The Computational Complexity of Delay Management. In *Graph-Theoretic Concepts in Computer Science*, volume 3787, pages 227–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. Series Title: Lecture Notes in Computer Science. `doi:10.1007/11604686_20`.

16 Andreas Ginkel and Anita Schöbel. To Wait or Not to Wait? The Bicriteria Delay Management Problem in Public Transportation. *Transportation Science*, 41(4):527–538, November 2007. `doi:10.1287/trsc.1070.0212`.

17 Géraldine Heilporn, Luigi De Giovanni, and Martine Labbé. Optimization models for the single delay management problem in public transportation. *European Journal of Operational Research*, 189(3):762–774, September 2008. `doi:10.1016/j.ejor.2006.10.065`.

18 A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, November 1962. `doi:10.1145/368996.369025`.

19 N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. Association for Computing Machinery. `doi:10.1145/800057.808695`.

20 Leo G. Kroon, Rommert Dekker, and Michiel J. C. M. Vromans. Cyclic Railway Timetabling: A Stochastic Optimization Approach. In Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos D. Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, volume 4359, pages 41–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. Series Title: Lecture Notes in Computer Science. `doi:10.1007/978-3-540-74247-0_2`.

21 David C. Kutner and Laura Larios-Jones. Temporal Reachability Dominating Sets: contagion in temporal graphs, May 2024. arXiv:2306.06999 [cs, math]. `doi:10.48550/arXiv.2306.06999`.

22 Eva König. A review on railway delay management. *Public Transport*, 12(2):335–361, June 2020. `doi:10.1007/s12469-020-00233-1`.

23 Federico Malucelli and Emanuele Tresoldi. Delay and disruption management in local public transportation via real-time vehicle and crew re-scheduling: a case study. *Public Transport*, 11(1):1–25, June 2019. `doi:10.1007/s12469-019-00196-y`.

24 Dániel Marx. NP-completeness of list coloring and precoloring extension on the edges of planar graphs. *Journal of Graph Theory*, 49(4):313–324, 2005. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jgt.20085. `doi:10.1002/jgt.20085`.

25 Hendrik Molter, Malte Renken, and Philipp Zschoche. Temporal reachability minimization: Delaying vs. deleting. *Journal of Computer and System Sciences*, 144:103549, September 2024. `doi:10.1016/j.jcss.2024.103549`.

26 B. M. E. Moret. Planar NAE3SAT is in P. *SIGACT News*, 19(2):51–54, June 1988. `doi:10.1145/49097.49099`.

27 Michael Schachtebeck. *Delay Management in Public Transportation: Capacities, Robustness, and Integration.* PhD thesis, Georg-August-University Göttingen, 2010. `doi:10.53846/goediss-2538`.

28 Anita Schöbel. A Model for the Delay Management Problem based on Mixed-Integer-Programming. *Electronic Notes in Theoretical Computer Science*, 50(1):1–10, August 2001. `doi:10.1016/S1571-0661(04)00160-4`.

29 Schöbel, Anita. *Optimization in Public Transportation*, volume 3 of *Springer Optimization and Its Applications.* Springer US, Boston, MA, 2006. `doi:10.1007/978-0-387-36643-2`.

30 Geoffrey Scozzaro, Clara Buire, Daniel Delahaye, and Aude Marzuoli. Optimizing air-rail travel connections: A data-driven delay management strategy for seamless passenger journeys. In *SESAR Innovation Days*, 2023.

31 Lucas P. Veelenturf, Martin P. Kidd, Valentina Cacchiani, Leo G. Kroon, and Paolo Toth. A Railway Timetable Rescheduling Approach for Handling Large-Scale Disruptions. *Transportation Science*, 50(3):841–862, August 2016. `doi:10.1287/trsc.2015.0618`.

32 Chuntian Zhang, Yuan Gao, Valentina Cacchiani, Lixing Yang, and Ziyou Gao. Train rescheduling for large-scale disruptions in a large-scale railway network. *Transportation Research Part B: Methodological*, 174:102786, August 2023. `doi:10.1016/j.trb.2023.102786`.

33 Yongqiu Zhu and Rob M. P. Goverde. Integrated timetable rescheduling and passenger reassignment during railway disruptions. *Transportation Research Part B: Methodological*, 140:282–314, October 2020. `doi:10.1016/j.trb.2020.09.001`.