Temporal Reachability Minimization: Delaying vs. Deleting

Hendrik Molter
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva, Israel
Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Malte Renken
Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Philipp Zschoche
Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Abstract

We study spreading processes in temporal graphs, i. e., graphs whose connections change over time. These processes naturally model real-world phenomena such as infectious diseases or information flows. More precisely, we investigate how such a spreading process, emerging from a given set of sources, can be contained to a small part of the graph. To this end we consider two ways of modifying the graph, which are (1) deleting connections and (2) delaying connections. We show a close relationship between the two associated problems and give a polynomial time algorithm when the graph has tree structure. For the general version, we consider parameterization by the number of vertices to which the spread is contained. Surprisingly, we prove W[1]-hardness for the deletion variant but fixed-parameter tractability for the delaying variant.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Network flows

Keywords and phrases Temporal Graphs, Temporal Paths, Disease Spreading, Network Flows, Parameterized Algorithms, NP-hard Problems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.76

Funding *Hendrik Molter*: Supported by the German Research Foundation (DFG), project MATE (NI 369/17), and by the Israeli Science Foundation (ISF), grant No. 1070/20. The main work was done while affiliated with TU Berlin.
Malte Renken: Supported by the German Research Foundation (DFG), project MATE (NI 369/17).

Acknowledgements This work was initiated at the research retreat of the Algorithmics and Computational Complexity group of TU Berlin in September 2020 in Zinnowitz.

1 Introduction

Reachability is a fundamental problem in graph theory and algorithmics [38, 39, 33, 17] and quite well-understood. With the emergence of temporal graphs,[1] the concept of reachability was extended to the dimension of time using *temporal paths* [34, 6]. For a vertex s to reach another vertex z in a temporal graph, there must not only be a path between them but the edges of this path have to appear in chronological order. This requirement makes temporal reachability non-symmetric and non-transitive, which stands in contrast to reachability in normal (static) graphs. Reachability is arguably one of the most central concepts in temporal graph algorithmics and has been studied under various aspects, such as path finding [40, 5, 9, 11], vertex separation [34, 28, 41], finding spanning subgraphs [12, 4], temporal graph exploration [25, 23, 26, 7, 24, 3], and others [35, 2, 10, 32].

[1] Temporal graphs are graphs whose edge set changes over discrete time.

Perhaps the most prominent application of temporal graph reachability is currently epidemiology, dealing with effective prevention or containment of disease spreading [1]. Here, minimizing the reachability of vertices in a temporal graph by manipulating the temporal graph corresponds to minimizing the spread of an infection in various networks by some countermeasures. Application instances for this scenario may be drawn from physical contacts [27, 19] or airline flights [8, 13], but also social networks [31, 14], cattle movements [36], or computer networks [37].

Enright et al. [21] studied the problem of deleting k time-edges[2] such that no single vertex can reach more than r other vertices and showed its NP-hardness and W[1]-hardness for the parameter k, even in very restricted settings. Here, we shift the focus to a set of multiple given sources, thus studying the following problem, which has not been considered for computational complexity analysis yet (to the best of our knowledge).

Minimizing Temporal Reachablity by Deleting (MinReachDelete)

Input: A temporal graph $\mathcal{G}$, a set of sources vertices S, and integers k, r.

Question: Can we delete at most k time-edges s.t. at most r vertices are reachable from S?

Imaginably, removing edges or vertices is not the most *infrastructure friendly* approach to restrict reachability. To address this, other operations have been studied. Enright et al. [22] considered restricting the reachability by just changing the relative order in which edges are active. Deligkas and Potapov [15] considered restricting the reachability by a merging operation of consecutive edge sets of the temporal graph and by a delay operation of time-edges by δ time steps, i.e., moving a time-edge from time t to $t + \delta$. In particular, they introduced a delay variant of MinReachDelete.

Minimizing Temporal Reachablity by Delaying (MinReachDelay)

Input: A temporal graph $\mathcal{G}$, a set of sources vertices $S \subseteq V$, and integers k, r, δ.

Question: Can we delay at most k time-edges by δ s.t. at most r vertices are reachable from S?

This is the central problem studied in this paper. Throughout the whole paper we assume for all instances of MinReachDelete and MinReachDelay that $0 < |S| \leq r$. We remark that technically Deligkas and Potapov [15] formulate the problem slightly differently, allowing delays of *up to* δ to appear. However, a simple argument can be given to see that this distinction is not significant: Clearly, delaying a time-edge reduces the number of reachable vertices only if the undelayed time-edge could be reached from some source $s \in S$. But when this is the case, increasing the delay of that time-edge can never increase the set of vertices reachable from S. The reason is that, while increasing the delay might enable some source $s' \in S \setminus \{s\}$ to reach a vertex v, that vertex v would be reached from s in any case.

Deligkas and Potapov [15] showed that MinReachDelay is NP-hard and W[1]-hard when parameterized by k, even if underlying graph has lifetime $\tau = 2$. A close look into the proof reveals that this also holds for MinReachDelete.

Our contribution. We study how MinReachDelete and MinReachDelay relate to each other. We show that both problems are polynomial-time solvable on trees. Moreover, there is an intermediate reduction from MinReachDelete to MinReachDelay indicating that MinReachDelay seems generally harder than MinReachDelete. However, surprisingly, this is no longer true when we parameterize the problems by the number r of reachable vertices. Here, we develop a max-flow-based branching strategy and obtain fixed-parameter

[2] That is, an edge at a point in time.

tractability for MinReachDelay while MinReachDelete remains W[1]-hard. This makes MinReachDelay particularly interesting for applications where the number of reachable vertices should be very small, e.g. when trying to contain the spread of dangerous diseases.

2 Preliminaries

We define $\mathbb{N}$ as the positive natural numbers, $[a, b] := \{i \in \mathbb{Z} \mid a \leq i \leq b\}$, and $[n] := [1, n]$. For a function $f\colon V \to \mathbb{Z}$ and subset $X \subseteq V$ we denote by $f(X)$ the sum $\sum_{x \in V} f(x)$. We use standard notation from graph theory [16]. We say for a (directed) graph G that $G - X := G[V(G) \setminus X]$ is the induced subgraph of G when the vertices in X are removed, and $G \setminus Y := (V(G), E(G) \setminus Y)$ is the subgraph when the edges in Y are removed, where X is a vertex set and Y is an edge set. For any predicate P, the Iverson bracket $[P]$ is 1 if P is true and 0 otherwise.

Parameterized complexity. Let Σ denote a finite alphabet. A parameterized problem $L \subseteq \{(x, k) \in \Sigma^* \times \mathbb{N} \cup \{0\}\}$ is a subset of all instances (x, k) from $\Sigma^* \times \mathbb{N} \cup \{0\}$, where k denotes the *parameter*. A parameterized problem L is in FPT (is *fixed-parameter tractable*) if there is an algorithm that decides every instance (x, k) for L in $f(k) \cdot |x|^{O(1)}$ time, where f is any computable function only depending on the parameter. If a parameterized problem L is W[1]-hard, then it is presumably not fixed-parameter tractable. We refer to Downey and Fellows [18] for details.

Temporal graphs. A *temporal graph* $\mathcal{G}$ consists of a set of vertices V (or $V(\mathcal{G})$), and a sequence of edge sets $(E_i)_{i \in [\tau]}$ where each E_i is a set of unordered pairs from V. The number τ is called the *lifetime* of $\mathcal{G}$. The elements of $\mathcal{E}(\mathcal{G}) := \bigcup_{i \in [\tau]} E_i \times \{i\}$ are called the *time-edges* of $\mathcal{G}$. Furthermore $\mathcal{G}$ has a *traversal time* function $\gamma : \mathcal{E}(\mathcal{G}) \to \mathbb{N}$ specifying the time it takes to traverse each time-edge. The temporal graph $\mathcal{G}$ is then written as the tuple $(V, (E_i)_{i \in [\tau]}, \gamma)$. Often we assume γ to be the constant function $\gamma \equiv 1$ and then simply write $\mathcal{G} = (V, (E_i)_{i \in [\tau]})$. The *underlying graph* of $\mathcal{G}$ is the graph $(V, \bigcup_{i=1}^{\tau} E_i)$. For a time-edge set Y and temporal graph $\mathcal{G}$, we denote by $\mathcal{G} \setminus Y$ the temporal graph where $V(\mathcal{G} \setminus Y) = V(\mathcal{G})$ and $\mathcal{E}(\mathcal{G} \setminus Y) = \mathcal{E}(\mathcal{G}) \setminus Y$. A *temporal s-z-path* in $\mathcal{G}$ is a sequence of time-edges $P = (e_i = (\{v_{i-1}, v_i\}, t_i))_{i=1}^{m}$ where

1. $v_0 = s$ and $v_m = z$,
2. the sequence of edges $(\{v_{i-1}, v_i\})_{i=1}^{m}$ forms an s-z-path in the underlying graph of $\mathcal{G}$, and
3. $t_{i+1} \geq t_i + \gamma(e_i)$ for all $i \in [m-1]$.

The *arrival time* of P is $t_m + \gamma(e_m)$. The set of vertices of P is denoted by $V(P) = \{v_i \mid 0 \leq i \leq m\}$. A vertex w is *reachable* from v in $\mathcal{G}$ (at time t) if there exists a temporal v-w-path in $\mathcal{G}$ (with arrival time at most t). In particular, every vertex reaches itself via a trivial path. Furthermore, w is reachable from $S \subseteq V$ if there is a temporal s-w-path for some $s \in S$, and the set of all vertices reachable from S is denoted the *reachable set* $R_{\mathcal{G}}(S)$. We drop the index $\mathcal{G}$ if it is clear form the context. *Delaying* a time-edge $(\{v, w\}, t)$ by δ refers to replacing it with the time-edge $(\{v, w\}, t + \delta)$. For a temporal graph $\mathcal{G}$ and a time-edge set $X \subseteq \mathcal{E}(\mathcal{G})$ we denote by $\mathcal{G} \nearrow_{\delta} X$ the temporal graph $\mathcal{G}$ where the time-edges in X are delayed by δ.

Preliminary observations. We present an intermediate polynomial-time reduction from the MinReachDelete to MinReachDelay.

▶ **Lemma 1.** *Given an instance $I = (\mathcal{G} = (V, (E_i)_{i\in[\tau]}), S, k, r)$ of* MinReachDelete*, we can compute in linear time, an instance $J = (\mathcal{G}' = (V', (E'_i)_{i=1}^{6\tau+1}), S', k, r', \delta = 3\tau)$ of* MinReachDelay *such that the feedback vertex number[3] of the underlying graph of $\mathcal{G}$ and $\mathcal{G}'$ is the same, and I is a* yes*-instance if and only if J is a* yes*-instance.*

Proof. We construct $\mathcal{G}' = (V', (E'_i)_{i=1}^{3\tau+\delta+1})$ in the following way. Set

$$V_\mathrm{e} := \{\mathrm{e}_{uv}, \mathrm{e}_{vu} \mid (\{v,u\},t) \in \mathcal{E}(\mathcal{G})\}, \qquad V_\mathrm{s} := \{\mathrm{s}_{vu} \mid \mathrm{e}_{vu} \in V_\mathrm{e}\},$$
$$V' := V \cup V_\mathrm{e} \cup V_\mathrm{s}, \text{ and} \qquad S' := S \cup V_\mathrm{s}\,.$$

Begin with $E'_i = \emptyset$ for all $i \in [3\tau]$. Then, add for each time-edge $(\{v,u\},t) \in \mathcal{E}(\mathcal{G})$, the time-edges $(\{v,\mathrm{e}_{vu}\}, 3t-2), (\{v,\mathrm{e}_{vu}\}, 3t), (\{\mathrm{e}_{vu},\mathrm{e}_{uv}\}, 3t-1), (\{u,\mathrm{e}_{uv}\}, 3t-2)$, and $(\{u,\mathrm{e}_{uv}\}, 3t)$ to $\mathcal{G}'$. Afterwards, add the time-edge $(\{\mathrm{s}_{vu},\mathrm{e}_{vu}\}, 6\tau+1 = 3\tau+\delta+1)$ for each $\mathrm{e}_{vu} \in V_\mathrm{e}$. Finally we set $r' := r + |V' \setminus V|$. Since we add for each time-edge of $\mathcal{G}$ a constant number of vertices and time-edges to $\mathcal{G}'$, we have that $|\mathcal{G}'| \in O(|\mathcal{G}|)$ and $\mathcal{G}'$ can be computed in linear time. Moreover, the underlying graph G' of $\mathcal{G}'$ is obtained from the underlying graph G of $\mathcal{G}$ by subdividing edges and adding leaves, thus G and G' have the same feedback vertex number. It remains to prove that the two instances are equivalent.

(⇒): Let X be a solution for I. Then, set $X' := \{(\{\mathrm{e}_{vu},\mathrm{e}_{uv}\}, 3t-1) \mid (\{v,u\},t) \in X\}$. Pick $s \in S$ and $v \in V$ arbitrary. Note that for each temporal s-v-path P' in $\mathcal{G}'$, we can construct a temporal s-v-path P in $\mathcal{G}$ which uses a time-edge $(\{u,w\},t)$ if and only if P' uses the time-edge $(\{\mathrm{e}_{uw},\mathrm{e}_{wu}\}, 3t-1)$. Furthermore, any temporal s-v-path in $\mathcal{G}' \nearrow_\delta X'$ cannot use any delayed time-edge. As s and v are arbitrary, this proves $R_{\mathcal{G}'\nearrow_\delta X'}(S') \cap V \subseteq R_{\mathcal{G}\setminus X}(S)$. Thus, at most $r + |V' \setminus V| = r'$ vertices are reachable from S' in $\mathcal{G}' \nearrow_\delta X'$, thus J is a *yes*-instance.

(⇐): Let X' be a solution for J. Begin by observing that delaying any time-edges between V_s and V_e has no effect. Consequently, $R_{\mathcal{G}'\nearrow_\delta X'}(V_\mathrm{s}) = V_\mathrm{s} \cup V_\mathrm{e}$ for every possible choice of X'. Therefore X' is a valid solution if and only if $|R_{\mathcal{G}'\nearrow_\delta X'}(S) \cap V| \leq r' - |V_\mathrm{s} \cup V_\mathrm{e}| = r$, i.e., we only need to study the reachability between vertices in V. Because of this, delaying a time-edge connecting two vertices of V_e has the same effect as deleting that time-edge. Next, observe that instead of delaying some time-edge $(\{v,\mathrm{e}_{vu}\},t)$ which connects vertices of V and V_e, the same or better reduction of reachability is achieved by instead delaying $(\{\mathrm{e}_{vu},\mathrm{e}_{uv}\},t')$, with $t' \in \{t-1, t+1\}$ chosen appropriately. Due to this, we may assume without loss of generality that $X' \subseteq \{(\{\mathrm{e}_{vu},\mathrm{e}_{uv}\}, 3t-1) \mid (\{v,u\},t) \in \mathcal{E}(\mathcal{G})\}$.

Set $X := \{(\{v,u\},t) \mid (\{\mathrm{e}_{uv},\mathrm{e}_{vu}\}, 3t-1) \in X'\}$. Let $s \in S$ and $v \in V$ be arbitrary. Note that for each temporal s-v-path P in $\mathcal{G}$, we can construct a temporal s-v-path P' in $\mathcal{G}'$ as above. Thus $R_{\mathcal{G}\setminus X}(S) \subseteq R_{\mathcal{G}'\nearrow_\delta X'}(S) \cap V$. Since we already observed that $|R_{\mathcal{G}'\nearrow_\delta X'}(S') \cap V| \leq r$, we conclude that I is a *yes*-instance. ◀

Note that the reduction in Lemma 1 preserves the size k of the solution and the feedback vertex number of the underlying graph. We remark that, in exchange for dropping the latter property, one can modify the reduction to instead have $|S'| = |S|$, by simply adding time-edges from S to V_s before all other time-edges. However, in any case the size r' of the reachable set in J is unbounded in terms of the size r of the reachable set in I. Unless FPT = W[1], this is unavoidable as we learn in the next section.

[3] That is, the minimum number of vertices needed to hit all cycles in an undirected graph.

3 Parameterized by the Reachable Set Size

In this section the study MinReachDelay and MinReachDelete parameterized by the reachable set size r. In particular, our main result in this section is the fixed-parameter tractability of MinReachDelay parameterized by r. This is in stark contrast to the W[1]-hardness of MinReachDelete parameterized by r which we show first.

▶ **Theorem 2.** *MinReachDelete parameterized by r is W[1]-hard, even if $\tau = 2$.*

Proof. We present a parameterized reduction from the W[1]-hard [18] Clique problem parameterized by ℓ, where given a graph $H = (U, F)$ we are asked whether H contains a clique of size ℓ.

Let $H = (U, F)$ be a graph, where $|F| = m$. We construct an instance $I = (\mathcal{G}, \{s\}, k = m - \binom{\ell}{2}, r = 1 + \ell + \binom{\ell}{2})$ of MinReachDelete, where $\mathcal{G} := (V, (E_i)_{i\in[2]})$ is the temporal graph given by

$$V := U \cup \{s\} \cup \{e_f \mid f \in F\},$$
$$E_1 := \{\{s, e_f\} \mid f \in F\}, \quad \text{and} \quad E_2 := \left\{\{e_{\{u,v\}}, u\}, \{e_{\{u,v\}}, v\} \;\middle|\; \{u, v\} \in F\right\}.$$

Note that I can be constructed in polynomial time.

($\Rightarrow$): Let $C = (V', F')$ be a clique of size ℓ in H. We set $X := \{(\{s, e_f\}, 1) \mid f \in F \setminus F'\}$. Note that $|X| \leq k$ and that for each edge $f \in F$ we can reach e_f from s if and only if $f \in F'$. Hence, by the construction of $\mathcal{G}$, a vertex $u \in U$ is reachable from s in $\mathcal{G} \setminus X$ if and only if $u \in V'$. Hence, we can reach $1 + \binom{\ell}{2} + \ell$ many vertices from s in $\mathcal{G} \setminus X$. Thus, I is a *yes*-instance.

($\Leftarrow$): Let $X \subseteq \mathcal{E}(\mathcal{G})$ be a solution for I. Without loss of generality, we can assume that X does not contain a time-edge $(\{e_f, u\}, 2)$, because it can be replaced by $(\{e_f, s\}, 1)$. Observe that at least $\binom{\ell}{2}$ vertices from $\{e_f \mid f \in F\}$ are reachable from s in $\mathcal{G} \setminus X$. Since $r = 1 + \binom{\ell}{2} + \ell$, we can reach from s at most ℓ vertices from U. Hence, $U \cap R_{\mathcal{G}\setminus X}(\{s\})$ must form a clique of size ℓ in H. ◀

Due to Theorem 2, we know that there is presumably no $f(r) \cdot |\mathcal{G}|^{O(1)}$-time algorithm to decide whether we can keep the reachable set of a vertex s of $\mathcal{G}$ small (at most r vertices), by deleting at most k time-edges. However, this changes when we delay (instead of deleting) at most k edges. Formally, we show the following.

▶ **Theorem 3.** *MinReachDelay is solvable in $O(r! \cdot k \cdot |\mathcal{G}|)$ time.*

The proof of Theorem 3 is structured as follows.

Step 1 (reduction to slowing): We reduce MinReachDelay to an auxiliary problem which we call MinReachSlow. Here, instead of *delaying* a time-edge (moving it δ layers forward in time) we *slow* it, i. e., increase the time required to traverse it by δ.

Step 2 (flow-based techniques): Our new target now is a fixed-parameter algorithm for MinReachSlow. Since we do not aim to preserve a specific temporal graph class, we simplify the input by replacing S with a single-source s. Then we transform the temporal graph $\mathcal{G}$ into a (non-temporal) directed graph D in which the deletion of an edge corresponds to slowing a temporal edge in $\mathcal{G}$. Using this, we derive a max-flow-based polynomial-time algorithm which checks whether the source s can be prevented from reaching any vertices outside of a given set R by slowing at most k time-edges in $\mathcal{G}$.

Step 3 (resulting search-tree): We are aiming for a search-tree algorithm for MinReachSlow. Let R be a set of vertices and suppose that our max-flow-based algorithm failed to prevent s from reaching any vertices outside of R. Now, if there exists a solution for the given instance of MinReachSlow, then we can identify less than $|R|$ vertices such that at least one of them will be always reached from s. We can then try adding each of them to R, gradually building a search-tree to find the solution.

Henceforth the details follow. Instead of solving MinReachDelay directly, we reduce it to an auxiliary problem introduced next. Let $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$ be a temporal graph. *Slowing* a time-edge $(\{v, w\}, t)$ by δ refers to increasing $\gamma((\{v, w\}, t))$ by δ. We define $\mathcal{G} \uparrow_\delta X := (V, (E_i)_{i\in[\tau]}, \gamma')$ where $\gamma'(e) := \gamma(e) + \delta \cdot [e \in X]$. Our auxiliary problem is the following.

Minimizing Temporal Reachablity by Slowing (MinReachSlow)

Input: A temporal graph $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$, a set of sources $S \subseteq V$, and integers k, r, δ.
Question: Is there a time-edge set $X \subseteq \mathcal{E}(\mathcal{G})$ of size at most k such that $|R_{\mathcal{G}\uparrow_\delta X}(S)| \leq r$?

By the following, solving an instance of MinReachSlow also solves MinReachDelay.

▶ **Lemma 4.** *An instance $I = (\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma), S, k, r, \delta)$ of MinReachDelay is a* yes*-instance if and only if $J = (\mathcal{G}, S, k, r, \delta)$ is a* yes*-instance of MinReachSlow.*

Proof.

(⇒): Let $X \subseteq \mathcal{E}(\mathcal{G})$ be a solution for I. Note that for every temporal v-w-path P in $\mathcal{G} \uparrow_\delta X$, we can construct a temporal v-w-path P' in $\mathcal{G} \nearrow_\delta X$ by replacing a time-edge $(e, t) \in P$ with $(e, t + \delta)$ whenever $(e, t) \in X$. Hence, the reachable set of $s \in S$ in $\mathcal{G} \nearrow_\delta X$ is a superset of the reachable set of s in $\mathcal{G} \uparrow_\delta X$. Thus, J is a *yes*-instance.

(⇐): Let $X \subseteq \mathcal{E}(\mathcal{G})$ be an inclusion-minimal solution for J. We claim that every vertex reachable from S in $\mathcal{G} \nearrow_\delta X$ by some time t is also reachable from S in $\mathcal{G} \uparrow_\delta X$ until time t. Suppose for contradiction that the claim does not hold true for some vertex z and let t be the time S reaches z in $\mathcal{G} \nearrow_\delta X$. We may assume z to be chosen to minimize t. Clearly $t > 0$, i. e., $z \notin S$. Let P be a temporal s-z-path in $\mathcal{G} \nearrow_\delta X$ with arrival time t and $s \in S$. Let u be the penultimate vertex of P and $(\{u, z\}, t')$ the last time-edge of P. By minimality of t, u must be reachable from S by time t' also in $\mathcal{G} \uparrow_\delta X$. Since all time-edges of $\mathcal{E}(\mathcal{G}) \setminus X$ appear in $\mathcal{G} \nearrow_\delta X$ and $\mathcal{G} \uparrow_\delta X$ with identical traversal times, the last time-edge $(\{u, z\}, t')$ of P must be in $\mathcal{E}(\mathcal{G} \nearrow_\delta X) \setminus (\mathcal{E}(\mathcal{G}) \setminus X)$. Thus $(\{u, z\}, t' - \delta) \in X$. By minimality of X, there must be a source $s' \in S$ and a temporal s'-u-path P' in $\mathcal{G} \uparrow_\delta X$ reaching either u or z at time $t' - \delta$. If P' reaches z, then this is clearly a contradiction. But if P' reaches u, then appending $(\{u, z\}, t - \delta)$ to P' produces a temporal s'-z-path in $\mathcal{G} \uparrow_\delta X$ arriving at time t, thus also a contradiction. ◀

In the reminder of this section, we show that MinReachSlow is fixed-parameter tractable, when parameterized by r. Formally, we aim for the following theorem, which in turn clearly implies Theorem 3 by the means of Lemma 4.

▶ **Theorem 5.** *MinReachSlow can be solved in $O(r! \cdot k \cdot |\mathcal{G}|)$ time.*

The remainder of this section is dedicated to proving Theorem 5. The advantage of considering MinReachSlow instead of MinReachDelay is that we do not have to deal with new time-edges appearing due to the delay operation. This allows us to translate the reachability of a temporal graph to a (non-temporal) directed graph specially tailored to MinReachSlow.

In particular, the removal of some edges in the directed graph corresponds to slowing the corresponding time-edges by δ in the temporal graph. Before giving the details of the construction, we first reduce to the case where S is a singleton.

▶ **Lemma 6.** *Given an instance $I = (\mathcal{G} = (V, (E_i)_{i\in[\tau]}), S, k, r, \delta)$ of* MinReachSlow*, we can construct in linear time a instance $J = (\mathcal{G}', \{s\}, k, r+1, \delta)$ of* MinReachSlow *such that I is a* yes*-instance if and only if J is a* yes*-instance.*

Proof. We set $\mathcal{G}' := (V \cup \{s\}, (E'_i)_{i\in[\tau+\delta+1]})$ where s is a new vertex, $E'_1 := \{\{s, s'\} \mid s' \in S\}$, $E'_i := \emptyset$ for all $i \in [\delta+1] \setminus \{1\}$, and $E'_{i+\delta+1} := E_i$ for all $i \in [\tau]$. Observe that slowing an edge in E'_1 has no effect. Thus, I is a *yes*-instance if and only if J is a *yes*-instance. Clearly, J can be computed in linear time. ◀

A *network* (D, c) consists of a directed graph $D = (V, A)$ and *edge capacities* $c\colon A \to \mathbb{N}_0 \cup \{\infty\}$. A function $f\colon A \to \mathbb{N} \cup \{0\}$ is an *s-z-flow* for two distinct vertices $s, z \in V$ if

- $\forall e \in A\colon f(e) \leq c(e)$ and
- $\forall v \in V \setminus \{s, z\}\colon \sum_{(u,v)\in A} f((u,v)) = \sum_{(v,u)\in A} f((v,u))$.

The *value* of f is denoted by $|f| := \sum_{(s,v)\in A} f((s,v))$. An arc set $C \subseteq A$ is an *s-z-cut* of a network $((V, A), c)$ if $s, z \in V$ and there is no s-z-path in $(V, A \setminus C)$. The *capacity* of the s-z-cut C is $c(C) := \sum_{e\in C} c(e)$.

Let $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$ be a temporal graph. We define the *temporal neighborhood* of a vertex $v \in V$ at time point $t \in [\tau]$ as the set $N_{\mathcal{G}}(v, t) := \bigcup_{i=t}^{\tau} N_{(V,E_i)}(v)$ containing all neighbors of v in the layers t through τ.

For any $s \in R \subseteq V$ and $\delta \in \mathbb{N}$, we define the flow network $\mathcal{F}(\mathcal{G}, s, R, \delta) := (D, c)$ where $D = (V', A)$ is the directed graph defined by

$$V' := \{s^0, \mathrm{z}\} \cup \left\{ \begin{array}{l} e_1, e_2, v^t, w^t, v^{t+\gamma(e)}, w^{t+\gamma(e)}, \\ v^{t+\gamma(e)+\delta}, w^{t+\gamma(e)+\delta} \end{array} \;\middle|\; v, w \in R \text{ and } e = (\{v, w\}, t) \in \mathcal{E}(\mathcal{G}) \right\}$$

and

$$A := \left\{ (v^t, v^{t'}) \;\middle|\; v^t \in V', t' = \min\{i \mid i > t \text{ and } v^i \in V'\} \neq \infty \right\} \tag{1}$$

$$\cup \left\{ \begin{array}{l} (v^t, e_1), (w^t, e_1), (e_1, e_2), \\ (e_2, v^{t+\gamma(e)}), (e_2, w^{t+\gamma(e)}), \\ (v^t, w^{t+\gamma(e)+\delta}), (w^t, v^{t+\gamma(e)+\delta}) \end{array} \;\middle|\; v, w \in R \text{ and } e = (\{v, w\}, t) \in \mathcal{E}(\mathcal{G}) \right\} \tag{2}$$

$$\cup \left\{ (v^t, \mathrm{z}) \mid v \in R, t = \max\{i \mid N_{\mathcal{G}}(v, i) \not\subseteq R\} \neq -\infty \right\}. \tag{3}$$

and we set $c((e_1, e_2)) = 1$ for all $e \in \mathcal{E}(\mathcal{G})$ and $c(a) = \infty$ for all other $a \in A$. Consider Figure 1 for an illustration.

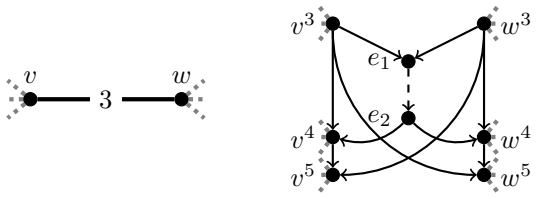

Figure 1 *Left*: An excerpt of a temporal graph $\mathcal{G}$ containing the time-edge $e = (\{v, w\}, 3)$ and $\gamma(e) = 1$. *Right*: An excerpt of the flow network $\mathcal{F}(\mathcal{G}, s, R, \delta = 1)$ showing the corresponding part for e, where solid arc have capacity ∞ and the dashed arc has capacity 1.

▶ **Lemma 7.** *For any given $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$, $s \in R \subseteq V$, and $k, \delta \in \mathbb{N}$, one can test in $O(k \cdot |\mathcal{G}|)$ time whether $\mathcal{F}(\mathcal{G}, s, R, \delta)$ has a s^0-z-flow of value at least $k+1$ and compute such a flow or a maximum flow otherwise.*

Proof. Note, that the flow network $\mathcal{F}(\mathcal{G}, s, R, \delta)$ can be computed in $O(|\mathcal{G}|)$ time by iterating over $\mathcal{E}(\mathcal{G})$ first forward and then backwards once. Then we can compute a flow of value $k+1$ or of maximum value, whichever is smaller, by running at most $k+1$ rounds of the Ford-Fulkerson algorithm [29]. This gives a overall running time of $O(k \cdot |\mathcal{G}|)$ time. ◀

▶ **Lemma 8.** *Let $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$, $s \in R \subseteq V$, $\delta \in \mathbb{N}$, and $((V', A), c) := \mathcal{F}(\mathcal{G}, s, R, \delta)$. Let $X \subseteq \mathcal{E}(\mathcal{G})$ and $C := \{(e_1, e_2) \in A \mid e \in X\}$. For any $x^t \in V'$, there is a s^0-x^t-path in $(V', A \setminus C)$ if and only if $\mathcal{G} \uparrow_\delta X$ contains a temporal s-x-path with arrival time at most t.*

Proof. Let γ' be the traversal time function of $\mathcal{G} \uparrow_\delta X$, i. e., $\gamma'(e) := \gamma(e) + \delta \cdot [e \in X]$.

(⇐): Let P be a temporal s-x-path in $\mathcal{G} \uparrow_\delta X$ for some vertex $x \in V$ and let t be the arrival time of P. Then we construct a s^0-x^t-path $\hat{P}$ in $(V', A \setminus C)$ as follows. Start with $\hat{P}$ being just the vertex s^0 and perform the following two steps for every time-edge $e = (\{v, w\}, b)$ of P in order.

1. Note that the currently last vertex of $\hat{P}$ is v^c for some $c \leq b$. As long as $c < b$, append to $\hat{P}$ the arc (v^c, v^d) where $d > c$ is chosen minimal (cf. (1)). Afterwards, the currently last vertex of $\hat{P}$ is v^b.
2. If $e \notin X$, i. e., $(e_1, e_2) \notin C$, then append to $\hat{P}$ the arcs (v^b, e_1), (e_1, e_2), and $(e_2, w^{b+\gamma(e)})$ (cf. (2)). Otherwise, if $e \in X$, i. e., $\gamma'(e) = \gamma(e) + \delta$, then append to $\hat{P}$ the arc $(v^b, w^{b+\gamma(e)+\delta})$. Note that in both cases the new last vertex of $\hat{P}$ is $w^{b+\gamma'(e)}$.

(⇒): Let P be a s^0-x^t-path in $(V', A \setminus C)$. Then we construct a s-x-path P' with arrival time at most t in $\mathcal{G} \uparrow_\delta X$ as follows. Start with P' being just the vertex s (with arrival time 0) and repeat the following steps until all arcs of P have been processed.

1. Let b be the arrival time of P' and v the last vertex. Note that the last vertex of P is v^c for some $c \geq b$. Ignore all arcs of P up to the last arc containing v^c for some $c \geq b$.
2. If the next three arcs in P are $(v^c, e_1), (e_1, e_2), (e_2, w^{c+\gamma(e)})$ for some time-edge $e = (\{v, w\}, c) \in \mathcal{E}(\mathcal{G})$, then append to P' that time-edge e. Note that, by assumption, $(e_1, e_2) \notin C$, thus $e \notin X$, and thus the arrival time of e is $c + \gamma'(e) = c + \gamma(e)$.
3. Otherwise the next arc in P must be $(v^c, w^{c+\gamma(e)+\delta})$ for some time-edge $e = (\{v, w\}, c) \in \mathcal{E}(\mathcal{G})$. Then append to P' that time-edge e. Note that the arrival time of e is $c + \gamma'(e) \leq c + \gamma(e) + \delta$. ◀

We now show that we can use Lemma 7 to check whether s can be prevented from reaching any vertices outside of R by slowing at most k time-edges by δ each.

▶ **Lemma 9.** *For any given $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$, $s \in R \subseteq V$, $\delta \in \mathbb{N}$, and $k \in \mathbb{N} \cup \{0\}$, the maximum s^0-z-flow in $\mathcal{F}(\mathcal{G}, s, R, \delta)$ has value at most k if and only if there is a set X of at most k time-edges such that s can not reach any vertices outside of R in $\mathcal{G} \uparrow_\delta X$.*

Proof. Write $((V', A), c) := F := \mathcal{F}(\mathcal{G}, s, R, \delta)$.

(⇒): Let the maximum s^0-z-flow f in F have value at most k. Moreover, let C be a s^0-z-cut of minimum capacity. From the max-flow min-cut theorem [30], we know that $c(C) \leq k$. Note that $C \subseteq \{(e_1, e_2) \in A \mid e \in \mathcal{E}(\mathcal{G})\}$, since all other edges have infinite capacity. Hence, $|C| \leq k$. Now set $X := \{e \in \mathcal{E}(\mathcal{G}) \mid (e_1, e_2) \in C\}$. Assume towards a contradiction that there

is a temporal s-x-path P in $\mathcal{G} \uparrow_\delta X$ for some $x \in V \setminus R$. We may take P to be minimal, thus the penultimate vertex y of P is contained in R. By Lemma 8 there is a s^0-y^t-path $\hat{P}$ in $(V', A \setminus C)$ where t is the time P reaches y. The fact that P afterwards proceeds to x and (3) in the definition of $\mathcal{F}$ imply that $(V', A \setminus C)$ contains a path from y^t to z. This contradicts C being a s^0-z-cut in (V', A).

($\Leftarrow$): Let X be a time-edge set as assumed. By assumption $R_{\mathcal{G}\uparrow_\delta X}(\{s\}) \subseteq R$. We claim that $C = \{(e_1, e_2) \mid e \in X\} \subseteq A$ is a s^0-z-cut in (V', A), which implies that the maximum value of an s^0-z-flow in F is at most $c(C) \leq k$ [30]. So suppose towards a contradiction that there is a s^0-z-path P in $(V', A \setminus C)$. Let $x^t \in V$ be the penultimate vertex of P. Then there is a s-x-path P' in $\mathcal{G} \uparrow_\delta X$ with arrival time at most t by Lemma 8. The final arc of P is (x^t, z). Hence, (x^t, z) must be contained in (3), i. e., we can extend P' by some time-edge to end at a vertex in $V \setminus R$. This contradicts our assumption $R_{\mathcal{G}\uparrow_\delta X}(\{s\}) \subseteq R$. ◀

If $\mathcal{F}(\mathcal{G}, s, R, \delta)$ contains a s^0-z-flow of value $k+1$, then we want to find a small set $Y \subseteq V(\mathcal{G}) \setminus R$ of vertices such that $Y \cap R_{\mathcal{G}\uparrow_\delta X}(s) \neq \emptyset$ for every $X \subseteq \mathcal{E}(\mathcal{G})$ with $|X| \leq k$ and $|R_{\mathcal{G}\uparrow_\delta X}(s)| \leq r$.

▶ **Lemma 10.** *Let $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$, $s \in R \subseteq V$, $\delta, r \in \mathbb{N}$, and $k \in \mathbb{N} \cup \{0\}$. Assume that $\mathcal{F}(\mathcal{G}, s, R, \delta)$ has a s^0-z-flow of value $k+1$. We can compute in $O(k \cdot |A|)$ time a set $Y \subseteq V \setminus R$ of size at most $|R|$ such that $Y \cap R_{\mathcal{G}\uparrow_\delta X}(s) \neq \emptyset$ holds for every $X \subseteq \mathcal{E}(\mathcal{G})$ with $|X| \leq k$ and $|R_{\mathcal{G}\uparrow_\delta X}(s)| \leq r$.*

Proof. Let f be a s^0-z-flow of value $k+1$ in $((V', A), c) := \mathcal{F}(\mathcal{G}, s, R, \delta)$. We may assume f to never use an arc $(v^t, w^{t'})$ whenever A contains some arc (v^b, z) with $b \geq t$, as we could otherwise redirect f to use that latter arc (of infinite capacity) instead. Note that performing this modification can be done in $O(k \cdot |A|)$ time. Now set

$$H := \left\{(v, t) \;\middle|\; v^t \in V' \text{ and } (v^t, \mathrm{z}) \in A, f((v^t, \mathrm{z})) > 0\right\}$$

By (3), we have $|H| \leq |R|$ and $N_{\mathcal{G}}(v, t) \setminus R \neq \emptyset$ for all $(v, t) \in H$. Construct the vertex set Y by picking for each $(v, t) \in H$ one arbitrary vertex from $N_{\mathcal{G}}(v, t) \setminus R$. Hence $|Y| \leq |H| \leq |R|$ and $Y \cap R = \emptyset$.

It remains to prove that $R_{\mathcal{G}\uparrow_\delta X}(s) \cap Y \neq \emptyset$, with $X \subseteq \mathcal{E}(\mathcal{G})$ being an arbitrary solution for the MinReachSlow-instance $(\mathcal{G}, \{s\}, k, r, \delta)$. Define $C := \{(e_1, e_2) \in A \mid e \in X\}$. Since f has value $k+1 > c(C)$, there is a s^0-v^t-path P in $(V', A \setminus C)$ where each edge $e \in E(P)$ has $f(e) > 0$ and $(v, t) \in H$. By Lemma 8, there is a temporal s-v-path P' in $\mathcal{G} \uparrow_\delta X$ with arrival time at most t. Note that through P', vertex s can reach v as well as all vertices of $N_{\mathcal{G}\uparrow_\delta X}(v, t) = N_{\mathcal{G}}(v, t)$. Hence, the vertex $u \in Y$ which we picked for $(v, t) \in H$ from $N_{\mathcal{G}}(v, t) \setminus R$ is in $R_{\mathcal{G}\uparrow_\delta X}(s)$ – thus the claim is proven. ◀

Now we are ready to prove of Theorem 5. The corresponding search-tree algorithm is listed as Algorithm 11.

Proof of Theorem 5. Let $I = (\mathcal{G}, S, k, r, \delta)$ be the given instance of MinReachSlow. By Lemma 6, we can assume $S = \{s\}$ after linear time preprocessing. We now prove that Algorithm 11 solves $(\mathcal{G}, \{s\}, k, r, \delta)$ in $O(r! \cdot k \cdot |\mathcal{G}|)$ time.

Let I be a *no*-instance. Observe that line 5 ensures that at all times $|R| \leq r$. Then, by Lemma 9, $\mathcal{F}(\mathcal{G}, s, R, \delta)$ will for all $s \in R \subseteq V$ have a s^0-z-flow of value $k+1$. Hence, line 4, and thus Algorithm 11 will never return *yes*.

Algorithm 11 Pseudocode of the algorithm behind Theorem 5.

Input: An instance $I = (\mathcal{G}, \{s\}, k, r, \delta)$ of MinReachSlow.
Output: *yes* if I is a *yes*-instance and otherwise *no*.

```
1 return g({s}), where
2 function g(R) is
3     Compute a s^0-z-flow f in F(G, s, R, δ) by Lemma 7
4     if f is of value at most k then return yes
5     if |R| ≥ r then return no
6     Compute a set Y ⊆ V \ R by Lemma 10
7     foreach v ∈ Y do
8         if g(R ∪ {v}) = yes then return yes
9     return no
```

Let I be a *yes*-instance. Thus there is a set X of at most k time-edges such that $|R_{\mathcal{G}\uparrow_\delta X}(s)| \leq r$. We claim that $g(R')$ returns *yes* for all R' with $s \in R' \subseteq R_{\mathcal{G}\uparrow_\delta X}(s)$. We prove this by reverse induction on $|R'|$. In the base case where $R = R_{\mathcal{G}\uparrow_\delta X}(s)$, $g(R)$ returns *yes* by Lemma 9. Now assume the claim to hold whenever $|R| = q$ and let R be of size $q-1$ with $s \in R \subseteq R_{\mathcal{G}\uparrow_\delta X}(s)$. Assume that $\mathcal{F}(\mathcal{G}, s, R, k, \delta)$ has a s^0-z-flow of value $k+1$, otherwise we are done (by line 4). By Lemma 10, the set Y computed in line 6 contains a vertex $u \in R_{\mathcal{G}\uparrow_\delta X}(s) \setminus R$. Thus, $g(R \cup \{u\})$ returns *yes* by induction hypothesis. Hence, by line 8, $g(R)$ return *yes*, completing the induction. In particular, $g(\{s\})$ returns *yes*, therefore Algorithm 11 is correct.

To bound the running time, note that each call $g(R)$ makes at most $|Y| \leq |R|$ recursive calls by Lemma 10. In each of these recursive calls the cardinality of R increases by one until $|R| = r$, so the recursion depth is at most r by line 5. Hence, we can observe by an inductive argument that the search tree has $d!$ many nodes at depth d, where the root is at depth 1. Hence, the search tree has at most $r!$ many leaves and thus $O(r!)$ many nodes in total. By Lemma 7 and Lemma 10, lines 3 and 6 take at most $O(k \cdot |\mathcal{G}|)$ time. Hence, the overall running time is bounded by $O(r! \cdot k \cdot |\mathcal{G}|)$. ◀

4 A Polynomial-Time Algorithm for Forests

In this section we present an algorithm that solves MinReachDelete and MinReachDelay in polynomial time on temporal graphs where the underlying graph is a tree or a forest. This is a quite severe yet well-motivated restriction of the input [20, 22, 21], since it could serve as the starting point for FPT-algorithms for "distance-to-forest"-parameterizations.

▶ **Theorem 12.** *MinReachDelete and MinReachDelay are polynomial-time solvable if the underlying graph is a forest.*

Actually we even provide an polynomial-time algorithm for a generalized version of MinReachDelay. Then, polynomial-time solvability of MinReachDelete follows from Lemma 1, since it is forest preserving. We define the generalized problem as follows:

Weighted MinReachDelay on Forests

Input: A temporal graph $\mathcal{G} = (V, (E_i)_{i\in[\tau]}, \gamma)$ whose underlying graph is a forest, a weight function $w\colon V \to \mathbb{N} \cup \{0, \infty\}$, a set $F \subseteq \mathcal{E}(\mathcal{G})$ of undelayable time-edges, a set of sources $S \subseteq V$, and integers k, r, δ.

Question: Does there exists a time-edge set $X \subseteq \mathcal{E}(\mathcal{G}) \setminus F$ of size at most k such that $w(R_{\mathcal{G}\nearrow_\delta X}(S)) \leq r$?

In the remainder of this section, we show how to solve this problem using dynamic programming in polynomial time. Informally speaking, our dynamic program works as follows. As a preprocessing step we unfold vertices of large degree, reducing to an equivalent instance of maximum degree 3. Then we root each underlying tree at an arbitrary vertex and build a dynamic programming table, starting at the leaves. More precisely, we compute a table entry for each combination of a vertex v, a budget k, a time step t, and a flag indicating whether v is first reached from a child or from its parent. This table entry then contains a minimum reachable subset of the subtree rooted at v that can be achieved by applying k delay operations to that subtree.

▶ **Theorem 13.** *Weighted MinReachDelay on Forests is polynomial-time solvable if the underlying graph is a forest.*

Proof. Assume for now that the underlying graph of $\mathcal{G}$ is a tree, rooted at an arbitrary leaf. We denote by T_v the subtree with root $v \in V$. We use the reaching time ∞ to denote "never". By convention, a vertex can reach itself at time 0. Define $\mathbb{N}^* := \mathbb{N} \cup \{0, \infty\}$.

We first show how to transform the underlying graph into a binary tree. This will highly simplify the description of the dynamic programming table. Replace each vertex v of degree $\deg(v) > 3$ by a path on $\deg(v) - 2$ new vertices, where each edge of that path is undelayable, appears at each time step and always has traversal time 0. Distribute the edges formerly incident to v among the new vertices such that each of them has degree 3. Set the weight of the path's first vertex to the weight of v, and the weight of all other path vertices to 0. Note that this modification produces an equivalent instance of maximum degree at most 3, while increasing the number of vertices only by a constant factor.

We extend the notion of reachability to vertex-time pairs $(s, t) \in S \times [\tau]$ by saying that (s, t) reaches $v \in V$ in $\mathcal{G}$ if there exists a temporal s-v-path starting at time t or later. For a set $A \subseteq V \times [\tau]$, $R_{\mathcal{G}}(A)$ is the set of all vertices, reachable from any member of A in $\mathcal{G}$. We say a vertex v is reached *through* another vertex w if there is a temporal path from a source $s \in S$ to v that uses w.

Let $v \in V$, $k \in \mathbb{N}$. Define $\mathcal{T}_{v,k}$ as the set of temporal graphs obtained from T_v by applying up to k delay operations. Partition $\mathcal{T}_{v,k}$ into $\{\mathcal{T}_{v,k,t} \mid t \in \mathbb{N}^*\}$, where $\mathcal{T}_{v,k,t}$ contains those graphs in which v is reached from $S \cap V(T_v)$ exactly at time t. Finally, we set for each $t \in \mathbb{N}^*$

$$D[v, k, t, \texttt{false}] = \min\left\{w\big(R_T(S \cap V(T_v))\big) \;\middle|\; T \in \mathcal{T}_{v,k,t}\right\} \quad \text{and}$$

$$D[v, k, t, \texttt{true}] = \min\left\{w\big(R_T(S \cap V(T_v)) \cup R_T(\{(v, t)\})\big) \;\middle|\; T \in \bigcup_{t' \geq t} \mathcal{T}_{v,k,t'}\right\}.$$

where the minimum of an empty set is ∞ by convention. It is convenient to also define these entries as ∞ whenever $k < 0$. Roughly speaking, $D[v, k, t, \iota]$ contains the minimal weight reached in T_v under the assumption that up to k delay operations are applied to T_v, that v is first reached at time t, and that

- v is reached by a source in $S \cap V(T_v)$ at time t if $\iota = \texttt{false}$,
- v is reached by a source in $S \setminus V(T_v)$ at time t if $\iota = \texttt{true}$.

Note that v might be reached simultaneously from $S \cap V(T_v)$ and $S \setminus V(T_v)$.

We next show how to compute $D[v,k,t,\iota]$ recursively, starting at the leaf vertices. Observe that $D[v,k,t,\iota] = \infty$ whenever $v \in S$ is a source and $t > 0$. Thus, this case shall be excluded in the following.

If v has no children. If $\iota = \texttt{false}$ and $v \notin S$ and $t < \infty$, then $D[v,k,t,\texttt{false}] = \infty$ as there is no way that v can be reached from a source in $S \cap V(T_v) = \emptyset$. Otherwise,

$$D[v,k,t,\iota] = w(v) \cdot [t < \infty] \,.$$

If v has exactly one child v'. If $\iota = \texttt{false}$ and $v \notin S$, then v must be reached through v' at time t. In this case the minimal total weight reached in $T_{v'}$ is

$$D_1 := \min_{t' \le t} D[v', k - \kappa(t',t), t', \texttt{false}],$$

where $\kappa(t',t)$ is the minimal number of delays that need to occur on the edge $\{v,v'\}$ to ensure that (v',t') reaches v at time t but not earlier. (Set $\kappa(t',t) = \infty$ if this is impossible.) Consequently, if $v \notin S$, then

$$D[v,k,t,\texttt{false}] = w(v) \cdot [t < \infty] + D_1 \,.$$

If $\iota = \texttt{true}$ or $v \in S$, then there are two possibilities. If v' is reached through v at time t', with t' being the first time v' is reached from S, then the minimal total weight reached in $T_{v'}$ is

$$D_2 := \min_{t' \ge t} D[v', k - \kappa(t,t'), t', \texttt{true}] \,.$$

Otherwise, v' must be reached from a source in $S \cap V(T_{v'})$ at time t', thus the minimal total weight reached in T'_v is

$$D_3 := \min_{t'} D[v', k - \hat{\kappa}(t',t), t', \texttt{false}],$$

where $\hat{\kappa}(t',t)$ is the minimal number of delays that need to occur on $\{v,v'\}$ to ensure that (v,t) cannot reach v' before time t' and (v',t') cannot reach v before time t. (Again, set $\hat{\kappa}(t',t) = \infty$ if this is impossible.) Thus we obtain for the case that $\iota = \texttt{true}$ or $v \in S$ that

$$D[v,k,t,\iota] = w(v) \cdot [t < \infty] + \min\{D_2, D_3\} \,.$$

If v has two children v', v''. The situation is similar to that of only one child vertex, although more possible cases have to be distinguished. We omit the tedious details. However, it is clear that $D[v,k,t,\iota]$ can be computed by simply trying all possible tuples $(t',t'',k',k'',i',i'',\iota',\iota'')$ where t',t'' are the times at which v',v'' are reached; k',k'' are the number of delays occurring in $T_{v'}, T_{v''}$; i',i'' are the number of delays occurring on the edges $\{v,v'\},\{v,v''\}$; and ι',ι'' describe whether v',v'' are reached from a source in their respective subtrees at time t' and t'', respectively. The number of such tuples and the time required to process each of them is clearly polynomial in $t + k + |\mathcal{G}|$.

After having computed all entries $D[v,k,t,\iota]$, the solution of the MinReachDelay instance $(\mathcal{G},k)$ can be found as the value of

$$R[\hat{v},k] := \min_t D[\hat{v},k,t,\texttt{false}],$$

where $\hat{v}$ is the root vertex of $\mathcal{G}$.

It remains to consider the case that the underlying graph of $\mathcal{G}$ is a disconnected forest. In this case simply apply the above algorithm to each connected component. Afterwards, determining the optimal way to split the overall budget between the connected components can be computed by a simple dynamic program. Define $X[i,k]$ as the minimum weight reached in the first i trees if up to k time-edges are delayed and use the fact that

$$X[1,k] = \min_{k' \leq k} (R[\hat{v}_1, k'])$$

and for all $i > 1$

$$X[i,k] = \min_{k' \leq k} (R[\hat{v}_i, k'] + X[i-1, k-k'])\,,$$

where $\hat{v}_j$ is the root of the j^{th} tree. ◀

5 Conclusion

While both problem variants, MinReachDelete and MinReachDelay, are polynomial-time solvable on forests and W[1]-hard when parameterized by k, even if the lifetime is $\tau = 2$, their complexities diverge when we parameterize by the number r of reachable vertices. Here, MinReachDelete is W[1]-hard while for MinReachDelay we found a fixed-parameter tractable algorithm. This makes MinReachDelay particularly interesting for applications where the number of reachable vertices should be very small, e.g. when trying to contain the spread of dangerous diseases.

On the practical side we want to point out that our algorithm for MinReachDelay parameterized by r uses only linear space, and its search-tree-based approach makes it fit for optimization techniques like further data reduction rules or pruning using lower bounds. Furthermore, our max-flow-based branching technique can be turned into a r-approximation for for minimizing the number r of reachable vertices by delaying k time-edges. To do so, instead of branching into all choices of $v \in Y$ in line 8 of Algorithm 11, simply invoke $g(R \cup Y)$. Refining the presented technique towards better approximation guarantees seems to be a promising research direction. Moreover, when focusing on specific applications, it is natural to exploit application-dependent graph properties towards designing more efficient algorithms. In particular: which well-motivated temporal graph classes beyond trees allow e.g. polynomial-time solvability of MinReachDelete or MinReachDelay? Finally, from the viewpoint of parameterized complexity the parameters k and r are settled, but the landscape of structural parameters is still waiting to be explored.

References

1 Infection prevention and control and preparedness for covid-19 in healthcare settings, 2020. URL: https://www.ecdc.europa.eu/en/publications-data/infection-prevention-and-control-and-preparedness-covid-19-healthcare-settings.

2 Eleni C. Akrida, George B. Mertzios, Sotiris E. Nikoletseas, Christoforos L. Raptopoulos, Paul G. Spirakis, and Viktor Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? *Journal of Computer and System Sciences*, 114:65–83, 2020. doi:10.1016/j.jcss.2020.05.005.

3 Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Christoforos L. Raptopoulos. The temporal explorer who returns to the base. *Journal of Computer and System Sciences*, 120:179–193, 2021. doi:10.1016/j.jcss.2021.04.001.

4 Kyriakos Axiotis and Dimitris Fotakis. On the size and the approximability of minimum temporally connected subgraphs. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 149:1–149:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.149.

5 Matthias Bentert, Anne-Sophie Himmel, André Nichterlein, and Rolf Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Applied Network Science*, 5(1):73, 2020. doi:10.1007/s41109-020-00311-0.

6 Kenneth A Berman. Vulnerability of scheduled networks and a generalization of menger's theorem. *Networks*, 28(3):125–134, 1996. doi:10.1002/(SICI)1097-0037(199610)28:3<125::AID-NET1>3.0.CO;2-P.

7 Hans L. Bodlaender and Tom C. van der Zanden. On exploring always-connected temporal graphs of small pathwidth. *Information Processing Letters*, 142:68–71, 2019. doi:10.1016/j.ipl.2018.10.016.

8 Dirk Brockmann and Dirk Helbing. The hidden geometry of complex, network-driven contagion phenomena. *Science*, 342(6164):1337–1342, 2013. doi:10.1126/science.1245200.

9 Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *Internation Journal of Foundations of Computer Science*, 14(02):267–285, 2003. doi:10.1142/S0129054103001728.

10 Sebastian Buß, Hendrik Molter, Rolf Niedermeier, and Maciej Rymar. Algorithmic aspects of temporal betweenness. In *Proceedings of the 26th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2084–2092, 2020. doi:10.1145/3394486.3403259.

11 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. In *Proceedings of the 31st International Symposium on Algorithms and Computation (ISAAC)*, volume 181, pages 30:1–30:18, 2020. To appear in Algorithmica. doi:10.4230/LIPIcs.ISAAC.2020.30.

12 Arnaud Casteigts, Joseph G. Peters, and Jason Schoeters. Temporal cliques admit sparse spanners. *Journal of Computer and System Sciences*, 121:1–17, 2021. doi:10.1016/j.jcss.2021.04.004.

13 Vittoria Colizza, Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences of the United States of America*, 103(7):2015–2020, 2006. doi:10.1073/pnas.0510525103.

14 Daryl J Daley and David G Kendall. Epidemics and rumours. *Nature*, 204(4963):1118–1118, 1964. doi:10.1038/2041118a0.

15 Argyrios Deligkas and Igor Potapov. Optimizing reachability sets in temporal graphs by delaying. In *Proceedings of the 34th Conference on Artificial Intelligence (AAAI)*, pages 9810–9817, 2020. doi:10.1609/aaai.v34i06.6533.

16 Reinhard Diestel. *Graph Theory*, volume 173. Springer, 5 edition, 2016. doi:10.1007/978-3-662-53622-3.

17 Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. doi:10.1007/BF01386390.

18 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

19 Ken TD Eames and Matt J Keeling. Contact tracing and disease control. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1533):2565–2571, 2003. doi:10.1098/rspb.2003.2554.

20 Jessica Enright and Kitty Meeks. Deleting edges to restrict the size of an epidemic: a new application for treewidth. *Algorithmica*, 80(6):1857–1889, 2018. doi:10.1007/s00453-017-0311-7.

21 Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021. doi:10.1016/j.jcss.2021.01.007.

22 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.

23 Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. *Journal of Computer and System Sciences*, 119:1–18, 2021. doi:10.1016/j.jcss.2021.01.005.

24 Thomas Erlebach, Frank Kammer, Kelin Luo, Andrej Sajenko, and Jakob T. Spooner. Two moves per time step make a difference. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 141:1–141:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.141.

25 Thomas Erlebach and Jakob T. Spooner. Faster exploration of degree-bounded temporal graphs. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 117, pages 36:1–36:13, 2018. doi:10.4230/LIPIcs.MFCS.2018.36.

26 Thomas Erlebach and Jakob T. Spooner. Non-strict temporal exploration. In *Proceedings of the 27th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 12156, pages 129–145, 2020. doi:10.1007/978-3-030-54921-3_8.

27 Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science*, 2020. doi:10.1126/science.abb6936.

28 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020. doi:10.1016/j.tcs.2019.03.031.

29 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

30 L. R. Ford, Jr. and D. R. Fulkerson. *Flows in networks*. Princeton University Press, 1962. doi:10.1515/9781400875184.

31 William Goffman and V Newill. Generalization of epidemic theory. *Nature*, 204(4955):225–228, 1964. doi:10.1038/204225a0.

32 Roman Haag, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Feedback edge sets in temporal graphs. In *Proceedings of the 46th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 12301, pages 200–212, 2020. doi:10.1007/978-3-030-60440-0_16.

33 John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973. doi:10.1145/362248.362272.

34 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. doi:10.1006/jcss.2002.1829.

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

36 Andrew Mitchell, David Bourn, J Mawdsley, William Wint, Richard Clifton-Hadley, and Marius Gilbert. Characteristics of cattle movements in britain–an analysis of records from the cattle tracing system. *Animal Science*, 80(3):265–273, 2005. doi:10.1079/ASC50020265.

37 Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200, 2001. doi:10.1103/PhysRevLett.86.3200.

38 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):1–24, 2008. doi:10.1145/1391289.1391291.

39 Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

40 Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016. doi:10.1109/TKDE.2016.2594065.

41 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020. doi:10.1016/j.jcss.2019.07.006.