Delay-Robust Routes in Temporal Graphs

Eugen Füchsle
Faculty IV, Algorithmics and Computational Complexity, TU Berlin, Germany

Hendrik Molter
Department of Industrial Engineering and Management,
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Rolf Niedermeier
Faculty IV, Algorithmics and Computational Complexity, TU Berlin, Germany

Malte Renken
Faculty IV, Algorithmics and Computational Complexity, TU Berlin, Germany

Abstract

Most transportation networks are inherently temporal: Connections (e.g. flights, train runs) are only available at certain, scheduled times. When transporting passengers or commodities, this fact must be considered for the the planning of itineraries. This has already led to several well-studied algorithmic problems on temporal graphs. The difficulty of the described task is increased by the fact that connections are often unreliable – in particular, many modes of transportation suffer from occasional delays. If these delays cause subsequent connections to be missed, the consequences can be severe. Thus, it is a vital problem to design itineraries that are *robust* to (small) delays. We initiate the study of this problem from a parameterized complexity perspective by proving its NP-completeness as well as several hardness and tractability results for natural parameterizations.

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

Keywords and phrases algorithms and complexity, parameterized complexity, time-varying networks, temporal paths, journeys

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.30

Related Version *Full Version*: `https://arxiv.org/abs/2201.05390` [17]

Funding *Hendrik Molter*: Supported by the ISF, grant No. 1070/20.
Malte Renken: Supported by the DFG, project MATE (NI 369/17).

1 Introduction

Finding a path between two vertices in a graph is one of the most fundamental problems in graph algorithms. In the rise in popularity of *temporal graphs* as a mathematical model [19, 20, 25, 24, 5], computing so-called *temporal paths* is one of the most important algorithmic problems in this area. Herein, a temporal graph is a graph whose edges are present only at certain, known points in time. For our purposes, it is specified by a set V of vertices and a set E of time arcs, where each time arc $(v, w, t, \lambda) \in E$ consists of a *start vertex* v, an *end vertex* w, a *time label* t, and a traversal time λ; this means that there is a (direct) connection from v to w starting at time t and arriving at time $t+\lambda$. Temporal graphs are prime models for many real-world networks: Social graphs, communication networks, and transportation networks are usually not static but vary over time.

The added dimension of time causes many aspects of connectivity to behave quite differently from static (i.e., non-temporal) graphs. In particular, the flow of goods or information through a temporal network has to respect time. More formally, it follows a *temporal walk* (or *path*, if every vertex is visited at most once), i.e., a sequence of time arcs

$(v_i, w_i, t_i, \lambda_i)_{i=1}^{\ell}$ where $v_{i+1} = w_i$ and $t_{i+1} \geq t_i + \lambda_i$ for all $i < \ell$. While inheriting many properties of their static counterparts, temporal walks exhibit certain characteristics that add a new level of complexity to algorithmic problems centered around them. For example, temporal connectivity is not transitive: the existence of a temporal walk from vertex u to w and a temporal walk from v to w does not imply the existence of a temporal walk from u to w. Furthermore, the temporal setting allows for several natural notions of an "optimal" temporal path [3].

As the finding of (optimal) temporal paths and walks constitutes the perhaps most important building block for (algorithmic) analysis of temporal networks, it has already been studied intensively [28, 3]. However, the temporal setting allows to model further natural constraints on temporal walks and paths that do not have a counterpart in the static setting. For example, recently the study of the computational complexity of finding temporal walks and paths that are subject to some waiting time constraints has been initiated [6, 1].

In this work, we investigate another very natural yet still unstudied temporal path variant, namely so-called *delay-robust* temporal paths. Real-world networks are often not perfect: Scheduled connections may be canceled or delayed. This immediately brings up the natural issue of robustness. To the best of our knowledge, this issue has so far only been analyzed with respect to cancellations [2], but not with respect to delays. We propose a model for delay-robust temporal paths and analyze natural structural and computational problems occurring in this context. Our main problem of interest is to determine whether there is a delay-robust temporal path between two vertices in a temporal graph.

Delay-Robust Route
Input: A temporal graph $\mathcal{G} = (V, E)$, two vertices $s, z \in V$ and $x, \delta \in \mathbb{N}$.
Question: Is there an x-delay-robust route from s to z in $\mathcal{G}$?

It remains to say how *delay-robustness* is understood. Although different notions are conceivable, we consider a sequence of vertices (called a *route*) in a temporal graph to be *x-delay-robust*, if there is a temporal path visiting the vertices in this sequence even if up to x time arcs are delayed by at most δ. We give a formal definition in Section 2.

This definition is motivated by the fact that changing the vertices may be costly for a number of reasons: storage or transhipment facilities may need to be newly allocated; if the new route passes through different jurisdictions, then new authorizations and documents have to be acquired; insurance policies might not cover alternative routes; the chosen packaging might no longer be adequate (e.g. when switching from rail to air transportation); or personnel might need to be moved. All these and many more issues are of much less concern when the chosen route can be kept and only the *schedule* has to be changed.

Related Work. Apart from the already mentioned work on finding temporal walks and paths, there has been extensive research on many other connectivity-related problems on temporal graphs [4, 14, 23]. Delays in temporal graphs have been considered as a modification operation to manipulate reachability sets [8, 26]. The individual delay operation considered in the mentioned work delays a single time arc and is similar to our notion of delays. The deletion of time arcs [26, 12, 11], the deletion of vertices [29, 16, 22], as well as reordering of time arcs [13] have also been considered as temporal graph modification operations to manipulate the connectivity properties of the temporal graph. The corresponding computational problems in all mentioned work are NP-hard and can be also considered as computing "robustness measures" for the connectivity in temporal graphs.

In companion work [18] we investigate the related problem where we ask whether two vertices remain connected even if up to x time arcs are delayed. Note that in this setting, the specific temporal path connecting the two vertices can visit different vertices for different delays. We show that this problem can be solved in polynomial time. We further investigate the problem variant where the delays occur dynamically during the "journey" from the start to the destination vertex. In this case the problem becomes PSPACE-complete if every vertex can be visited at most once and stays polynomial-time solvable, otherwise.

Our Contribution. We introduce the computational problem of finding routes that are robust under delays. We investigate its computational complexity with a focus on parameterized algorithms and hardness [10, 7].

We first give some structural results in Section 3, including that Delay-Robust Path is solvable in polynomial time if the underlying graph[1] is a forest. In Section 4, we show that Delay-Robust Path is NP-hard even if the underlying graph has constant bandwidth, which implies that it also has constant treewidth. We further show that Delay-Robust Path is W[1]-hard when parameterized by the combination of the feedback vertex number of the underlying graph and the number of delays. In Section 5, we present our general algorithmic results where we explore how the polynomial-time algorithm for underlying forests can be generalized. We give a polynomial-time algorithm for the case where we have a constant number of delays. We further give two FPT algorithms: one for the underlying feedback edge set number as a parameter and one for the combination of the so-called *timed* feedback vertex number [6] and the number of delays as a parameter.

Due to the lack of space, several proofs (marked with ($\star$)) had to be deferred to the full version [17].

2 Preliminaries

We abbreviate $\{1, 2, \ldots, n\}$ as $[n]$ and $\{n, n+1, \ldots, m\}$ as $[n, m]$. For any time arc $e = (v, w, t, \lambda_e)$, we denote the starting and ending vertices as $\mathrm{start}(e) = v$ an $\mathrm{end}(e) = w$, the time label as $\mathrm{t}(e) = t$, and the traversal time as $\lambda(e) = \lambda_e$. Furthermore, for any vertex v, τ_v^+ denotes the set of time steps where v has outgoing time arcs, and τ_v^- denotes the time steps with incoming time arcs. We set $\tau_v := \tau_v^+ \cup \tau_v^-$.

Given a temporal graph $\mathcal{G}$, we denote by T the maximum time label of all time arcs in $\mathcal{G}$. When removing all time information and directions from the time arcs of a temporal graph $\mathcal{G} = (V, E)$, the resulting (static & undirected) graph $G_{\mathrm{u}}(\mathcal{G}) = (V, E')$ with $E' = \{\{v, w\} \mid (v, w, t, \lambda) \in E\}$ is called the *underlying graph* of $\mathcal{G}$.

Delays. We distinguish two different types of delays. Both are applied to a single time arc e and delay it by a natural number δ. A *starting delay* increases the time label $\mathrm{t}(e)$ by δ while a *traversal delay* increases the traversal time $\lambda(e)$ by δ. In the example of a railway network, a starting delay would correspond to a delayed departure at a station whereas a traversal delay would describe a delay occurring on the way between two stations.

For a given set $D \subseteq E$ of *delayed arcs*, a sequence of time arcs $(v_i, w_i, t_i, \lambda_i)_{i=1}^{\ell}$ is called a D-starting-delayed temporal walk resp. a D-traversal-delayed temporal walk if it is a temporal walk in the temporal graph obtained from $\mathcal{G}$ by applying starting delays resp. traversal delays to all time arcs in D. (We omit D as well as the type of delay when they are clear from context.) Note that a traversal-delayed temporal walk is always also a temporal walk in $\mathcal{G}$, which is not necessarily true for a starting-delayed temporal walk.

[1] The *underlying graph* of a temporal graph is the undirected static graph obtained by connecting all vertices that are connected by a time arc.

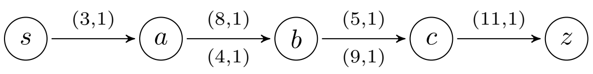


Figure 1 An example temporal graph where (s, a, b, c, z) is a traversal- and starting-delay-robust temporal route for $x = 1$ and $\delta = 3$. No matter which time arc is delayed, there is a temporal path through these vertices in that order. On the other hand, the route ceases to be delay-robust for $\delta \geq 5$, as delaying the first arc demonstrates.

As an example consider the following temporal walk with edges labeled by $(\mathrm{t}(e), \lambda(e))$:

$$\begin{tikzcd} a \arrow[r, "{(1,1)}"] & b \arrow[r, "{(3,1)}"] & c \end{tikzcd}$$

When delaying the first time arc by 1, i.e. when setting $\delta = 1$ and $D = \{(a, b, 1, 1)\}$, then this is also a starting-delayed as well as a traversal-delayed temporal walk: Due to the delay, the first time arc arrives in b at time step $2 + \delta = 3$ which is no later than the departure of the second time arc. However, if we instead set $\delta = 2$ and $D = \{(a, b, 1, 1), (b, c, 3, 1)\}$, then it is still a starting-delayed temporal walk but no longer a traversal-delayed temporal walk, because the first time arc only reaches b at time 4.

We say a sequence R of vertices forms a *(delayed) route* if there is a (delayed) temporal walk which *follows* R, that is, which visits exactly the vertices of R in the given order. Generally, a temporal walk or route from vertex s to vertex z is also called a *temporal* (s, z)*-walk* or (s, z)*-route*. A *(delayed) temporal path* is a (delayed) temporal walk where no vertex is visited twice.

Robustness. We say that a temporal route is *traversal-delay-robust* resp. *starting-delay-robust* for a given number x of delays if it is a D-traversal-delayed resp. D-starting-delayed temporal route for all delay sets D of size $|D| \leq x$. Of course, this also depends on the value of δ. An example can be seen in Figure 1.

We now have all the ingredients for the formal definition of our main problem, Delay-Robust Route, as given in Section 1. In this definition, we did not specify whether traversal- or starting-delay is used. The reason for that is that we will show in Section 3 that the distinction is meaningless because both problem variants are equivalent. In the meantime, however, we will refer to them as TD-Delay-Robust Route and SD-Delay-Robust Route.

3 Structural Results and Recognizing Robust Routes

In this section, we derive some important properties of delay-robust routes.

3.1 Structural Results

We begin by investigating the distinction between walks and paths. Clearly, from any temporal walk one can obtain a temporal path by eliminating all circular subwalks. This leads to the following lemma, which holds for traversal as well as starting delays, and for all delay sizes x and delay times δ and will come in handy later.

▶ **Lemma 1.** *Let s and z be two vertices. If there is a delay-robust (s, z)-route, then there is a delay-robust (s, z)-route without repeated vertices.*

Proof. If there is a delay-robust (s, z)-route $R = (v_i)_{i=1}^k$, then for each delay of size at most x there is a delayed temporal walk traversing $v_1, v_2, \dots, v_k$ in that order. Each of these delayed temporal walks can be turned into a delayed temporal walk by eliminating circular subwalks. All the delayed temporal paths obtained in this way follow the same sequence of vertices, making this sequence a delay-robust (s, z)-route without repeated vertices. ◀

By virtue of Lemma 1, we will subsequently assume routes to not contain repeated vertices.

Next, we turn towards proving the equivalence of SD-Delay-Robust Route and TD-Delay-Robust Route. We start with some important observations. The first one is that every traversal-delayed temporal walk is also a starting-delayed temporal walk.

▶ **Lemma 2.** *Let P be a traversal-delayed temporal walk for some delay set D and $\delta \in \mathbb{N}$. Then P is also a starting-delayed temporal walk for D and δ.*

Proof. Let $P = (v_i, w_i, t_i, \lambda_i)_{i=1}^{\ell}$. This means that

$$t_i + \lambda_i + [e_i \in D] \cdot \delta \leq t_{i+1}$$

for all $i \leq \ell - 1$, where $[e_i \in D] = \begin{cases} 1 & \text{if } e_i \in D \\ 0 & \text{otherwise} \end{cases}$ denotes the Iverson bracket. Thus

$$t_i + \lambda_i + [e_i \in D] \cdot \delta \leq t_{i+1} + [e_{i+1} \in D] \cdot \delta$$

which shows that P is a starting-delayed temporal walk. ◀

While the converse of Lemma 2 is generally not true, the following weaker statement holds.

▶ **Lemma 3.** *Let R be a route, $\delta \in \mathbb{N}$ and D a minimal delay set such that R is not a D-traversal-delayed route. Then R is not a D-starting-delayed route either.*

Proof. Suppose for contradiction that R was a D-starting-delayed route. Then there is a D-starting-delayed temporal walk $P = (e_i)_{i=1}^{\ell} = (v_i, w_i, t_i, \lambda_i)_{i=1}^{\ell}$ that follows R, i.e.,

$$t_i + \lambda_i + [e_i \in D] \cdot \delta \leq t_{i+1} + [e_{i+1} \in D] \cdot \delta$$

for all $i \leq \ell - 1$. Since R is not a traversal-delayed route, P is not a traversal-delayed temporal path. Thus there exists an index $j \leq \ell - 1$ with

$$t_j + \lambda_j + [e_j \in D] \cdot \delta > t_{j+1}$$

and we may assume j to be chosen maximally. This implies that

$$t_{j+1} < t_j + \lambda_j + [e_j \in D] \cdot \delta \leq t_{j+1} + [e_{j+1} \in D] \cdot \delta,$$

which in turn implies that $e_{j+1} \in D$. By maximality of j, $P' = (e_i)_{i=j+1}^{\ell}$ is a traversal-delayed temporal path. Thus, for any traversal-delayed temporal path $Q = (v_i, w_i, t'_i, \lambda'_i)_{i=1}^{j}$ following $(v_i)_{i=1}^{j+1}$, we must have $t'_j + \lambda'_j + [(v_j, w_j, t'_j, \lambda'_j) \in D] \cdot \delta > t_{j+1}$, for otherwise its concatenation with P' would contradict the fact that R is not a traversal-delayed route. Therefore, R is also not a D'-traversal-delayed temporal vertex walk, where $D' = D \setminus \{e_{j+1}\}$. This contradicts the minimality of D. ◀

Using Lemmas 2 and 3, we can now prove the following.

▶ **Theorem 4.** *TD-Delay-Robust Route = SD-Delay-Robust Route.*

Proof. Let $\mathcal{G} = (V, E)$ be a temporal graph, $s, z \in V$ be a start and an end vertex, and $x, \delta \in \mathbb{N}$. If $(\mathcal{G}, s, z, x, \delta)$ is a no-instance of SD-Delay-Robust Route, then for every (s, z)-route R, there exists a set D of $|D| \leq x$ time arcs such that there is no D-starting-delayed temporal path following R. By Lemma 2, there is then also no D-traversal-delayed temporal path following R, thus $(\mathcal{G}, s, z, x, \delta)$ is a no-instance of TD-Delay-Robust Route.

Conversely, if $(\mathcal{G}, s, z, x, \delta)$ is a no-instance of TD-Delay-Robust Route, then for every (s, z)-route R there exists a set D of $|D| \leq x$ time arcs such that R is no D-traversal-delayed route. We may assume D to be minimal to that respect. Then Lemma 3 gives us that R is no D-starting-delayed route, making $(\mathcal{G}, s, z, x, \delta)$ a no-instance of SD-Delay-Robust Route. ◀

Theorem 4 allows us now to drop the distinction between the two delay types and speak simply of Delay-Robust Route. For the remainder of this paper we will mostly work with traversal delays, for they are slightly easier to handle.

3.2 Recognizing Robust Routes

Since a route can be followed by an exponential number of different temporal walks, it is not immediately clear whether delay-robustness can be efficiently checked. The following theorem says that this is the case, and Delay-Robust Route is thus contained in NP.

▶ **Theorem 5** ($\star$). *For any given $x, \delta \in \mathbb{N}$, one can determine in $\mathcal{O}(nmx^2 + m \log m)$ time whether a given route R in a temporal graph $\mathcal{G}$ is x-delay-robust, where n is the number of vertices of R and m is the number of time arcs connecting consecutive vertices of R.*

Theorem 5 also gives us a polynomial-time algorithm to solve Delay-Robust Route on temporal graphs with underlying forest: As any vertex pair (s, z) is connected by at most one route, we only need to test the delay robustness of that route.

In the remainder of this section, we will prove Theorem 5. The basic idea is that a route is delay-robust for a worst-case delay if and only if it is delay-robust for all delays. This worst-case delay can be computed in polynomial time using a dynamic program.

First, we introduce the term *earliest arrival time* for a given route. A route $R = (v_i)_{i=1}^k$ requires that there is at least one temporal path following R. The *earliest arrival time* is then the arrival time of the temporal path that arrives earliest. Formally, we define the earliest arrival time as follows. Let $\mathcal{P} = \{P_i\}_{i=1}^{\ell}$ be the set of temporal paths following R with $P_i = (e_1^{(i)}, e_2^{(i)}, \ldots, e_{k-1}^{(i)})$. The *earliest arrival time* of R then is defined as the earliest arrival time of any temporal path in $\mathcal{P}$, i.e., as $\min\left\{t(e_{k-1}^{(i)}) + \lambda(e_{k-1}^{(i)}) \,\middle|\, i \leq \ell\right\}$. Analogously, if $R = (v_i)_{i=1}^k$ is a delayed route for the delay set $D \subseteq E$ and delay time $\delta \in \mathbb{N}$, and if $\mathcal{P}$ as above is the set of delayed temporal paths following R, then the *earliest delayed arrival time* of R is $\min\left\{t(e_{k-1}^{(i)}) + \lambda(e_{k-1}^{(i)}) + [e_{k-1}^{(i)} \in D] \cdot \delta \,\middle|\, i \leq \ell\right\}$.

We then define the *worst-case arrival time* of a route $R = (v_i)_{i=1}^k$ for a given delay size x and delay time δ as the maximum earliest delayed arrival time of R, taken over all delay sets D with $|D| \leq x$. (If R is not x-delay-robust, then we define the worst-case arrival time to be ∞.)

Now that we defined the worst-case arrival time, we show how to compute it. Let $R_j = (v_i)_{i=1}^j$ denote the prefix routes of R. The dynamic program computes table entries $A_{R_j}[y]$ iteratively for all $j \leq k$ and $y \leq x$, where $A_{R_j}[y]$ stores the worst-case arrival time of R_j for y delays.

We begin with the single-vertex route $R_1 = (v_1)$, setting $A_{R_1}[y] = 0$ for all y since the empty temporal path is always available to go from v_1 to v_1. Our goal is then to inductively compute A_{R_j} from $A_{R_{j-1}}$.

Consider the situation where we want to get from v to w in a single step, starting at time t. Then the set of available time arcs is $E(v, w, t) = \{(v, w, t', \lambda) \in E \mid t' \geq t\}$. Suppose $E(v, w, t) = \{a_i\}_{i=1}^{\ell}$ where $\mathrm{t}(a_i) + \lambda(a_i) \leq \mathrm{t}(a_{i+1}) + \lambda(a_{i+1})$ for all i. Now if up to y delays occur, then the latest time at which we will reach w is

$$\alpha(v, w, t, y) := \min\{\mathrm{t}(a_1) + \lambda(a_1) + \delta, \mathrm{t}(a_{y+1}) + \lambda(a_{y+1})\}.$$

Here, the worst case occurs if a_1 through a_y are all delayed.

Using this fact, we can now compute the table entries A_{R_i} from $A_{R_{i-1}}$ as follows.

$$A_{R_i}[y] = \max_{0 \leq y' \leq y} \{\alpha(v_{i-1}, v_i, A_{R_{i-1}}[y'], y - y')\}.$$

The idea here is that some number $y' \leq y$ of delays will occur between v_{i-1} and v_i, while the other $y - y'$ delays can occur somewhere along R_{i-1}.

The formal proof that $A_R[x]$ contains the solution to the Delay-Robust Route instance and that it can be computed in the specified time is deferred to the full version [17].

4 A Reduction Framework for Delay-Robust Route

In this section, we investigate the computational hardness of Delay-Robust Route with a particular attention to parameterized hardness with respect to "distance to forest" parameters. The goal is to lay out the ground for potential generalization of the algorithm presented in Section 3.2. We introduce a new problem Multi-Colored Monotone SAT in Section 4.1 and design a polynomial-time reduction to Delay-Robust Route. We will use this as an intermediate problem for reductions from 3-SAT and Multi-Colored Clique in Section 4.2 to show NP-hardness and parameterized hardness results.

4.1 Multi-Colored Monotone SAT

The problem Multi-Colored Monotone SAT is a Satisfiability variant where the variables are partitioned into "color classes" and only one variable from each color may be set to true. Furthermore, we do not make any assumptions on the Boolean formula other than that all variables appear non-negated. Formally, the we define the problem as follows.

Multi-Colored Monotone SAT (MCMSAT)

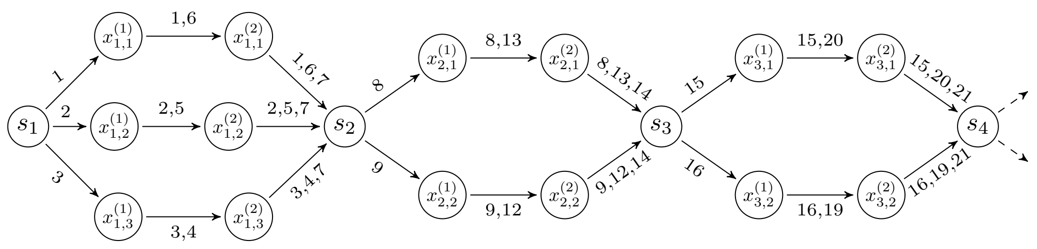
Input: Disjoint sets of variables $X_1, X_2, \ldots, X_n$ and a boolean formula Φ only consisting of positive literals and the operators $\wedge$ and $\vee$.

Question: Is there a satisfying truth assignment for Φ where exactly one variable from each X_i for $i \in [n]$ is true?

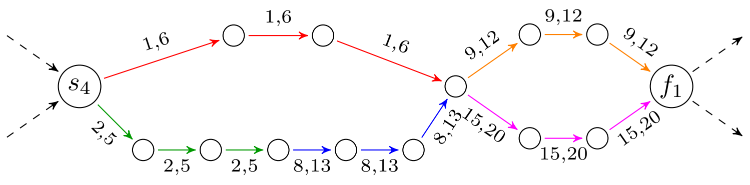
We have the following theorem.

▶ **Theorem 6** ($\star$). *MCMSAT $\leq_{\mathrm{m}}^{\mathrm{poly}}$ Delay-Robust Route.*

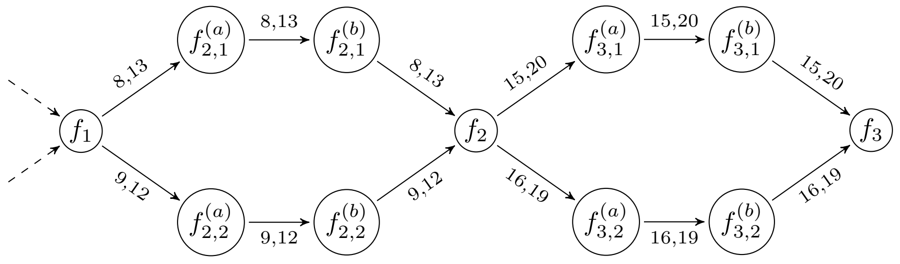
We describe the reduction behind Theorem 6 here, but defer most of the formal correctness proofs to the full version [17].



(a) Selection gadgets for the variable sets with $|X_1| = 3$ and $|X_2| = |X_3| = 2$.



(b) Validation gadgets for $\Phi = (x_{1,1} \vee (x_{1,2} \wedge x_{2,1})) \wedge (x_{2,2} \vee x_{3,1})$. The time arcs belonging to a literal are highlighted in the corresponding color.



(c) Finalization gadgets for $n = 3$.

Figure 2 An example temporal graph resulting from a MULTI-COLORED MONOTONE SAT reduction. Dummy time arcs are omitted. The instance has $n = 3$ disjoint variable sets.

Let $I = ((X_1, X_2, \ldots, X_n), \Phi)$ be an instance of MCMSAT. We will construct a temporal graph $\mathcal{G} = (V, E)$ and an instance $I' = (\mathcal{G}, s, z, \delta, x)$ of DELAY-ROBUST ROUTE so that I is a yes-instance of MCMSAT if and only if I' is a yes-instance of DELAY-ROBUST ROUTE. All time arcs in $\mathcal{G}$ have a traversal time of 0. Thus, we abbreviate time arcs as 3-tuples (v, w, t). In figures we omit the traversal time and label arcs only with their time step.

Let $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,|X_i|}\}$ for all $i \in [n]$. Furthermore, let $m := \max_i |X_i|$ denote the largest cardinality of a variable set X_i. The temporal graph consists of chained *selection gadgets* for each variable set X_i, a recursively constructed *validation gadget* and chained *finalization gadgets* for each variable set X_i. The selection gadgets are used to select the variable from X_i that is assigned to true, for each $i \in [n]$. Then the validation gadgets check whether the formula is satisfied under the selected truth assignment. If this is not the case, then a connection breaks at latest in the finalization gadgets and the target vertex can not be reached. The gadgets use an offset $o_i := (2m + 1) \cdot (i - 1)$. We set the delay time to $\delta = 1$ and the number of delays to $x = 2 \cdot n - 1$. Figure 2 shows examples for all gadget types.

Selection Gadgets. The selection gadgets are used to select one variable $x_{i,a}$ from each set X_i. For each set X_i, we add a vertex s_i to V and one additional vertex s_{n+1}. For each set X_i and each variable $x_{i,a} \in X_i$, we add the vertices $x_{i,a}^{(1)}$ and $x_{i,a}^{(2)}$ to V. Moreover, we add the following time arcs to E so that there is one route from s_i to s_{i+1} for this variable $x_{i,a}$:

$$s_i \xrightarrow{o_i+a} x_{i,a}^{(1)} \xrightarrow{o_i+a,o_{i+1}-a} x_{i,a}^{(2)} \xrightarrow{o_i+a,o_{i+1}-a,o_{i+1}} s_{i+1}$$

Taking this sub-route corresponds to setting the variable $x_{i,a}$ to true. Additionally, for each of the three underlying arcs we add a *dummy time arc* for each time step $t \in [o_{i+1}-1]$. If the sub-route $s_i \to x_{i,a}^{(1)} \to x_{i,a}^{(2)} \to s_{i+1}$ is chosen, then the worst-case arrival time from s_1 to s_{i+1} is $o_i + a$ for $2 \cdot (i-1)$ delays and $o_{i+1} - a = o_i + 2 \cdot m + 1 - a$ for $2 \cdot (i-1) + 1$ delays. Any sub-route is a Pareto optimum: while one arrives earlier for $2 \cdot (i-1)$ delays, another arrives earlier for $2 \cdot (i-1) + 1$ delays. An example for chained selection gadgets can be seen in Figure 2a.

Validation Gadgets. The validation gadgets are used to check whether the formula Φ is satisfied under the selected truth assignment. We will add a fresh vertex f_1 to V which is the start of the validation gadgets. The validation gadget for Φ will be constructed with s_{n+1} as a start vertex and f_1 as an end vertex. Given a start vertex v and an end vertex w, we can recursively construct the validation gadget for a formula Φ in the following way:

1. $\Phi = x_{i,a}$ is a single positive literal.
   We add two fresh vertices $\ell_{i,a}^{(1)}$ and $\ell_{i,a}^{(2)}$ to V. We add the following time arcs, so that there is a connection from v to w:

   $$v \xrightarrow{o_i+a,o_{i+1}-a} \ell_{i,a}^{(1)} \xrightarrow{o_i+a,o_{i+1}-a} \ell_{i,a}^{(2)} \xrightarrow{o_i+a,o_{i+1}-a} w$$

   Additionally for all three underlying arcs we add a *dummy time arc* for each time step $t \in [o_{n+1}-1] \setminus [o_i, o_{i+1}-1]$. We call this constructed part of the validation gadget a *literal gadget*. If the variable $x_{i,a}$ has been selected in the selection gadgets, then traversing this literal gadget does not affect the worst-case arrival time with respect to the number of delays. However, if $x_{i,a}$ has not been selected there is a delay that breaks the connection at latest in the finalization gadgets.
2. $\Phi = \Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_k$ is a conjunction of k sub-formulae.
   We add a fresh vertex c_i to V for all $i \in [k-1]$. Then the validation gadgets for all sub-formulae Φ_i are constructed, with c_{i-1} as the start and c_i as the end vertex, where $c_0 = v$ and $c_k = w$. Thus, the gadgets for the sub-formulae are connected in a row, and to traverse the temporal graph from v to w all gadgets for the sub-formulae have to be traversed.
3. $\Phi = \Phi_1 \vee \Phi_2 \vee \ldots \vee \Phi_k$ is a disjunction of k sub-formulae.
   We construct the validation gadgets for all sub-formulae Φ_i with v as the start and w as the end vertex. Thus, the gadgets for the sub-formulae are connected in parallel, and to traverse the temporal graph from v to w one gadget for a sub-formulae has to be traversed.

An example for a valid gadget can be seen in Figure 2b.

Finalization Gadgets. The finalization gadgets are similar to the selection gadgets for all sets X_2 to X_n. For each variable set X_i for $i \in [2, n]$ we add a vertex f_i to V. For each variable $x_{i,a} \in X_i$ we add the vertices $f_{i,a}^{(1)}$ and $f_{i,a}^{(2)}$ to V and add the following time arcs:

$$f_{i-1} \xrightarrow{o_i+a,o_{i+1}-a} f_{i,a}^{(1)} \xrightarrow{o_i+a,o_{i+1}-a} f_{i,a}^{(2)} \xrightarrow{o_i+a,o_{i+1}-a} f_i$$

Again for all three underlying arcs and each time step $t \in [o_{n+1} - 1] \setminus [o_i, o_{i+1} - 1]$ we add a dummy time arc. An example for finalization gadgets can be seen in Figure 2c.

The start and end vertices for our Delay-Robust Route-instance are s_1 and f_n, respectively.

We defer the proof that the constructed Delay-Robust Route instance is equivalent to the given Multi-Colored Monotone SAT instance to the full version [17].

4.2 Applications of the Framework

Next, we use our previous result that MCMSAT $\leq_{\mathrm{m}}^{\mathrm{poly}}$ Delay-Robust Route (Theorem 6) to show that Delay-Robust Route is NP-complete even if the underlying graph has bandwidth 3. The *bandwidth* $\mathrm{bw}(G)$ of a graph G is the smallest number b such that the vertices of G can be placed at distinct integer points along a line so that the length of the longest edge is b. The bandwidth of a graph upper-bounds both the graph's pathwidth and treewidth [27]. Formally, we show the following result by using an appropriate polynomial-time reduction from the NP-complete 3-SAT problem [21] to MCMSAT.

▶ **Theorem 7** ($\star$). *Delay-Robust Route is NP-complete for all fixed $\delta \geq 1$, maximum traversal times $\lambda_{\max} \geq 0$, and bandwidths of the underlying graph $\mathrm{bw}(G_{\mathrm{u}}(\mathcal{G})) \geq 3$.*

Next, we show W[1]-hardness of Delay-Robust Route for the feedback vertex set of the underlying graph, the length of a delay-robust temporal path, and the number of delays combined. To this end, we give a parameterized polynomial-time reduction from Multi-Colored Clique [15] to Delay-Robust Route. Again we use MCMSAT as an intermediate problem and use Theorem 6. Formally, we show the following result.

▶ **Theorem 8** ($\star$). *Delay-Robust Route is W[1]-hard with respect to $x + L + f$ where x is the number of delays, L is the length of a longest s-z path in $G_{\mathrm{u}}(\mathcal{G})$, and f is the feedback vertex number of $G_{\mathrm{u}}(\mathcal{G})$.*

The presented hardness results show that we presumably cannot generalize Theorem 5 to an FPT result for parameters such as the treewidth of the underlying graph or the feedback vertex number of the underlying graph.

5 Parameterized Algorithms

In Section 4, we presented several hardness results. Here, we present our algorithmic results for general input graphs which can be seen as different ways to generalize Theorem 5. We start with an XP-algorithm for the number of delays as a parameter and then present two FPT algorithms for "distance to forest" parameters.

5.1 Number of Delays

In what follows, we present an algorithm similar to Dijkstra's algorithm [9]. Starting at the source vertex s, it finds all optimal temporal (s, v)-routes by expanding each optimum by one step per iteration. However, as we have seen in the polynomial-time reductions in Section 4, there can be many (s, z)-routes that are Pareto-optimal with respect to the arrival time for a given number of delays. We use the dynamic program from Theorem 5 to extend the paths by a single time arc. Our main result of this section is that Delay-Robust Route admits an XP-algorithm with respect to the number x of delays. Theorem 8 implies that we presumably cannot improve this to an FPT result for this parameter. Formally, we show the following (in the remainder of this subsection, we provide a sketch of proof).

▶ **Theorem 9** (⋆). *Delay-Robust Route can be solved in $\mathcal{O}(|V|^3 \cdot |E|^{2x} \cdot x^2)$ time, where x is the number of allowed delays.*

For each route, its *arrival time vector* $\vec{t} = (t_0, t_1, \dots, t_x)$ is a vector of $x+1$ time steps where t_y is the worst-case arrival time for y delays. We define a partial order $\preceq$ to compare arrival time vectors. For $\vec{t} = (t_0, t_1, \dots, t_x)$ and $\vec{t'} = (t'_0, t'_1, \dots, t'_x)$, set $\vec{t} \preceq \vec{t'}$ if and only if $t_y \leq t'_y$ for all $y \leq x$. This partial order can be used to decrease the set of prefix paths that need to be considered due to the following observation.

▶ **Observation 10.** *Let $\mathcal{G} = (V, E)$ be a temporal graph, let $s, v, z \in V$ be three vertices, and P_1 and P_2 be two delay-robust (s, v)-routes with the arrival time vectors $\vec{t_1} \preceq \vec{t_2}$. If there is a delay-robust (s, z)-route P so that $P = P_2 \circ P'$, then $P_1 \circ P'$ is also a delay-robust route. Additionally, if there is a delay-robust (s, v)-route, then there is one whose arrival time vector is minimal among all (s, v)-routes.*

Since for any delay one can arrive earlier in vertex v by using the route P_1 compared to P_2, replacing the prefix P_2 by P_1 still guarantees delay-robustness.

We define a table A with entries for every vertex of $\mathcal{G}$. The table entry $A[v]$ contains a set of arrival time vectors for (s, v)-routes. We will only store vectors that are minimal with respect to $\preceq$, since we do not need to consider others due to Observation 10. Thus, the set $A[v]$ will represent the Pareto front of routes from s to v.

Furthermore, we define a priority queue Q that contains tuples $(v, \vec{t})$ of vertices and arrival time vectors. The queue is sorted according by the arrival time vectors according to $\preceq$. The queue elements $(v, \vec{t})$ contain the prefix routes from where a search should be expanded.

We initialize the table A as follows:

$$A[v] = \begin{cases} \{(0, \dots, 0)\}, & \text{if } v = s \\ \emptyset, & \text{otherwise.} \end{cases}$$

The start vertex s can always be reached through the empty path. For all other vertices there is initially no route stored. Furthermore, we initialize the queue Q with the tuple $(s, (0, \dots, 0))$.

To compute the table entries we repeatedly pop the first element $(v, \vec{t})$ from Q and propagate possible delay-robust routes from there. If $(v, \vec{t})$ is in the queue, then this means that there is a delay-robust (s, v)-route P with the arrival time vector $\vec{t}$.

Let $\text{next}_v := \{w \mid (v, w, t, \lambda) \in V\}$ denote the set of vertices reachable from v by a single time arc. For all $w \in \text{next}_v$, we compute the arrival time vector $\vec{t'} = (t'_0, t'_1, \dots, t'_x)$ of $P' = P \circ (w)$ using the dynamic program described in Section 3.2: The arrival time vector of P' is simply the table row $A_{P'}$ and P' is y-delay-robust if and only if $A_{P'}[y] < \infty$.

As an optimization, we can round up the arrival time entries to the next time step in τ_w^+, i.e. replace t'_y by

$$\hat{t'_y} = \min_t \{t \in \tau_w^+ \mid t \geq t'_y\}.$$

This rounding does not change the delay-robustness of any route since no temporal walk can leave w between time t'_y and $\hat{t'_y}$.

If P' is x-delay-robust, then we can add $\vec{t'}$ to the set $A[w]$, unless $A[w]$ already contains a smaller arrival time vector. We then delete all $\vec{t''}$ with $\vec{t'} \preceq \vec{t''}$ from $A[w]$ and also remove the corresponding elements $(w, \vec{t''})$ from the queue Q. Finally, we insert $(w, \vec{t'})$ into Q.

Once the queue Q is empty, we have investigated all x-robust prefix routes that might eventually lead to z. There is then a delay-robust (s, v)-route if and only if $A[z] \neq \emptyset$.

We defer the correctness proof for the presented algorithm as well as the running time analysis to the full version [17].

5.2 Timed Feedback Vertex Number

In this section, we explore another way to generalize Theorem 5. We present an FPT algorithm for the so-called *timed feedback vertex number* (introduced by Casteigts et al. [6]) and the number x of delays combined. Intuitively, the timed feedback vertex number is the minimum number of "vertex appearances" that need to be removed from the temporal graph to turn its underlying graph into a forest. Formally, it is defined as follows.

Let $\mathcal{G}$ be a temporal graph and $X \subseteq V \times [T]$ a set of *vertex appearances*. Then we write $\mathcal{G} - X := (V, E')$, where $E' = E \setminus \{(v, w, t, \lambda) \mid (v, t) \in X \vee (w, t) \in X\}$. A *timed feedback vertex set* of $\mathcal{G}$ is a set $X \subseteq V \times [T]$ of vertex appearances such that $G_{\mathrm{u}}(\mathcal{G} - X)$ is cycle-free. The *timed feedback vertex number* of a temporal graph $\mathcal{G}$ is the minimum cardinality of a timed feedback vertex set of $\mathcal{G}$.

▶ **Theorem 11** ($\star$). *Delay-Robust Route can be solved in* $2^{\mathcal{O}(xf \log f)} \cdot (|V| + |E|)^{\mathcal{O}(1)}$ *time, where f is the timed feedback vertex number of the underlying graph.*

In the following, we give a description of the main steps of the algorithm we use to obtain the above result. The algorithm follows a simple "guess and check"-approach.

1. Compute a minimum timed feedback vertex set X of the input graph using an algorithm provided by Casteigts et al. [6].
2. Let $\hat{X} = \{v \mid (v, t) \in X\}$. Iterate over all partitions $\hat{X}_0 \uplus \hat{X}_1 \uplus \hat{X}_2 \uplus \hat{X}_3 = \hat{X}$ of $\hat{X}$. We distinguish two types of neighbors of a vertex. A neighbor connected by a time arc that is preserved in $\mathcal{G} - X$ is called a "forest neighbor", while other neighbors are called "feedback neighbors". Intuitively, in this step we guess for each vertex whether its predecessor resp. successor in the route is a feedback neighbor or a forest neighbor, leading to the following four cases:
   - The route does not contain v or the predecessor and successor of v in the route are forest neighbors of v (then $v \in \hat{X}_0$),
   - the predecessor of v in the route is a forest neighbor v, and the successor of v in the route is a feedback neighbor of v (then $v \in \hat{X}_1$),
   - the predecessor of v in the route is a feedback neighbor v, and the successor of v in the route is a forest neighbor of v (then $v \in \hat{X}_2$), or
   - the predecessor and successor of v in the route are feedback neighbors of v (then $v \in \hat{X}_3$).
3. Iterate over all orders on $\hat{X}_1 \cup \hat{X}_2 \cup \hat{X}_3$. Intuitively, in this step we guess in which order the vertices appear in the route.
4. Let $\hat{T} = \{t, t + \delta \mid \exists w \in V : (w, t) \in X\} \cup \{\infty\}$ be the *relevant* time steps. For each vertex $v \in \hat{X}_1 \cup \hat{X}_2 \cup \hat{X}_3$, iterate over all *delay profiles* $(t_1, t_2, \ldots, t_x) \in \hat{T}^x$. Intuitively, here we guess for each delay size i the smallest relevant time t_i which is at least the worst-case arrival time at v.
5. Use Theorem 5 to find route segments that respect the guessed delay profiles between consecutive vertices in $\hat{X}_1 \cup \hat{X}_2 \cup \hat{X}_3$ and which can be combined to an x-delay-robust (s, z)-route.

A detailed description of the last step and a sketch of the main ideas for the correctness proof and running time analysis are deferred to the full version [17].

5.3 Underlying Feedback Edge Number

In this section, we show that Delay-Robust Route admits an FPT-algorithm with respect to the feedback edge number of the underlying graph. Given a (static) undirected graph $G = (V, E)$, a feedback edge set $F \subseteq E$ is a set of edges, so that $G - F$ is acyclic. The *feedback edge number* is the cardinality of a minimum feedback edge set of G. Formally, we show the following.

▶ **Theorem 12** ($\star$). *Delay-Robust Route can be solved in* $2^{\mathcal{O}(f)} \cdot (|V| \cdot |E| \cdot x^2) + \mathcal{O}(|E| \cdot \log |E|)$ *time, where* f *is the feedback edge number of the underlying graph.*

Casteigts et al. [6] designed an FPT-algorithm for the so-called Restless Temporal Path parameterized by the feedback edge number of the underlying graph. This algorithm can be applied to Delay-Robust Route as well with minor modifications in order to prove Theorem 12.

6 Conclusion

We modeled a naturally motivated path-finding problem taking into account delays by means of (algorithmic) temporal graph theory. For our central problem, Delay-Robust Route, we found computational hardness already for some tree-like underlying (static) graphs. While having provided a few encouraging parameterized tractability results, we leave plenty of room for further investigations into this direction. In particular, we left open what happens for the special case when the number of time labels per edge is bounded from above (in parameterized complexity terms, taking this as a parameter). Recall that our central hardness reduction needs many time labels. Moreover, the parameters vertex cover number or timed feedback vertex set number [6] (as a single parameter) deserve investigations as well. Rather from a modeling perspective, one might vary the basic problem by e.g. considering a *global* delay budget or other variations of the delay concept.

References

1 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.

2 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.

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

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

5 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.

6 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9):2754–2802, 2021. doi:10.1007/s00453-021-00831-w.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015. doi:10.1007/978-3-319-21275-3.

8 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.

9 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.

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

11 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.

12 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.

13 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.

14 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.

15 Michael R Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.

16 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.

17 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Delay-robust routes in temporal graphs, 2022. arXiv:2201.05390.

18 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Temporal connectivity: Coping with foreseen and unforeseen delays, 2022. To appear at SAND 2022. arXiv:2201.05011.

19 Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234, 2015. doi:10.1140/epjb/e2015-60657-4.

20 Petter Holme and Jari Saramäki, editors. *Temporal Network Theory.* Springer, 2019. doi:10.1007/978-3-030-23495-9.

21 Richard M Karp. *Reducibility among combinatorial problems*, pages 85–103. Springer, 1972. doi:10.1007/978-1-4684-2001-2_9.

22 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.

23 Nina Klobas, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Interference-free walks in time: Temporally disjoint paths. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 4090–4096. ijcai.org, 2021. doi:10.24963/ijcai.2021/563.

24 Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61, 2018. doi:10.1007/s13278-018-0537-7.

25 Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016. doi:10.1080/15427951.2016.1177801.

26 Hendrik Molter, Malte Renken, and Philipp Zschoche. Temporal reachability minimization: Delaying vs. deleting. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 76:1–76:15, 2021. doi:10.4230/LIPIcs.MFCS.2021.76.

27 Manuel Sorge and Mathias Weller. The graph parameter hierarchy. unpublished manuscript, 2019. URL: https://manyu.pro/assets/parameter-hierarchy.pdf.

28 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.

29 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.