Finding Temporal Paths Under Waiting Time Constraints

**Arnaud Casteigts[1] · Anne-Sophie Himmel[2] · Hendrik Molter[2] ·
Philipp Zschoche[2]** (ID)

Abstract

Computing a (short) path between two vertices is one of the most fundamental prim-
itives in graph algorithmics. In recent years, the study of paths in temporal graphs,
that is, graphs where the vertex set is fixed but the edge set changes over time,
gained more and more attention. A path is time-respecting, or *temporal*, if it uses
edges with non-decreasing time stamps. We investigate a basic constraint for tem-
poral paths, where the time spent at each vertex must not exceed a given duration Δ,
referred to as Δ-*restless temporal paths*. This constraint arises naturally in the mod-
eling of real-world processes like packet routing in communication networks and
infection transmission routes of diseases where recovery confers lasting resistance.
While finding temporal paths without waiting time restrictions is known to be doa-
ble in polynomial time, we show that the "restless variant" of this problem becomes
computationally hard even in very restrictive settings. For example, it is W[1]-hard
when parameterized by the *distance to disjoint path* of the underlying graph, which
implies W[1]-hardness for many other parameters like feedback vertex number
and pathwidth. A natural question is thus whether the problem becomes tractable
in some natural settings. We explore several natural parameterizations, presenting
FPT algorithms for three *kinds* of parameters: (1) output-related parameters (here,
the maximum length of the path), (2) classical parameters applied to the underlying
graph (e.g., feedback edge number), and (3) a new parameter called *timed feedback
vertex number*, which captures finer-grained temporal features of the input temporal
graph, and which may be of interest beyond this work.

Keywords Temporal graphs · Disease spreading · Waiting-time policies · Restless
temporal paths · Timed feedback vertex set · NP-hard problems · Parameterized
algorithms

✉ Philipp Zschoche
zschoche@tu-berlin.de

Extended author information available on the last page of the article

1 Introduction

A highly successful strategy to control (or eliminate) outbreaks of infectious diseases is contact tracing [25]—whenever an individual is diagnosed positively, every person who is possibly infected by this individual is put into quarantine. However, the viral spread can be too fast to be traced manually, e.g., if the disease is transmittable in a pre-symptomatic (or asymptomatic) stage, then it seems likely that an individual already caused infection chains when diagnosed positively. Hence, large-scale digital systems are recommended which use physical proximity networks based on location and contact data [29]—this allows fast and precise contact tracing while avoiding the harmful effect of mass quarantines to society [29]. Physical proximity networks can be understood as temporal graphs[1] [18, 37, 39, 46, 51], that is, graphs where the vertex set (individuals) remains static but the edge set (physical contacts) may change over time. In this paper, we extend the literature on reachability in temporal graphs [5, 7, 11, 14, 17, 44, 50, 61] by a computational complexity analysis of an important variation of one of the most fundamental combinational problems arising in the above mentioned scenario: given a temporal graph and two individuals s and z, is a chain of infection from s to z possible, that is, is there a *temporal path* from s to z? In particular, we use a reachability concept that captures the standard 3-state *SIR-model* (Susceptible-Infected-Recovered), a canonical spreading model for diseases where recovery confers lasting resistance [8, 45, 54].

In temporal graphs, the basic concepts of paths and reachability are defined in a time-respecting way [44]: a (strict) temporal path, also called "journey", is a path that uses edges with non-decreasing (increasing) time steps. To represent infection chains in the SIR-model, we restrict the time of "waiting" or "pausing" at each intermediate vertex to a prescribed duration. We call these paths *restless* temporal paths. They model infection transmission routes of diseases that grant immunity upon recovery [38]: An infected individual can transmit the disease until it is recovered (reflected by bounded waiting time) and it cannot be infected a second time afterwards since then it is immune (reflected by considering path instead of walk: every vertex can only be visited at most once). Another natural example of restless temporal paths is delay-tolerant networking among mobile entities, where the routing of a packet is performed over time and space by storing the packet for a limited time at intermediate nodes.

In the following we give an example to informally describe our problem setting.[2] In Fig. 1 we are given the depicted temporal graph, vertices s and z, and the time bound $\Delta = 2$. We are asked to decide whether there is a restless temporal path from s to z, that is, a path which visits each vertex at most once and pauses at most Δ units of time between consecutive hops. Here, (s, d, b, z) is a feasible solution, but (s, b, z) is not because the waiting time at b exceeds Δ. The walk (s, b, c, d, b, z) is not a valid solution because it visits vertex b twice. Finally (s, a, c, d, b, z) is also a feasible solution.

---

[1] Also known as time-varying graphs, evolving graphs, or link streams.

[2] We refer to Sect. 2 for a formal definition.

Fig. 1 Example of a temporal graph whose edges are labeled with time stamps. Bold edges depict a 2-restless temporal (s, z)-path. (In general, multiple time stamps per edge are possible)
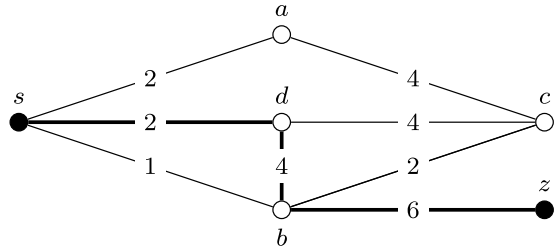
1.1 Related Work

Several types of waiting time constraints have been considered in the temporal graph literature. An empirical study by Pan and Saramäki [56] based on phone calls datasets observed a threshold in the correlation between the duration of pauses between calls and the ratio of the network reached over a spreading process. Casteigts et al. [19] showed a dramatic impact of waiting time constraints to the expressivity of a temporal graph, when considering such a graph as an automaton and temporal paths as words. In the context of temporal flows, Akrida et al. [4] considered a concept of "vertex buffers", which however pertains to the quantity of information that a vertex can store, rather than a duration. Enright et al. [26] considered deletion problems for reducing temporal connectivity. More closely related to our work, Himmel et al. [11] studied a variant of restless temporal paths where several visits to the same vertex are allowed, i.e., restless temporal *walks*. They showed, among other things, that such walks can be computed in polynomial time.

Many path-related problems have been studied in the temporal setting and the nature of temporal paths significantly increases the computational complexity of many of them (compared to their static counterparts). In the temporal setting, reachability is not an equivalence relation among vertices, which makes many problems more complicated. For example, finding a maximum temporally connected component is NP-hard [14]. We further have that in a temporal graph, spanning trees may not exist. In fact, even the existence of *sparse* spanners (i.e., subgraphs with $o(n^2)$-many edges ensuring temporal connectivity) is not guaranteed [7], unless the underlying graph is complete [20], and computing a minimum-cardinality spanner is APX-hard [3, 50]. Yet another example is the problem of deciding whether there are k disjoint temporal paths between two given vertices. In a seminal article, Kempe et al. [44] showed that this problem, whose classical analogue is (again) polynomial-time solvable, becomes NP-hard. They further investigated the related problem of finding temporal separators, which is also NP-hard [30, 44, 62]. Deciding whether there exists a separator of a given size that cuts all *restless* temporal paths is known to be Σ_2^P-complete [52], that is, the problem is located in the second level of the polynomial time hierarchy.
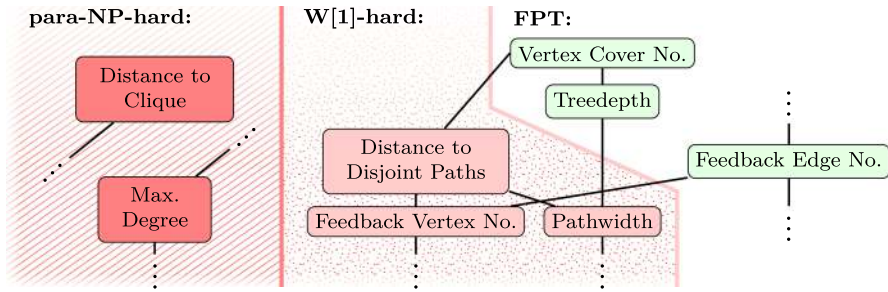
Fig. 2 Relevant part of the hierarchy among classic parameters of the underlying graph (cf. Sorge et al. [57]) for our results for RESTLESS TEMPORAL PATH

1.2 Our Contributions

We introduce the problem RESTLESS TEMPORAL PATH. To get a finer understanding of the computational complexity of this problem, we turn our attention to its parametrized complexity. In stark contrast to both restless temporal walks and non-restless temporal paths, we show that this problem is NP-hard even in very restricted settings—in particular, even when the lifetime is restricted to only three time steps—and W[1]-hard when parameterized by the (vertex deletion) *distance to disjoint paths* of the underlying graph, which implies W[1]-hardness with respect to many other parameters like feedback vertex number and pathwidth (Sect. 3). This is tight in the sense that the problem can be solved in polynomial time when the underlying graph is a forest. On the positive side, we explore parameters of three different natures. First, we show that the problem is fixed-parameter tractable (FPT) for the *length* (in number of hops) of the temporal path (Sect. 4). We further show that the problem is FPT when parameterized by the *feedback edge number* of the underlying graph (Sect. 5). Additionally, we show that the problem presumably does not admit a polynomial kernel under the previously mentioned parameterizations where the problem is in FPT. Our results provide a fine-grained characterization of the tractability boundary of the computation of restless temporal paths for parameters of the underlying graph, as illustrated by the vicinity of the corresponding parameters in Fig. 2. Then, going beyond parameters related to the output and to the underlying graph, we define a novel temporal version of the classic *feedback vertex number* called *timed feedback vertex number*. Intuitively, it counts the number of vertex *appearances* that have to be removed from the temporal graph such that its underlying graph becomes cycle-free. We show that finding restless temporal paths is FPT when parameterized by this parameter (Sect. 6). We believe that the latter is an interesting turn of events compared to our hardness results.

1.3 Strict Versus Non-strict Temporal Paths

In this paper, we focus mainly on the case of non-strict temporal paths, i.e., the times along a path are required to be non-decreasing. We expect most of the algorithms

and reductions to be extendable to a strict setting, albeit with some change in the results themselves. For instance, a similar NP-hardness reduction as for non-strict temporal paths may apply, but requires more than a constant lifetime to be adapted. In fact, the length of a strict temporal path is trivially bounded by the lifetime itself, thus an FPT algorithm for the *length* parameter implies one for the lifetime parameter as well.

2 Preliminaries

Here, we formally introduce the most important concepts related to temporal graphs and paths, and give the formal problem definition of (SHORT) RESTLESS TEMPORAL (s, z) -PATH.

An *interval* is an ordered set $[a, b] := \{n \mid n \in \mathbb{N} \wedge a \le n \le b\}$, where $a, b \in \mathbb{N}$. Further, let $[a] := [1, a]$.

2.1 Static Graphs

We use standard notation from (static) graph theory [24]. Unless stated otherwise, we assume graphs in this paper to be *undirected* and *simple*. To clearly distinguish them from temporal graphs, they are sometimes referred to as *static* graphs. Given a (static) graph $G = (V, E)$ with $E \subseteq \binom{V}{2}$, we denote by $V(G) := V$ and $E(G) := E$ the sets of its vertices and edges, respectively.

We call two vertices $u, v \in V$ *adjacent* if $\{u, v\} \in E$. Two edges $e_1, e_2 \in E$ are *adjacent* if $e_1 \cap e_2 \ne \emptyset$. For a vertex $v \in V$, we denote by $\deg_G(v)$ the *degree* of the vertex, that is, $\deg_G(v) = |\{w \in V \mid \{v, w\} \in E\}|$. For some vertex subset $V' \subseteq V$, we denote by $G[V']$ the *induced* subgraph of G on the vertex set V', that is, $G[V'] = (V', E')$ where $E' = \{\{v, w\} \mid \{v, w\} \in E \wedge v \in V' \wedge w \in V'\}$. For some vertex subset $V' \subseteq V$, we denote by $G - V'$ the subgraph of G without the vertices in V', that is, $G - V' = G[V \setminus V']$. For some edge subset $E' \subseteq E$, we denote by $G - E'$ the subgraph of G without the edges E', that is, $G - E' = (V, E \setminus E')$.

An (s, z)-*path* of length k is a sequence $P = (\{s = v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k = z\})$ of edges such that for all $i \in [k]$ we have that $\{v_{i-1}, v_i\} \in E$ and $v_i \ne v_j$ for all $i, j \in [k]$. We denote v_0 and v_k as the endpoints of P. We further denote by $E(P)$ the set of edges of path P, that is, $E(P) = \{\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}\}$ and by $V(P)$ the set of vertices visited by the path, that is, $V(P) = \bigcup_{e \in E(P)} e$. If $v_0 = v_k$ and P is of length at least three, then P is a *cycle*.

2.2 Temporal Graphs

An (undirected, simple) *temporal graph* is a tuple $\mathcal{G} = (V, E_1, \dots, E_\ell)$ (or $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ for short), with $E_i \subseteq \binom{V}{2}$ for all $i \in [\ell]$. We call $\ell(\mathcal{G}) := \ell$ the *lifetime* of $\mathcal{G}$. As with static graphs, we assume all temporal graphs in this paper to be undirected and simple. We call the graph $G_i(\mathcal{G}) = (V, E_i(\mathcal{G}))$ the *layer i* of $\mathcal{G}$

where $E_i(\mathcal{G}) := E_i$. If $E_i = \emptyset$, then G_i is a *trivial* layer. We call layers G_i and G_{i+1} *consecutive*. We call i a *time step*. If an edge e is present at time i, that is, $e \in E_i$, we say that e has *time stamp* i. We further denote $V(\mathcal{G}) := V$. The *underlying graph* $G_\downarrow(\mathcal{G})$ of $\mathcal{G}$ is defined as $G_\downarrow(\mathcal{G}) := (V, \bigcup_{i=1}^{\ell(\mathcal{G})} E_i(\mathcal{G}))$. To improve readability, we remove $(\mathcal{G})$ from the introduced notations whenever it is clear from the context. For every $v \in V$ and every time step $t \in [\ell]$, we denote the *appearance of vertex v at time t* by the pair (v, t). For every $t \in [\ell]$ and every $e \in E_t$ we call the pair (e, t) a *time edge*. For a time edge $(\{v, w\}, t)$ we call the vertex appearances (v, t) and (w, t) its *endpoints*. We assume that the *size* (for example when referring to input sizes in running time analyzes) of $\mathcal{G}$ is $|\mathcal{G}| := |V| + \sum_{i=1}^{\ell} \min\{1, |E_i|\}$, that is, we do not assume that we have compact representations of temporal graphs. Finally, we write n for $|V|$.

A *temporal (s, z)-walk* (or *temporal walk*) of length k from vertex $s = v_0$ to vertex $z = v_k$ in a temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a sequence $P = \left((v_{i-1}, v_i, t_i) \right)_{i=1}^k$ of triples that we call *transitions* such that for all $i \in [k]$ we have that $\{v_{i-1}, v_i\} \in E_{t_i}$ and for all $i \in [k-1]$ we have that $t_i \le t_{i+1}$. Moreover, we call P a *temporal (s, z)-path* (or *temporal path*) of length k if $v_i \ne v_j$ for all $i, j \in \{0, \dots, k\}$ with $i \ne j$. Given a temporal path $P = \left((v_{i-1}, v_i, t_i) \right)_{i=1}^k$, we denote the set of vertices of P by $V(P) = \{v_0, v_1, \dots, v_k\}$. Moreover, we say that P *visits* the vertex v_i at time t if $t \in [t_i, t_{i+1}]$, where $i \in [k-1]$. A *restless* temporal path is not allowed to wait an arbitrary amount of time in a vertex, but has to leave any vertex it visits within the next Δ time steps, for some given value of Δ. Analogously to the non-restless case, a restless temporal walk may visit a vertex multiple times.

Definition 1 A temporal path (walk) $P = \left((v_{i-1}, v_i, t_i) \right)_{i=1}^k$ is *Δ-restless* if $t_i \le t_{i+1} \le t_i + \Delta$, for all $i \in [k-1]$. We say that P *respects* the waiting time Δ.

Having this definition at hand, we are ready to define the main decision problem of this work.

RESTLESS TEMPORAL PATH
Input: A temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$, two distinct vertices $s, z \in V$, and an integer $\Delta \le \ell$.
Question: Is there a Δ-restless temporal (s, z)-path in $\mathcal{G}$?

Note the waiting time at the source vertex s is ignored. This is without loss of generality, since one can add an auxiliary degree one source vertex which is only in the first layer adjacent to s. We also consider a variant, where we want to find Δ-restless paths of a certain maximum length. In the SHORT RESTLESS TEMPORAL PATH problem, we are additionally given a integer $k \in \mathbb{N}$ and the question is whether there is a Δ-restless temporal path of length at most k from s to z in $\mathcal{G}$? Note that RESTLESS TEMPORAL PATH is the special case of SHORT RESTLESS TEMPORAL PATH for $k = |V| - 1$ and that both problems are in NP.

2.3 Parameterized Complexity

We use standard notation and terminology from parameterized complexity theory [23] and give here a brief overview of the most important concepts that are used in this paper. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. We call the second component the *parameter* of the problem. A parameterized problem is *fixed-parameter tractable* (in the complexity class FPT) if there is an algorithm that solves each instance (I, r) in $f(r) \cdot |I|^{O(1)}$ time, for some computable function f. A decidable parameterized problem L admits a *polynomial kernel* if there is a polynomial-time algorithm that transforms each instance (I, r) into an instance (I', r') such that $(I, r) \in L$ if and only if $(I', r') \in L$ and $|(I', r')| = r^{O(1)}$. If a parameterized problem is hard for the parameterized complexity class W[1], then it is (presumably) not in FPT. The complexity classes W[1] is closed under parameterized reductions, which may run in FPT-time and additionally set the new parameter to a value that exclusively depends on the old parameter.

2.4 Basic Observations

If there is a Δ-restless temporal (s, z)-path $\left((v_{i-1}, v_i, t_i)\right)_{i=1}^{k}$ in a temporal graph $\mathcal{G}$, then $(\{v_0, v_1\}, \ldots, \{v_{k-1}, v_k\})$ is an (s, z)-path in the underlying graph $G_{\downarrow}$. The other direction does not necessarily hold. However, we now show that for any (s, z)-path in $G_{\downarrow}$ we can decide in linear time whether this path is the support of a Δ-restless temporal (s, z)-path in $\mathcal{G}$. As a consequence, we can decide RESTLESS TEMPORAL PATH in linear time for any temporal graph where there exists a unique (s, z)-path in the underlying graph, in particular, if the underlying graph is a forest.

Lemma 1 *Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph where the underlying graph $G_{\downarrow}$ is an (s, z)-path with $s, z \in V$. Then there is an algorithm which computes in $O(|\mathcal{G}|)$ time the set*

$$\mathcal{A} = \{t \mid \text{there is a } \Delta\text{-restless temporal}(s, z) \text{ with arrival time } t\}.$$

Proof Let $V(G_{\downarrow}) = \{s = v_0, \ldots, v_n = z\}$ be the vertices and $E(G_{\downarrow}) = \{e_1 = \{v_0, v_1\}, \ldots, e_n = \{v_{n-1}, v_n\}\}$ be the edges of the underlying path. We further define L_i as the set of layers of $\mathcal{G}$ in which the edge $e_i \in E(G_{\downarrow})$ exists, that is, $L_i := \{t \mid e_i \in E_t\}$.

In the following, we construct a dynamic program on the path. We compute for every vertex v_i the table entry $T[v_i]$ which is defined as the set of all layers t such that there exists a Δ-restless temporal (s, v_i)-path with arrival time t. It holds that $T[v_1] = L_1$. Then, for all $i \in [2, \ell]$, we compute the table entry $T[v_i]$ by checking for each layer $t \in L_i$ whether there exists a Δ-restless temporal (s, v_{i-1})-path that arrives in a layer $t' \in T[v_{i-1}]$ such that we can extend the path to the vertex v_i in layer t without exceeding the maximum waiting time Δ, that is, $0 \leq t - t' \leq \Delta$. Formally, we have

$$T[v_i] := \{t \in L_i \mid \text{there is a } t' \in T[v_{i-1}] \text{ with } 0 \leq t - t' \leq \Delta\}.$$

It is easy to verify that $T[v_i]$ contains all layers t such that there exists a Δ-restless temporal (s, v_i)-path with arrival time t. After computing the last entry $T[v_n]$, this entry contains the set $\mathcal{A}$ of all layers t such that there exists a Δ-restless temporal (s, z)-path with arrival time t.

In order to compute a table entries $T[v_i]$ in linear time, we will need sorted lists of layers for L_i and $T[v_{i-1}]$ in ascending order. The sorted lists L_i of layers can be computed in $O(|\mathcal{G}|)$: For every $t \in [\ell]$, we iterate over each $e_i \in E_t$ and add t to L_i. Now assume that L_i and $T[v_{i-1}]$ are lists of layers both in ascending order, then we can compute the table entry $T[v_i]$ in $O(|T[v_{i-1}]| + |L_i|)$ time as follows. Let $T[v_i]$ be initially empty. Let t be the first element in L_i and let t' be the first element in $T[v_{i-1}]$:

1. If $t' > t$, then replace t with the next layer in L_i and repeat.
2. If $t - t' \leq \Delta$, then add t to $T[v_i]$, replace t with the next layer in L_i and repeat.
3. Else, replace t' with the next layer in $T[v_{i-1}]$ and repeat.

This is done until all elements in one of the lists are processed.

The resulting list $T[v_i]$ is again sorted. Due to this and $T[v_1](= L_1)$ being sorted, we can assume that $T[v_{i-1}]$ is given as a sorted list of layers when computing $T[v_i]$. Hence, we can compute each table entry $T[v_i]$ in $O(|T[v_{i-1}]| + |L_i|)$ time. It further holds that $|T[v_i]| \leq |L_i|$ and $\sum_{i=1}^{n} |L_i| = \sum_{i=1}^{\ell} |E_i|$. Hence, the dynamic program runs in $O(|\mathcal{G}|)$ time. $\qquad\square$

Furthermore, it is easy to observe that computational hardness of RESTLESS TEMPORAL PATH for some fixed value of Δ implies hardness for all larger finite values of Δ. This allows us to construct hardness reductions for small fixed values of Δ and still obtain general hardness results.

Observation 1 *Given an instance* $I = (\mathcal{G}, s, z, k, \Delta)$ *of* SHORT RESTLESS TEMPORAL PATH, *we can construct in linear time an instance* $I' = (\mathcal{G}', s, z, k, \Delta + 1)$ *of* SHORT RESTLESS TEMPORAL PATH *such that I is a yes-instance if and only if I' is a yes-instance.*

Proof The result immediately follows from the observation that a temporal graph $\mathcal{G}$ contains a Δ-restless temporal (s, z)-path if and only if the temporal graph $\mathcal{G}'$ contains a $(\Delta + 1)$-restless temporal (s, z)-path, where $\mathcal{G}'$ is obtained from $\mathcal{G}$ by inserting one trivial (edgeless) layer after every Δ consecutive layers. $\qquad\square$

However, for some special values of Δ we can solve RESTLESS TEMPORAL PATH in polynomial time.

Observation 2 RESTLESS TEMPORAL PATH *on instances* $(\mathcal{G}, s, z, \Delta)$ *can be solved in polynomial time, if* $\Delta = 0$ *or* $\Delta \geq \ell$.

Proof Considering $\Delta = 0$ implies that the entirety of a path between s and z must be realized in a single layer. Thus, the problem is equivalent to testing if at least one of the layers G_i contains a (static) path between s and z. □

If $\Delta \geq \ell$, then Δ-restless temporal paths correspond to unrestricted temporal paths, whose computation can be made using any of the (polynomial time) algorithms in Bui-Xuan et al. [17].

3 Hardness Results for Restless Temporal Paths

In this section we present a thorough analysis of the computational hardness of RESTLESS TEMPORAL PATH which also transfers to SHORT RESTLESS TEMPORAL PATH.

3.1 NP-hardness for few layers

We start by showing that RESTLESS TEMPORAL PATH is NP-complete even if the lifetime of the input temporal graph is constant. The reduction is similar in spirit to the classic NP-hardness reduction for 2-DISJOINT PATHS in directed graphs by Fortune et al. [33].

Theorem 1 RESTLESS TEMPORAL PATH *is NP-complete for all finite* $\Delta \geq 1$ *and* $\ell \geq \Delta + 2$ *even if every edge has only one time stamp.*

Proof We show this result by a reduction from the NP-complete EXACT $(3, 4)$-SAT problem [59]. The problem EXACT $(3, 4)$-SAT asks whether a formula ϕ is satisfiable, assuming that it is given in conjunctive normal form, each clause having exactly three literals and each variable appearing in exactly four clauses.

Let ϕ be an instance of EXACT $(3, 4)$-SAT with n variables and m clauses. We construct a temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ with $\ell = 3$ consisting of a series of variable gadgets followed by dedicated vertices s_n and s' and then a series of clause gadgets. It is constructed in such a way that for $\Delta = 1$, any Δ-restless temporal (s, z)-path has to visit a vertex s_n and each possible Δ-restless temporal (s, s_n)-path represents exactly one variable assignment for the formula ϕ. Further we show that for any Δ-restless temporal (s, s_n)-path it holds that it can be extended to a Δ-restless temporal (s, z)-path if and only if the Δ-restless temporal (s, s_n)-path represents a satisfying assignment for the formula ϕ.

Variable Gadget. We start by adding a vertex s to the vertex set V of $\mathcal{G}$. For each variable x_i with $i \in [n]$ of ϕ, we add 9 fresh vertices to V: $x_i^{(1)}$, $x_i^{(2)}$, $x_i^{(3)}$, $x_i^{(4)}$, $\bar{x}_i^{(1)}$,
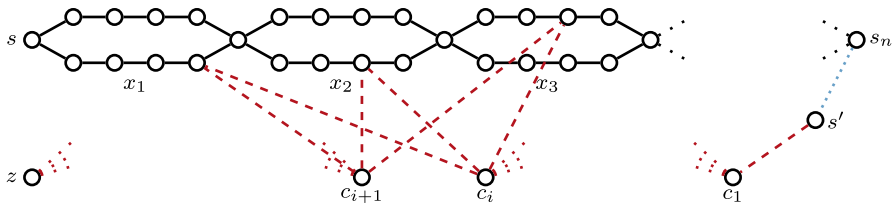
Fig. 3 Illustration of the temporal graph constructed by the reduction in the proof of Theorem 1. An excerpt is shown with variable gadgets for x_1, x_2, and x_3 and the clause gadget for $c_i = (x_1 \vee x_2 \vee \neg x_3)$, where x_1 appears for the fourth time, x_2 appears for the third time, and x_3 also appears for the third time. Black edges appear at time step one, the blue (dotted) edge $\{s_n, s'\}$ appears at time step two, and the red (dashed) edges appear at time step three

$\bar{x}_i^{(2)}$, $\bar{x}_i^{(3)}$, $\bar{x}_i^{(4)}$, and s_i. Each variable x_i is represented by a gadget consisting two disjoint path segments of four vertices each. One path segment is formed by $x_i^{(1)}$, $x_i^{(2)}$, $x_i^{(3)}$, and $x_i^{(4)}$ in that order and the second path segment is formed by $\bar{x}_i^{(1)}$, $\bar{x}_i^{(2)}$, $\bar{x}_i^{(3)}$, and $\bar{x}_i^{(4)}$ in that order. The connecting edges all appear exclusively at time step one, that is, $\{x_i^{(1)}, x_i^{(2)}\}$, $\{x_i^{(2)}, x_i^{(3)}\}$, and $\{x_i^{(3)}, x_i^{(4)}\}$ are added to E_1. Analogously for the edges connecting $\bar{x}_i^{(1)}$, $\bar{x}_i^{(2)}$, $\bar{x}_i^{(3)}$, and $\bar{x}_i^{(4)}$. Intuitively, if a Δ-restless temporal (s, z)-path passes the first segment, then this corresponds to setting the variable x_i to false. If it passes the second segment, then the variable is set to true. For all $i \in [n-1]$ we add the edges $\{x_i^{(4)}, s_i\}$, $\{\bar{x}_i^{(4)}, s_i\}$, $\{s_i, \bar{x}_{i+1}^{(1)}\}$, and $\{s_i, \bar{x}_{i+1}^{(1)}\}$ to E_1 and, additionally, we add $\{s, x_1^{(1)}\}$, $\{s, \bar{x}_1^{(1)}\}$, $\{x_n^{(4)}, s_n\}$, and $\{\bar{x}_n^{(4)}, s_n\}$ to E_1.

We can observe that there are exactly 2^n different temporal (s, s_n)-paths at time step one. Intuitively, each path represents exactly one variable assignment for the formula ϕ.

Clause Gadget. We add a vertex z to V. For each clause c_j with $j \in [m]$ we add a fresh vertex c_j to V. We further add a vertex s' to V and add the edge $\{s_n, s'\}$ to E_2. Let x_i (or $\bar{x}_i$) be a literal that appears in clause c_j and let this be the kth appearance of variable x_i in ϕ. Then, we add the edges $\{c_j, x_i^{(k)}\}$, $\{x_i^{(k)}, c_{j+1}\}$ (or $\{c_j, \bar{x}_i^{(k)}\}$, $\{\bar{x}_i^{(k)}, c_{j+1}\}$) to E_3 (where $c_{m+1} = z$). Finally, we add the edge $\{s', c_1\}$ to E_3.

Hence, there are exactly 3^m different temporal (s', z)-paths at time step three. Each path must visit the clause vertices $c_1, \ldots, c_m$ in the given order by construction.

Finally, we set $\Delta = 1$. This finishes the construction, for a visualization see Fig. 3. It is easy to check that every edge in the constructed temporal graph has only one time step and that the temporal graph can be computed in polynomial time.

Correctness. Now we can show that ϕ is satisfiable if and only if $\mathcal{G}$ has a Δ-restless temporal (s, z)-path.

($\Rightarrow$): Let us assume there is a satisfying assignment for formula ϕ. Then we construct a Δ-restless temporal path from vertex s to z as follows. Starting from s, for

each variable x_i of ϕ the Δ-restless temporal path passes through the variables $x_i^{(1)}$, $x_i^{(2)}$, $x_i^{(3)}$, and $x_i^{(4)}$, if x_i is set to false, and $\bar{x}_i^{(1)}$, $\bar{x}_i^{(2)}$, $\bar{x}_i^{(3)}$, and $\bar{x}_i^{(4)}$, if x_i is set to true, at time step one. The Δ-restless temporal path arrives at time step one in the vertex s_n. In time step two it goes from s_n to s'.

At time step three, the Δ-restless temporal path can be extended to c_1. In each clause c_j for $j \in [m]$ there is at least one literal x_i (or $\bar{x}_i$) that is evaluated to true. Let c_j be the kth clause in which x_i appears. We have that, depending on whether x_i is set to true (or false), the vertex $x_i^{(k)}$ (or $\bar{x}_i^{(k)}$) has not been visited so far. Hence, the Δ-restless temporal path can be extended from c_j to c_{j+1} (or to z for $j = m$) at time step three via $x_i^{(k)}$ (or $\bar{x}_i^{(k)}$). Thus, there exists a Δ-restless temporal (s, z)-path in $\mathcal{G}$.

($\Leftarrow$): Let us assume that there exists a Δ-restless temporal (s, z)-path in the constructed temporal graph $\mathcal{G}$. Note that any Δ-restless temporal (s, z)-path must reach s_n in time step one because the variable gadget has only edges at time step one and the waiting time $\Delta = 1$ prevents the path to enter the clause gadget (which only has edges at time step three) before using the edge $\{s_n, s'\}$ at time step two.

It is easy to see that for the first part of the Δ-restless temporal graph from s to s_n it holds that for each $i \in [n]$, it visits either vertices $x_i^{(1)}$, $x_i^{(2)}$, $x_i^{(3)}$, and $x_i^{(4)}$, or vertices $\bar{x}_i^{(1)}$, $\bar{x}_i^{(2)}$, $\bar{x}_i^{(3)}$, and $\bar{x}_i^{(4)}$. In the former case we set x_i to false and in the latter case we set x_i to true. We claim that this produces a satisfying assignment for ϕ.

In time step three, the part of the Δ-restless temporal path from s' to z has to pass vertices $c_1, c_2, \ldots, c_m$ to reach z. The Δ-restless temporal path passes exactly one variable vertex $x_i^{(k)}$ (or $\bar{x}_i^{(k)}$) when going from c_j to c_{j+1} (and finally from c_m to z) that has not been visited so far and that corresponds to a variable that appears in the clause c_j for the kth time. The fact that the variable vertex was not visited implies that we set the corresponding variable to a truth value that makes it satisfy clause c_j. This holds for all $j \in [m]$. Hence, each clause is satisfied by the constructed assignment and, consequently, ϕ is satisfiable. □

The reduction used in the proof of Theorem 1 also yields a running time lower bound assuming the Exponential Time Hypothesis (ETH) [40, 41].

Corollary 1 RESTLESS TEMPORAL PATH *does not admit a* $f(\ell)^{o(|\mathcal{G}|)}$*-time algorithm for any computable function f unless the ETH fails.*

Proof First, note that any 3-SAT formula with m clauses can be transformed into an equisatisfiable EXACT (3, 4)-SAT formula with $O(m)$ clauses [59]. The reduction presented in the proof of Theorem 1 produces an instance of RESTLESS TEMPORAL PATH with a temporal graph of size $|\mathcal{G}| = O(m)$ and $\ell = 3$. Hence an algorithm for RESTLESS TEMPORAL PATH with running time $f(\ell)^{o(|\mathcal{G}|)}$ for some computable function f would imply the existence of an $2^{o(m)}$-time algorithm for 3-SAT. This is a contradiction to the ETH [40, 41]. □

Furthermore, the reduction behind Theorem 1 can be modified such that it also yields that RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph has constant maximum degree or the underlying graph is a clique where one edge ($\{s, z\}$) is missing. Note that in the latter case the underlying graph contains all edges except the one edge which would turn the instance into a trivial *yes*-instance.

Corollary 2 *RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph has all but one edge or maximum degree six.*

Proof That RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph has maximum degree six follows directly from the construction used in the proof of Theorem 1. To show that that RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph has all edges except $\{s, z\}$, we reduce from RESTLESS TEMPORAL PATH. Let $I = (\mathcal{G} = (V, (E_i)_{i \in [\ell]}), s, z, \Delta)$ be an instance of RESTLESS TEMPORAL PATH with $\ell = 3$. We construct an instance $I' := (\mathcal{G}' = (V, E_1', E_2', E_3', E_4', E_5'), s, z, \Delta)$ of RESTLESS TEMPORAL PATH, where $E_1' = \binom{V \setminus \{s\}}{2}$, $E_2' := E_1$, $E_3' := E_2$, $E_4' := E_3$, and $E_5' = \binom{V \setminus \{z\}}{2}$. Observe that none of the edges in $E_1 \cup E_5$ can be used in temporal (s, z)-path. Hence, I is a *yes*-instance if and only if I' is a *yes*-instance. Furthermore, $E_1 \cup E_5$ contain all possible edges except $\{s, z\}$. □

3.2 W[1]-Hardness for Distance to Disjoint Paths

In the following, we show that parameterizing RESTLESS TEMPORAL PATH with structural graph parameters of the underlying graph of the input temporal graph presumably does not yield fixed-parameter tractability for a large number of popular parameters. In particular, we show that RESTLESS TEMPORAL PATH parameterized by the distance to disjoint paths of the underlying graph is W[1]-hard. The *distance to disjoint paths* of a graph G is the minimum number of vertices we have to remove from G such that the reminder of G is a set of disjoint paths. Many well-known graph parameters can be upper-bounded in the distance to disjoint paths, e.g., pathwidth, treewidth, and feedback vertex number [57]. Hence, the following theorem also implies that RESTLESS TEMPORAL PATH is W[1]-hard when parameterized by the pathwidth or the feedback vertex number of the underlying graph.

Theorem 2 *RESTLESS TEMPORAL PATH parameterized by the distance to disjoint path of the underlying graph is W[1]-hard for all $\Delta \geq 1$ even if every edge has only one time stamp.*

Proof We present a parameterized reduction from MULTICOLORED CLIQUE where, given a k-partite graph $H = (U_1 \uplus U_2 \uplus \ldots \uplus U_k, F)$, we are asked to decide whether H contains a clique of size k. MULTICOLORED CLIQUE is known to be W[1]-hard when parameterized by the clique size k [23, 28].

Let $(H = (U_1 \uplus U_2 \uplus \ldots \uplus U_k, F), k)$ be an instance of MULTICOLORED CLIQUE. For each $i, j \in [k]$ with $i < j$ let $F_{i,j} = \{\{u, v\} \in F \mid u \in U_i \wedge v \in U_j\}$ be the set of edges between vertices in U_i and U_j. We can assume that $k \geq 3$, otherwise we can solve the instance in polynomial time. Without loss of generality, we assume that for all $i, j, i', j' \in [k]$ with $i < j$ and $i' < j'$ we have that $|F_{i,j}| = |F_{i',j'}| = m$ for some $m \in \mathbb{N}$. Note that if this is not the case, we add new vertices and single edges to increase the cardinality of some set $F_{i,j}$ and this does not introduce new cliques since $k \geq 3$. We further assume without loss of generality that $|U_1| = |U_2| = \cdots = |U_k| = n$ for some $n \in \mathbb{N}$. If this is not the case, we can add additional isolated vertices to increase the cardinality of some set U_i. We construct a temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ with two distinct vertices $s, z \in V$ such that there is a Δ-restless temporal (s, z)-path in $\mathcal{G}$ if and only if H contains a clique of size k. Furthermore, we show that the underlying graph $G_\downarrow$ of $\mathcal{G}$ has a distance to disjoint paths of $O(k^2)$.

Vertex Selection Gadgets. For each set U_i with $i \in [k]$ of the vertex set of H we create the following gadget. Let $U_i = \{u_1^{(i)}, u_2^{(i)}, \ldots, u_n^{(i)}\}$. We create a path of length $k \cdot n + n + 1$ on fresh vertices $w_1^{(i)}, v_{1,1}^{(i)}, v_{1,2}^{(i)}, \ldots, v_{1,k}^{(i)}, w_2^{(i)}, v_{2,1}^{(i)}, \ldots, v_{n,k}^{(i)}, w_{n+1}^{(i)}$. Intuitively, this path contains a segment of length k for each vertex in U_i which are separated by the vertices $w_j^{(i)}$, and the construction will allow a Δ-restless temporal (s, z)-path to skip exactly one of these segments, which is going to correspond to selecting this vertex for the clique.

Formally, for each vertex $u_j^{(i)} \in U_i$ we create k vertices $v_{j,1}^{(i)}, v_{j,2}^{(i)}, \ldots, v_{j,k}^{(i)}$, which we call the *segment corresponding to* $u_j^{(i)}$. We further create vertices $w_1^{(i)}, w_2^{(i)}, \ldots, w_{n+1}^{(i)}$. For all $j \in [n]$ and $x \in [k-1]$ we connect vertices $v_{j,x}^{(i)}$ and $v_{j,x+1}^{(i)}$ with an edge at time $(i-1) \cdot n + j$ and we connect $w_j^{(i)}$ with $v_{j,1}^{(i)}$ and $w_{j+1}^{(i)}$ with $v_{j,k}^{(i)}$ at time $(i-1) \cdot n + j$ each.

Lastly, we introduce a "skip vertex" $s^{(i)}$ that will allow a Δ-restless temporal (s, z)-path to skip one path segment of length k that corresponds to one of the vertices in U_i. For each $j \in [n+1]$, we connect vertices $s^{(i)}$ and $w_j^{(i)}$ with an edge at time $(i-1) \cdot n + j$.

Now we connect the gadgets for all U_i's in sequence, that is, a Δ-restless temporal (s, z)-path passes through the gadgets one after another, selecting one vertex of each part U_i. Formally, for all $i \in [k-1]$, we connect vertices $w_{n+1}^{(i)}$ and $w_1^{(i+1)}$ with an edge at time $i \cdot n + 1$. It is easy to check that after the removal of the vertices $\{s^{(1)}, s^{(2)}, \ldots, s^{(k)}\}$, the vertex selection gadget is a path. The vertex selection gadget is visualized in Fig. 4.

Validation Gadgets. A Δ-restless temporal (s, z)-path has to pass through the validation gadgets after it passed through the vertex selection gadgets. Here, we are forced to choose a point in time where we visit two vertices of two different vertex selection gadgets. This choice corresponds to the selection of an edge. Intuitively,
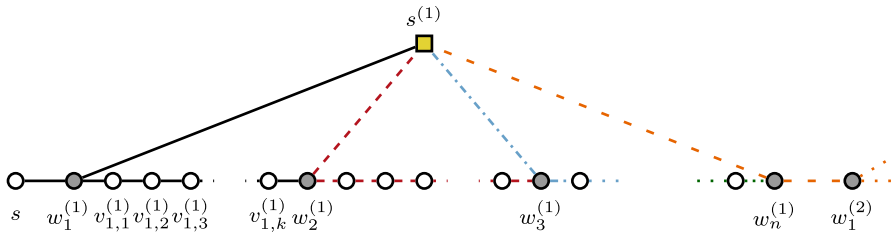
Fig. 4 Visualization of the vertex selection gadget for U_1 from the reduction of Theorem 2. Black edges appear at time step one, red edges (densely dashed) at time step two, blue edges (dashdotted) at time step three, green edges (dotted) at time step $n - 1$, and orange edges (loosely dashed) at time step n. For the segment corresponding to $u_1^{(1)} \in U_1$ all vertex names are presented, for the other segments the names are analogous but omitted. The auxiliary $w_1^{(1)}, \dots, w_n^{(1)}, \dots$ vertices are colored gray. The "skip vertex" $s^{(1)}$ is depicted as a yellow square. Note that after the removal of $s^{(1)}$ the vertex selection gadget for U_1 is a path

this should only be possible if the selected vertices form a clique. We construct the gadget in the following way.

For each $i, j \in [k]$ with $i < j$ let the edges in $F_{i,j}$ be ordered in an arbitrary way, that is, $F_{i,j} = \{e_1^{(i,j)}, e_2^{(i,j)}, \dots, e_m^{(i,j)}\}$. We create two paths of length $2m$ on fresh vertices $v_{1,1}^{(i,j)}, v_{1,2}^{(i,j)}, v_{2,1}^{(i,j)}, v_{2,2}^{(i,j)}, \dots, v_{m,2}^{(i,j)}$ and $v_{1,3}^{(i,j)}, v_{1,4}^{(i,j)}, v_{2,3}^{(i,j)}, v_{2,4}^{(i,j)}, \dots, v_{m,4}^{(i,j)}$, respectively. Intuitively, the *first path* selects an edge from U_i to U_j and the transition to the *second path* should only be possible if the two endpoints of the selected edge are selected in the corresponding vertex selection gadgets.

Formally, for each edge $e_h^{(i,j)} \in F_{i,j}$ we create four vertices $v_{h,1}^{(i,j)}, v_{h,2}^{(i,j)}, v_{h,3}^{(i,j)}, v_{h,4}^{(i,j)}$. Furthermore, we introduce three extra vertices $s_1^{(i,j)}, s_2^{(i,j)}, s_3^{(i,j)}$. For all $h \in [m]$ we connect vertices $v_{h,1}^{(i,j)}$ and $v_{h,2}^{(i,j)}$ with an edge at time $y_{i,j} + 2h - 1$, we connect vertices $v_{h,1}^{(i,j)}$ and $s_1^{(i,j)}$ with an edge at time $y_{i,j} + 2h - 1$, we connect vertices $v_{h,3}^{(i,j)}$ and $v_{h,4}^{(i,j)}$ with an edge at time $y_{i,j} + 2h - 1$, we connect vertices $v_{h,3}^{(i,j)}$ and $s_3^{(i,j)}$ with an edge at time $y_{i,j} + 2h - 1$, and if $h < m$, we connect vertices $v_{h,2}^{(i,j)}$ and $v_{h+1,1}^{(i,j)}$ with an edge at time $y_{i,j} + 2h$ and we connect vertices $v_{h,4}^{(i,j)}$ and $v_{h+1,3}^{(i,j)}$ with an edge at time $y_{i,j} + 2h$, where $y_{i,j} = k \cdot n + 2m \cdot (i \cdot j + \frac{1}{2} \cdot i \cdot (i-1) - 1)$ (the value of $y_{i,j}$ can be interpreted as a "time offset" for the validation gadget for $F_{i,j}$, the value is computed by adding all time steps needed in validation gadget for $F_{i',j'}$ with $i' < j'$, $i' \leq i$, $j' \leq j$, and $(i',j') \neq (i,j)$). Next, for each edge $e_h^{(i,j)} = \{u_a^{(i)}, u_b^{(j)}\} \in F_{i,j}$ we connect vertices $s_1^{(i,j)}$ and $v_{a,j}^{(i)}$ (from the vertex selection gadget for U_i) with an edge at time $y_{i,j} + 2h - 1$, we connect vertices $s_2^{(i,j)}$ and $v_{a,j}^{(i)}$ with an edge at time $y_{i,j} + 2h - 1$, we connect vertices $s_2^{(i,j)}$ and $v_{b,i}^{(j)}$ (from the vertex selection gadget for U_j) with an edge at time $y_{i,j} + 2h - 1$, and we connect vertices $s_3^{(i,j)}$ and $v_{b,i}^{(j)}$ with an edge at time $y_{i,j} + 2h - 1$.

Intuitively, the time labels on the edges and the waiting time restrictions enforce that when arriving at $s_1^{(i,j)}$ there is only one way to continue to $s_2^{(i,j)}$ for which is it necessary to visit a vertex in the vertex selection gadget that corresponds to an endpoint
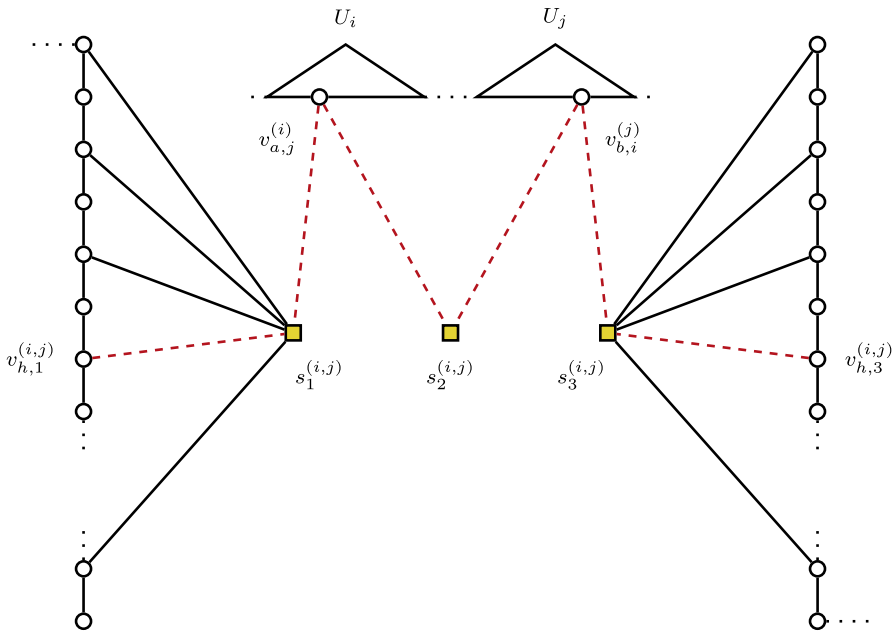
Fig. 5 Visualization of the validation gadget for $F_{i,j}$ from the reduction of Theorem 2. The "first path" of the gadget is depicted vertically on the left, the "second path" on the right. The connections to the vertex selection gadgets for the edge $e_h^{(i,j)} = \{u_a^{(i)}, u_b^{(j)}\} \in F_{i,j}$ are depicted. The edges in red (dashed) correspond to the path through the gadget if edge $e_h^{(i,j)}$ is "selected" and all these edges have the same time stamp. The vertex selection gadgets corresponding to U_i and U_j are depicted as triangles in the upper center part. The three vertices $s_1^{(i,j)}$, $s_2^{(i,j)}$, and $s_3^{(i,j)}$ are colored yellow (squared). Note that after the removal of $s_1^{(i,j)}$, $s_2^{(i,j)}$, and $s_3^{(i,j)}$, the validation gadget for $F_{i,j}$ is a set of disjoint paths

of the selected edge. Similarly, from $s_2^{(i,j)}$ there is only one way to continue to $s_3^{(i,j)}$ for which it is necessary to visit a vertex in the vertex selection gadget that corresponds to the other endpoint of the selected edge. For a visualization of the validation gadget see Fig. 5, where the red dashed path corresponds to the selection of an edge.

Now we connect the gadgets for all $F_{i,j}$'s in sequence, that is, a Δ-restless temporal (s,z)-path passes through the gadgets one after another, selecting one edge of each part $F_{i,j}$ of the edge set F. Formally, for each $i,j \in [k]$ with $i < j$, if $i < j - 1$, we connect vertices $v_{m,4}^{(i,j)}$ and $v_{1,1}^{(i+1,j)}$ with an edge at time $y_{i+1,j}$, and if $i = j - 1 < k - 1$, we connect vertices $v_{m,4}^{(i,j)}$ and $v_{1,1}^{(i,j+1)}$ with an edge at time $y_{i,j+1}$. It is easy to check that after the removal of $3 \cdot \binom{k}{2}$ many vertices $\{s_1^{(1,2)}, s_2^{(1,2)}, s_3^{(1,2)}, s_1^{(1,3)}, \ldots, s_3^{(1,k)}, \ldots s_3^{(k-1,k)}\}$, the validation gadgets are a set of disjoint paths, see Fig. 5.

Finally, we create two new vertices s and z, we connect vertices s and $w_1^{(1)}$ (the "first" vertex of the vertex selection gadgets) with an edge at time one, we connect vertices s and $s^{(1)}$ (the "skip vertex" of the first vertex selection gadget) with an edge at time one, and we connect z and $v_{m,4}^{(k-1,k)}$ (the "last" vertex of the validation gadgets)

with an edge at time $k \cdot n + m \cdot (3k^2 + 5k + 3)$. We further connect vertices $w_{n+1}^{(k)}$ and $v_{1,1}^{(1,2)}$ (connecting the vertex selection gadgets and the validation gadgets) with an edge at time $k \cdot n$. Finally, we set $\Delta = 1$. This completes the construction. It is easy to check that $\mathcal{G}$ can be constructed in polynomial time and that the distance to disjoint paths of $G_\downarrow$ is at most $k + 3 \cdot \binom{k}{2}$ and that every edge has only one time stamp.

Correctness. Now we show that H contains a clique of size k if and only if there is a Δ-restless temporal path from s to z in $\mathcal{G}$.

($\Rightarrow$): Assume that H contains a clique of size k and let $X \subseteq V(H)$ with $|X| = k$ be the set of vertices that form the clique in H. Now we show how to construct a Δ-restless temporal (s, z)-path in $\mathcal{G}$. Note that since H is k-partite, we have that $|U_i \cap X| = 1$ for all $i \in [k]$. The temporal path starts at vertex s in $\mathcal{G}$ and then first passes through the vertex selection gadgets. If $u_j^{(i)} \in X$ for some $i \in [k]$ and $j \in [n]$, then the temporal path skips the segment corresponding to $u_j^{(i)}$ in the vertex selection gadget for U_i. More formally, the temporal path follows the vertices $w_1^{(i)}, v_{1,1}^{(i)}, v_{1,2}^{(i)}, \ldots, v_{1,k}^{(i)}, w_2^{(i)}, \ldots, v_{j-1,k}^{(i)}, w_j^{(i)}, s^{(i)}, w_{j+1}^{(i)}, v_{j+1,1}^{(i)}, \ldots, v_{n,k}^{(i)}, w_{n+1}^{(i)}$ in that order, that is, the path skips vertices $v_{j,1}^{(i)}, v_{j,2}^{(i)}, \ldots, v_{j,k}^{(i)}$. It is easy to check that the time labels of the edges in the vertex selection gadget allow for a restless temporal path as described that respects the waiting time Δ.

In the validation gadget for $F_{i,j}$ with $i < j$, the path "selects" the edge $(U_i \cap X) \cup (U_j \cap X) \in F_{i,j}$ that connects the vertices from the parts U_i and U_j that are contained in the clique X. Let $(U_i \cap X) \cup (U_j \cap X) = \{u_a^{(i)}, u_b^{(j)}\} = e_h^{(i,j)} \in F_{i,j}$. Formally, the path follows vertices $v_{1,1}^{(i,j)}, v_{1,2}^{(i,j)}, v_{2,1}^{(i,j)}, v_{2,2}^{(i,j)}, \ldots, v_{h,1}^{(i,j)}, s_1^{(i,j)}, v_{a,j}^{(i)}, s_2^{(i,j)}, v_{b,i}^{(j)}, s_3^{(i,j)}, v_{h,4}^{(i,j)}, v_{h+1,3}^{(i,j)}, v_{h+1,4}^{(i,j)}, \ldots, v_{m,4}^{(i,j)}$ in that order. Note that vertices $v_{a,j}^{(i)}$ and $v_{b,i}^{(j)}$ have not been used by the path in the vertex selection gadgets, since they appear in the segments that were skipped by the temporal path in the corresponding vertex selection gadgets. Furthermore, since the clique in H only contains one edge that connects vertices from U_i and U_j, the vertices $v_{a,j}^{(i)}$ and $v_{b,i}^{(j)}$ have not been used by the temporal path in an earlier validation gadget. It is easy to check that the time labels of the edges in the validation gadget allow for a Δ-restless temporal path as described. After the last validation gadget the path arrives at vertex z. Hence, we have found a Δ-restless temporal (s, z)-path in $\mathcal{G}$.

($\Leftarrow$): Assume that we are given a Δ-restless temporal (s, z)-path in $\mathcal{G}$. We now show that H contains a clique of size k.

After starting at s, the Δ-restless temporal path first passes the vertex selection gadgets. Here, we need to make the important observation, that for each $i \in [k]$, any Δ-restless temporal (s, z)-path has to "skip" at least one segment corresponding to one vertex $u_j^{(i)} \in U_i$ in the vertex selection gadget corresponding to U_i, otherwise the temporal path cannot traverse the validation gadgets. More formally, assume for contradiction that there is a Δ-restless temporal (s, z)-path and an $i \in [k]$ such that the

temporal path visits all vertices in the vertex selection gadget corresponding to U_i. Let $j \in [k]$ with $j \neq i$. Assume that $i < j$ (the other case works analogously). We claim that the temporal path cannot traverse the validation gadget for $F_{i,j}$. For the temporal path to go from $s_1^{(i,j)}$ to $s_2^{(i,j)}$ by construction it has to visit at least one vertex from the vertex selection gadget for U_i. If all vertices have already been visited, that would mean the Δ-restless temporal (s, z)-path visits one vertex twice—a contradiction.

The waiting time Δ prevents the temporal path from "skipping" more than one segment. More formally, any Δ-restless temporal (s, z)-path arrives at the "skip vertex" $s^{(i)}$ of the vertex selection gadget for U_i at time $(i - 1) \cdot n + j$, for some $j \in [k - 1]$. By construction this means the path visits $w_j^{(i)}$, then $s^{(i)}$, and then has to continue with $w_{j+1}^{(i)}$ since there is only one time edge the path can use without violating the waiting time Δ. It follows that the temporal path skips exactly the segment corresponding to $u_j^{(i)} \in U_i$.

This implies that any Δ-restless temporal (s, z)-path that traverses the vertex selection gadgets leaves exactly one segment of every vertex selection gadget unvisited. Let the set $X = \{u_j^{(i)} \in U_i \mid i \in [k] \wedge j \in [n] \wedge v_{j,1}$ is an unvisited vertex.$\}$ be the set of vertices corresponding to the segments that are "skipped" by the given Δ-restless temporal (s, z)-path. It is easy to check that $|X| = k$. We claim that X is a clique in H.

Assume for contradiction that it is not. Then there are two vertices $u_{i'}^{(i)}, u_{j'}^{(j)} \in X$ such that the edge $\{u_{i'}^{(i)}, u_{j'}^{(j)}\}$ is not in F. Assume that $i < j$. We show that then the Δ-restless temporal (s, z)-path is not able to pass through the validation gadget for $F_{i,j}$. By assumption we have that $\{u_{i'}^{(i)}, u_{j'}^{(j)}\} \notin F_{i,j}$. Note that the validation gadget is designed in a way that the first path "selects" an edge from $F_{i,j}$ and then the waiting time of one enforces that a Δ-restless temporal (s, z)-path can only move from the first path to the second path of a validation gadget if the two endpoints of the selected edge are vertices whose corresponding segments in the vertex selection gadget were skipped. We have seen that for every U_i with $i \in [k]$, the path segment corresponding to exactly one vertex of that set was skipped. Since $\{u_{i'}^{(i)}, u_{j'}^{(j)}\} \notin F_{i,j}$, we have that for every edge in $F_{i,j}$ that the segment corresponding to at least one of the two endpoints of the edge was *not* skipped. Hence, we have that the Δ-restless temporal path cannot pass through the validation gadget of $F_{i,j}$ and cannot reach z—a contradiction. $\qquad\square$

4 An FPT-Algorithm for Short Restless Temporal Path

In this section, we discuss how to find *short* restless temporal paths. Recall that in SHORT RESTLESS TEMPORAL PATH, we are given an additional integer k as input and are asked whether there exists a Δ-restless temporal (s, z)-path that uses at most k

time edges. By Theorem 1 this problem is NP-hard. Note that in the contact tracing scenario from the beginning, we can expect to have a small k and a large temporal graph.

Theorem 3 SHORT RESTLESS TEMPORAL PATH is

(i)   solvable in $2^k \cdot |\mathcal{G}|^{O(1)}$ time with a constant one-side error,[3]
(ii)   deterministically solvable in $2^{O(k)} \cdot |\mathcal{G}|\Delta$ time,

Note that we can solve SHORT RESTLESS TEMPORAL PATH such that the running time is independent from the lifetime ℓ of the temporal graph. To show Theorem 3, we first reduce the problem to a specific path problem in directed graphs. Then, we apply known algebraic tools for multilinear monomials detection. Here, Theorem 3 (i) is based on Williams [60]. To get a deterministic algorithm with a running time almost linear in $|\mathcal{G}|$, we show a different approach based on representative sets [31] which results in Theorem 3 (ii).

Reduction to directed graphs. We introduce a so-called Δ-(s, z)-*expansion* for two vertices s and z of a temporal graph with waiting times. That is, a time-expanded version of the temporal graph which reduces reachability questions to directed graphs. While similar approaches have been applied several times [4, 12, 50, 61, 62], to the best of our knowledge, this is the first time that waiting-times are considered. In a nutshell, the Δ-(s, z)-expansion has for each vertex v at most ℓ many copies $v^1, \ldots, v^\ell$ and if an (s, z)-dipath visits v^i, it means that the corresponding Δ-restless temporal (s, z)-walk visits v at time i.

Definition 2 (Δ-(s, z)-Expansion) Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph with two distinct vertices $s, z \in V$ such that $\{s, z\} \notin E_t$, for all $t \in [\ell]$. Let $\Delta \le \ell$. The Δ-(s, z)-*expansion* of $\mathcal{G}$ is the directed graph $D = (V', E')$ with

(i) $V' := \{s, z\} \cup \{v^t \mid v \in e, e \in E_t, v \notin \{s, z\}\}$,
(ii) $E_s := \{(s, v^t) \mid \{s, v\} \in E_t\}$,
(iii) $E_z := \{(v^i, z) \mid v^i \in V', \{v, z\} \in E_t, 0 \le t - i \le \Delta\}$, and
(iv) $E' := E_s \cup E_z \cup \{(v^i, w^t) \mid v^i \in V' \setminus \{s, z\}, \{v, w\} \in E_t, 0 \le t - i \le \Delta\}$.

Furthermore, we define $V'(s) := \{s\}, V'(z) := \{z\}$, and $V'(v) := \{v^t \in V' \mid t \in [\ell]\}$, for all $v \in V \setminus \{s, z\}$.

Next, we show that a Δ-(s, z)-expansion of a temporal graph can be computed efficiently.

---

[3] The algorithm always outputs *no* if there is no Δ-restless temporal (s, z)-path and outputs otherwise *yes* with constant probability.

Lemma 2 *Given a temporal graph* $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$, *two distinct vertices* $s, z \in V$, *and* $\Delta \leq \ell$, *we can compute its* Δ-(s, z)-*expansion* D *with* $|V(D)| \in O(|\mathcal{G}|)$ *in* $O(|\mathcal{G}| \cdot \Delta)$ *time.*

Proof Let $V' := \{s, z\}$ and E' be empty in the beginning. We will fill up V' and E' simultaneously. In order to do that efficiently, we will maintain for each vertex $v \in V$ a ordered list L_v such that $t \in L_v$ if and only if $v^t \in V'$. We assume that $|V| \leq \sum_{i=1}^{\ell} |E_i|$, because vertices which are isolated in every layer are irrelevant for the Δ-(s, z)-expansion and can be erased in linear time.

We proceed as follows. For each $t \in \{1, \ldots, \ell\}$ (in ascending order), we iterate over E_t. For each $\{v, w\} \in E_t$, we distinguish three cases.

$(w = s)$:	We add v^t to V', (s, v^t) to E', and add t to L_v. This can be done in constant time.
$(w = z)$:	We add v^t to V', and add t to L_v. Now we iterate over all $i \in L_v$ (in descending order) and add (v^i, z) to E' until $t - i > \Delta$. This can be done in $O(\Delta)$ time.
$(\{s, z\} \cap \{v, w\} = \emptyset)$:	We add v^t, w^t to V', and add t to L_v and L_w. Now we iterate over $i \in L_v$ (in descending order) and add (v^i, w^t) to E' until $t - i > \Delta$. Afterwards, we iterate over $i \in L_w$ (in descending order) and add (w^i, v^t) to E' until $t - i > \Delta$. This can be done in $O(\Delta)$ time.

Observe that after this procedure the digraph $D = (V', E')$ is the Δ-(s, z)-expansion of $\mathcal{G}$ and that we added at most 2 vertices for each time-edge in $\mathcal{G}$. Hence, $V' \leq |\mathcal{G}|$. This gives a overall running time of $O(|\mathcal{G}| \cdot \Delta)$. $\qquad \square$

It is easy to see that there is a Δ-restless temporal (s, z)-walk in the temporal graph if and only if there is an (s, z)-dipath in the Δ-(s, z)-expansion. Next, we identify the necessary side constraint to identify Δ-restless temporal (s, z)-paths in the Δ-(s, z)-expansion.

Lemma 3 *Let* $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ *be a temporal graph,* $s, z \in V$ *two distinct vertices,* $\Delta \leq \ell$, *and* $D = (V', E')$ *the* Δ-(s, z)-*expansion of* $\mathcal{G}$. *There is a* Δ-*restless temporal* (s, z)-*path in* $\mathcal{G}$ *of length* k *if and only if there is an* (s, z)-*dipath* P' *in* D *of length* k *such that for all* $v \in V$ *it holds that* $|V'(v) \cap V(P')| \leq 1$.

Proof ($\Rightarrow$): Let $P = \left(((s, v_1, t_1), (v_1, v_2, t_2), \ldots, (v_{k'-1}, z, t_{k'})) \right)$ be a Δ-restless temporal (s, z)-path in $\mathcal{G}$ of length k. We can inductively construct an (s, z)-dipath P' in D. Observe that $P'_1 := ((s, v_1^{t_1}))$ is an $(s, v_1^{t_1})$-dipath of length 1 in D, because the arc

$(s, v_1^{t_1})$ is in E_s of D. Now let $i \in [k' - 2]$ and P_i' be an $(s, v_i^{t_i})$-dipath of length i such that

   (i)   for all $j \in [i]$, we have that $|V'(v_j) \cap V(P_i')| = 1$, and
   (ii) for all $v \in V \setminus \{s, v_1, \dots, v_i\}$, we have that $|V'(v) \cap V(P_i')| = 0$.

In order to get an $(s, v_{i+1}^{t_{i+1}})$-dipath P_{i+1}' of length $i + 1$, we extend P_i' by the arc $(v_i^{t_i}, v_{i+1}^{t_{i+1}})$. Observe, that $v_{i+1}^{t_{i+1}} \in V'$ because of the time-edge $(\{v_i, v_{i+1}\}, t_{i+1})$ in $\mathcal{G}$ and that the arc $(v_i^{t_i}, v_{i+1}^{t_{i+1}}) \in E'$, because we have $0 \leq t_{i+1} - t_i \leq \Delta$. Observe that

   (i)   for all $j \in [i + 1]$, we have that $|V'(v_j) \cap V(P_{i+1}')| = 1$, and
   (ii) for all $v \in V \setminus \{s, v_1, \dots, v_{i+1}\}$, we have that $|V'(v) \cap V(P_{i+1}')| = 0$.

Hence, we have an $(s, v_{k'-1}^{t_{k'-1}})$-dipath P_{k-1}' of length $k - 1$ satisfying (i) and (ii) which can be extended (in a similar way) to an (s, z)-dipath of length k such that for all $v \in V$ it holds that $|V'(v) \cap V(P')| \leq 1$.

($\Leftarrow$): Let P' be a (s, z)-dipath in D of length k such that for all $v \in V$ it holds that $|V'(v) \cap V(P')| \leq 1$. Let $V(P') = \{s, v_1^{t_1}, \dots, v_{k-1}^{t_{k-1}}, z\}$. Observe that an arc from s to $v_1^{t_1}$ in D implies that there is a time-edge $(\{s, v_1\}, t_1)$ in $\mathcal{G}$. Similarly, an arc from $v_i^{t_i}$ to $v_{i+1}^{t_{i+1}}$ implies that there is a time-edge $(\{v_i, v_{i+1}\}, t_{i+1})$ in $\mathcal{G}$ and that $0 \leq t_{i+1} - t_i \leq \Delta$, for all $i \in [k - 2]$. Moreover, an arc from $v_{k-1}^{t_{k-1}}$ to z implies that there is some t_k such that there is a time-edge $(\{v_k, z\}, t_k)$ in $\mathcal{G}$ with $0 \leq t_k - t_{k-1} \leq \Delta$. Hence, $P = ((s, v_1, t_1), (v_1, v_2, t_2), \dots, (v_{k'-1}, z, t_{k'}))$ is a Δ-restless temporal (s, z)-walk of length k in $\mathcal{G}$. Finally, $|V'(v) \cap V(P')| \leq 1$, for all $v \in V$, implies that $v_i \neq v_j$ for all $i, j \in \{0, \dots, k\}$ with $i \neq j$. Thus, P is a Δ-restless temporal (s, z)-path of length k. $\qquad\square$

Obtaining Theorem 3 (i) We now adapt the algorithm of Williams [60] to our specific needs. To this end, we introduce some standard notation from algebraic theory.

An *arithmetic circuit C* over a commutative ring R is a simple labelled directed acyclic graph with its internal nodes labeled by + (sum gates) or × (product gates) and its nodes of in-degree zero (input gates) labeled with elements from $R \cup X$, where X is a set of variables. There is one node of out-degree zero, called the output gate. The size of C is the number of vertices in the graph. An arithmetic circuit C over R computes a polynomial $P(X)$ over R in the natural way: an input gate represents the polynomial it is labeled by. A sum (product) gate represents the sum (product) of the polynomials represented by its in-degree neighbors. We say C represents $P(X)$ if the polynomial of the output gate of C is equivalent to $P(X)$.

Lemma 4 *Let $k \in \mathbb{N}$ and $D = (V, A)$ be a directed graph with partition $V = \biguplus_{i=0}^n V_i$, where $V_0 = \{s\}$ and $V_n = \{z\}$. Then, there is an arithmetic circuit C representing a polynomial $Q(X)$ of degree at most $k + 1$ such that $Q(X)$ has a multilinear[4] monomial*

---

[4] No variable occurs to a power of two or higher.

of degree at most $k + 1$ if and only if there is an (s, z)-path P of length at most k in D where $|V(P) \cap V_i| \leq 1$ for all $i \in [n]$. Moreover, $|X| = n + 1$, C is of size $O(k(n + |A|))$, has no scalar multiplication, and all product gates in have in-degree two.

The idea of the polynomial is similar to the one of Williams [60], but here instead of having one variable for each vertex we just have one variable for all vertices in one part of the partition of V.

Proof We define the polynomial recursively as $Q(X) = x_\perp Q_z^k$ over variables $X = \{x_\perp, x_0\} \cup \{x_i \mid i \in [n]\}$, where

$$
\begin{aligned}
Q_s^0 &:= x_0, \\
\forall v \in V \setminus \{s\} : Q_v^0 &:= x_\perp, \qquad\qquad\qquad \text{and} \\
\forall v \in V, \forall j \in [k] : Q_v^j &:= \sum_{(u,v)\in A} Q_u^{j-1} x_i, \quad \text{where } v \in V_i.
\end{aligned}
\tag{1}
$$

Note that we can, by simply following (1), construct an arithmetic circuit C which represents $Q(X)$ in $O(k(n + |A|))$ time such that each product gate has an in-degree of two. Furthermore, observe that $Q(X)$ has no scalar multiplication and is of degree at most $k + 1$.

The following induction completes the proof: We claim that for all $v \in V$ and $j \in [k] \cup \{0\}$, $Q(X)$ has a multilinear monomial M of degree at most $j + 1$ if and only if there is an (s, v)-path P of length at most j in D where $|V(P) \cap V_i| \leq 1$ for all $i \in [n]$. Moreover, M contains the variable x_i if and only if $|V(P) \cap V_i| = 1$, for all $i \in [n]$. Is is easy to verify that the claim is true for $j = 0$.

Now assume as induction hypothesis that for all $u \in V$ and all $j' < j \in [k]$, the polynomial $x_\perp Q_u^{j'}$ has a multilinear monomial M of degree at most $j' + 1$ if and only if there is an (s, u)-path P of length at most j' in D where $|V(P) \cap V_i| \leq 1$, for all $i \in [n]$. Moreover, M contains the variable x_i if and only if $|V(P) \cap V_i| = 1$, for all $i \in [n]$. Let $v \in V_p$.

($\Rightarrow$): Assume there is a multilinear monomial M of degree at most $j + 1$ in $x_\perp Q_v^j$. Since $x_\perp Q_v^j = \sum_{(u,v)\in A} x_p(x_\perp Q_u^{j-1})$, we know that M contains x_p and there is a $(u, v) \in A$ such that $x_\perp Q_u^{j-1}$ contains a multilinear monomial M' which does not contain x_p. By induction hypothesis, there is an (s, u)-path P' of length at most $j - 1$ such that $|V(P') \cap V_i| = 1$ if and only if M' contains x_i for all $i \in [n]$. Hence, there is an (s, v)-path P (obtained by extending P' with v) such that $|V(P) \cap V_i| \leq 1$ for all $i \in [n]$. Furthermore, we have that $|V(P) \cap V_i| = 1$ if and only if M contains x_i for all $i \in [n]$.

($\Leftarrow$): Assume there is an (s, v)-path P of length at most j in D where $|V(P) \cap V_i| \leq 1$ for all $i \in [n]$. Let P' be the (s, u)-path obtained by removing v from P. Hence, P' is

of length at most $j - 1$, and $|V(P) \cap V_i| \leq 1$ for all $i \in [n]$. By induction hypothesis $x_{\perp} Q_u^{j-1}$ contains a multilinear monomial M of degree at most j which does not contain x_p. Since $x_{\perp} Q_v^j = \sum_{(u,v) \in A} x_p(x_{\perp} Q_u^{j-1})$, we know that $x_{\perp} Q_v^j$ contains a M multiplied by x_p as monomial. Thus, $x_{\perp} Q_v^j$ has a multilinear monomial of degree at most $j + 1$ which contains variable x_i if and only if $|V(P') \cap V_i| = 1$, for all $i \in [n]$. $\qquad\square$

Now we can apply the following result of Williams [60].

Theorem 4 [60] *Let $Q(X)$ be a polynomial of degree at most k, represented by an arithmetic circuit of size n with no scalar multiplications and where all product gates have in-degree two. There is a randomized algorithm that runs in $2^k n^{O(1)}$ time, outputs yes with high probability ($\geq 1/5$) if there is a multilinear term in the sum-product expansion of Q, and always outputs no if there is no multilinear term.*

Theorem 3 (i) follows from Lemmas 2 to 4 and Theorem 4. This can be derandomized by Theorem 5.2 of Fomin et al. [32] resulting in $O(3.841^k \cdot (|\mathcal{G}|\Delta)^2 |V| \log |V|)$ time algorithm. We now show how to improve the polynomial part of a deterministic algorithm.

Obtaining Theorem 3 (ii) To show Theorem 3 (ii), we first note that in the (s, z)-expansion of an (s, z)-path P in the directed graph describes a Δ-restless temporal (s, z)-path exactly when $V(P)$ is an *independent set of some specific matroid*. We then show an algorithm to find such a path P (if there is one). To this end, we introduce a problem, INDEPENDENT PATH, and some standard terminology from matroid theory [55]. A pair $(U, \mathcal{I})$, where U is the *ground set* and $\mathcal{I} \subseteq 2^U$ is a family of *independent sets*, is a *matroid* if the following holds: $\emptyset \in \mathcal{I}$; if $A' \subseteq A$ and $A \in \mathcal{I}$, then $A' \in \mathcal{I}$; and if $A, B \in \mathcal{I}$ and $|A| < |B|$, then there is an $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$. An inclusion-wise maximal independent set $A \in \mathcal{I}$ of a matroid $M = (U, I)$ is a *basis*. The cardinality of the bases of M is called the *rank* of M. The *uniform matroid of rank r* on U is the matroid $(U, \mathcal{I})$ with $\mathcal{I} = \{S \subseteq U \mid |S| \leq r\}$. A matroid $(U, \mathcal{I})$ is *linear* or *representable over a field* $\mathbb{F}$ if there is a matrix A with entries in $\mathbb{F}$ and the columns labeled by the elements of U such that $S \in \mathcal{I}$ if and only if the columns of A with labels in S are linearly independent over $\mathbb{F}$. Such a matrix A is called a *representation* of $(U, \mathcal{I})$. Now we are ready to state the INDEPENDENT PATH problem.

INDEPENDENT PATH

Input: A digraph $D = (V, E)$, two distinct vertices $s, z \in V$, a representation A_M of a matroid $M = (V, \mathcal{I})$ of rank r over a finite field $\mathbb{F}$.

Question: Is there an (s, z)-dipath P of length at most k in D such that $V(P) \in \mathcal{I}$?

For the remainder of this section, whenever we speak about *independent sets*, these are independent sets of a matroid and not a set of vertices which induce an edgeless graph.

Agrawal et al. [1] studied, independently from us, a similar problem where the edges of the path shall be an independent set of a matroid. To show Theorem 3 (ii), we need a single-exponential algorithm which has only a linear dependency on the input size. To this end, we show the following, based on representative families.

Theorem 5 *An instance (D, s, z, A_M) of* INDEPENDENT PATH *can be solved in time of $O(2^{\omega r} m)$ operations over the field $\mathbb{F}$, where $\mathbb{F}$ is the field of A_M, r is rank of M, m is the number of edges in D, and $2 < \omega < 2.373$ is an upper-bound for the matrix multiplication exponent.*[5]

In this section, we provide a fixed-parameter algorithm for INDEPENDENT PATH parameterized by rank r of the matroid. Since the rank r is at most $|V(D)|$, this algorithm is asymptotically optimal, see Corollary 3. To show Theorem 5, we provide an algorithm (Algorithm 4.1), show its correctness (Lemma 5), and prove the running time upper-bound (Lemma 6). The idea of our algorithm is based on the algorithm of Fomin et al. [31] for k-PATH and independently from us Agrawal et al. [1] showed an algorithm which runs in $2^{O(r)} n^{O(1)}$ time for INDEPENDENT PATH and Lokshtanov et al. [48] provided a dynamic program, running in $5.18^r n^{O(1)}$ time, for the special case of INDEPENDENT PATH when the matroid given in the input is a transversal matroid. However, in contrast to Agrawal et al. [1] and Lokshtanov et al. [48], we pay attention to the detail that the algorithm behind Theorem 5 runs in linear time, if we can perform one field operation in constant time.

Algorithm 4.1: INDEPENDENT PATH parameterized by the rank r.

Input: An instance $(D = (V, E), s, z, A_M)$ of INDEPENDENT PATH, where A_M is a representation of matroid $M = (V(D), \mathcal{I})$ over field $\mathbb{F}$ and of rank r.
Output: Determines whether $(D = (V, E), s, z, A_M)$ is a *yes-* or *no-instance*.

1   $T[v, i] \leftarrow \emptyset$, for all $v \in V$ and $i \in [r - 1]$.
2   $T[s, 0] \leftarrow \{s\}$.
3   **for** $i \leftarrow 1$ **to** $r - 1$ **do**
4   **foreach** $w \in V$ **do**
5         $\mathcal{N}_{w,i} \leftarrow \emptyset$.
6   **foreach** $(v, w) \in E$ *with* $T[v, i - 1] \neq \emptyset$ *and* $\{w\} \in \mathcal{I}$ **do**
7           $\mathcal{N}_{w,i} \leftarrow \mathcal{N}_{w,i} \cup (T[v, i - 1] \bullet_M \{\{w\}\})$.
8         $T[w, i] \leftarrow (r - i - 1)$-representative of $\mathcal{N}_{w,i}$.   (using Theorem 6)
9   **if** $T[z, i] \neq \emptyset$ **then return** $(D = (V, E), s, z, A_M)$ *is a yes-instance*.
10 **return** $(D = (V, E), s, z, A_M)$ *is a no-instance*.

---

[5] Note that we require $2 < \omega$ even though this might be not true. We do this to upper-bound the polynomial part in r. The bound $\omega < 2.373$ is known [6].

The main tool of our algorithm are representative families of independent sets.

Definition 3 (Representative family) Given a matroid $(U, \mathcal{I})$, and a family $\mathcal{S} \subseteq 2^U$,

we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is a *q-representative for* $\mathcal{S}$ if, for each set $Y \subseteq U$ of size at most q, it holds that:

- if there is a set $X \in \mathcal{S}$ with $X \uplus Y \in \mathcal{I}$,
- then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ with $\widehat{X} \uplus Y \in \mathcal{I}$.

A *p-family* is a family $\mathcal{F}$ such that each set $S \in \mathcal{F}$ is of size exactly p. For linear matroids, we can compute small representative families efficiently. Formally, the following is known.

Theorem 6 (Fomin et al. [31, Theorem 1.1]) *Let* $M = (U, \mathcal{I})$ *be a linear matroid of rank* $r = p + q$ *given together with its representation* A_M *over field* $\mathbb{F}$. *Let* $\mathcal{S}$ *be a p-family of independents of M. Then a q-representative family* $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ *of size at most* $\binom{r}{p}$ *can be found in* $O\left(\binom{r}{p} t p^\omega + t \binom{r}{q}^{\omega-1}\right)$ *operations over* $\mathbb{F}$, *where* $\omega < 2.373$ *is the matrix multiplication exponent.*

We are now ready to give the pseudo-code of the algorithm behind Theorem 5 (see Algorithm 4.1).

In Algorithm 4.1, $\mathcal{A} \bullet_M \mathcal{B}$ is defined as $\{A \cup B \mid A \in \mathcal{A}, B \in \mathcal{B}, A \cap B = \emptyset, A \cup B \in \mathcal{I}\}$ for families $\mathcal{A}, \mathcal{B} \subseteq \mathcal{I}$ and matroid $M = (U, \mathcal{I})$.

Lemma 5 *Algorithm 4.1 is correct.*

Proof Let $\mathcal{P}_{w,i} := \{X \in \mathcal{I} \mid \text{there is an } (s, w)\text{-dipath } P \text{ of length } i \text{ such that } V(P) = X\}$, for all $w \in V$ and $i \in [r-1]$. Observe that $\mathcal{P}_{w,i}$ is an $(i+1)$-family of independent sets. We show by induction that after iteration i of the for-loop in Line (3) the entry $T[w, i]$ is an $(r-i)$-representative of $\mathcal{P}_{w,i}$, for all $w \in V$ and $i \in [r-1]$. Then the correctness follows, since we check after each of these iterations whether $T[w, i]$ is nonempty (Line (9)). Observe that $\mathcal{P}_{s,0} = \{s\}$ and $\mathcal{P}_{v,0} = \emptyset$ for all $v \in V \setminus \{s\}$. Hence, the entries of T computed in Lines (1) and (2) fulfill our induction hypothesis.

Now let $i \in [r-1]$ be the current iteration of the for-loop in Line (3) and assume that for all $j < i$ we have that $T[w, j]$ is an $(r-j)$-representative of $\mathcal{P}_{w,j}$, for all $w \in V$. Fix a vertex $w \in V$. We first show that if there is an $X \in T[i, w]$, then there is an (s, w)-dipath P_w of length i such that $X = V(P_w) \in \mathcal{I}$. Observe that in Lines (5)–(7) we look at each possible predecessor $v \in V$ of w in an (s, w)-dipath of length i, take each set $X' \in T[v, i-1]$ and check whether $X' \cup \{w\}$ is an independent set of size $i + 1$. If this is the case, we add it to $\mathcal{N}_{w,i}$. After Line (8), we have that $T[w, i] \subseteq \mathcal{N}_{w,i}$. Since $X' \in T[v, i-1]$, we know that there is an (s, v)-dipath P_v of length $i - 1$ with $X' = V(P)$. Thus, if there is an $X \in T[i, w]$, then there is an (s, w)-dipath P_w of length i such that $X = V(P_w) \in \mathcal{I}$

Now let $X \in \mathcal{P}_{w,i}$ and $Y \subseteq V(D)$ be a set of vertices of size at most $r - i - 1$ such that $X \cap Y = \emptyset$ and $X \uplus Y \in \mathcal{I}$. Hence, there is an (s, w)-dipath P of length i such that $V(P) = X$. Let v be the predecessor of w in P. Let P_v be the (s, v)-dipath of length $i - 1$ induced by P without w. Hence, $V(P_v) \in \mathcal{P}_{v,i-1}$. Moreover, $V(P_v) \cap (Y \cup \{w\}) = \emptyset$ and $V(P_v) \uplus (Y \cup \{w\}) \in \mathcal{I}$. Since $T[v, i - 1]$ is an $(r - i + 1)$-representative family of $\mathcal{P}_{v,i-1}$, we know that there is an $\widehat{X} \in T[v, i - 1]$ such that $\widehat{X} \cap (Y \cup \{w\}) = \emptyset$ and $\widehat{X} \cup (Y \cup \{w\}) \in \mathcal{I}$. In Lines (5)–(7) we add $\widehat{X} \cup \{w\}$ to $\mathcal{N}_{w,i}$. Let $X' := \widehat{X} \cup \{w\}$ and note that $X' \cap Y = \emptyset$ and $X' \uplus Y \in \mathcal{I}$. Since $T[w, i]$ is an $(r - i)$-representative family of $\mathcal{N}_{w,i}$, we know that there is an $\widehat{X'} \in T[w, i]$ such that $\widehat{X'} \cap Y = \emptyset$ and $X' \uplus Y \in \mathcal{I}$. Thus, $T[w, i]$ is an $(r - i)$-representative of $\mathcal{P}_{w,i}$. $\qquad\square$

Next, we show that Algorithm 4.1 is actually a fixed-parameter algorithm parameterized by the length of a shortest Δ-restless temporal (s, z)-path.

Lemma 6 *Algorithm* 4.1 *runs in time of* $O(2^{\omega r} \cdot m)$ *operations over* $\mathbb{F}$, *where* $\mathbb{F}$ *is the field of* A_M, r *is the rank of the matroid,* m *is the number of edges, and* $2 < \omega < 2.373$ *is an upper-bound for the matrix multiplication exponent.*

Proof Without loss of generality we assume to have a total ordering on V. We represent a subset of V as a sorted string. Hence, union and intersection of two sets of size at most r takes $O(r)$ time. We can thus look up and store sets of size at most r in a trie (or radix tree) in $O(r)$ time [21]. Note that we do not have the time to completely initialize the arrays of size $|V|$ in each trie node. Instead, we will initialize each array cell of a trie node at its first access. To keep track of the already initialized cells, we use *sparse sets* over V which allows membership test, insertion, and deletion of elements in constant time [16].

We denote the *in-neighborhood* of a vertex w by $N^-(w) := \{v \in V \mid (v, w) \in E\}$. Furthermore, let $H_{i,w}$ be the running time of Lines (5)–(7) in iteration i of the for-loop in Line (3), and $R_{i,w}$ be the number of operations over $\mathbb{F}$ of Line (8) in iteration i of the for-loop in Line (3). Then we can run Algorithm 4.1 in time of $O\left(\sum_{i=1}^{r-1} \sum_{w \in V} H_{i,w} + \sum_{i=1}^{r-1} \sum_{w \in V} R_{i,w}\right)$ operations over $\mathbb{F}$—that is, the running time respecting the time needed for operations over $\mathbb{F}$. Let $i \in [r - 1]$ and $w \in V$. In the i-th iteration of the for-loop in Line (3), $|T[v, j]| \leq \binom{r}{j+1}$ for all $j < i$ and $v \in V$, since we used Theorem 6 in prior iterations. Let $2 < \omega < 2.373$ be an upper-bound for the matrix multiplication exponent. Hence, $|\mathcal{N}_{w,i}| \leq \binom{r}{i+1}|N^-(w)|$ and $H_{i,w} \in O(\binom{r}{i+1}|N^-(w)| \cdot r^\omega)$, because the independence test can be done via matrix multiplication. Thus,

$$O\left(\sum_{i=1}^{r-1}\sum_{w\in V}H_{i,w}\right)\subseteq O\left(\sum_{i=1}^{r-1}\sum_{w\in V}|N^-(w)|\binom{r}{i+1}\cdot r^\omega\right)\subseteq O(2^{r+o(r)}m).$$

Moreover, by Theorem 6, we have

$$O\left(\sum_{i=1}^{r-1}\sum_{w\in V}R_{i,w}\right)\subseteq O\left(\sum_{i=1}^{r-1}m\binom{r}{i}\binom{r}{i+1}(i+1)^\omega+\sum_{i=1}^{r-1}m\binom{r}{i}\binom{r}{r-i-1}^{\omega-1}\right)$$
$$\subseteq O\left(2^r m(2^r r^\omega+2^{r(\omega-1)})\right)\subseteq O\left(m(2^{2r+\log_2(r)\omega}+2^{r\omega})\right)\subseteq O(2^{\omega r}m),$$

where the last inclusion is true because we assume $2 < \omega$.

Thus, we can run Algorithm 4.1 in time of $O(2^{\omega r}m)$ operations over $\mathbb{F}$. □

Now Theorem 5 follows from Lemmas 6 and 5.

Observe that by Lemma 3, there is a Δ-restless temporal (s, z)-path in the temporal graph $\mathcal{G}$ if and only if there is an (s, z)-path P in the Δ-(s, z)-expansion $D = (V', E')$ of $\mathcal{G}$ such that $V(P)$ is an independent set in the *partition matroid*[6] $M = (V', \{X \subseteq V' \mid \forall v \in V : |X \cap V'(v)| \leq 1\})$. Note that M is of rank $|V|$ and hence too large to show Theorem 3 with Theorem 5.

A *k-truncation* of a matroid $(U, \mathcal{I})$ is a matroid $(U, \{X \in \mathcal{I} \mid |X| \leq k\})$ such that all independent sets are of size at most k. The k-truncation of a linear matroid is also a linear matroid [49]. In our reduction from SHORT RESTLESS TEMPORAL PATH to INDEPENDENT PATH we use a $(k + 1)$-truncation of matroid M. Two general approaches are known to compute a representation for a k-truncation of a linear matroid—one is randomized [49] and one is deterministic [47].[7] Both approaches require a large field implying that one operation over that field is rather slow. However, for our specific matroid we employ the Vandermonde matrix to compute a representation over a small finite field. Note that we would not get a running time linear in the input size by applying the algorithm of Lokshtanov et al. [47] or Marx [49] on M.

Lemma 7 *Given a universe U of size n, a partition $P_1 \uplus \cdots \uplus P_q = U$, and an integer $k \in \mathbb{N}$, we can compute in $O(kn)$ time a representation A_M for the matroid $M = \left(U, \left\{X \subseteq U \mid |X| \leq k \text{ and } \forall i \in [q] : |X \cap P_i| \leq 1\right\}\right)$, where A_M is defined over a finite field $\mathbb{F}$ and one operation over $\mathbb{F}$ takes constant time.*

Proof For this running time analysis we assume the *Word RAM model of computation*, introduced by [34], which is similar to the *RAM model of computation* but one

---

[6] Partition matroids are linear [49].

[7] For both algorithms, a representation of the original matroid must be given.

memory cell can store only $O(\log n)$ many bits, where n is the input size. This avoids abuse of the unit cost random access machine by for example multiplying very large numbers in constant time.

Without loss of generality we assume that $q \leq n$. Let p be a prime number with $q \leq p \leq 2q$. Such a prime exists by the folklore Betrand-Chebyshev Theorem [2] and can be computed in $O(n)$ time using Lagarias-Odlyzko Method [58]. To perform one operation on the prime field $\mathbb{F}_p$, one can first perform the primitive operation in $\mathbb{Z}$ and them take the result modulo p. Since $p \leq 2q \in O(n)$, each element of $\mathbb{F}_p$ fits into one cell of the *Word RAM model of computation*. Thus, we can perform one operation over $\mathbb{F}_p$ in constant time.

Let $x_1, \ldots, x_q$ be pair-wise distinct elements from $\mathbb{F}_p$. To compute an $(k \times n)$-matrix A_M as representation for M over $\mathbb{F}_p$, we compose (column-wise) for each element $u \in P_i$ the vector $\mathbf{v}_i := \left(x_i^0 \ x_i^1 \ \ldots \ x_i^{k-1} \right)^T$, where $i \in [q]$. That gives a running time of $O(k \cdot n)$ operations over $\mathbb{F}_p$, since we can compute $\mathbf{v}_i$ in $O(k)$ operations over $\mathbb{F}_p$.

It remains to show that A_M is a representation of M. Let $X \subseteq U$. If there is an $i \in [p]$ such that $|X \cap P_i| > 1$, then the corresponding columns of A_M are linearly dependent, because we have the vector $\mathbf{v}_i$ twice. Now we assume that for all $i \in [q]$ we have $|X \cap P_i| \leq 1$. Furthermore, if $|X| > k$, then we know that the corresponding columns of A_M are linearly dependent, because A_M is an $(k \times n)$-matrix. We can observe that if $|X| = k$, then the corresponding columns in A_M form a Vandermonde matrix, whose determinate is known to be non-zero. Hence, if $|X| \leq k$, then the corresponding columns in A_M are linearly independent. Thus, A_M is a representation of M. □

We now show a reduction from SHORT RESTLESS TEMPORAL PATH to INDEPENDENT PATH using Lemmas 2, 3 and 7

Lemma 8 *Given an instance $(\mathcal{G}, s, z, k, \Delta)$ of SHORT RESTLESS TEMPORAL PATH, we can compute in $O(\max\{k, \Delta\} \cdot |\mathcal{G}|)$ time an instance (D, s, z, A_M) of INDEPENDENT PATH such that M has rank $k + 1$, and $(\mathcal{G}, s, z, k, \Delta)$ is a yes-instance if and only if (D, s, z, A_M) is a yes-instance, where one operation over the finite field of A_M takes constant time.*

Proof Let $(\mathcal{G} = (V, (E_i)_{i \in [\ell]}), s, z, k, \Delta)$ be an instance of SHORT RESTLESS TEMPORAL PATH. We construct an instance (D, s, z, A_M, k) of INDEPENDENT PATH in the following way. Let digraph $D = (V', E')$ be the Δ-(s, z)-expansion of $\mathcal{G}$ which can be computed, by Lemma 2, in $O(|\mathcal{G}| \cdot \Delta)$ time such that $V' \in O(|\mathcal{G}|)$. Observe that $\bigcup_{v \in V} V'(v)$ is a partition of V'. Now, we construct a representation A_M (over a finite field where we can perform one operation in constant time) of the matroid

$$M = \left(V', \left\{ X \subseteq V' \mid |X| \le k + 1 \text{ and } \forall v \in V : |X \cap V'(v)| \le 1 \right\} \right)$$

in $O(k \cdot |\mathcal{G}|)$ time by Lemma 7. Note that M is an $(k + 1)$-truncated partition matroid and hence has rank $k + 1$. This completes the construction and gives us an overall running time of $O(\max\{k, \Delta\} \cdot |\mathcal{G}|)$.

We now claim $(\mathcal{G}, s, z, k, \Delta)$ is a *yes*-instance if and only if (D, s, z, A_M) is a *yes*-instance and contains an independent (s, z)-dipath of length at most k.

($\Leftarrow$): Let P be a Δ-restless temporal (s, z)-path of length $k' \le k$ in $\mathcal{G}$. Then, by Lemma 3 there is an (s, z)-dipath P' of length k' such that for all $v \in V$ it holds that $|V'(v) \cap V(P')| \le 1$. Since $|V(P')| = k' + 1 \le k + 1$, we know that $V(P')$ is an independent set of M. Thus, P' is a witness of length at most k for (D, s, z, A_M) being a *yes*-instance.

($\Rightarrow$): Let P' be an (s, z)-dipath of length $k' \le k$ in D such that $V(P')$ is an independent set of M. Clearly, for $v \in V$ it holds that $|V'(v) \cap V(P')| \le 1$. Then, by Lemma 3, there is a Δ-restless temporal (s, z)-path of length k' in $\mathcal{G}$. □

Proof of Theorem 3 (ii) Let $I = (\mathcal{G} = (V, (E_i)_{i \in [\ell]}), s, z, k, \Delta)$ be an instance of SHORT RESTLESS TEMPORAL PATH.

To decide whether there is a witness of length k of I being a *yes*-instance, we first use Lemma 8 to compute an instance $I' = (D, s, z, A_M)$ of INDEPENDENT PATH in $O(|\mathcal{G}| \cdot \max\{\Delta, k\})$ time, where we can compute one operation over the field $\mathbb{F}$ of A_M in constant time and the matroid M which is represented by A_M is of rank $k + 1$. Note that I' is a *yes*-instance if and only if there is witness of length k for I being a *yes*-instance. Second, we solve I' by Theorem 5 in $O(2^{\omega(k+1)} \cdot |\mathcal{G}| \cdot \Delta)$ time.

Thus, we have an overall running time of $2^{O(k)} \cdot |\mathcal{G}| \cdot \Delta$. □

Moreover, from Lemma 8 it is intermediately clear that the lower-bounds of Corollary 1 and Theorem 1 translate to INDEPENDENT PATH.

Corollary 3 *INDEPENDENT PATH is NP-hard and unless the ETH fails there is no $2^{o(n)}$ -time algorithm for it, where n is the number of vertices.*

Note that from Theorem 1 we can further deduce that there is not much hope for *fast* or *early* restless temporal paths, that is, restless temporal path that have a small duration or an early arrival time. The instance constructed in the reduction has lifetime $\ell = 3$ and hence the duration as well as the arrival time of any restless temporal path in this instance is at most three. This implies that we presumably cannot find fast or early restless temporal paths efficiently.

5 Computational Complexity Landscape for the Underlying Graph

In this section we investigate the parameterized computational complexity of RESTLESS TEMPORAL PATH when parameterized by structural parameters of the underlying graph. We start by observing that whenever a parametrization forbids path of unbounded length, then we can use Theorem 3 to show fixed-parameter tractability. For example, if we consider the vertex cover number $vc_\downarrow$ of the underlying graph, then we can deduce that any path in the underlying graph and hence any restless temporal path can have length at most $2vc_\downarrow + 1$. Thus, by Theorem 3, we get fixed-parameter tractability of RESTLESS TEMPORAL PATH when parameterized by the vertex cover number of the underlying graph.

Observation 3 RESTLESS TEMPORAL PATH *parameterized by the vertex cover number* $vc_\downarrow$ *of the underlying graph is fixed-parameter tractable.*

From a classification standpoint, we can improve this a little further by observing that the length of a path in the underlying graph can be bounded by $2^{O(td_\downarrow)}$ [53], where $td_\downarrow$ is the treedepth of the underlying graph.

Observation 4 RESTLESS TEMPORAL PATH *parameterized by the treedepth* $td_\downarrow$ *of the underlying graph is fixed-parameter tractable.*

One of the few dark spots of the landscape is the *feedback edge number*[8] of the underlying graph which is resolved in the following way.

Theorem 7 RESTLESS TEMPORAL PATH *can be solved in* $2^{O(f)} \cdot |\mathcal{G}|$ *time, where f is the feedback edge number of the underlying graph.*

By Corollary 1 we know that Theorem 7 is asymptotically optimal, unless ETH fails. In a nutshell, our algorithm to prove Theorem 7 has the following five steps:

1. Exhaustively remove all degree-1 vertices from $G_\downarrow$ (except for s and z).
2. Compute a minimum-cardinality feedback edge set F of the graph $G_\downarrow$.
3. Compute a set $\mathcal{P}$ of $O(f)$ many paths in $G_\downarrow - F$ such that every path in $G_\downarrow - F$ is a concatenation of some paths in $\mathcal{P}$.
4. "Guess" the feedback edges in F and paths in $\mathcal{P}$ of an (s, z)-path in $G_\downarrow$.
5. Verify whether the "guessed" (s, z)-path is a Δ-restless temporal (s, z)-path in $\mathcal{G}$.

First, we show that we can safely remove all (except s and z) degree-one vertices from the underlying graphs $G_\downarrow$.

---

[8] For a given graph $G = (V, E)$ a set $F \subseteq E$ is a *feedback edge set* if $G - F$ does not contain a cycle. The *feedback edge number* of a graph G is the size of a minimum feedback edge set for G.

Reduction Rule 1 (Low Degree Rule) Let $I = (\mathcal{G} = (V, (E_i)_{i \in [\ell]}), s, z, \Delta)$ be an instance of RESTLESS TEMPORAL PATH, $G_\downarrow$ be the underlying graph of $\mathcal{G}$, $v \in V \setminus \{s, z\}$, and $\deg_{G_\downarrow}(v) \le 1$. Then, output $(\mathcal{G} - \{v\}, s, z, \Delta)$.

Lemma 9 *Reduction Rule 1 is safe and can be applied exhaustively in $O(|\mathcal{G}|)$ time.*

Proof Let $I = (\mathcal{G} = (V, (E_i)_{i \in [\ell]}), s, z, \Delta)$ be an instance of RESTLESS TEMPORAL PATH, For the safeness we can observe that a vertex $v \in V \setminus \{s, z\}$ with $\deg_{G_\downarrow}(v) \le 1$ cannot be visited by any Δ-restless temporal (s, z)-path. To apply Reduction Rule 1 exhaustively, we iterate once over the set of time edges to store for each vertex $v \in V \setminus \{s, z\}$ its degree in a counter c_v. Afterwards, we collect all vertices of degree 0 in X and all vertices of degree 1 in V_1. Now we iterate over each vertex $v \in V_1$, remove v from V_1, add v to X, decrement the counter c_u of its neighbor u. If c_u becomes 1 we add u to V_1. Note that this procedure ends after $O(|V|)$ time.

Finally, we iterate one last time over the temporal graph $\mathcal{G}$ to construct the temporal graph $\mathcal{G}' := \mathcal{G} - X$. The instance $(\mathcal{G}', s, z, \Delta)$ of RESTLESS TEMPORAL PATH is the resulting instance when we apply Reduction Rule 1 exhaustively on I. □

Next, we consider a static graph G with no degree-one or degree-zero vertices. Let F be an minimum feedback edge set of G and let V_F be the endpoints of the edges in F, that is $V_F = \{v \in e \mid e \in F\}$. Let $V^{\ge 3}$ be the set of all vertices with a degree greater than two in $G - F$. We can partition the graph $G - F$ into a set $\mathcal{P}$ of $V_F \cup V^{\ge 3}$-*connecting paths*, that are, all paths in $G - F$ who start and end in $V_F \cup V^{\ge 3}$ and have no internal vertices in that set of vertices. Note that all degree-one vertices of $G - F$ are in V_F. Hence, the graph $G - F$ can be partitioned into $V_F \cup V^{\ge 3}$-connecting paths. We can show that $|\mathcal{P}| \in O(f)$.

Lemma 10 *Let G be a graph with no degree-one vertices and F be an minimum feedback edge set of G. The set $\mathcal{P}$ of $V_F \cup V^{\ge 3}$-connecting paths of $G - F$ has size $O(|F|)$ and can be computed in $O(|G|)$ time.*

Proof We can compute the set $\mathcal{P}$ in $O(|G|)$ time as follows. We start with $\mathcal{P} = \emptyset$ and pick any leaf $v \in V(G - F)$ of degree one. Recall that $v \in V_F$ and that $G - F$ is cycle-free. There is at most one vertex $w \in (V^{\ge 3} \cup V_F) \setminus \{v\}$ such that there is a path P between v and w which does not contain internal vertices from $V^{\ge 3} \cup V_F$. Note that also P is unique. We add P to $\mathcal{P}$ and remove $V(P) \setminus \{w\}$ from the graph. Now we repeat this procedure with the next leaf of degree one until the graph has no edges.

It is easy to verify that the number of paths is bounded by the number of vertices in $V^{\ge 3} \cup V_F$. We know that $|V_F|$ is upper-bounded by $2|F|$. It remains to show that $|V^{\ge 3}|$ is in $O(|F|)$.

As shown by Bentert et al. [9, Lemma 2], the number of vertices with degree greater or equal to three is bounded by $3|F|$ in a graph with no degree-one vertices. Hence, the number of $V_F \cup V^{\geq 3}$-connecting paths is bounded by $5|F|$. □

With Lemmas 9 and 10 we can prove Theorem 7.

Proof of Theorem 7 Let $I = (\mathcal{G} = (V, (E_i)_{i \in [\ell]}), s, z, \Delta)$ be an instance of RESTLESS TEMPORAL PATH and $G_{\downarrow}$ be the underlying graph of $\mathcal{G}$. Without loss of generality, we can assume that all vertices in $V(G_{\downarrow}) \setminus \{s, z\}$ have a degree greater than one. If this is initially not the case, then we safely remove all degree-one vertices of the underlying graph exhaustively in $O(|\mathcal{G}|)$ time by Lemma 9.

First we compute an minimum feedback edge set F of $G_{\downarrow}$ in $O(|G_{\downarrow}|)$ time. Then, we compute the set $\mathcal{P}$ of $V_F \cup V^{\geq 3} \cup \{s, z\}$-connecting paths of $G_{\downarrow} - F$ in $O(|G_{\downarrow}|)$ time by Lemma 10. Note that the additional vertices s and z can increase the size of $\mathcal{P}$ by at most four. Now, for any subset of feedback edges $F' \subseteq F$ and $\mathcal{P}' \subseteq \mathcal{P}$, we check whether $F' \cup E(\mathcal{P}')$ form an (s, z)-path P in $G_{\downarrow}$ where $E(\mathcal{P}') := \bigcup_{P' \in \mathcal{P}'} E(P')$. This can be decided in $O(|G_{\downarrow}|)$ time by a simple breadth-first search on $G_{\downarrow}$ starting at the vertex s and using only edges in $F' \cup E(\mathcal{P}')$. Last, we verify whether P forms a Δ-restless temporal (s, z)-path in $\mathcal{G}$. Therefore, we consider the temporal graph $\mathcal{G}_P = (V, (E_i \cap E(P))_{i \in [\ell]})$ which has P as underlying graph. Note that we can construct $\mathcal{G}_P$ in $O(|\mathcal{G}|)$ time, by iterating once over the set of time edges of $\mathcal{G}$. By Lemma 1 we can decide in $O(|\mathcal{G}_P|)$ time whether $\mathcal{G}_P$ has a Δ-restless temporal (s, z)-path.

It is easy to check that the algorithm described above runs in $2^{O(|F|)}|\mathcal{G}|$ time.

Correctness. It remains to show the correctness of the algorithm.

($\Rightarrow$): If our algorithm outputs *yes*, then there is a Δ-restless temporal (s, z)-path in $\mathcal{G}_P$. The temporal graph $\mathcal{G}_P$ contains a subset of the time edges of $\mathcal{G}$, hence the Δ-restless temporal (s, z)-path in $\mathcal{G}_P$ is also present in $\mathcal{G}$. It follows that I is a yes-instance.

($\Leftarrow$): Assume I is a *yes*-instance. Then there exists a Δ-restless temporal (s, z)-path in the temporal graph $\mathcal{G}$. Let $P = \big((v_0, v_1, t_1), \ldots, (v_{k-1}, v_k, t_k)\big)$ be such a path. Hence, $P' = \big(\{v_0, v_1\}, \ldots, \{v_{n-1}, v_n\}\big)$ is an (s, z)-path in the underlying graph $G_{\downarrow}$. Let $F' = F \cap E(P')$. If we remove the edges in F' from P' then what remains is a collection of paths where each path is a concatenation of paths in $\mathcal{P}$. Hence, there exists a subset $\mathcal{P}' \subseteq \mathcal{P}$ such that $F' \cup E(\mathcal{P}') = E(P)$. Thus, we will find P' in $G_{\downarrow}$ and, by Lemma 1, we will correctly verify that this P' forms a Δ-restless temporal (s, z)-path in $\mathcal{G}$. □

The results from Sects. 3 to 5 provide a good picture of the parameterized complexity landscape for RESTLESS TEMPORAL PATH, meaning that for most of the

widely known (static) graph parameters we know whether the problem is in FPT or W[1]-hard or para-NP-hard, see Fig. 2.

Our understanding of the class of temporal graphs where we can solve REST-LESS TEMPORAL PATH efficiently narrows down to the following points. We can check efficiently whether there is a Δ-restless temporal (s, z)-path P in temporal graph $\mathcal{G}$ if

1. there is a bounded number of (s, z)-path in $G_{\downarrow}$ (cf. Theorem 7 and Lemma 1),
2. there is a bound on the length of P (cf. Theorem 3 and Observations 4 and 3).

Apart from that we established with Theorems 1, 2 and Corollary 2 hardness results for temporal graphs having restricted underlying graphs, see Fig. 2.

Finally, we show that we presumably cannot expect to obtain polynomial kernels for all parameters considered so far and most structural parameters of the underlying graph.

Proposition 1 *RESTLESS TEMPORAL PATH parameterized by the number n of vertices does not admit a polynomial kernel for all $\Delta \geq 1$ unless NP $\subseteq$ coNP/poly.*

We employ the OR-cross-composition framework by Bodlaender, Jansen, and Kratsch [15] to refute the existence of a polynomial kernel for a parameterized problem under the assumption that NP $\not\subseteq$ coNP/poly, the negation of which would cause a collapse of the polynomial-time hierarchy to the third level. In order to formally introduce the framework, we need some definitions.

An equivalence relation R on the instances of some problem L is a *polynomial equivalence relation* if

1. one can decide for each two instances in time polynomial in their sizes whether they belong to the same equivalence class, and
2. for each finite set S of instances, R partitions the set into at most $(\max_{x \in S} |x|)^{O(1)}$ equivalence classes.

Using this, we can now define OR-cross-compositions.

Definition 4 An *OR-cross-composition* of a problem $L \subseteq \Sigma^*$ into a parameterized problem P (with respect to a polynomial equivalence relation R on the instances of L) is an algorithm that takes n R-equivalent instances $x_1, \ldots, x_n$ of L and constructs in time polynomial in $\sum_{i=1}^{n} |x_i|$ an instance (x, k) of P such that

1. k is polynomially upper-bounded in $\max_{1 \leq i \leq n} |x_i| + \log(n)$ and
2. (x, k) is a *yes*-instance of P if and only if there is an $i \in [n]$ such that x_i is a *yes*-instance of L.

If an NP-hard problem L OR-cross-composes into a parameterized problem P, then P does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly [15].

Proof of Proposition 1 We provide an OR-cross-composition from RESTLESS TEMPORAL PATH onto itself. We define an equivalence relation R as follows: Two instances $(\mathcal{G} = (V, (E_i)_{i \in [\ell]}), s, z, \Delta)$ and $(\mathcal{G}' = (V', (E'_i)_{i \in [\ell']}), s', z', \Delta')$ are equivalent under R if and only if $|V| = |V'|$ and $\Delta = \Delta'$. Clearly, R is a polynomial equivalence relation.

Now let $(\mathcal{G}_1 = (V_1, (E_{1,i})_{i \in [\ell_1]}), s_1, z_1, \Delta), \ldots, (\mathcal{G}_n = (V_n, (E_{n,i})_{i \in [\ell_n]}), s_n, z_n, \Delta)$ be R-equivalent instances of RESTLESS TEMPORAL PATH. We construct a temporal graph $\mathcal{G}^\star = (V^\star, (E_i^\star)_{i \in [\ell^\star]})$ as follows. Let $|V^\star| = |V_1|$ and $s^\star, z^\star \in V^\star$. We identify all vertices s_i with $i \in [n]$ with each other and with $s^\star$, that is, $s^\star = s_1 = \ldots = s_n$. Analogously, we identify all vertices z_i with $i \in [n]$ with each other and with $z^\star$, that is, $z^\star = z_1 = \ldots = z_n$. We arbitrarily identify the remaining vertices of the instances with the remaining vertices from $V^\star$, that is, let $V^\star \setminus \{s^\star, z^\star\} = V_1 \setminus \{s_1, z_1\} = \ldots = V_n \setminus \{s_n, z_n\}$. Now let $E_1^\star = E_{1,1}, E_2^\star = E_{1,2}, \ldots, E_{\ell_1}^\star = E_{1,\ell_1}$. Intuitively, the first instance $(\mathcal{G}_1 = (V_1, (E_{1,i})_{i \in [\ell_1]})$ essentially forms the first ℓ_1 layers of $\mathcal{G}^\star$. Then we introduce $\Delta + 1$ trivial layers, that is, $E_{\ell_1+1}^\star = E_{\ell_1+2}^\star = \ldots = E_{\ell_1+\Delta+1}^\star = \emptyset$. Then we continue in the same fashion with the second instance and so on. We have that $\ell^\star = \sum_{i \in [n]} \ell_i + (n-1) \cdot (\Delta + 1)$.

This instance can be constructed in polynomial time and the number of vertices is the same as the vertices in the input instances, hence $|V^\star|$ is polynomially upper-bounded by the maximum size of an input instance. Furthermore, it is easy to check that $\mathcal{G}^\star$ contains a Δ-restless temporal $(s^\star, z^\star)$-path if and only if there is an $i \in [n]$ such that $\mathcal{G}_i$ contains a Δ-restless temporal (s_i, z_i)-path. This follows from the fact that all instances are separated in time by $\Delta + 1$ trivial layers, hence no Δ-restless temporal $(s^\star, z^\star)$-path can use time edges from different original instances. Since RESTLESS TEMPORAL PATH is NP-hard (Theorem 1) the result follows. □

6 Timed Feedback Vertex Number

In this section we introduce a new temporal version of the well-studied "feedback vertex number"-parameter. Recall that by Theorem 2 we know that RESTLESS TEMPORAL PATH is W[1] -hard when parameterized by the feedback vertex number of the underlying graph. This motivates studying larger parameters with the goal to obtain tractability results. We propose a new parameter called *timed feedback vertex number* which, intuitively, quantifies the number of vertex appearances that need to be removed from a temporal graph such that its underlying graph becomes cycle-free. Note that having vertex appearances in the deletion set allows us to "guess" *when* we want to enter and leave the deletion set with a Δ-restless temporal (s, z)-path in addition to guessing in which order the vertex appearances are visited.

We remark that there also have been studies of removing (time) edges from temporal graph to destroy *temporal* cycles [36], that is, temporal paths from a

vertex back to itself. Similarly, one also could remove vertex appearances to destroy temporal cycles, resulting in a parameter that is smaller than the timed feedback vertex number and incomparable to the feedback vertex number of the underlying graph. Note that the mentioned parameters aiming at destroying temporal cycles are unbounded in our reductions. We leave the parameterized complexity of RESTLESS TEMPORAL PATH with respect to those parameters open for future research.

Before defining timed feedback vertex number formally, we introduce notation for removing vertex appearances from a temporal graph. Intuitively, when we remove a vertex appearance from a temporal graph, we do not change its vertex set, but remove all time edges that have the removed vertex appearance as an endpoint. Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph and $X \subseteq V \times [\ell]$ a set of vertex appearances. Then we write $\mathcal{G} - X := (V, (E_i')_{i \in [\ell]})$, where $E_i' = E_i \setminus \{e \in E_i \mid \exists (v, i) \in X \text{ with } v \in e\}$. Formally, the timed feedback vertex number is defined as follows.

Definition 5 (Timed Feedback Vertex Number) Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph. A *timed feedback vertex set* of $\mathcal{G}$ is a set $X \subseteq V \times [\ell]$ of vertex appearances such that $G_\downarrow(\mathcal{G} - X)$ 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 $\mathcal{G}$.

We can observe that for any temporal graph the timed feedback vertex number is as least as large as the feedback vertex number of the underlying graph and upper-bounded by the product of the feedback vertex number of the underlying graph and the lifetime. We further remark that the timed feedback vertex number is invariant under reordering the layers. At the end of this section we show how a timed feedback vertex set can be computed efficiently.

The main result of this section is that RESTLESS TEMPORAL PATH is fixed-parameter tractable when parameterized by the timed feedback vertex number of the input temporal graph. To this end, we show the following.

Theorem 8 *Given a timed feedback vertex set X of size x for a temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$, we can decide in $O(6^x x! \cdot \max\{|\mathcal{G}|^3, |V|^4 x^2\})$ time, whether there is a Δ-restless temporal (s, z)-path in $\mathcal{G}$, where $s, z \in V, \Delta \in \mathbb{N}$.*

The algorithm we present to show Theorem 8 solves CHORDAL MULTICOLORED INDEPENDENT SET, where given a chordal graph[9] $G = (V, E)$ and a vertex coloring $c : V \to [k]$, we are asked to decide whether G contains an independent set of size k that contains exactly one vertex of each color. This problem is known to be NP-complete [13, Lemma 2] and solvable in $O(3^k \cdot |V|^2)$ time [10, Proposition 5.6]. Our algorithm for RESTLESS TEMPORAL PATH roughly follows these computation steps:

---

[9] A graph is *chordal* if it does not contain induced cycles of length four or larger.

1. "Guess" which of and in which order the vertex appearances from the timed feedback vertex set appear in the Δ-restless temporal (s,z)-path.
2. Compute the path segments between two timed feedback vertex set vertices by solving a CHORDAL MULTICOLORED INDEPENDENT SET instance.

Algorithm 6.1: FPT algorithm for RESTLESS TEMPORAL PATH parameterized by timed feedback vertex set.

Input: Temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ with $s, z \in V$, timed feedback vertex set X with $s, z \notin \{v \mid (v, t) \in X\}$, and $\Delta \in \mathbb{N}$.
Output: *yes*, if there is a Δ-restless temporal (s, z)-path, otherwise *no*.

1 **for** *each valid partition* $O \uplus I \uplus U = X$ **do**
2 $\quad$ $\mathcal{G}' \leftarrow \mathcal{G} - U$ and $x \leftarrow |I \cup O|$.
3 $\quad$ $\mathcal{T} \leftarrow \mathcal{G}' - (\{v \in V \mid (v, t) \in O \cup I\} \cup \{s, z\})$.
4 $\quad$ **for** *each* Δ-*ordering* $(v_0, t_0) \leq \cdots \leq (v_{x+1}, t_{x+1})$ *of* $I \cup O \cup (\{s, z\} \times \{\bot\})$ **do**
5 $\quad\quad$ $\mathcal{P}_i \leftarrow \emptyset$, for all $i \in [0, x+1]$.
6 $\quad\quad$ **for** $i \leftarrow 1$ **to** $x+1$ **do**
7 $\quad\quad\quad$ **if** $v_{i-1} = v_i$ **then** $\mathcal{P}_i = \{\emptyset\}$.
8 $\quad\quad\quad$ **for** *each* $e_1 = (\{v_{i-1}, w\}, t)$, $e_2 = (\{u, v_i\}, t')$ *of* $\mathcal{G}'$ *where* $v_{i-1} \neq v_i$ **do**
9 $\quad\quad\quad\quad$ $\mathcal{T}' \leftarrow \mathcal{T} + \{e_1, e_2\}$.
10 $\quad\quad\quad\quad$ **if** $\exists$ (t_{i-1}, t_i)-*valid* Δ-*restless temporal* (v_{i-1}, v_i)-*path* P *in* $\mathcal{T}'$ **then** $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{V(P) \setminus \{v_{i-1}, v_i\}\}$.
11

12 $\quad\quad$ $G \leftarrow$ intersection graph of the multiset $\{P^{(i)} \in \mathcal{P}_i \mid i \in [x+1]\}$.
13 $\quad\quad$ Define $c \colon V(G) \rightarrow [x+1], P^{(i)} \mapsto i$.
14 $\quad\quad$ **if** (G, c) *has a multicolored independent set of size* $x + 1$ **then return** *yes*.

15 **return** *no*.

We give a precise description of our algorithm in Algorithm 6.1. Here, a partition $O \uplus I \uplus U$ of a set of vertex appearances X is *valid* if we have $v \neq v'$, for all distinct $(v, t), (v', t') \in I$ and for all distinct $(v, t), (v', t') \in O$. A vertex appearance $(v, t) \in I$ signals that a Δ-restless temporal (s, z)-path arrives in v at time t and $(v, t) \in O$ signals that it departs from v at time t. Let $M := O \cup I \cup (\{s, z\} \times \{\bot\})$. We call a linear ordering $(v_0, t_0) \leq_M \cdots \leq_M (v_{x+1}, t_{x+1})$ of M a Δ-*ordering* if $(v_0, t_0) = (s, \bot)$, $(v_{x+1}, t_{x+1}) = (t, \bot)$, $t_i \leq t_j$ if and only if $i < j \in [x]$, and for all $v \in V$ with $(v, t_i) \in I$ and $(v, t_j) \in O$ it holds that $i + 1 = j$ and $t_i \leq t_j \leq t_i + \Delta$. Moreover, observe that for a vertex appearance $(v, t) \in I$, the Δ-restless temporal (s, z)-path has to depart from v not later than $t + \Delta$ and for vertex appearance $(v, t) \in O$, it has to arrive in v not earlier than $t - \Delta$. To this end, we define the notion of a valid path between two consecutive vertex appearances:

Definition 6 Let $O \uplus I \uplus U$ be a valid partition of X, and let $(v_i, t_i), (v_{i+1}, t_{i+1}) \in I \cup O \cup (\{s, z\} \times \{\bot\})$ with $v_i \neq v_{i+1}$, and P a Δ-restless temporal (v_i, v_{i+1})-path with departure time t_d and arrival time t_a. Then P is (t_i, t_{i+1}, I, O)-*valid* if the following holds true
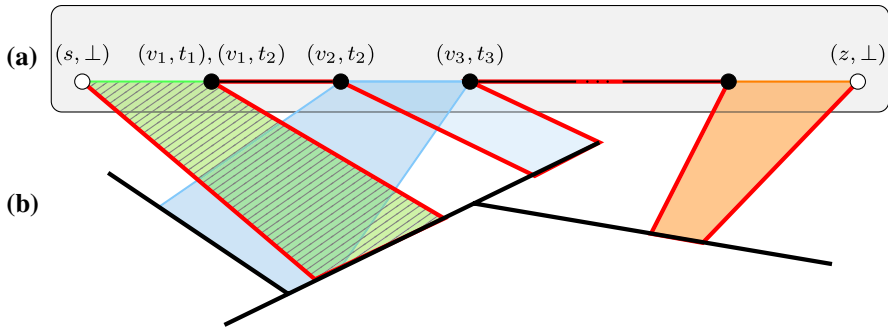
Fig. 6 Illustration of Algorithm 6.1, where (**a**) depicts the set $(\{s, z\} \times \{\bot\}) \cup I \cup O$ and (**b**) sketches the underlying graph of the temporal graph $\mathcal{T}$ which is a forest. The back solid dots correspond to one or two vertex appearances. The Δ-restless temporal (s, z)-path is the red thick path which uses valid (Definition 6) Δ-restless temporal (s, v_1)- and (v_1, v_2)-paths over $\mathcal{T}$

(i) $(v_i, t_i) \in I \implies t_i \leq t_d \leq t_i + \Delta$,

(ii) $(v_i, t_i) \in O \implies t_d = t_i$,

(iii) $(v_{i+1}, t_{i+1}) \in I \implies t_a = t_{i+1}$, and

(iv) $(v_{i+1}, t_{i+1}) \in O \implies t_a \leq t_{i+1} \leq t_a + \Delta$.

If it is clear from context, then we write (t_i, t_{i+1})-valid.

Note that if there exists a (t_i, t_{i+1})-valid Δ-restless temporal (v_i, v_{i+1})-path P_{i+1} and (t_{i+1}, t_{i+2})-valid Δ-restless temporal (v_{i+1}, v_{i+2})-path P_{i+2}, then we can "glue" them together and get a (t_i, t_{i+2})-valid Δ-restless (v_i, v_{i+2})-walk (not necessarily a path). Thus if there exist a valid Δ-restless temporal path between all consecutive pairs in a Δ-ordering which are pairwise vertex disjoint (except for the endpoints), then there exist a Δ-restless temporal (s, z)-path.

The idea of Algorithm 6.1 is that a Δ-restless temporal (s, z)-path P induces a valid partition of the timed feedback vertex set X such that $(v, t) \in I$ if P arrives v at time t, $(v, t) \in O$ if P leaves v at time t, or otherwise $(v, t) \in U$. Furthermore, if we order $M := I \cup O \cup (\{s, z\} \times \{\bot\})$ according to the traversal of P (from s to z), then this is a Δ-ordering such that a subpath P' of P corresponding to consecutive $(v, t), (v', t') \in M$ with $v \neq v'$ is (t, t', I, O)-valid in some temporal graph $\mathcal{T}'$ of Line (9), see Fig. 6.

The algorithm tries all possible partitions of X and all corresponding Δ-orderings. For each of these, we store the vertices $V(P_i) \cap V(\mathcal{T})$ in the family $\mathcal{P}_i$, for all valid Δ-restless temporal (v_{i-1}, v_i)-path, where $(v_{i-1}, t), (v_i, t')$ are two consecutive vertex appearances in the Δ-ordering. Here, we assume without loss of generality that no vertex appearance of s, z is in X. More specifically, for each two consecutive vertex appearances $(v_{i-1}, t), (v_i, t')$ in the Δ-ordering our algorithm iterates over all pairs of time edges leading from (v_{i-1}, t) into the "forest" and from the "forest" back to (v_i, t'). Since this fixes the entry points into the forest in each iteration, any two (t_i, t_{i+1})-valid Δ-restless temporal (v_i, v_{i+1})-paths present in the iteration use the same vertices of the underlying graph. Hence it suffices to check whether one exists. Note that, if we have $|\mathcal{P}_i| \geq 0$ for all $i \in \{1, \ldots, x+1\}$, then

there is a Δ-restless (s, z)-walk in $\mathcal{G}$. Hence, to find a Δ-restless temporal (s, z)-path, we have to find $x + 1$ pair-wise disjoint sets $P_1^{(1)}, \ldots, P_{x+1}^{(x+1)}$ such that $P_i \in \mathcal{P}_i$. Here, we observe that the intersection graph in Line (12) is *chordal* [35] and use an algorithm of Bentert et al. [10] for CHORDAL MULTICOLORED INDEPENDENT SET as a subroutine to find such pairwise-disjoint $P_1^{(1)}, \ldots, P_{x+1}^{(x+1)}$.

Lemma 11 *Algorithm 6.1 runs in $O(6^x x! x^2 \cdot \max\{|\mathcal{G}|^3, |V|^4\})$ time, if $x = |X| \leq |V|$.*

Proof Let $(\mathcal{G}, s, z, X, \Delta)$ be the input of Algorithm 6.1 and $x := |X|$. There are at most 3^x many iterations of the loop in Line (1) and we can check in $O(|\mathcal{G}|)$ time whether a given partition $O \uplus I \uplus U = X$ is valid. Since there are $O(x!)$ are many Δ-orderings of $I \uplus O \uplus (\{s, z\} \times \{\perp\})$, the number of iterations of the loop in Line (4) is also bounded by $O(x!)$. Furthermore, we can check in $O(|\mathcal{G}|)$ time whether a given permutation $((v_i, t_i))_{i=1}^{|U \uplus I|}$ of $U \uplus I$ is a Δ-ordering where $s = v_0, z = v_{|U \uplus I|+1}, t_0 = t_{|U \uplus I|+1} = \perp$. Note that during one iteration of the loop in Line (4) we consider an time edge of $\mathcal{G}$ at most two times as e_1 and two times as e_2 in Line (8). Hence, we have $O(|\mathcal{G}|^2)$ many iteration of the loop in Line (8), during one iteration of the loop in Line (4). Observe that Lemma 1 implies that we can compute a Δ-restless temporal (s, z)-paths in linear time if the underlying graph is a forest. Moreover, each Δ-restless temporal (v_{i-1}, v_i)-path in $\mathcal{T}'$ departs at time t and arrives at t' as e_1 and e_2 are in any temporal path from v_{i-1} and v_i. Hence, Line (11) can be computed in $O(|\mathcal{G}|)$ time. Thus, we can compute Lines (5)–(11) in $O(|\mathcal{G}|^3)$ time. Observe that each set in $\mathcal{P}_i$ is either an empty set or contains the vertices of a path in the forest $G_{\downarrow}(\mathcal{T})$, for all $i \in [x]$. Hence, the intersection graph G has at most $|V|^2 \cdot x$ vertices and is chordal. Thus, Line (14) can be computed in $O(3^x |V|^4 \cdot x^2)$ time with an algorithm of Bentert et at. [10, Proposition 5.6]. This gives an overall running time of $O(6^x x! \cdot \max\{|\mathcal{G}|^3, |V|^4 x^2\})$. □

Lemma 12 *Algorithm 6.1 is correct.*

Proof Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph with $s, z \in V$ and let X be a timed feedback vertex set of $\mathcal{G}$. We assume without loss of generality that s and z have no vertex appearance in X, that is, $s, z \notin \{v \mid (v, t) \in X\}$. If this is not the case, then we can add a new vertex $\hat{s}$ to $\mathcal{G}$ and for each edge $\{s, v\} \in E_i$, we add $\{\hat{s}, s\}$ to E_i. It is clear that there exists a Δ-restless temporal (s, z)-path P if and only if there exists a Δ-restless temporal $(\hat{s}, z)$-path $\hat{P}$. The set X remains a time feedback vertex set because $\hat{s}$ has degree one in the underlying graph $G_{\downarrow}$. Hence, we can now ask for a Δ-restless temporal $(\hat{s}, z)$-path in $\mathcal{G}$. The same holds for the vertex z by a symmetric argument.

We show now that Algorithm 6.1 outputs *yes* if and only if there is a Δ-restless temporal (s, z)-path in $\mathcal{G}$.

($\Rightarrow$): We claim that if we find a multicolored independent set in (G, c), then there is a Δ-restless temporal (s, z)-path in $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$. Let $D = \{P_1^{(1)}, \ldots, P_{x+1}^{(x+1)}\}$ be such an multicolored independent set, let $(v_0, t_0) \leq \cdots \leq (v_{x+1}, t_{x+1})$ be the respective

Δ-ordering when the set D was found, and let $I \uplus O \uplus X$ be the valid partition of X. Hence, P_i represents a (t_{i-1}, t_i, I, O)-valid Δ-restless temporal (v_{i-1}, v_i)-path. Due to D being an independent set, it holds that $P_i^{(i)} \cap P_j^{(j)} = \emptyset$ for all $i \neq j \in [x + 1]$. For all $i \in [x + 1]$ it further holds that if $(v_i, t_i) \in I$, then P_{i-1} arrives in v_i at time t_i and P_i departs from v_i not later than t with $t_i \leq t \leq t_i + \Delta$. If $(v_i, t_i) \in O$, then P_{i-1} arrives in v_i at time t with $t \leq t_i \leq t + \Delta$ and P_i departs from v_i at time t_i. Hence, $P_{i-1} \cdot P_i$ is a Δ-restless temporal (v_{i-1}, v_{i+1})-path in $\mathcal{G}$. Consequently, $P_1 \cdots P_{x+1}$ is a Δ-restless temporal (s, z)-path in $\mathcal{G}$.

($\Leftarrow$): Assume $\mathcal{G}$ contains a Δ-restless temporal (s, z)-path P, then let $I \uplus O \uplus U = X$ be the partition of X that is induced by P. That is, for all $(v, t) \in I$ there exists a time edge (w, v, t) in P, for all $(v, t) \in O$ there exists a time edge (v, w, t) in P, and for all $(v, t) \in U$ there exist no time edge (v, w, t) or (w, v, t) in P. The partition $I \uplus O \uplus U$ is a valid partition. Otherwise there exist two distinct vertex appearances $(v, t), (v, t') \in O$ such that there exist two time edges $(w, v, t), (u, v, t')$ in P indicating that P visits the vertex v twice. The same argument works for two vertex appearances of the same vertex in I. Let $(v_1, t_1) \leq \cdots \leq (v_x, t_x)$ be the vertex appearances in the order in which they are visited by P. It holds that $t_1 \leq \ldots \leq t_x$ and for $i < j \in [x]$ if $v_i = v_j$, then there cannot exist a vertex appearance between v_i and v_j (otherwise P would visit v_i twice). Thus $j = i + 1$, $(v_i, t_i) \in I$, $(v_j, t_j) \in O$, and $t_i \leq t_j \leq t_i + \Delta$. It follows that $(s, \bot) = (v_0, t_0) \leq (v_1, t_1) \leq \cdots \leq (v_x, t_x) \leq (v_{x+1}, t_{x+1}) = (z, \bot)$ is a Δ-ordering of $I \uplus O \uplus (\{s, z\} \times \{\bot\})$. Let P_i be the subpath of P starting in vertex v_{i-1} and ending in v_i for $i \in [x + 1]$. If $v_{i-1} = v_i$, then it holds that P_i is empty and $\mathcal{P}_i = \{\emptyset\}$ (Line (7)). Otherwise, let $P_i = (e_i^{(1)} = (v_{i-1}, v_i^{(1)}, t_i^{(1)}), \ldots, e_i^{(p_i)} = (v_i^{(p_i)}, v_i, t_i^{(p_i)}))$. Note that if $(v_{i-1}, t_{i-1}) \in O$, then $t_i^{(1)} = t_{i-1}$; if $(v_{i-1}, t_{i-1}) \in I$, then $t_{i-1} \leq t_i^{(1)} \leq t_{i-1} + \Delta$; if $(v_i, t_i) \in I$, then $t_i^{(p_i)} = t_i$; and if $(v_i, t_i) \in O$, then $t_i^{(p_i)} \leq t_i \leq t_i^{(p_i)} + \Delta$. Thus path P_i is a (t_{i-1}, t_i, I, O)-valid path in $\mathcal{T} + \{e_i^{(1)}, e_i^{(p_i)}\}$, and hence $V(P_i) \setminus \{v_{i-1}, v_i\} \in \mathcal{P}_i$ (Line (11)). Let $Q_i = V(P_i) \setminus \{v_{i-1}, v_i\}$. It holds that for $i \neq j \in [x + 1]$ the paths P_i and P_j can intersect only in their endpoints because P does not visit a vertex twice and thus $Q_i \cap Q_j = \emptyset$. For each P_i there exists a vertex $P_i^{(i)}$ in the intersection graph G representing with $c(P_i^{(i)}) = i$. For $i, j \in [x + 1]$, there exist no edge $\{P_i^{(i)}, P_j^{(j)}\}$ in G because $Q_i \cap Q_j = \emptyset$. Hence, G has a multicolored independent set $D = \{P_1^{(1)}, \ldots, P_{x+1}^{(x+1)}\}$ of size $x + 1$ and Algorithm 6.1 outputs *yes*. □

To conclude from Theorem 8 the fixed-parameter tractability of Restless Temporal Path parameterized the timed feedback vertex number, we need to compute a timed feedback vertex set efficiently. This is clearly NP-hard, since it generalizes the NP-complete Feedback Vertex Set problem [43]. However, we establish the following possibilities to compute a Feedback Vertex Set.

Theorem 9 *A minimum timed feedback vertex set of temporal graph $\mathcal{G}$ can be computed in $4^x \cdot |\mathcal{G}|^{O(1)}$ time, where x is the timed feedback vertex number of $\mathcal{G}$. Furthermore, there is a polynomial-time 8-approximation for timed feedback vertex set.*

To prove Proposition 9, we first show that a timed feedback vertex set of a temporal graph can be computed via the following problem.

SUBSET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES (SFVS-UV)

Input: Graph $G = (V, E)$, subsets $V^\infty, T \subseteq V$ and integer $k \in \mathbb{N}$.
Question: Is there a $X \subseteq (V \setminus V^\infty)$ of size at most k such that no simple cycle in $G - X$ contains a vertex in T?

Then the 8-approximation for timed feedback vertex set follows from the 8-approximation of Even et al. [27] for WEIGHTED SUBSET FEEDBACK SET.[10] Now we see two ways in the literature to deduce a FPT-algorithm for timed feedback vertex set. One is via a reduction from SFVS-UV to the more general problem GROUP SUBSET FEEDBACK VERTEX SET. The other is through Cygan et al. [22] who claim that SFVS-UV is equivalent (under parameterized reductions for k) to SUBSET FEEDBACK VERTEX SET. The latter is SFVS-UV where $V^\infty = \emptyset$. While the arguments of Cygan et al. [22] only work if $V^\infty \cap T = \emptyset$, we here provide the missing arguments to show that the statement itself is true and hence fill a gap in the literature.

We start with the reduction to SFVS-UV.

Lemma 13 *Given a temporal graph $\mathcal{G}$ and an integer $x \in \mathbb{N}$, we can construct in $O(|\mathcal{G}| + |V|\ell^2)$ time an instance $I = (G, V^\infty, T, x)$ of SFVS-UV such that I is a yes-instance if and only if $\mathcal{G}$ has a timed feedback vertex set of size at most x.*

Construction 1 Given a temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ with underlying graph $G_\downarrow = (V, E)$, we construct the instance $I = (G = (V', E'), V^\infty, T, x)$ of SUBSET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES, where $V' := \bigcup_{v \in V} V_v \cup \bigcup_{e \in E} \cup V_e$ and $E' := \bigcup_{v \in V} E_v \cup \bigcup_{e \in E} E_e \cup \bigcup_{t \in [\ell]} \bigcup_{e \in E_t} E_{(e,t)}$. Here,

$$\forall v \in V : V_v := \{v_i \mid i \in [\ell], v \in e, e \in E_i\},$$

$$\forall e = \{u, w\} \in E : V_e := \{e^{(u)}, e^{(T)}, e^{(w)}\},$$

$$T := \{e^{(T)} \mid e = \{u, w\} \in E\},$$

$$\forall v \in V : E_v := \{\{v_i, v_j\} \mid v_i, v_j \in V_v, v_i \neq v_j\},$$

$$\forall e = \{u, w\} \in E : E_e := \{\{e^{(u)}, e^{(T)}\}, \{e^{(T)}, e^{(w)}\}\}, \text{ and}$$

$$\forall t \in [\ell], \forall e = \{u, w\} \in E_t : E_{(e,t)} := \{\{e^{(u)}, u_t\}, \{w_t, e^{(w)}\} \mid u_t \in V_u, w_t \in V_w\}.$$

Finally we set $V^\infty := \bigcup_{e \in E} \cup V_e$. Consider Fig. 7 for an example.   ♦
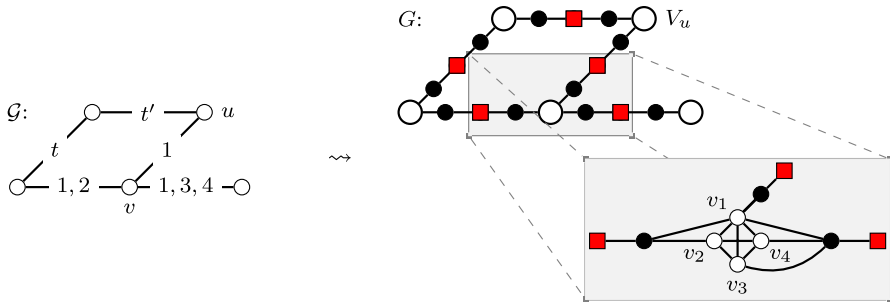

Fig. 7 An illustration of Construction 1 for a temporal graph $\mathcal{G}$ (left) to graph G (right). The set V_u in G of a vertex u in $\mathcal{G}$ is depicted by a large circle. The vertices in V_e of an edge e in the underlying graph of $\mathcal{G}$ are filled. The vertices in T are squared (red)

Proof of Lemma 13 Let $\mathcal{G} = (V, E_1, \ldots, E_\ell)$ be a temporal graph, $x \in \mathbb{N}$, and $I = (G, V^\infty, T, x)$ be the resulting instance from Construction 1. It is easy to check that Construction 1 can be computed in $O(|\mathcal{G}| + |V|\ell^2)$ time.

We now claim that there is a timed feedback vertex set X of size at most x in $\mathcal{G}$ if and only if there is a subset feedback vertex X of size at most x in G such that $X \cap V^\infty = \emptyset$.

($\Rightarrow$): Let X be a timed feed back vertex for $\mathcal{G}$. Then, set $Y := \{v_t \in V_v \mid (v, t) \in X\}$. Hence, $|Y| \leq x$ and $Y \cap V^\infty = \emptyset$. We claim that Y is a subset feedback vertex set for I. Assume towards a contradiction that there is a simple cycle C in $G - Y$ which contains a vertex of T. Furthermore, we assume without loss of generality that there is no shorter cycle in $G - Y$ which contains a vertex of T. Observe that this implies that C does not visit three distinct vertices $v_a, v_b, v_c \in V_v$, for any $v \in V$, because otherwise there is a shorter cycle using one of the edges $\{v_a, v_b\}, \{v_a, v_c\}$ or $\{v_c, v_b\}$ in E_v. Moreover if C visits two distinct vertices $v_a, v_b \in V_v$, then $\{v_a, v_b\} \in E_v$ is part of C, for all $v \in V$, because otherwise there is a shorter cycle using the edge $\{v_a, v_b\}$. Furthermore, for all edge $e \in E$ we have that $V_e \subseteq V(C)$ or $V_e \cap V(C) = \emptyset$, because $G[V_e]$ induces a P_3 and hence using only an endpoint of that P_3 would imply that C visits two vertices $v_a, v_b \in V_v$ without the edge $\{v_a, v_b\}$, for some $v \in V$. Since T only contains the middle vertex $e^{(T)} \in E_e$ of the P_3 induced by $G[V_e]$, we can observe that C is a subdivision of a cycle in $G_\downarrow(\mathcal{G} - X)$ which contradicts that X is a timed feedback vertex set for $\mathcal{G}$.

($\Leftarrow$): Let $Y \subseteq (V' \setminus V^\infty)$ be of size at most x such that no simple cycle in $G' - X$ which contains a vertex in T. We set $X := \{(v, t) \mid v_t \in \bigcup_{u \in V} V_u \cap Y\}$. Hence, X is of size at most x. We claim that X is a timed feedback vertex set for $\mathcal{G}$. Assume towards a contradiction that this is not the case and there is a cycle C in $G_\downarrow(\mathcal{G} - X)$. We now build a cycle in $G - Y$ containing a vertex from T. Note that for each edge e used in C none of the vertices in V_e are in Y, otherwise $V^\infty \cap Y \neq \emptyset$. Hence, set $V_C := \bigcup_{e \in E(C)} E_e$, where $E(C)$ is the edge set of C. Since any two incident edges

$e_1, e_2 \in E(C)$ are in the underlying graph of $\mathcal{G} - X$, we know that there are $t_1, t_2 \in [\ell]$ such that (e_1, t_1) and (e_2, t_2) are time edges of $\mathcal{G} - X$. Hence, for two incident edges $e_1, e_2 \in E(C)$ with $\{v\} = e_1 \cap e_2$ we pick $t_1, t_2 \in [\ell]$ such that (e_1, t_1) and (e_2, t_2) are time edges of $\mathcal{G} - X$ add $v_{t_1}, v_{t_2} \in V_v$ to V_C. Observe that $G[V_C]$ contains a cycle and that $V_C \cap T \neq \emptyset$. Since we constructed V_C from a cycle in the underlying graph of $\mathcal{G} - X$ we have $V_C \cap Y = \emptyset$. This is a contradiction. □

Now we can use the polynomial-time 8-approximation of Even et al. [27] for WEIGHTED SUBSET FEEDBACK VERTEX SET and Lemma 13 to conclude the following.[11]

Corollary 4 *There is a polynomial-time 8-approximation for timed feedback vertex set.*

In the remainder of this section we prove the following.

Lemma 14 *Given an instance $I = (G, V^\infty, T, k)$ of SUBSET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES we can construct in $O(k^2(|V| + |E|))$ time an instance $I' = (G, T', k')$ of SUBSET FEEDBACK VERTEX SET with $k' \leq k$ such that I is a yes-instance if and only if I' is a yes-instance.*

Note that the running time of the algorithm behind Lemma 14 depends only linearly on the size of the graph. The proof of Lemma 14 is deferred to the end of this section. First, we introduce two data reductions rules and then perform the reduction behind Lemma 14 in two steps. We use these data reduction rules to get an equivalent instance where $G[T \cap V^\infty]$ is an independent set. We start by detecting some *no*-instances.

Reduction Rule 2 Let $I = (G, V^\infty, T, k)$ be an instance of SUBSET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES such that there is a vertex u and a simple cycle C intersecting T where $V(C) \setminus (T \cap V^\infty) \subseteq \{u\} \subseteq V^\infty$. Then output a trivial *no*-instance.

We now show that we can detect in linear-time whether Reduction Rule 2 is applicable and that it is *safe*. The latter means that the application of Reduction Rule 2 does not turn a *yes*-instance into a *no*-instance or vice versa.

Lemma 15 *Reduction Rule 2 is safe and can be applied in linear time.*

Proof Since C is a witness that I is a *no*-instance, Reduction Rule 2 is safe. We check whether there is cycle C such that $V(C) \setminus (T \cap V^\infty) = \emptyset$ by simply checking whether $G[T \cap V^\infty]$ is a forest. Assume that $G[T \cap V^\infty]$ is a forest, otherwise we are done and output a trivial *no*-instance. First, we partition $V(G[T \cap V^\infty]) = Q_1 \uplus \cdots \uplus Q_c$

---

[11] Here, vertices get weight ∞ if there are undeletable, and one otherwise.

such that Q_i is a maximal connected component of $G[T \cap V^\infty]$. Clearly, this can be done in linear time. For each connected component Q_i of $G[T \cap V^\infty]$ we first unmark all vertices in $V \setminus (T \cap V^\infty)$. Then we iterate over all vertices $v \in Q_i$ and mark all vertices in $N_G(v) \cap (V^\infty \setminus T)$. If we find a vertex $w \in Q_i$ such that there is a vertex $u \in N_G(w) \cap (V^\infty \setminus T)$ which is already marked, then the path from v to w in Q_i together with u is a cycle C where $V(C) \setminus (T \cap V^\infty) \subseteq \{u\} \subseteq V^\infty$. Hence, we output a trivial *no*-instance in this case. Moreover, if there is some simple cycle C where $V(C) \setminus (T \cap V^\infty) \subseteq \{u\} \subseteq V^\infty$, then all vertices $V(C) \setminus \{u\}$ belong to the same connected component of $G[T \cap V^\infty]$. Thus, the above described procedure will find C. A simple application of the Handshaking Lemma shows that this procedure ends after linear time. $\qquad\square$

The purpose of the next data reduction rule is to merge undeletable terminal vertices which do share an edge.

Reduction Rule 3 Let $I = (G, V^\infty, T, k)$ be an instance of SUB-SET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES such that there is $\{v, w\} \in E$ with $\{v, w\} \subseteq T \cap V^\infty$ and $N_G(v) \cap N_G(w) \cap V^\infty = \emptyset$. Then set $\hat{G} = (V(G - w), E(G - w) \cup \{\{v, u\} \mid \{w, u\} \in E\})$ and $G' := \hat{G} - (N_G(v) \cap N_G(w))$. Output $I' = (G', V^\infty \cap V(G'), T \cap V(G'), k - |N_G(v) \cap N_G(w)|)$.

Note that for each edge in $G[T \cap V^\infty]$, either Reduction Rule 2 or Reduction Rule 3 is applicable. While it is easy to apply Reduction Rule 3 exhaustively in polynomial time, more effort is required to do the same in linear time. To this end, we will construct in linear time an equivalent instance such that Reduction Rules 2 and 3 are not applicable.

Lemma 16 *Given an instance I of* SUBSET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES, *we can compute an equivalent instance I′ of* SUBSET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES *in linear time such that Reduction Rules 2 and 3 are not applicable to I′.*

Proof We first check in linear time by Lemma 15 whether Reduction Rule 2 is applicable. Assume that Reduction Rule 2 is not applicable, otherwise we are done. Hence, $G[T \cap V^\infty]$ is a forest.

We now aim to apply Reduction Rule 3 for all edges in $G[T \cap V^\infty]$ at once. First, we partition $V(G[T \cap V^\infty]) = Q_1 \uplus \cdots \uplus Q_c$ such that Q_i is a maximal connected component of $G[T \cap V^\infty]$. Then we replace Q_i with a fresh vertex q_i. To this end, we construct the graph $G' := (V', E')$ where

$V' := (V \setminus (T \cap V^\infty)) \cup \{q_i \mid i \in [c]\}$ and

$E' := \{\{a, b\} \in E \mid \{a, b\} \subseteq V'\} \cup \{\{q_i, w\} \mid w \in V' \text{ and } \exists v \in Q_i : \{v, w\} \in E\}.$

Note that G' and hence $V'^\infty = (V^\infty \cap V') \cup \{q_i \mid i \in [c]\}$ and $T' = (T \cap V') \cup \{q_i \mid i \in [c]\}$ can be computed in linear time. To compute the

remaining budget k', we set $K = \emptyset$. Then, for each connected component Q_i of $G[T \cap V^\infty]$, we first unmark all vertices. Second, we iterate over all vertices $v \in Q_i$ and mark all vertices in $N_G(v) \setminus (T \cap V^\infty)$. If we find a vertex $w \in Q_i$ such that a vertex $u \in N_G(w) \setminus (T \cap V^\infty)$ which is already marked, then we add u to K, because $G[Q_i \cup \{u\}]$ contains a cycle intersecting T where u is the only deletable vertex. Recall that $u \notin V^\infty$, since Reduction Rule 2 was not applicable. A simple application of the Handshaking Lemma show that this procedure ends after linear time. If $k < |K|$, then we return a trivial *no*-instance, because for each vertex $v \in K$ there is a simple cycle intersecting T where v is the unique vertex not in $V^\infty \cap T$. Otherwise, we output $I' = (G' - K, V'^\infty, T' \setminus K, k' = k - |K|)$. It is easy to verify that Reduction Rule 3 is not applicable in I'. We now claim that I is a *yes*-instance if and only if I' is a *yes*-instance.

($\Rightarrow$): Assume that X is a solution for I. Note that for each vertex $v \in K$, there is a simple cycle intersecting T where v is the unique vertex not in $V^\infty \cap T$. Hence, $K \subseteq X$. Set $X' := X \setminus K$ and observe that $X' \subseteq V(G')$. The set X' is a solution for I', because it is of size at most k' and for each cycle in $G' - K$ which intersects to $T' \setminus K$, we can construct a cycle in G which intersects to T by replacing a vertex $q_j \in \{q_i \mid i \in [c]\}$ with a path in $G[Q_j]$.

($\Leftarrow$): Assume that X' is a solution for I'. We set $X := X' \cup K$ and note that X is of size at most k. We may assume towards a contradiction that X is not a solution for I. Hence there is a simple cycle C in $G - X$ which contains a vertex of T. We now construct a closed walk of C' in $G' - T$ from C by replacing a subpath on vertices in Q_i with q_i, for all $i \in [c]$. Since we only replaced vertices from $T \cap V^\infty$ of I with vertices in $T' \cap V'^\infty$, the closed walk C' contains a simple cycle in $G' - X'$ containing a vertex from T'—a contradiction. $\qquad\square$

We now show an algorithm to dispose all vertices undeletable vertices in T such that the running time dependence only linearly on the size of the graph.

Lemma 17 *Let $I = (G, V^\infty, T, k)$ be an instance of* Subset Feedback Vertex Set with Undeletable Vertices. *We can construct in $O(k(|V| + |E|))$ time an instance $I' = (G', V'^\infty, T', k')$ of* Subset Feedback Vertex Set with Undeletable Vertices *such that*

1. $k' \leq k$,
2. $V'^\infty \setminus T' = \emptyset$,
3. *I is a yes-instance if and only if I' is a yes-instance.*

Proof Let $I = (G, V^\infty, T, k)$ be an instance of Subset Feedback Vertex Set with Undeletable Vertices. Since we aim for a running time of $O(k(|V| + |E|))$, we can assume that Reduction Rules 2 and 3 are not applicable on I, see Lemma 16.

The goal now is to duplicate each vertex in $V^\infty \setminus T$ $k + 1$ times, such that we cannot delete all of them even if there are not in V^∞. Note that we might create many copies of an edge doing this naïvely. To avoid this, we observe that it suffices to replace each maximal connected component in $G[V^\infty \setminus T]$ with $k + 1$ vertices. We partition $V(G[V^\infty \setminus T]) = Q_1 \uplus \cdots \uplus Q_c$ such that Q_i is a maximal connected component of $G[V^\infty \setminus T]$ and we construct $G' := (V', E')$, where

$$V' := (V \setminus (V^\infty \setminus T)) \cup \{q_i^j \mid i \in [c], j \in [k+1]\} \text{ and}$$

$$E' := E(G[V \setminus (V^\infty \setminus T)]) \cup \{\{q_i^j, w\} \mid \exists v \in Q_i : \{v, w\} \in E\}.$$

Since $|\{q_i^j \mid i \in [c], j \in [k+1]\}| \leq |V|(k+1)$ and we copy each edge at most $k+1$ times, G' is constructed after $O(k(|V| + |E|))$ time.

We now claim that I is a *yes*-instance if and only if $I' := (G', T' := V^\infty \cap V(G'), T, k)$ is a *yes*-instance of SUBSET FEEDBACK VERTEX SET WITH UNDELETABLE VERTICES.

($\Rightarrow$): Let X be a solution for I. Then X is also a solution for I', because $X \cap V^\infty = \emptyset$ and for each cycle C' in G' containing a vertex from T, we can construct a closed walk in G by replacing a vertex q_i^j by a path in $G[Q_i]$. This closed walk induces a simple cycle C in G containing a vertex in T. Hence, C' also contains a vertex from X.

($\Leftarrow$): Let X be a solution for I'. We assume without loss of generality that $X \cap \{q_i^j \mid i \in [c], j \in [k+1]\} = \emptyset$. Suppose towards a contradiction that X is not a solution for I. Let C be an cycle in G satisfying $V(C) \cap X = \emptyset \neq V(C) \cap T$.

We construct a closed walk C' in G' from C by replacing each maximal consecutive subpath in C containing only vertices from Q_i with q_i^1, for all $i \in [c]$. Note that C contains a simple cycle which intersects T' but not X—a contradiction. $\square$

Now we are finally ready to prove Lemma 14.

Proof of Lemma 14 First, we apply Lemma 17 on I and hence assume that $V^\infty \setminus T = \emptyset$. Furthermore, by Lemma 16 we assume that Reduction Rules 3 and 2 are not applicable. Thus, $G[V^\infty \cap T]$ is an independent set. We now create $k + 1$ copies of each vertex in $V^\infty \cap T$ such that we cannot remove all of them, even if there are deletable. However, we have to be careful what kind of new cycles this creates. We do the following. Let $V^\infty \cap T = \{v_1, v_2, \ldots, v_p\}$ and take a fresh set of vertices $Q_i := \{q_i^j \mid j \in [k+1]\}$ for each $i \in [p]$. We construct $G' := (V', E')$, where

$$V' := (V \setminus (V^\infty \cap T)) \cup \bigcup_{i=1}^{p} Q_i \cup \{w_i', w_i \mid \{v_i, w\} \in E\}, \text{ and}$$

$$E' := \{\{v, w\} \in E \mid v, w \in V'\} \cup$$
$$\{\{w_i, w\}, \{w_i, w_i'\}, \{w_i', q_i^j\} \mid \{v_i, w\} \in E, i \in [p], j \in [k+1]\}.$$

Output $I' = (G', T', k)$, where $T' := (T \setminus V^\infty) \cup \{w_i \mid \{v_i, w\} \in E\}$. Since $|\{q_i^j \mid i \in [c], j \in [k+1]\}| \le |V|(k+1)$ and we create for each edge at most $k+2$ new edges, G' is constructed after $O(k(|V| + |E|))$ time. Together with the preprocessing of Lemma 17 this gives an overall running time of $O(k^2(|V| + |E|))$.

We now claim that I is a *yes*-instance if and only if I' is a *yes*-instance of SUBSET FEEDBACK VERTEX SET.

($\Rightarrow$): Let X be a solution for I. Observe that $X \subseteq V'$, because $V \setminus V' \subseteq V^\infty$. Assume towards a contradiction that there is a cycle C' in G' such that $V(C') \cap X = \emptyset$. Assume without loss of generality that C' is a shortest of the set of cycles satisfying $V(C') \cap X = \emptyset$. Hence, for each $i \in [p]$ we have that $|V(C') \cap Q_i| \le 1$, and since $G[V^\infty \cap T]$ is an independent set, a vertex $q_i^j \in V(C') \cap Q_i$ has two neighbors w_i', u_i', where $w, u \in V$. Thus, we can construct a cycle C in G by replacing each subpath w_i', q_i^j, u_i' with the vertex $v_i \in T \cap V^\infty$, where $q_i^j \in V(C') \cap Q_i$ and $i \in [p], j \in [k+1]$. This contradicts X being a solution for I.

($\Leftarrow$): Let X' be a solution for I'. For all $i \in [p]$, we assume without loss of generality that

$X' \cap Q_i = \emptyset$:	This can be assumed, because $|Q_i| > k \ge |X'|$ and all vertices in Q_i have the same neighborhood.
$X' \cap \{w_i \mid \{v_i, w\} \in E\} = \emptyset$:	This can be assumed, because these vertices are of degree two and thus a vertex w_i can be replaced by its origin w.
$X' \cap \{w_i' \mid \{v_i, w\} \in E\} = \emptyset$:	Such a vertex w_i' can be replaced by its origin w as well, because for each cycle C' in G' passing through T' which does not include w, we know that $v_i' q_i^j w_i' q_i^{j'} u_i'$ is a subpath of C' for some $v, u \in V$, $q_i^j, q_i^{j'} \in Q_i$, and $j, j' \in [k+1]$. Hence $C' - \{q_i^j, w_i\}$ is also a cycle in G' that contains a vertex from T'. Thus, there is $V(C') \cap (X \setminus \{w_i'\}) \ne \emptyset$.

Hence $X' \subseteq V \setminus V^\infty$. Now assume towards a contradiction that there is a cycle C in G which does not contain a vertex in X'. Hence there $v_i \in V(C) \cap (T \cap V^\infty)$, otherwise C is also a cycle in G'. We construct a cycle C' in G' from C by replacing each subpath u, v_i, w in C with $u, u_i, u_i', v_i, w_i', w_i, w$, for all $v_i \in V(C) \cap (T \cap V^\infty)$. This contradicts X' being a solution because $u_i \in T'$. $\qquad \square$

By using Lemmas 13 and 14 we now can compute a minimum timed feedback vertex set by any known SUBSET FEEDBACK VERTEX SET algorithms. Thus, Lemmas 13 and 14 and Corollary 4 together with the algorithm of Iwata et al. [42] imply Proposition 9.

7 Conclusion

We have analyzed the (parameterized) computational complexity of RESTLESS TEMPORAL PATH, a canonical variant of the problem of finding temporal paths, where the waiting time at every vertex is restricted. Unlike its non-restless counterpart or the "walk-version", this problem turns out to be computationally hard, even in quite restricted cases. On the positive side, we give an efficient algorithm to find short restless temporal paths and we could identify structural parameters of the underlying graph and of the temporal graph itself that allow for fixed-parameter algorithms.

References

1. Akanksha, A., Pallavi, J., Lawqueen, K., Saket, S.: Parameterized complexity of conflict-free matchings and paths. Algorithmica **82**, 1939–1965 (2020)
2. Aigner, M., Ziegler, G.M., Hofmann, K.H., Paul ErdosPaul ErdosErdos, P.: Proofs from the Book. Springer, Berlin (2010)
3. Akrida, E.C., Gąsieniec, L., Mertzios, G.B., Spirakis, P.G.: The complexity of optimal design of temporally connected graphs. Theory Comput. Syst. **61**(3), 907–944 (2017)
4. Akrida, E.C., Czyzowicz, J., Gąsieniec, L., Kuszner, Ł., Spirakis, P.G.: Temporal flows in temporal networks. J. Comput. Syst. Sci. **103**, 46–60 (2019)
5. Akrida, E.C., Mertzios, G.B., Nikoletseas, S., Raptopoulos, C., Spirakis, P.G., Zamaraev, V.: How fast can we reach a target vertex in stochastic temporal graphs? J. Comput. Syst. Sci. **114**, 65–83 (2020)
6. Alman, J., Williams, V.V.: A refined laser method and faster matrix multiplication. In: Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA '21), pp. 522–539. SIAM (2021)
7. Axiotis, K., Fotakis, D.: On the size and the approximability of minimum temporally connected subgraphs. In: Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP '16), pp. 149:1–149:14 (2016)

8. Barabási, A.-L.: Network Science. Cambridge University Press, Cambridge (2016)
9. Bentert, M., Dittmann, A., Kellerhals, L., Nichterlein, A., Niedermeier, R.: An adaptive version of Brandes' algorithm for betweenness centrality. In: 29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16–19, 2018, Jiaoxi, Yilan, Taiwan, pp. 36:1–36:13 (2018)
10. Bentert, M., van Bevern, R., Niedermeier, R.: Inductive k-independent graphs and c-colorable subgraphs in scheduling: a review. J. Sched. **22**(1), 3–20 (2019)
11. Bentert, M., Himmel, A.-S., Nichterlein, A., Niedermeier, R.: Efficient computation of optimal temporal walks under waiting-time constraints. Appl. Netw. Sci. **5**(1), 1–26 (2020)
12. Berman, K.A.: Vulnerability of scheduled networks and a generalization of Menger's theorem. Netw. An Int. J. **28**(3), 125–134 (1996)
13. van Bevern, R., Mnich, M., Niedermeier, R., Weller, M.: Interval scheduling and colorful independent sets. J. Sched. **18**(5), 449–469 (2015)
14. Bhadra, S., Ferreira, F.: Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In: International Conference on Ad-Hoc Networks and Wireless, pp. 259–270. Springer (2003)
15. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM J. Discrete Math. **28**(1), 277–305 (2014)
16. Briggs, P., Torczon, L.: An efficient representation for sparse sets. ACM Lett. Program. Lang. Syst. (LOPLAS) **2**(1–4), 59–69 (1993)
17. Bui-Xuan, B.-M., Ferreira, A., Jarry, A.: Computing shortest, fastest, and foremost journeys in dynamic networks. Int. J. Found. Comput. Sci. **14**(02), 267–285 (2003)
18. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. Int. J. Parallel Emerg. Distrib. Syst. **27**(5), 387–408 (2012)
19. Casteigts, A., Flocchini, P., Godard, E., Santoro, N., Yamashita, M.: On the expressivity of time-varying graphs. Theor. Comput. Sci. **590**, 27–37 (2015)
20. Casteigts, A., Peters, J., Schoeters, J.: Temporal cliques admit sparse spanners. In: Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP '19), volume 132 of LIPIcs, pp. 134:1–134:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2019)
21. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, London (2009)
22. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Subset feedback vertex set is fixed-parameter tractable. SIAM J. Discrete Math. **27**(1), 290–309 (2013)
23. Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Pilipczuk, M., Saurabh, S., Marx, D., Pilipczuk, M.: Parameterized Algorithms. Springer, Berlin (2015)
24. Diestel, R.: Graph Theory, 5th edn, volume 173 of Graduate Texts in Mathematics. Springer, Berlin (2016)
25. Eames, K.T.D., Keeling, M.J. (2003) Contact tracing and disease control. Proc. R. Soc. Lond. Ser. B Biol. Sci. **270**(1533):2565–2571
26. Enright, J., Meeks, K., Mertzios, G., Zamaraev, V.: Deleting edges to restrict the size of an epidemic in temporal networks. In: Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science (MFCS '19), volume 138 of LIPIcs, pp. 57:1–57:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2019)
27. Even, G., Naor, J., Zosin, L.: An 8-approximation algorithm for the subset feedback vertex set problem. SIAM J. Comput. **30**(4), 1231–1252 (2000)
28. Fellows, M.R., Hermelin, D., Rosamond, F., Vialette, S.: On the parameterized complexity of multiple-interval graph problems. Theor. Comput. Sci. **410**(1), 53–61 (2009)
29. Ferretti, L., Wymant, C., Kendall, M., Zhao, L., Nurtay, A., Abeler-Dörner, L., Parker, M., Bonsall, D., Fraser, C.: Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. Science (2020)
30. Fluschnik, T., Molter, H., Niedermeier, R., Renken, M., Zschoche, P.: Temporal graph classes: a view through temporal separators. Theor. Comput. Sci. **806**, 197–218 (2020)
31. Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.. Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4):29:1–29:60 (2016)
32. Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: Representative families of product families. ACM Trans. Algorithms **13**(3), 36:1–36:29 (2017)
33. Fortune, S., Hopcroft, J.E., Wyllie, J.: The directed subgraph homeomorphism problem. Theor. Comput. Sci. **10**, 111–121 (1980)

34. Fredman, M.L., Willard, D.E.: Blasting through the information theoretic barrier with fusion trees. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC '90), pp. 1–7 (1990)

35. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. J. Combin. Theory Ser. B **16**(1), 47–56 (1974)

36. Haag, R., Molter, H., Niedermeier, R., Renken, M.: Feedback edge sets in temporal graphs. In: Proceedings of the 46th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '20), volume 12301 of Lecture Notes in Computer Science, pp. 200–2012. Springer (2020)

37. Holme, P.: Modern temporal network theory: a colloquium. Eur. Phys. J. B **88**(9), 234 (2015)

38. Holme, P.: Temporal network structures controlling disease spreading. Phys. Rev. E **94**(2), 022305 (2016)

39. Holme, P., Saramäki, J. (eds.): Temporal Network Theory. Springer, Berlin (2019)

40. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. **62**(2), 367–375 (2001)

41. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**(4), 512–530 (2001)

42. Iwata, Y., Wahlström, M., Yoshida, Y.: Half-integrality, LP-branching, and FPT algorithms. SIAM J. Comput. **45**(4), 1377–1411 (2016)

43. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103. Springer, Berlin (1972)

44. Kempe, D., Kleinberg, J., Kumar, A.: Connectivity and inference problems for temporal networks. J. Comput. Syst. Sci. **64**(4), 820–842 (2002)

45. Kermack, W.O., McKendrick, A.G.: A contribution to the mathematical theory of epidemics. Proc. R. Soc. Lond. Ser. A **115**(772), 700–721 (1927)

46. Latapy, M., Viard, T., Magnien, C.: Stream graphs and link streams for the modeling of interactions over time. Soc. Netw. Anal. Min. **8**(1), 61 (2018)

47. Lokshtanov, D., Misra, P., Panolan, F., Saurabh, S.:. Deterministic truncation of linear matroids. ACM Trans. Algorithms **14**(2), 14:1–14:20 (2018a)

48. Lokshtanov, D., Misra, P., Panolan, F., Saurabh, S., Zehavi, M.: Quasipolynomial representation of transversal matroids with applications in parameterized complexity. In: Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS '18), pp. 32:1–32:13 (2018b)

49. Marx, D.: A parameterized view on matroid optimization problems. Theor. Comput. Sci. **410**(44), 4471–4479 (2009)

50. Mertzios, G.B., Michail, O., Spirakis, P.G.: Temporal network optimization subject to connectivity constraints. Algorithmica **81**(4), 1416–1449 (2019)

51. Michail, O.: An introduction to temporal graphs: an algorithmic perspective. Internet Math. **12**(4), 239–280 (2016)

52. Molter, H.: Classic graph problems made temporal—a parameterized complexity analysis. Ph.D. thesis, Technische Universität Berlin, December 2020. http://dx.doi.org/10.14279/deposonce-10551

53. Nešetřil, J., Mendez, P.O. De.: Sparsity: Graphs, Structures, and Algorithms. Springer, Berlin (2012)

54. Newman, M.E.J.: Networks. Oxford University Press, Oxford (2018)

55. Oxley, J.G.: Matroid Theory. Oxford University Press, Oxford (1992)

56. Pan, R.K., Saramäki, J.: Path lengths, correlations, and centrality in temporal networks. Phys. Rev. E **84**(1), 016105 (2011)

57. Sorge, M., Weller, M. et al.: The graph parameter hierarchy, 2018. https://manyu.pro/assets/parameter-hierarchy.pdf (2020)

58. Tao, T., Croot III, E., Helfgott, H.: Deterministic methods to find primes. Math. Comput. **81**(278), 1233–1246 (2012)

59. Tovey, C.A.: A simplified NP-complete satisfiability problem. Discret. Appl. Math. **8**(1), 85–89 (1984)

60. Williams, R.: Finding paths of length k in $O^*(2^k)$ time. Inf. Process. Lett. **109**(6), 315–318 (2009)

61. Wu, H., Cheng, J., Ke, Y., Huang, S., Huang, Y., Wu, H.: Efficient algorithms for temporal path computation. IEEE Trans. Knowl. Data Eng. **28**(11), 2927–2942 (2016)

62. Zschoche, P., Fluschnik, T., Molter, H., Niedermeier, R.: The complexity of finding separators in temporal graphs. J. Comput. Syst. Sci. **107**, 72–92 (2020)

Authors and Affiliations

Arnaud Casteigts[1] · Anne-Sophie Himmel[2] · Hendrik Molter[2] · Philipp Zschoche[2]

Arnaud Casteigts
arnaud.casteigts@labri.fr

Anne-Sophie Himmel
anne-sophie.himmel@tu-berlin.de

Hendrik Molter
h.molter@tu-berlin.de

[1]   LaBRI, CNRS, Bordeaux INP, Université de Bordeaux, Bordeaux, France

[2]   Algorithmics and Computational Complexity, Technische Universität Berlin, Berlin, Germany