

1 Finding Temporal Paths under Waiting Time 2 Constraints

3
4
5 **Arnaud Casteigts** 

6 LaBRI, Université de Bordeaux, CNRS, Bordeaux INP, France
7 arnaud.casteigts@labri.fr

8 **Anne-Sophie Himmel** 

9 Technische Universität Berlin, Algorithmics and Computational Complexity, Berlin, Germany
10 anne-sophie.himmel@tu-berlin.de

11 **Hendrik Molter** 

12 Technische Universität Berlin, Algorithmics and Computational Complexity, Berlin, Germany
13 h.molter@tu-berlin.de

14 **Philipp Zschoche** 

15 Technische Universität Berlin, Algorithmics and Computational Complexity, Berlin, Germany
16 zschoche@tu-berlin.de

17 — Abstract —

18 Computing a (short) path between two vertices is one of the most fundamental primitives in graph
19 algorithmics. In recent years, the study of paths in temporal graphs, that is, graphs where the
20 vertex set is fixed but the edge set changes over time, gained more and more attention. A path is
21 time-respecting, or *temporal*, if it uses edges with non-decreasing time stamps.

22 We investigate a basic constraint for temporal paths, where the time spent at each vertex must
23 not exceed a given duration Δ , referred to as *Δ -restless temporal paths*. This constraint arises
24 naturally in the modeling of real-world processes like packet routing in communication networks and
25 infection transmission routes of diseases where recovery confers lasting resistance.

26 While finding temporal paths without waiting time restrictions is known to be doable in
27 polynomial time, we show that the “restless variant” of this problem becomes computationally hard
28 even in very restrictive settings. For example, it is $W[1]$ -hard when parameterized by the feedback
29 vertex number or the pathwidth of the underlying graph. The main question thus is whether the
30 problem becomes tractable in some natural settings. We explore several natural parameterizations,
31 presenting FPT algorithms for three *kinds* of parameters: (1) output-related parameters (here,
32 the maximum length of the path), (2) classical parameters applied to the underlying graph (e.g.,
33 feedback edge number), and (3) a new parameter called *timed feedback vertex number*, which captures
34 finer-grained temporal features of the input temporal graph, and which may be of interest beyond
35 this work.

36 **2012 ACM Subject Classification** Mathematics of computing → Graph algorithms

37 **Keywords and phrases** Temporal graphs, disease spreading, waiting-time policies, restless temporal
38 paths, timed feedback vertex set, NP-hard problems, parameterized algorithms

39 **Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2020.30

40 **Related Version** Full version on arXiv: <https://arxiv.org/abs/1909.06437>.

41 **Funding** *Arnaud Casteigts*: Supported by the ANR, project ESTATE (ANR-16-CE25-0009-03).

42 *Anne-Sophie Himmel*: Supported by the DFG, project FPTinP (NI 369/16).

43 *Hendrik Molter*: Supported by the DFG, project MATE (NI 369/17).



© Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, Philipp Zschoche;
licensed under Creative Commons License CC-BY

31st International Symposium on Algorithms and Computation (ISAAC 2020).

Editors: Yixin Cao, Siu-Wing Cheng, and Minming Li; Article No. 30; pp. 30:1–30:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

44 **1 Introduction**

45 A highly successful strategy to control (or eliminate) outbreaks of infectious diseases is contact
 46 tracing [18]—whenever an individual is diagnosed positively, every person who is possibly
 47 infected by this individual is put into quarantine. However, the viral spread can be too fast
 48 to be traced manually, e.g., if the disease is transmittable in a pre-symptomatic (or asymp-
 49 tomatic) stage, then it seems likely that an individual already caused infection chains when
 50 diagnosed positively. Hence, large-scale digital systems are recommended which use physical
 51 proximity networks based on location and contact data [20]—this allows fast and precise
 52 contact tracing while avoiding the harmful effect of mass quarantines to society [20]. Physical
 53 proximity networks can be understood as temporal graphs¹ [13, 27, 29, 35, 39], that is, graphs
 54 where the vertex set (individuals) remains static but the edge set (physical contacts) may
 55 change over time. In this paper, we extend the literature on reachability in temporal graphs
 56 [4, 5, 10, 11, 26, 33, 38, 47] by a computational complexity analysis of an important variation
 57 of one of the most fundamental combinatorial problems arising in the above mentioned
 58 scenario: given a temporal graph and two individuals s and z , is a chain of infection from s
 59 to z possible, that is, is there a *temporal path* from s to z ? In particular, we use a reachability
 60 concept that captures the standard 3-state *SIR-model* (Susceptible-Infected-Recovered), a
 61 canonical spreading model for diseases where recovery confers lasting resistance [6, 34, 41].

62 In temporal graphs, the basic concepts of paths and reachability are defined in a time-
 63 respecting way [33]: a (strict) temporal path, also called “journey”, is a path that uses edges
 64 with non-decreasing (increasing) time steps. To represent infection chains in the SIR-model,
 65 we restrict the time of “waiting” or “pausing” at each intermediate vertex to a prescribed
 66 duration². We call these paths *restless* temporal paths. They model infection transmission
 67 routes of diseases that grant immunity upon recovery [28]: An infected individual can transmit
 68 the disease until it is recovered (reflected by bounded waiting time) and it cannot be infected a
 69 second time afterwards since then it is immune (reflected by considering path instead of walk:
 70 every vertex can only be visited at most once). Another natural example of restless temporal
 71 paths is delay-tolerant networking among mobile entities, where the routing of a packet is
 72 performed over time and space by storing the packet for a limited time at intermediate nodes.

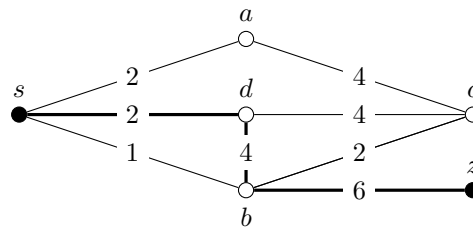
73 In the following we give an example to informally describe our problem setting³. In
 74 Figure 1 we are given the depicted temporal graph, vertices s and z , and the time bound
 75 $\Delta = 2$. We are asked to decide whether there is a restless temporal path from s to z , that
 76 is, a path which visits each vertex at most once and pauses at most Δ units of time between
 77 consecutive hops. Here, (s, d, b, z) is a feasible solution, but (s, b, z) is not because the waiting
 78 time at b exceeds Δ . The walk (s, b, c, d, b, z) is not a valid solution because it visits vertex b
 79 twice. Finally, (s, a, c, d, b, z) is also a feasible solution.

80 **Related work.** Several types of waiting time constraints have been considered in the
 81 temporal graph literature. An empirical study by Pan and Saramäki [43] based on phone
 82 calls datasets observed a threshold in the correlation between the duration of pauses between
 83 calls and the ratio of the network reached over a spreading process. Casteigts et al. [12]
 84 showed a dramatic impact of waiting time constraints to the expressivity of a temporal
 85 graph, when considering such a graph as an automaton and temporal paths as words. In
 86 the context of temporal flows, Akrida et al. [2] considered a concept of “vertex buffers”,

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

² Note that the latent period of an infectious disease can be modeled by subdividing time edges.

³ We refer to Section 2 for a formal definition.



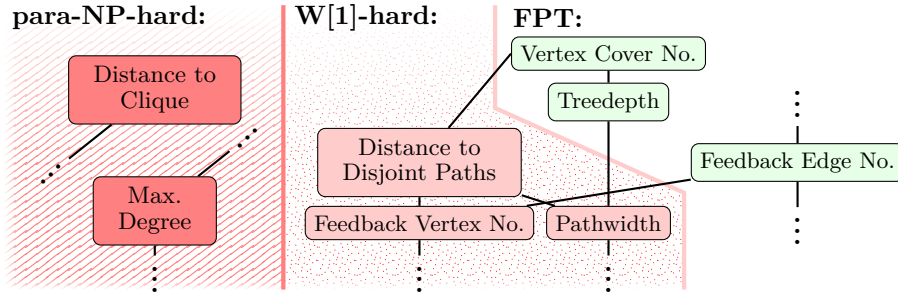
■ **Figure 1** Example of a temporal graph whose edges are labeled with time stamps. Bold edges depict a 2-restless temporal (s, z) -path. (In general, multiple time stamps per edge are possible.)

87 which however pertains to the quantity of information that a vertex can store, rather than a
 88 duration. Enright et al. [19] considered deletion problems for reducing temporal connectivity.
 89 Closer related to our work, Himmel et al. [26] studied a variant of restless temporal paths
 90 where several visits to the same vertex are allowed, i.e., restless temporal *walks*. They showed,
 91 among other things, that such walks can be computed in polynomial time.

92 Many path-related problems have been studied in the temporal setting and the nature
 93 of temporal paths significantly increase the computational complexity of many of them
 94 (compared to their static counterparts). In the temporal setting, reachability is not an
 95 equivalence relation among vertices which makes many problems more complicated. For
 96 example, finding a maximum temporally connected component is NP-hard [10]. We further
 97 have that in a temporal graph, spanning trees may not exist. In fact, even the existence
 98 of *sparse* spanners (i.e., subgraphs with $o(n^2)$ -many edges ensuring temporal connectiv-
 99 ity) is not guaranteed [5], unless the underlying graph is complete [14], and computing a
 100 minimum-cardinality spanner is APX-hard [3, 38]. Yet another example is the problem
 101 of deciding whether there are k disjoint temporal paths between two given vertices. In a
 102 seminal article, Kempe et al. [33] showed that this problem, whose classical analogue is
 103 (again) polynomial-time solvable, becomes NP-hard. They further investigated the related
 104 problem of finding temporal separators, which is also NP-hard [21, 33, 48].

105 **Our contributions.** We introduce the problem RESTLESS TEMPORAL PATH. To get a finer
 106 understanding of the computational complexity of this problem, we turn our attention to its
 107 parametrized complexity. In stark contrast to both restless temporal walks and non-restless
 108 temporal paths, we show that this problem is NP-hard and W[1]-hard when parameterized
 109 by the (vertex deletion) *distance to disjoint paths* of the underlying graph⁴ (Section 3). This
 110 is tight in the sense that the problem can be solved in polynomial time when the underlying
 111 graph is a forest. On the positive side, we explore parameters of three different natures. First,
 112 we show that the problem is fixed-parameter tractable (FPT) for the *number of hops* of the
 113 temporal path (Section 4). We further show that the problem is FPT when parameterized by
 114 the *feedback edge number* of the underlying graph (Section 5). Additionally, we show that the
 115 previously mentioned parameterizations that allow for FPT results, the problem presumably
 116 does not admit a polynomial kernel. Our results provide a fine-grained characterization of
 117 the tractability boundary of the computation of restless temporal paths for parameters of the
 118 underlying graph, as illustrated by the vicinity of the corresponding parameters in Figure 2.
 119 Then, going beyond parameters related to the output and to the underlying graph, we define
 120 a novel temporal version of the classic *feedback vertex number* called *timed feedback vertex*
 121 *number*. Intuitively, it counts the number of vertex *appearances* that have to be removed from

⁴ The *underlying graph* of a temporal graph is a static graph that contains every edge that appears at least once in the temporal graph. A formal definition is given in Section 2.



■ **Figure 2** Relevant part of the hierarchy among classic parameters of the underlying graph (cf. Sorge et al. [44]) for our results for RESTLESS TEMPORAL PATH.

122 the temporal graph such that its underlying graph becomes cycle-free. We show that finding
 123 restless temporal paths is FPT when parameterized by this parameter (Section 6). We believe
 124 that the latter is an interesting turn on events compared to our hardness results. Due to space
 125 constraints, most proofs (marked with ★) are deferred to the related full version on arXiv.⁵

126 2 Preliminaries

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

129 An *interval* is an ordered set $[a, b] := \{n \mid n \in \mathbb{N} \wedge a \leq n \leq b\}$, where $a, b \in \mathbb{N}$. Further,
 130 let $[a] := [1, a]$. We use standard notation and terminology from (static) graph theory [16]
 131 and parameterized complexity theory [15, 17].

132 **Temporal graphs.** An (undirected, simple) *temporal graph* is a tuple $\mathcal{G} = (V, E_1, \dots, E_\ell)$
 133 (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
 134 \mathcal{G} . As with static graphs, we assume all temporal graphs in this paper to be undirected and
 135 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 $E_i = \emptyset$,
 136 then G_i is a *trivial* layer. We call layers G_i and G_{i+1} *consecutive*. We call i a *time step*. If
 137 an edge e is present at time i , that is, $e \in E_i$, we say that e has *time stamp i* . We further
 138 denote $V(\mathcal{G}) := V$. The *underlying graph* $G_\downarrow(\mathcal{G})$ of \mathcal{G} is defined as $G_\downarrow(\mathcal{G}) := (V, \bigcup_{i=1}^{\ell(\mathcal{G})} E_i(\mathcal{G}))$.
 139 To improve readability, we remove (\mathcal{G}) from the introduced notations whenever it is clear
 140 from the context. For every $v \in V$ and every time step $t \in [\ell]$, we denote the *appearance of*
 141 *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)
 142 a *time edge*. For a time edge $(\{v, w\}, t)$ we call the vertex appearances (v, t) and (w, t) its
 143 *endpoints*. We assume that the *size* (for example when referring to input sizes in running
 144 time analyzes) of \mathcal{G} is $|\mathcal{G}| := |V| + \sum_{i=1}^{\ell} |E_i|$, that is, we do not assume that we have compact
 145 representations of temporal graphs. Finally, we write n for $|V|$.

146 A *temporal (s, z) -walk* (or *temporal walk*) of length k from vertex $s = v_0$ to vertex $z = v_k$
 147 in a temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a sequence $P = ((v_{i-1}, v_i, t_i))_{i=1}^k$ of triples that we
 148 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
 149 have that $t_i \leq t_{i+1}$. Moreover, we call P a *temporal (s, z) -path* (or *temporal path*) of length
 150 k if $v_i \neq v_j$ for all $i, j \in \{0, \dots, k\}$ with $i \neq j$. Given a temporal path $P = ((v_{i-1}, v_i, t_i))_{i=1}^k$,
 151 we denote the set of vertices visited by P by $V(P) = \{v_0, v_1, \dots, v_k\}$. A *restless* temporal
 152 path is not allowed to wait an arbitrary amount of time in a vertex, but has to leave any

⁵ <https://arxiv.org/abs/1909.06437>

153 vertex it visits within the next Δ time steps, for some given value of Δ . Analogously to the
 154 non-restless case, a restless temporal walk may visit a vertex multiple times.

155 ► **Definition 1.** A temporal path (walk) $P = ((v_{i-1}, v_i, t_i))_{i=1}^k$ is Δ -restless if $t_i \leq t_{i+1} \leq$
 156 $t_i + \Delta$, for all $i \in [k-1]$. We say that P respects the waiting time Δ .

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

RESTLESS TEMPORAL PATH

158 *Input:* A temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$, two distinct vertices $s, z \in V$, and an
 integer $\Delta \leq \ell$.

Question: Is there a Δ -restless temporal (s, z) -path in \mathcal{G} ?

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

166 **Parameterized complexity.** We use standard notation and terminology from parameter-
 167 ized complexity theory [15] and give here a brief overview of the most important concepts
 168 that are used in this paper. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where
 169 Σ is a finite alphabet. We call the second component the *parameter* of the problem. A
 170 parameterized problem is *fixed-parameter tractable* (in the complexity class FPT) if there
 171 is an algorithm that solves each instance (I, r) in $f(r) \cdot |I|^{O(1)}$ time, for some computable
 172 function f . A decidable parameterized problem L admits a *polynomial kernel* if there is a
 173 polynomial-time algorithm that transforms each instance (I, r) into an instance (I', r') such
 174 that $(I, r) \in L$ if and only if $(I', r') \in L$ and $|(I', r')| \in r^{O(1)}$. If a parameterized problem
 175 is hard for the parameterized complexity class W[1], then it is (presumably) not in FPT.
 176 The complexity classes W[1] is closed under parameterized reductions, which may run in
 177 FPT-time and additionally set the new parameter to a value that exclusively depends on the
 178 old parameter.

179 **Basic observations.** If there exists a Δ -restless temporal (s, z) -path $P = ((v_{i-1}, v_i, t_i))_{i=1}^k$
 180 in a temporal graph \mathcal{G} , then $P' = (\{v_0, v_1\}, \dots, \{v_{k-1}, v_k\})$ is an (s, z) -path in the underlying
 181 graph G_\downarrow . The other direction does not necessarily hold, but for any (s, z) -path in G_\downarrow we
 182 can decide in linear time whether this path forms a Δ -restless temporal (s, z) -path in \mathcal{G} . As
 183 a consequence, we can decide RESTLESS TEMPORAL PATH in linear time for any temporal
 184 graph where there exists a unique (s, z) -path in the underlying graph, in particular, if the
 185 underlying graph is a forest. Some further basic observations are deferred to the full version
 186 of this paper.

187 ► **Lemma 2.** Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph where the underlying graph G_\downarrow is
 188 an (s, z) -path with $s, z \in V$. Then there is an algorithm which computes in $O(|\mathcal{G}|)$ time the
 189 set $\mathcal{A} = \{t \mid \text{there is a } \Delta\text{-restless temporal } (s, z)\text{-path with arrival time } t\}$.

190 **Proof.** Let $V(G_\downarrow) = \{s = v_0, \dots, v_n = z\}$ be the vertices and $E(G_\downarrow) = \{e_1 = \{v_0, v_1\}, \dots, e_n =$
 191 $\{v_{n-1}, v_n\}\}$ be the edges of the underlying path. We further define L_i as the set of layers of
 192 \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

30:6 Finding Temporal Paths under Waiting Time Constraints

a Δ -restless temporal (s, v_i) -path with arrival time t . It holds that $T[v_1] = L_1$. Now we can compute the table entries successively:

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

193 For a table entry $T[v_i]$, we check for each layer $t \in L_i$ whether there exists an Δ -restless
 194 temporal (s, v_{i-1}) -path that arrives in a layer $t' \in T[v_{i-1}]$ such that we can extend the
 195 path to the vertex v_i in layer t without exceeding the maximum waiting time Δ , that is,
 196 $0 \leq t - t' \leq \Delta$. It is easy to see that $T[v_i]$ contains all layers t such that there exists a
 197 Δ -restless temporal (s, v_i) -path with arrival time t . After computing the last entry $T[v_n]$, this
 198 entry contains the set \mathcal{A} of all layers t such that there exists a Δ -restless temporal (s, z) -path
 199 with arrival time t .

200 In order to compute a table entries $T[v_i]$ in linear time, we will need sorted lists of layers
 201 for L_i and $T[v_{i-1}]$ in ascending order. The sorted lists L_i of layers can be computed in $O(|\mathcal{G}|)$:
 202 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
 203 $T[v_{i-1}]$ are lists of layers both in ascending order, then we can compute the table entry $T[v_i]$
 204 in $O(|T[v_{i-1}]| + |L_i|)$ time.

205 Let $T[v_i]$ be initially empty. Let t be the first element in L_i and let t' be the first element
 206 in $T[v_{i-1}]$:

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

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

211 The resulting list $T[v_i]$ is again sorted. Due to this and $T[v_1](= L_1)$ being sorted, we
 212 can assume that $T[v_{i-1}]$ is given as a sorted list of layers when computing $T[v_i]$. Hence, we
 213 can compute each table entry $T[v_i]$ efficiently in $O(|T[v_{i-1}]| + |L_i|)$ time. It further holds
 214 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 $O(|\mathcal{G}|)$
 215 time. \blacktriangleleft

216 2.1 Further Basic Observations

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

221 \triangleright **Observation 3.** Given an instance $I = (\mathcal{G}, s, z, k, \Delta)$ of SHORT RESTLESS TEMPORAL
 222 PATH, we can construct in linear time an instance $I' = (\mathcal{G}', s, z, k, \Delta + 1)$ of SHORT RESTLESS
 223 TEMPORAL PATH such that I is a *yes*-instance if and only if I' is a *yes*-instance.

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

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

230 \triangleright **Observation 4.** RESTLESS TEMPORAL PATH on instances $(\mathcal{G}, s, z, \Delta)$ can be solved in
 231 polynomial time, if $\Delta = 0$ or $\Delta \geq \ell$.

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

235 If $\Delta \geq \ell$, the computation of temporal paths without waiting time constraints was
 236 solved for three possible metrics by Bui-Xuan, Ferreira, and Jarry [11]. Any of the three
 237 corresponding algorithms apply in this case. ◀

238 3 Hardness results for restless temporal paths

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

241 **NP-hardness for few layers.** We start by showing that RESTLESS TEMPORAL PATH is
 242 NP-complete even if the lifetime of the input temporal graph is constant. The reduction
 243 is similar in spirit to the classic NP-hardness reduction for 2-DISJOINT PATHS in directed
 244 graphs by Fortune et al. [24].

245 ▶ **Theorem 5 (★).** RESTLESS TEMPORAL PATH is NP-complete for all finite $\Delta \geq 1$ and
 246 $\ell \geq \Delta + 2$ even if every edge has only one time stamp.

247 The reduction used in the proof of Theorem 5 also yields a running time lower bound
 248 assuming the Exponential Time Hypothesis (ETH) [30, 31].

249 ▶ **Corollary 6.** RESTLESS TEMPORAL PATH does not admit a $f(\ell)^{o(|\mathcal{G}|)}$ -time algorithm for
 250 any computable function f unless the ETH fails.

251 **Proof.** First, note that any 3-SAT formula with m clauses can be transformed into an
 252 equisatisfiable EXACT (3, 4)-SAT formula with $O(m)$ clauses [45]. The reduction presented
 253 in the proof of Theorem 5 produces an instance of RESTLESS TEMPORAL PATH with a
 254 temporal graph of size $|\mathcal{G}| = O(m)$ and $\ell = 3$. Hence an algorithm for RESTLESS TEMPORAL
 255 PATH with running time $f(\ell)^{o(|\mathcal{G}|)}$ for some computable function f would imply the existence
 256 of an $2^{o(m)}$ -time algorithm for 3-SAT. This is a contradiction to the ETH [30, 31]. ◀

257 Furthermore, the reduction behind Theorem 5 can be modified such that it also yields
 258 that RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph has constant
 259 maximum degree or the underlying graph is a clique where one edge ($\{s, z\}$) is missing.

260 ▶ **Corollary 7.** RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph has
 261 all but one edge or maximum degree six.

262 **Proof.** That RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph has
 263 maximum degree six follows directly from the construction used in the proof of Theorem 5.
 264 To show that that RESTLESS TEMPORAL PATH is NP-hard, even if the underlying graph
 265 has all edges except $\{s, z\}$, we reduce from RESTLESS TEMPORAL PATH. Let $I = (\mathcal{G} =$
 266 $(V, (E_i)_{i \in [l]}, s, z, \Delta)$ be an instance of RESTLESS TEMPORAL PATH with $\ell = 3$. We construct
 267 an instance $I' := (\mathcal{G}' = (V, E'_1, E'_2, E'_3, E'_4, E'_5), s, z, \Delta)$ of RESTLESS TEMPORAL PATH, where
 268 $E'_1 = \binom{V \setminus \{s\}}{2}$, $E'_2 := E_1$, $E'_3 := E_2$, $E'_4 := E_3$, and $E'_5 = \binom{V \setminus \{z\}}{2}$. Observe that none of the
 269 edges in $E_1 \cup E_5$ can be used in Δ -restless temporal (s, z) -path. Hence, I is a *yes*-instance
 270 if and only if I' is a *yes*-instance. Furthermore, $E_1 \cup E_5$ contain all possible edges except
 271 $\{s, z\}$. ◀

272 **W[1]-hardness for distance to disjoint paths.** In the following, we show that parame-
 273 terizing RESTLESS TEMPORAL PATH with structural graph parameters of the underlying

274 graph of the input temporal graph presumably does not yield fixed-parameter tractability for
 275 a large number of popular parameters. In particular, we show that RESTLESS TEMPORAL
 276 PATH parameterized by the distance to disjoint paths of the underlying graph is W[1]-hard.
 277 The *distance to disjoint paths* of a graph G is the minimum number of vertices we have to
 278 remove from G such that the remainder of G is a set of disjoint paths. Many well-known
 279 graph parameters can be upper-bounded in the distance to disjoint paths, e.g., pathwidth,
 280 treewidth, and feedback vertex number [44]. Hence, the following theorem also implies that
 281 RESTLESS TEMPORAL PATH is W[1]-hard when parameterized by the pathwidth or the
 282 feedback vertex number of the underlying graph.

283 ► **Theorem 8 (★).** RESTLESS TEMPORAL PATH *parameterized by the distance to disjoint*
 284 *path of the underlying graph is W[1]-hard for all $\Delta \geq 1$ even if every edge has only one time*
 285 *stamp.*

286 4 An FPT-algorithm for short restless temporal path

287 In this section, we discuss how to find *short* restless temporal paths. Recall that in SHORT
 288 RESTLESS TEMPORAL PATH, we are given an additional integer k as input and are asked
 289 whether there exists a Δ -restless temporal (s, z) -path that uses at most k time edges. By
 290 Theorem 5 this problem is NP-hard. Note that in the contact tracing scenario from the
 291 beginning, we can expect to have a small k and a large temporal graph.

292 ► **Theorem 9.** SHORT RESTLESS TEMPORAL PATH *is*
 293 *(i) solvable in $2^k \cdot |\mathcal{G}|^{O(1)}$ time with a constant one-side error,*
 294 *(ii) deterministically solvable in $2^{O(k)} \cdot |\mathcal{G}| \Delta$ time,*

295 Note that we can solve SHORT RESTLESS TEMPORAL PATH such that the running time is
 296 independent from the lifetime ℓ of the temporal graph. To show Theorem 9, we first reduce
 297 the problem to a specific path problem in directed graphs. Then, we apply known algebraic
 298 tools for multilinear monomials detection. Here, Theorem 9 (i) is based on Williams [46]. To
 299 get a deterministic algorithm with a running time almost linear in $|\mathcal{G}|$, we show a different
 300 approach based on representative sets [22] which results in Theorem 9 (ii).

301 **Reduction to directed graphs.** We introduce a so-called Δ - (s, z) -*expansion* for two
 302 vertices s and z of a temporal graph with waiting times. That is, a time-expanded version of
 303 the temporal graph which reduces reachability questions to directed graphs. While similar
 304 approaches have been applied several times [2, 8, 38, 47, 48], to the best of our knowledge,
 305 this is the first time that waiting-times are considered. In a nutshell, the Δ - (s, z) -expansion
 306 has for each vertex v at most ℓ many copies v^1, \dots, v^ℓ and if an (s, z) -dipath visits v^i , it
 307 means that the corresponding Δ -restless temporal (s, z) -walk visits v at time i .

308 ► **Definition 10 (Δ - (s, z) -Expansion).** Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph with two
 309 distinct vertices $s, z \in V$ such that $\{s, z\} \notin E_t$, for all $t \in [\ell]$. Let $\Delta \leq \ell$. The Δ - (s, z) -
 310 expansion of \mathcal{G} is the directed graph $D = (V', E')$ with

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

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

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

318 ► **Lemma 11.** *Given a temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$, two distinct vertices $s, z \in V$, and*
 319 *$\Delta \leq \ell$, we can compute its Δ -(s, z)-expansion D with $|V(D)| \in O(|\mathcal{G}|)$ in $O(|\mathcal{G}| \cdot \Delta)$ time.*

320 **Proof.** Let $V' := \{s, z\}$ and E' be empty in the beginning. We will fill up V' and E'
 321 simultaneously. In order to that efficiently, we will maintain for each vertex $v \in V$ a ordered
 322 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
 323 vertices which are isolated in every layer are irrelevant for the Δ -(s, z)-expansion and can be
 324 erased in linear time.

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

327 ($w = s$): We add v^t to V' , (s, v^t) to E' , and add t to L_v . This can be done in constant
 328 time.

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

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

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

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

341 ► **Lemma 12.** *Let $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ be a temporal graph, $s, z \in V$ two distinct vertices,*
 342 *$\Delta \leq \ell$, and $D = (V', E')$ the Δ -(s, z)-expansion of \mathcal{G} . There is a Δ -restless temporal (s, z)-*
 343 *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*
 344 *all $v \in V$ it holds that $|V'(v) \cap V(P')| \leq 1$.*

345 **Proof.** (\Rightarrow): Let $P = ((s, v_1, t_1), (v_1, v_2, t_2), \dots, (v_{k-1}, z, t_{k'}))$ be a Δ -restless temporal
 346 (s, z)-path in \mathcal{G} of length k . We can inductively construct an (s, z)-dipath P' in D . Observe
 347 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
 348 D . Now let $i \in [k' - 2]$ and P'_i be an ($s, v_i^{t_i}$)-dipath of length i such that

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

351 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 $(v_i^{t_i}, v_{i+1}^{t_{i+1}})$.
 352 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
 353 $(v_i^{t_i}, v_{i+1}^{t_{i+1}}) \in E'$, because we have $0 \leq t_{i+1} - t_i \leq \Delta$. Observe that

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

356 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
 357 be extended (in a similar way) to an (s, z)-dipath of length k such that for all $v \in V$ it holds
 358 that $|V'(v) \cap V(P')| \leq 1$.

359 (\Leftarrow): Let P' be a (s, z)-dipath in D of length k such that for all $v \in V$ it holds that
 360 $|V'(v) \cap V(P')| \leq 1$. 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
 361 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
 362 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 $i \in [k - 2]$.

30:10 Finding Temporal Paths under Waiting Time Constraints

363 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
 364 $(\{v_k, z\}, t_k)$ in \mathcal{G} with $0 \leq t_k - t_{k-1} \leq \Delta$. Hence, $P = ((s, v_1, t_1), (v_1, v_2, t_2), \dots, (v_{k-1}, z, t_k))$
 365 is a Δ -restless temporal (s, z) -walk of length k in \mathcal{G} . Finally, $|V'(v) \cap V(P')| \leq 1$, for all
 366 $v \in V$, 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
 367 temporal (s, z) -path of length k . ◀

368 **Obtaining Theorem 9 (i).** We now adapt the algorithm of Williams [46] to our specific
 369 needs. To this end, we introduce some standard notation from algebraic theory.

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

378 ▶ **Lemma 13 (★).** *Let $k \in \mathbb{N}$ and $D = (V, A)$ be a directed graph with partition $V = \bigsqcup_{i=0}^n V_i$,
 379 where $V_0 = \{s\}$ and $V_n = \{z\}$. Then, there is an arithmetic circuit C representing a
 380 polynomial $Q(X)$ of degree at most $k + 1$ such that $Q(X)$ has a multilinear⁶ monomial of
 381 degree at most $k + 1$ if and only if there is an (s, z) -path P of length at most k in D where
 382 $|V(P) \cap V_i| \leq 1$ for all $i \in [n]$. Moreover, $|X| = n + 1$, C is of size $O(k(n + |A|))$, has no
 383 scalar multiplication, and all product gates in have in-degree two.*

384 The polynomial in Lemma 13 is recursively defined as $Q(X) = x_\perp Q_z^k$ over variables $X =$
 385 $\{x_\perp, x_0\} \cup \{x_i \mid i \in [n]\}$, where $Q_s^0 := x_0$, $Q_v^0 := x_\perp$ for all $v \in V \setminus \{s\}$, and for
 386 all $v \in V$ and $j \in [k]$ we define $Q_v^j = \sum_{(u,v) \in A} Q_u^{j-1} x_i$, where $v \in V_i$. The idea of the
 387 polynomial is similar to the one of Williams [46], but here instead of having one variable for
 388 each vertex we just have one variable for all vertices in one part of the partition of V . Now
 389 we can apply the following result of Williams [46].

390 ▶ **Theorem 14 ([46]).** *Let $Q(X)$ be a polynomial of degree at most k , represented by an
 391 arithmetic circuit of size n with no scalar multiplications and where all product gates have
 392 in-degree two. There is a randomized algorithm that runs in $2^k n^{O(1)}$ time, outputs yes with
 393 high probability ($\geq 1/5$) if there is a multilinear term in the sum-product expansion of Q , and
 394 always outputs no if there is no multilinear term.*

395 Theorem 9 (i) follows from Lemmata 11 to 13 and Theorem 14. This can be derandomized
 396 by Theorem 5.2 of Fomin et al. [23] resulting in $O(3.841^k \cdot (|\mathcal{G}| \Delta)^2 |V| \log |V|)$ time algorithm.
 397 We now show how to improve the polynomial part of a deterministic algorithm.

398 **Obtaining Theorem 9 (ii).** To show Theorem 9 (ii), we first note that in the (s, z) -
 399 expansion of an (s, z) -path P in the directed graph describes a Δ -restless temporal (s, z) -path
 400 exactly when $V(P)$ is an *independent set of some specific matroid*. We then show an algorithm
 401 to find such a path P (if there is one). To this end, we introduce a problem, INDEPENDENT
 402 PATH, and some standard terminology from matroid theory [42]. A pair (U, \mathcal{I}) , where U is
 403 the *ground set* and $\mathcal{I} \subseteq 2^U$ is a family of *independent sets*, is a *matroid* if the following holds:
 404 $\emptyset \in \mathcal{I}$; 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

⁶ No variable occurs to a power of two or higher.

405 $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$. An inclusion-wise maximal independent set $A \in \mathcal{I}$ of a
 406 matroid $M = (U, \mathcal{I})$ is a *basis*. The cardinality of the bases of M is called the *rank* of M .
 407 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
 408 matroid (U, \mathcal{I}) is *linear* or *representable over a field \mathbb{F}* if there is a matrix A with entries in \mathbb{F}
 409 and the columns labeled by the elements of U such that $S \in \mathcal{I}$ if and only if the columns of A
 410 with labels in S are linearly independent over \mathbb{F} . Such a matrix A is called a *representation*
 411 of (U, \mathcal{I}) . Now we are ready to state the INDEPENDENT PATH problem. Given a digraph
 412 $D = (V, E)$, two distinct vertices $s, z \in V$, and a representation A_M of a matroid $M = (V, \mathcal{I})$
 413 of rank r over a finite field \mathbb{F} , we are asked whether there is an (s, z) -dipath P of length at
 414 most k in D such that $V(P) \in \mathcal{I}$.

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

417 Agrawal et al. [1] studied, independently from us, a similar problem where the edges
 418 of the path shall be an independent set of a matroid. To show Theorem 9, we need a
 419 single-exponential algorithm which has only a linear dependency on the input size. To this
 420 end, we based on representative families [22] show the following.

421 **► Theorem 15 (★).** *An instance (D, s, z, A_M) of INDEPENDENT PATH can be solved in*
 422 *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*
 423 *is the number of edges in D , and $\omega < 2.373$ is the matrix multiplication exponent.*

424 Observe that by Lemma 12, there is a Δ -restless temporal (s, z) -path in the temporal
 425 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}
 426 such that $V(P)$ is an independent set in the *partition matroid*⁷ $M = (V', \{X \subseteq V' \mid \forall v \in$
 427 $V: |X \cap V'(v)| \leq 1\})$. Note that M is of rank $|V|$ and hence too large to show Theorem 9
 428 with Theorem 15.

429 A *k-truncation* of a matroid (U, \mathcal{I}) is a matroid $(U, \{X \in \mathcal{I} \mid |X| \leq k\})$ such that all
 430 independent sets are of size at most k . The k -truncation of a linear matroid is also a linear
 431 matroid [37]. In our reduction from SHORT RESTLESS TEMPORAL PATH to INDEPENDENT
 432 PATH we use a $(k + 1)$ -truncation of matroid M . Two general approaches are known to
 433 compute a representation for a k -truncation of a linear matroid—one is randomized [37] and
 434 one is deterministic [36].⁸ Both approaches require a large field implying that one operation
 435 over that field is rather slow. However, for our specific matroid we employ the Vandermonde
 436 matrix to compute a representation over a small finite field. Note that we would not get a
 437 running time linear in the input size by applying the algorithm of Lokshtanov et al. [36] or
 438 Marx [37] on M .

439 **► Lemma 16 (★).** *Given a universe U of size n , a partition $P_1 \uplus \dots \uplus P_q = U$, and*
 440 *an integer $k \in \mathbb{N}$, we can compute in $O(kn)$ time a representation A_M for the matroid*
 441 $M = \left(U, \left\{ X \subseteq U \mid |X| \leq k \text{ and } \forall i \in [q]: |X \cap P_i| \leq 1 \right\} \right)$, *where A_M is defined over a finite*
 442 *field \mathbb{F} and one operation over \mathbb{F} takes constant time.*

443 Now Theorem 9 (ii) follows from Lemmata 11, 12 and 16 and Theorem 15. We refer to the
 444 full version for more details.

⁷ Partition matroids are linear [37].

⁸ For both algorithms, a representation of the original matroid must be given.

445 **5 Computational complexity landscape for the underlying graph**

446 In this section we investigate the parameterized computational complexity of RESTLESS
 447 TEMPORAL PATH when parameterized by structural parameters of the underlying graph. We
 448 start by showing fixed-parameter tractability results for parameterizations that are directly
 449 implied by Theorem 15. We can observe that any path of a graph can contain at most
 450 twice as many vertices as the vertex cover number of the graph (plus one), since we cannot
 451 visit two vertices outside of the vertex cover directly after another. Essentially the same
 452 observation can be made in the temporal setting. If we consider the vertex cover number
 453 vc_{\downarrow} of the underlying graph, we can deduce that any restless temporal path can have length
 454 at most $2vc_{\downarrow} + 1$. From a classification standpoint, we can improve this a little further by
 455 observing that the length of any restless temporal path is bounded by the length of any
 456 path of the underlying graph. The length of a path in the underlying graph can be bounded
 457 by $2^{O(td_{\downarrow})}$ [40], where td_{\downarrow} is the treedepth of the underlying graph.

458 \triangleright **Observation 17.** RESTLESS TEMPORAL PATH parameterized by the treedepth td_{\downarrow} of the
 459 underlying graph is fixed-parameter tractable.

460 One of the few dark spots of the landscape is the *feedback edge number*⁹ of the underlying
 461 graph which is resolved in the following way.

462 \blacktriangleright **Theorem 18 (★).** RESTLESS TEMPORAL PATH can be solved in $2^{O(f)} \cdot |\mathcal{G}|$ time, where f
 463 is the feedback edge number of the underlying graph.

464 We note that, by Corollary 6, Theorem 18 is asymptotically optimal, unless ETH fails. In a
 465 nutshell, our algorithm to prove Theorem 18 has the following five steps:

- 466 1. Exhaustively remove all degree-1 vertices from G_{\downarrow} (except for s and z).
- 467 2. Compute a minimum-cardinality feedback edge set F of the graph G_{\downarrow} .
- 468 3. Compute a set \mathcal{P} of maximal paths in $G_{\downarrow} - F$ and note that $|\mathcal{P}| = O(f)$.
- 469 4. “Guess” the feedback edges in F and paths in \mathcal{P} of an (s, z) -path in G_{\downarrow} .
- 470 5. Verify whether the “guessed” (s, z) -path is a Δ -restless temporal (s, z) -path in \mathcal{G} .

471 The results from Sections 3 to 5 provide a good picture of the parameterized complexity
 472 landscape for RESTLESS TEMPORAL PATH, meaning that for most of the widely known (static)
 473 graph parameters we know whether the problem is in FPT or W[1]-hard or para-NP-hard,
 474 see Figure 2.

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

- 478 1. there is a bounded number of (s, z) -path in G_{\downarrow} , or (see Theorem 18 and Lemma 2)
- 479 2. there is a bound on the length of P . (see Theorem 9 and Observation 17)

480 Apart from that we established with Theorems 5 and 8 and Corollary 7 hardness results for
 481 temporal graphs having restricted underlying graphs, see Figure 2.

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

484 \blacktriangleright **Proposition 19 (★).** RESTLESS TEMPORAL PATH parameterized by the number n of
 485 vertices does not admit a polynomial kernel for all $\Delta \geq 1$ unless $NP \subseteq coNP/poly$.

⁹ 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 .

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 8 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.

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 20 (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 21 (★). *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 21 solves CHORDAL MULTICOLORED INDEPENDENT SET, where given a chordal graph¹⁰ $G = (V, E)$ and a vertex coloring $c: V \rightarrow [k]$, 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 [9, Lemma 2] and solvable in $O(3^k \cdot |V|^2)$ time [7, Proposition 5.6]. Our algorithm for RESTLESS TEMPORAL PATH roughly follows these computation steps:

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.

¹⁰A graph is *chordal* if it does not contain induced cycles of length four or larger.

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    $\mathcal{G}' \leftarrow \mathcal{G} - U$  and  $x \leftarrow |I \cup O|$ .
3    $\mathcal{T} \leftarrow \mathcal{G}' - (\{v \in V \mid (v, t) \in O \cup I\} \cup \{s, z\})$ .
4   for each  $\Delta$ -ordering  $(v_0, t_0) \leq \dots \leq (v_{x+1}, t_{x+1})$  of  $I \cup O \cup (\{s, z\} \times \{\perp\})$  do
5      $\mathcal{P}_i \leftarrow \emptyset$ , for all  $i \in [x+1]$ .
6     for  $i \leftarrow 1$  to  $x+1$  do
7       if  $v_{i-1} = v_i$  then  $\mathcal{P}_i = \{\emptyset\}$ .
8       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          $\mathcal{T}' \leftarrow \mathcal{T} + \{e_1, e_2\}$ .
10        if  $\exists (t_{i-1}, t_i)$ -valid  $\Delta$ -restless temporal  $(v_{i-1}, v_i)$ -path  $P$  in  $\mathcal{T}'$  then
11           $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{V(P) \setminus \{v_{i-1}, v_i\}\}$ .
12       $G \leftarrow$  intersection graph of the multiset  $\{P^{(i)} \in \mathcal{P}_i \mid i \in [x+1]\}$ .
13      Define  $c: V(G) \rightarrow [x+1], P^{(i)} \mapsto i$ .
14      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 $M := O \cup I \cup (\{s, z\} \times \{\perp\})$. We call a linear ordering $(v_0, t_0) \leq_M \dots \leq_M (v_{x+1}, t_{x+1})$ of M a Δ -ordering if $(v_0, t_0) = (s, \perp)$, $(v_{x+1}, t_{x+1}) = (t, \perp)$, $t_i \leq t_j$ 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 t 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 22.** Let $O \uplus I \uplus U$ be a valid partition of X , and let $(v_i, t_i), (v_{i+1}, t_{i+1}) \in I \cup O \cup (\{s, z\} \times \{\perp\})$ with $v_i \neq v_{i+1}$ and $t_i \leq t_{i+1}$, and P_i a Δ -restless temporal (v_i, v_{i+1}) -path with departure time t_d and arrival time t_a . Then P_i is (t_{i-1}, t_i, I, O) -valid if the following holds true

(i) $(v_{i-1}, t_{i-1}) \in I \implies t_{i-1} \leq t_d \leq t_{i-1} + \Delta$,

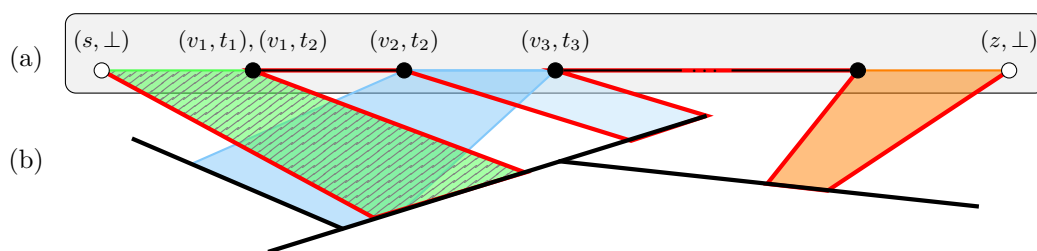
(ii) $(v_{i-1}, t_{i-1}) \in O \implies t_d = t_{i-1}$,

(iii) $(v_i, t_i) \in I \implies t_a = t_i$, and

(iv) $(v_i, t_i) \in O \implies t_a \leq t_i \leq t_a + \Delta$.

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



■ **Figure 3** Illustration of Algorithm 6.1, where (a) depicts the set $(\{s, z\} \times \{\perp\}) \cup I \cup O$ 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 22) Δ -restless temporal (s, v_1) - and (v_1, v_2) -paths over \mathcal{T} .

553 are pairwise vertex disjoint (except for the endpoints), then there exist a Δ -restless temporal
554 (s, z) -path.

555 The idea of Algorithm 6.1 is that a Δ -restless temporal (s, z) -path P induces a valid
556 partition of the timed feedback vertex set X such that $(v, t) \in I$ if P arrives v at time
557 t , $(v, t) \in O$ if P leaves v at time t , or otherwise $(v, t) \in U$. Furthermore, if we order
558 $M := I \cup O \cup (\{s, z\} \times \{\perp\})$ according to the traversal of P (from s to z), then this is a
559 Δ -ordering such that a subpath P' of P corresponding to consecutive $(v, t), (v', t') \in M$ with
560 $v \neq v'$ is (t, t', I, O) -valid in some temporal graph \mathcal{T}' of Line (9), see Figure 3.

561 The algorithm, tries all possible partitions of X and all corresponding Δ -orderings. For
562 each of these, we store for all valid Δ -restless temporal (u, w) -path P' of two consecutive
563 $(u, t), (w, t')$ the vertices $V(P) \cap V(\mathcal{T})$ in the family \mathcal{P}_i . Here, we assume without loss of
564 generality that no vertex appearance of s, z is in X . Note that, if we have $|\mathcal{P}_i| \geq 0$ for
565 all $i \in \{1, \dots, x+1\}$, then there is Δ -restless (s, z) -walk in \mathcal{G} . Hence, to find a Δ -restless
566 temporal (s, z) -path, we have to find $x+1$ pair-wise disjoint sets $P_1^{(1)}, \dots, P_{x+1}^{(x+1)}$ such that
567 $P_i \in \mathcal{P}_i$. Here, we observe that the intersection graph of in Line (12) is *chordal* [25] and use
568 an algorithm of Bentert et al. [7] for CHORDAL MULTICOLORED INDEPENDENT SET as a
569 subroutine to find such pairwise-disjoint $P_1^{(1)}, \dots, P_{x+1}^{(x+1)}$.

570 To conclude from Theorem 21 the fixed-parameter tractability of RESTLESS TEMPORAL
571 PATH parameterized the timed feedback vertex number, we need to compute a timed feedback
572 vertex set efficiently. This is clearly NP-hard, since it generalizes the NP-complete FEEDBACK
573 VERTEX SET problem [32]. However, we establish the following possibilities to compute a
574 FEEDBACK VERTEX SET.

575 ► **Theorem 23** (★). *A minimum timed feedback vertex set of temporal \mathcal{G} can be computed*
576 *in $4^x \cdot |\mathcal{G}|^{O(1)}$ time, where x is the timed feedback vertex number of \mathcal{G} . Furthermore, there is*
577 *a polynomial-time 8-approximation for timed feedback vertex set.*

578 7 Conclusion

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

586 — References

- 587 1 Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, and Saket Saurabh. Parameterized
588 complexity of conflict-free matchings and paths. *Algorithmica*, pages 1–27, 2020. 11
- 589 2 Eleni C. Akrida, Jurek Czyzowicz, Leszek Gąsieniec, Łukasz Kuszner, and Paul G. Spirakis.
590 Temporal flows in temporal networks. *Journal of Computer and System Sciences*, 103:46–60,
591 2019. 2, 8
- 592 3 Eleni C Akrida, Leszek Gąsieniec, George B Mertzios, and Paul G Spirakis. The complexity of
593 optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944,
594 2017. 3
- 595 4 Eleni C. Akrida, George B. Mertzios, Sotiris Nikolettseas, Christoforos Raptopoulos, Paul G.
596 Spirakis, and Viktor Zamaraev. How fast can we reach a target vertex in stochastic temporal
597 graphs? *Journal of Computer and System Sciences*, 114:65 – 83, 2020. 2
- 598 5 Kyriakos Axiotis and Dimitris Fotakis. On the size and the approximability of minimum
599 temporally connected subgraphs. In *Proceedings of the 43rd International Colloquium on*
600 *Automata, Languages, and Programming (ICALP '16)*, pages 149:1–149:14, 2016. 2, 3
- 601 6 Albert-László Barabási. *Network Science*. Cambridge University Press, 2016. 2
- 602 7 Matthias Bentert, René van Bevern, and Rolf Niedermeier. Inductive k -independent graphs
603 and c -colorable subgraphs in scheduling: a review. *Journal of Scheduling*, 22(1):3–20, 2019.
604 13, 15
- 605 8 Kenneth A Berman. Vulnerability of scheduled networks and a generalization of Menger’s
606 theorem. *Networks: An International Journal*, 28(3):125–134, 1996. 8
- 607 9 René van Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. Interval scheduling
608 and colorful independent sets. *Journal of Scheduling*, 18(5):449–469, 2015. 13
- 609 10 Sandeep Bhadra and Afonso Ferreira. Complexity of connected components in evolving graphs
610 and the computation of multicast trees in dynamic networks. In *International Conference on*
611 *Ad-Hoc Networks and Wireless*, pages 259–270. Springer, 2003. 2, 3
- 612 11 B.-M. Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost
613 journeys in dynamic networks. *International Journal of Foundations of Computer Science*,
614 14(02):267–285, 2003. 2, 7
- 615 12 Arnaud Casteigts, Paola Flocchini, Emmanuel Godard, Nicola Santoro, and Masafumi Ya-
616 mashita. On the expressivity of time-varying graphs. *Theoretical Computer Science*, 590:27–37,
617 2015. 2
- 618 13 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying
619 graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed*
620 *Systems*, 27(5):387–408, 2012. 2
- 621 14 Arnaud Casteigts, Joseph Peters, and Jason Schoeters. Temporal cliques admit sparse spanners.
622 In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*
623 *(ICALP '19)*, volume 132 of *LIPICs*, pages 134:1–134:14. Schloss Dagstuhl–Leibniz-Zentrum
624 für Informatik, 2019. 3
- 625 15 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin
626 Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. 4,
627 5
- 628 16 Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate Texts in Mathematics*.
629 Springer, 2016. 4
- 630 17 Rodney G Downey and Michael R Fellows. *Fundamentals of Parameterized Complexity*.
631 Springer, 2013. 4
- 632 18 Ken TD Eames and Matt J Keeling. Contact tracing and disease control. *Proceedings of the*
633 *Royal Society of London. Series B: Biological Sciences*, 270(1533):2565–2571, 2003. 2
- 634 19 Jessica Enright, Kitty Meeks, George Mertzios, and Viktor Zamaraev. Deleting edges to
635 restrict the size of an epidemic in temporal networks. In *Proceedings of the 44th International*
636 *Symposium on Mathematical Foundations of Computer Science (MFCS '19)*, volume 138 of
637 *LIPICs*, pages 57:1–57:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019. 3

- 638 20 Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner,
639 Michael Parker, David Bonsall, and Christophe Fraser. Quantifying SARS-CoV-2 transmission
640 suggests epidemic control with digital contact tracing. *Science*, 2020. 2
- 641 21 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche.
642 Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*,
643 806:197–218, 2020. 3
- 644 22 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation
645 of representative families with applications in parameterized and exact algorithms. *Journal of*
646 *the ACM*, 63(4):29:1–29:60, 2016. 8, 11
- 647 23 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative
648 families of product families. *ACM Transactions on Algorithms*, 13(3):36:1–36:29, 2017. 10
- 649 24 Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism
650 problem. *Theor. Comput. Sci.*, 10:111–121, 1980. 7
- 651 25 Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs.
652 *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974. 15
- 653 26 Anne-Sophie Himmel, Matthias Bentert, André Nichterlein, and Rolf Niedermeier. Efficient
654 computation of optimal temporal walks under waiting-time constraints. In *Proceedings of the*
655 *8th International Conference on Complex Networks and their Applications*, volume 882 of *SCI*,
656 pages 494–506. Springer, 2019. 2, 3
- 657 27 Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal*
658 *B*, 88(9):234, 2015. 2
- 659 28 Petter Holme. Temporal network structures controlling disease spreading. *Physical Review E*,
660 94.2:022305, 2016. 2
- 661 29 Petter Holme and Jari Saramäki (eds.). *Temporal Network Theory*. Springer, 2019. 2
- 662 30 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k -SAT. *Journal of*
663 *Computer and System Sciences*, 62(2):367–375, 2001. 7
- 664 31 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly
665 exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 7
- 666 32 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer*
667 *Computations*, pages 85–103. Springer, 1972. 15
- 668 33 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for
669 temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. 2, 3
- 670 34 William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical
671 theory of epidemics. *Proceedings of the Royal Society of London, Series A.*, 115(772):700–721,
672 1927. 2
- 673 35 Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for
674 the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61, 2018. 2
- 675 36 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic
676 truncation of linear matroids. *ACM Trans. Algorithms*, 14(2):14:1–14:20, 2018. 11
- 677 37 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer*
678 *Science*, 410(44):4471–4479, 2009. 11
- 679 38 George B Mertzios, Othon Michail, and Paul G Spirakis. Temporal network optimization
680 subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019. 2, 3, 8
- 681 39 Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet*
682 *Mathematics*, 12(4):239–280, 2016. 2
- 683 40 Jaroslav Nešetřil and Patrice Ossona De Mendez. *Sparsity: Graphs, Structures, and Algorithms*.
684 Springer, 2012. 12
- 685 41 Mark E J Newman. *Networks*. Oxford University Press, 2018. 2
- 686 42 James G. Oxley. *Matroid Theory*. Oxford University Press, 1992. 10
- 687 43 Raj Kumar Pan and Jari Saramäki. Path lengths, correlations, and centrality in temporal
688 networks. *Physical Review E*, 84(1):016105, 2011. 2

30:18 Finding Temporal Paths under Waiting Time Constraints

- 689 **44** Manuel Sorge and Mathias Weller et al. The graph parameter hierarchy, 2018. 2020. URL:
690 <https://manyu.pro/assets/parameter-hierarchy.pdf>. 4, 8
- 691 **45** Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*,
692 8(1):85 – 89, 1984. 7
- 693 **46** Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318,
694 2009. 8, 10
- 695 **47** H. Wu, J. Cheng, Y. Ke, S. Huang, Y. Huang, and H. Wu. Efficient algorithms for temporal
696 path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942,
697 2016. 2, 8
- 698 **48** Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of
699 finding separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92,
700 2020. 3, 8