Deleting edges to restrict the size of an epidemic in temporal networks*

Jessica Enright† Kitty Meeks‡ George B. Mertzios§ Viktor Zamaraev¶ ‖

February 5, 2021

Abstract

Spreading processes on graphs are a natural model for a wide variety of real-world phenomena, including information spread over social networks and biological diseases spreading over contact networks. Often, the networks over which these processes spread are dynamic in nature, and can be modeled with temporal graphs. Here, we study the problem of deleting edges from a given temporal graph in order to reduce the number of vertices (temporally) reachable from a given starting point. This could be used to control the spread of a disease, rumour, etc. in a temporal graph. In particular, our aim is to find a temporal subgraph in which a process starting at any single vertex can be transferred to only a limited number of other vertices using a temporally-feasible path. We introduce a natural edge-deletion problem for temporal graphs and provide positive and negative results on its computational complexity and approximability.

1 Introduction and motivation

A temporal graph is, loosely speaking, a graph that changes with time. A great variety of modern and traditional networks can be modeled as temporal graphs; social networks, wired or wireless networks which may change dynamically, transportation networks, and several physical systems are only a few examples of networks that change over time [35, 43]. Due to its vast applicability in many areas, this notion of temporal graphs has been studied from different perspectives under various names such as *time-varying* [1, 27, 50], *evolving* [11, 17, 25], *dynamic* [14, 30], and *graphs over time* [37]; for an attempt to integrate existing models, concepts, and results from the distributed computing perspective see the survey papers [12–14] and the references therein. Mainly motivated by the fact that, due to causality, entities and information in temporal graphs can "flow" only along sequences of edges whose time-labels are increasing, most temporal graph parameters and optimisation problems that have been studied so far are based on the notion of temporal paths (see Definition 1 below) and other path-related notions, such as temporal analogues of distance, diameter, reachability, exploration, and centrality [2–4, 23, 24, 39, 42]. Recently, non-path temporal graph problems have also been addressed theoretically, including for example temporal variations of vertex cover [5], vertex coloring [41], matching [40], and maximal cliques [34, 55, 56].

We adopt a simple model for such time-varying networks, in which the vertex set remains unchanged while each edge is equipped with a set of time-labels. This formalism originates in the foundational work of Kempe et al. [36].

Definition (temporal graph)**.** *A* temporal graph *is a pair (G, λ), where $G = (V, E)$ is an underlying (static) graph and $\lambda : E \to 2^{\mathbb{N}}$ is a* time-labeling *function which assigns to every edge of G a set of discrete-time labels.*

*Kitty Meeks was supported by a Personal Research Fellowship from the Royal Society of Edinburgh, funded by the Scottish Government. George B. Mertzios and Viktor Zamaraev were partially supported by the EPSRC Grant EP/P020372/1. The results of this paper previously appeared as an extended abstract in the proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019 [22].

†School of Computing Science, University of Glasgow, UK. Email: jessica.enright@glasgow.ac.uk

‡School of Computing Science, University of Glasgow, UK. Email: kitty.meeks@glasgow.ac.uk

§Department of Computer Science, Durham University, UK. Email: george.mertzios@durham.ac.uk

¶Department of Computer Science, University of Liverpool, UK. Email: viktor.zamaraev@liverpool.co.uk

‖The main part of this paper was prepared while the author was affiliated at the Department of Computer Science, Durham University, UK.

Throughout this paper we restrict our attention to graphs in which every edge is *active* at exactly one time, so that $\lambda(e)$ is a singleton set for every $e \in E(G)$; abusing notation slightly, we will write $\lambda(e) = t$ to indicate that the edge e is (only) present at time t.

Spreading processes on networks or graphs are a topic of significant research across network science [7], and a variety of application areas [31, 32], as well as inspiring more theoretical algorithmic work [26]. Part of the motivation for this interest is the usefulness of spreading processes for modelling a variety of natural phenomena, including biological diseases spreading over contact networks, rumours or news (both fake and real) spreading over information-passing networks, memes and behaviours, etc. The rise of quantitative approaches in modelling these phenomena is supported by the increasing number and size of network datasets that can be used as denominator graphs on which processes can spread (e.g. human mobility and contact networks [48], agricultural trade networks [44], and social networks [38]). Typically, a vertex in one of these networks represents some entity that has a state in the process (for example, being infected with a disease, or holding a belief), and edges represent contacts over which the state can spread to other vertices.

Our work is partly motivated by the need to control contagion (be it biological or informational) that may spread over contact networks. Data specifying timed contacts that could spread an infectious disease are recorded in a variety of settings, including movements of humans via commuter patterns and airline flights [18], and fine-grained recording of livestock movements between farms in most European nations [45]. There is very strong evidence that these networks play a critical role in large and damaging epidemics, including the 2009 H1N1 influenza pandemic [10] and the 2001 British foot-and-mouth disease epidemic [31]. Because of the key importance of timing in these networks to their capacity to spread disease, methods to assess the susceptibility of temporal graphs and networks to disease incursion have recently become an active area of work within network epidemiology in general, and within livestock network epidemiology in particular [9, 47, 53, 54].

Here, similarly to [21], we focus our attention on deleting edges from (G, λ) in order to limit the temporal connectivity of the remaining temporal subgraph. To this end, the following temporal extension of the notion of a path in a static graph is fundamental [36, 39].

Definition (Temporal path)**.** *A* temporal path *from u to v in a temporal graph (G, λ) is a path from u to v in G, composed of edges $e_0, e_1, \ldots, e_k$ such that each edge e_i is assigned a time $t(e_i) \in \lambda(e_i)$, where $t(e_i) < t(e_{i+1})$ for $0 \leq i < k$.*

Our contribution. We consider a natural deletion problem for temporal graphs, namely Temporal Reachability Edge Deletion (for short, TR Edge Deletion), as well as its optimisation version, and study its computational complexity, both in the traditional and the parameterised sense, subject to natural parameters. Given a temporal graph (G, λ) and two natural numbers k, h, the goal is to delete at most k edges from (G, λ) such that, for every vertex v of G, there exists a temporal path to at most $h - 1$ other vertices.

In Section 3, we show that TR Edge Deletion is NP-complete, even on a very restricted class of graphs. We give two different reductions. The first shows that, assuming the Exponential Time Hypothesis, we cannot improve significantly on a brute-force approach when considering how the running-time depends on the input size and the number of permitted deletions. The second demonstrates that TR Edge Deletion is *para-NP-hard* (i.e. NP-hard even for constant-valued parameters) with respect to each one of the parameters h, maximum degree Δ_G, or lifetime of (G, λ) (i.e. the maximum label assigned by λ to any edge of G).

In Section 4, we turn our attention to approximation algorithms for the optimisation version of the problem, Min TR Edge Deletion, in which the goal is to find a minimum-size set of edges to delete. We begin by describing a polynomial-time algorithm to compute a h-approximation to Min TR Edge Deletion on arbitrary graphs, then show how similar techniques can be applied to compute a c-approximation on input graphs of cutwidth at most c. We conclude our consideration of approximation algorithms by showing that there is unlikely to be a polynomial-time algorithm to compute any constant-factor approximation in general, even on temporal graphs of lifetime two.

In Section 5, we consider exact fixed-parameter tractable (FPT) algorithms. Our hardness results show that the problem remains intractable when parameterised by h or Δ_G alone; here we obtain an FPT algorithm by parameterising simultaneously by h, Δ_G and the treewidth $\mathrm{tw}(G)$ of the underlying (static) graph G. In doing so, we demonstrate a general framework in which a celebrated result by

Courcelle, concerning relational structures with bounded treewidth (see Theorem 5.2) can be applied to solve problems in temporal graphs.

Finally, in Section 6 we consider a natural generalization of TR Edge Deletion by restricting the notion of a temporal path, as follows. Given two numbers $\alpha, \beta \in \mathbb{N}$, where $\alpha \leq \beta$, we require that the time between arriving at and leaving any vertex on a temporal path is between α and β; we refer to such a path as an (α, β)-*temporal path*. The resulting problem, incorporating this restricted version of a temporal path, is called (α, β)-TR Edge Deletion. This (α, β)-extension of the deletion problem is well motivated when considering the spread of disease: an upper bound β on the permitted time between entering and leaving a vertex might represent the time within which an infection would be detected and eliminated (thus ensuring no further transmission), while a lower bound α might in different contexts represent either the time between an individual being infected and becoming infectious, or the minimum time individuals must spend together (i.e. in the same vertex) for there to be a non-trivial probability of disease transmission. We show that all of our results can be generalised in a natural way to this "clocked" setting.

2 Preliminaries

Given a (static) graph G, we denote by $V(G)$ and $E(G)$ the sets of its vertices and edges, respectively. An edge between two vertices u and v of G is denoted by uv, and in this case u and v are said to be *adjacent* in G. For a subset $S \subseteq V(G)$ we denoted by $G[S]$ the subgraph of G induced by S. Given a temporal graph (G, λ), where $G = (V, E)$, the maximum label assigned by λ to an edge of G, called the *lifetime* of (G, λ), is denoted by $T(G, \lambda)$, or simply by T when no confusion arises. That is, $T(G, \lambda) = \max\{\lambda(e) : e \in E\}$. Throughout the paper we consider temporal graphs with *finite lifetime* T. Furthermore, we assume that the given labeling λ is arbitrary, i.e. (G, λ) is given with an explicit label for every edge. We say that an edge $e \in E$ *appears at time* t if $\lambda(e) = t$, and in this case we call the pair (e, t) a *time-edge* in (G, λ). Given a subset $E' \subseteq E$, we denote by $(G, \lambda) \setminus E'$ the temporal graph (G', λ'), where $G' = (V, E \setminus E')$ and λ' is the restriction of λ to $E \setminus E'$.

We say that a vertex v is *temporally reachable* from u in (G, λ) if there exists a temporal path from u to v. Furthermore we adopt the convention that every vertex v is temporally reachable from itself. The *temporal reachability set* of a vertex u, denoted by $\text{reach}_{G,\lambda}(u)$, is the set of vertices which are temporally reachable from vertex u. The *temporal reachability* of u is the number of vertices in $\text{reach}_{G,\lambda}(u)$. Furthermore, the *maximum temporal reachability* of a temporal graph is the maximum of the temporal reachabilities of its vertices.

In this paper we mainly consider the following problem.

Temporal Reachability Edge Deletion (TR Edge Deletion)
Input: A temporal graph (G, λ), and $k, h \in \mathbb{N}$.
Output: Is there a set $E' \subseteq E(G)$, with $\lvert E' \rvert \leq k$, such that the maximum temporal reachability of $(G, \lambda) \setminus E'$ is at most h?

Note that the problem clearly belongs to NP as a set of edges acts as a certificate (the reachability set of any vertex in a given temporal graph can be computed in polynomial time [2, 36, 39]). It is worth noting here that the (similarly-flavored) deletion problem for finding small separators in temporal graphs was studied recently; namely the problem of removing a small number of vertices from a given temporal graph such that two fixed vertices become temporally disconnected [29, 57].

3 Computational hardness

The main results of this section demonstrate that TR Edge Deletion is NP-complete even under very strong restrictions on the input. Our first result shows that the trivial brute-force algorithm, running in time $n^{\mathcal{O}(k)}$, in which we consider all possible sets of k edges to delete, cannot be significantly improved in general.

Theorem 3.1. TR Edge Deletion *is W[1]-hard when parameterised by the maximum number k of edges that can be removed, even when the input temporal graph has the lifetime 2. Moreover, assuming*

the Exponential Time Hypothesis (ETH), there is no $f(k)\tau^{o(k)}$ time algorithm for TR Edge Deletion, *where τ is the size of the input temporal graph.*

Proof. We provide a standard parameterised m-reduction from the following W[1]-complete problem.

Clique
Input: A graph $G = (V, E)$.
Parameter: $r \in \mathbb{N}$.
Question: Does G contain a clique on at least r vertices?

First note that, without loss of generality, we may assume that $r \geq 3$, as otherwise the problem is trivial. Let $(G = (V_G, E_G), r)$ be the input to an instance of Clique; we denote $n = |V_G|$ and $m = |E_G|$. We will construct an instance $((H, \lambda), k, h)$ of TR Edge Deletion, which is a yes-instance if and only if (G, r) is a yes-instance for Clique. Note that, without loss of generality we may assume that $m > r + \binom{r}{2}$; otherwise there cannot be more than $r + 3$ vertices of degree at least $r - 1$ in G, and thus we can check all possible sets of r vertices with degree at least $r - 1$ in time $\mathcal{O}(r^3)$.

We begin by defining $H = (V_H, E_H)$. The vertex set of H is $V_H = \{s\} \cup V_G \cup E_G$. The edge set is

$$E_H = \{sv : v \in V_G\} \cup \{ve : e \in E_G, v \in e\}.$$

We complete the construction of the temporal graph (H, λ) by setting

$$\lambda(e) = \begin{cases} 1 & \text{if } e \text{ incident to } s, \\ 2 & \text{otherwise.} \end{cases}$$

Finally, we set $k = r$ and $h = 1 + (n - r) + (m - \binom{r}{2})$.

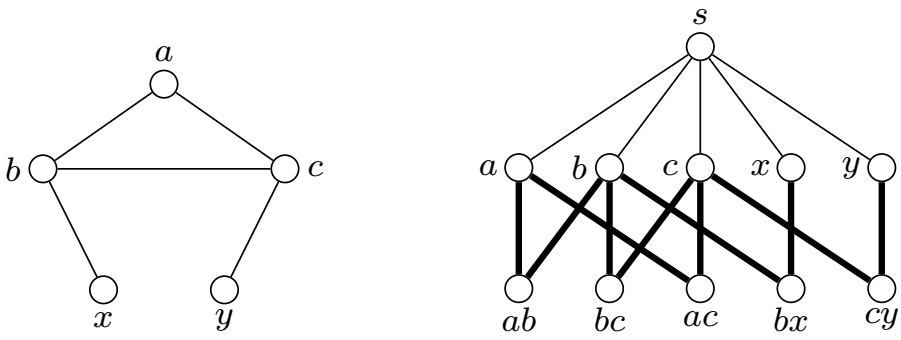

Figure 1: Graph G (left) and the corresponding temporal graph (H, λ) (right). The thin edges of (H, λ) appear in time step 1, and the thick edges appear in time step 2.

We begin by observing that s is the only vertex in (H, λ) whose temporal reachability is more than h. Note that $|\operatorname{reach}_{H,\lambda}(e)| = 3$ for all $e \in E_G$, and $|\operatorname{reach}_{H,\lambda}(v)| \leq n + 1$ for all $v \in V_G$. Thus, as

$$h = 1 + n - r + m - \binom{r}{2} > 1 + n - r + r + \binom{r}{2} - \binom{r}{2} = n + 1,$$

the temporal reachability of any vertex other than s is less than h. Hence, we see that for any $E' \subseteq E_H$ the maximum temporal reachability of $(H, \lambda) \setminus E'$ is at most h if and only if the temporal reachability of s in the modified graph is at most h.

Now suppose that G contains a set $U \subseteq V_G$ of r vertices that induce a clique. Let $E' = \{sv : v \in U\}$ and $(H', \lambda') = (H, \lambda) \setminus E'$. Consider a vertex $v \in V_G$: it is clear that v can only belong to $\operatorname{reach}_{H',\lambda'}(s)$ if $sv \in E_H \setminus E'$, so no element of U belongs to $\operatorname{reach}_{H',\lambda'}(s)$. Moreover, for any $e \in E_G$, any temporal path from s to e in (H, λ) must contain precisely two edges, and so must include an endpoint of e; thus, for any edge e with both endpoints in U, we have $e \notin \operatorname{reach}_{H',\lambda'}(s)$. Since U induces a clique, there are

precisely $\binom{r}{2}$ such edges. It follows that

$$\begin{aligned} |\operatorname{reach}_{H',\lambda'}(s)| &\leq 1 + n + m - |U| - |\{uv \in E_G : u, v \in U\}| \\ &= 1 + n + m - r - \binom{r}{2} \\ &= h, \end{aligned}$$

as required.

Conversely, suppose that we have a set $E' \subseteq E_H$, with $|E'| \leq k = r$, such that $|\operatorname{reach}_{H',\lambda'}(s)| \leq h$, where $(H', \lambda') = (H, \lambda) \setminus E'$.

We begin by arguing that we may assume, without loss of generality, that every element of E' is incident to s. Let $W \subset V_G$ be the set of vertices in V_G which are incident to some element of E'; we claim that deleting the set of edges $E'' = \{sw : w \in W\}$ instead of E' would also reduce the maximum temporal reachability of (H, λ) to at most h. To see this, consider a vertex $x \notin \operatorname{reach}_{H',\lambda'}(s)$. If $x \in V_G$, then we must have $sx \in E'$, and so $sx \in E''$ implying that there is no temporal path from s to x when E'' is deleted. If, on the other hand, $x = u_1u_2 \in E_G$, then E' must contain at least one edge from each of the two temporal paths from s to x in (H, λ), namely su_1x and su_2x. Hence E' contains at least one edge incident to each of u_1 and u_2, so $su_1, su_2 \in E''$ and deleting all edges in E'' destroys all temporal paths from s to x.

Thus we may assume that $E' \subseteq \{sv : v \in V_G\}$. We define $U \subseteq V_G$ to be the set of vertices in V_G incident to some element of E', and claim that U induces a clique of cardinality r in G. First note that $|U| \leq r$. Now observe that the only vertices in V_G that are not temporally reachable from s in (H', λ') are the elements of U, and the only elements of E_G that are not temporally reachable from s are those corresponding to edges with both endpoints in U. Thus, if m' denotes the number of edges in $G[U]$, we have

$$|\operatorname{reach}_{H',\lambda'}(s)| \geq 1 + n + m - |U| - m'.$$

By our assumption that this quantity is at most h, we see that

$$\begin{aligned} 1 + n + m - r - \binom{r}{2} &\geq 1 + n + m - |U| - m' \\ \Leftrightarrow |U| + m' &\geq r + \binom{r}{2}. \end{aligned}$$

Since $|U| \leq r$, we have that $m' \leq \binom{r}{2}$, with equality if and only if $G[U]$ is a clique of size r. Thus, in order to satisfy the inequality above, we must have that $|U| = r$ and that U induces a clique in G, as required.

To prove the lower complexity bound, assume there exists a $f(k)\tau^{o(k)}$ time algorithm for TR Edge Deletion. Then using the above reduction and the fact that the size of the temporal graph (H, λ) is at most $2n^2$ we conclude that Clique can be solved in $f(r)n^{o(r)}$ time which is not possible unless ETH fails [16]. □

The W[1]-hardness reduction of Theorem 3.1 also implies that the problem TR Edge Deletion is NP-complete, even on temporal graphs with lifetime at most two. We note that, for temporal graphs of lifetime one, the problem is solvable in polynomial time: in this setting, the reachability set of each vertex is precisely its closed neighbourhood, so the problem reduces to that of deleting a set of at most k edges so that every vertex has degree at most $h - 1$ which is solvable in polynomial time [49, Theorem 33.4].

We now demonstrate that TR Edge Deletion remains NP-complete on temporal graphs of lifetime two even if the underlying graph has bounded degree and the maximum permitted size of a temporal reachability set is bounded by a constant.

Theorem 3.2. TR Edge Deletion *is NP-complete, even when the maximum temporal reachability h is at most* 6 *and the input temporal graph (G, λ) has:*

1. *maximum degree Δ_G of the underlying graph G at most 5, and*
2. *lifetime at most 2.*

Therefore TR Edge Deletion *is para-NP-hard with respect to the combination of parameters h, Δ_G, and lifetime $T(G, \lambda)$.*

Proof. As we mentioned in Section 2, the problem trivially belongs to NP. Now we give a reduction from the following well-known NP-complete problem [52].

3,4-SAT
Input: A CNF formula Φ with exactly 3 variables per clause, such that each variable appears in at most 4 clauses.
Output: Does there exists a truth assignment satisfying Φ?

Let Φ be an instance of $3,4$-SAT with variables $x_1, \ldots, x_n$, and clauses $C_1, \ldots, C_m$. We may assume without loss of generality that every variable x_i appears at least once negated and at least once unnegated in Φ. Indeed, if a variable x_i appears only negated (resp. unnegated) in Φ, then we can trivially set $x_i = 0$ (resp. $x_i = 1$) and then remove from Φ all clauses where x_i appears; this process would provide an equivalent instance of 3,4-SAT of smaller size. Now we construct an instance $((G, \lambda), k, h)$ of TR Edge Deletion which is a yes-instance if and only if Φ is satisfiable.

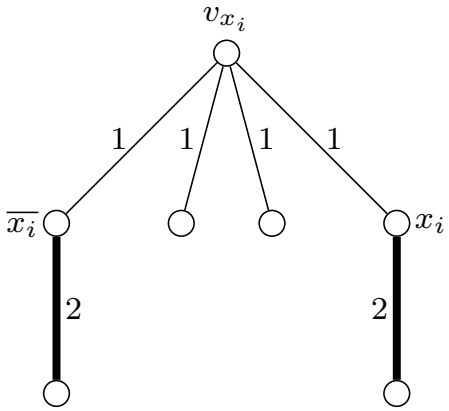

Figure 2: The gadget corresponding to variable x_i. The number beside an edge is the time step at which that edge appears. The bold edges are the ones we refer to as *literal edges*.

We construct (G, λ) as follows. For each variable x_i we introduce in G a copy of the subgraph shown in Figure 2, which we call an *x_i-gadget*. There are three special vertices in an x_i-gadget: x_i and $\overline{x_i}$, which we call *literal vertices*, and v_{x_i} which we call *the head vertex* of x_i-gadget. All the edges incident to v_{x_i} appear in time step 1, the other two edges of x_i-gadget, which we call *literal edges*, appear in time step 2. Additionally, for every clause C_s we introduce in G a *clause vertex* C_s that is adjacent to the three literal vertices corresponding to the literals of C_s. All the edges incident to C_s appear in time step 1. See Figure 3 for illustration. Finally, we set $k = n$ and $h = 6$.

First recall that, in Φ, every variable x_i appears at least once negated and at least once unnegated. Therefore, since every variable x_i appears in at most four clauses in Φ, it follows that each of the two vertices corresponding to the literals $x_i, \overline{x_i}$ is connected with at most three clause vertices. Therefore the degree of each vertex corresponding to a literal in the constructed temporal graph (G, λ) (see Figure 3) is at most five. Moreover, it can be easily checked that the same also holds for every other vertex of (G, λ), and thus $\Delta_G \leq 5$.

We continue by observing temporal reachabilities of the vertices of (G, λ). A literal vertex can only temporally reach its neighbours and so, by the argument above, has temporal reachability at most 6 (including the vertex itself). The head vertex of a gadget temporally reaches only the vertices of the gadget, hence the temporal reachability of any head vertex in (G, λ) is 7. Any other vertex belonging to a gadget can temporally reach only its unique neighbour in G. Every clause vertex can reach only the corresponding literal vertices and their neighbours incident to the literal edges. Hence the temporal reachability of every clause vertex in (G, λ) is 7. Therefore in our instance of TR Edge Deletion we only need to care about temporal reachabilities of the clause and head vertices.

Now we show that, if there is a set E of n edges such that the maximum temporal reachability of the modified graph $(G, \lambda) \setminus E$ is at most 6, then Φ is satisfiable. First, notice that since the temporal reachability of every head vertex is decreased in the modified graph and the number of gadgets is n, the set E contains exactly one edge from every gadget. Hence, as the temporal reachability of every

clause vertex C_s is also decreased, set E must contain at least one literal edge that is incident to a literal neighbour of C_s. We now construct a truth assignment as follows: for every literal edge in E we set the corresponding literal to TRUE. If there are unassigned variables left we set them arbitrarily, say, to TRUE.

Since E has one edge in every gadget, every variable was assigned exactly once. Moreover, by the above discussion, every clause has a literal that is set to TRUE by the assignment. Hence the assignment is well-defined and satisfies Φ.

To show the converse, given a truth assignment $(\alpha_1, \ldots, \alpha_n)$ satisfying Φ we construct a set E of n edges such that the maximum temporal reachability of $(G, \lambda) \setminus E$ is at most 6. For every $i \in [n]$ we add to E the literal edge incident to x_i if $\alpha_i = 1$, and the literal edge incident to $\overline{x_i}$ otherwise. By the construction, E has exactly one edge from every gadget. Moreover, since the assignment satisfies Φ, for every clause C_s set E contains at least one literal edge corresponding to one of the literals of C_s. Hence, by removing E from (G, λ), we strictly decrease temporal reachability of every head and clause vertex. □

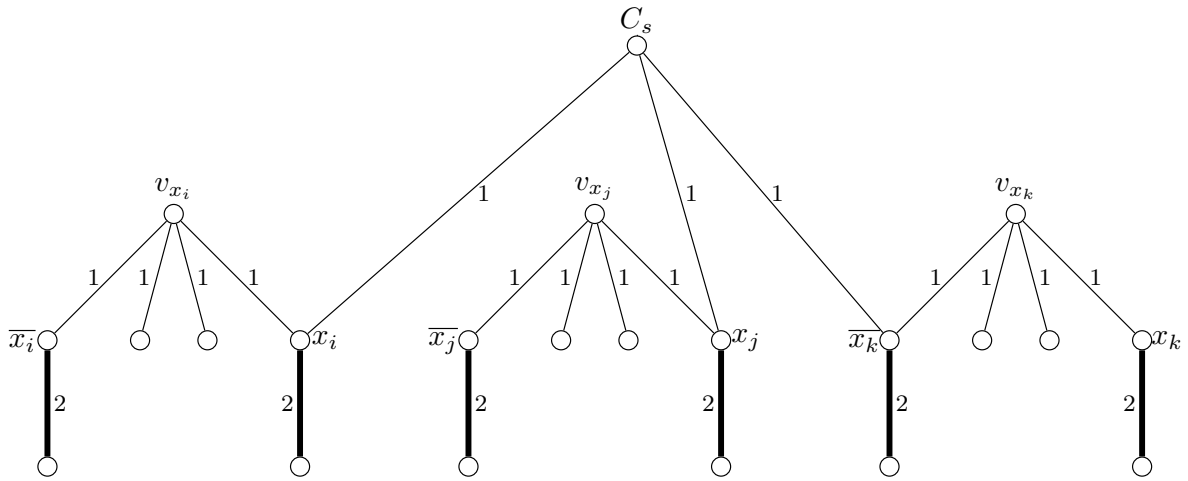

Figure 3: A subgraph of a temporal graph corresponding to an instance of 3,4-SAT.

4 Approximability

Given the strength of the hardness results proved in Section 3, it is natural to ask whether the problem admits efficient approximation algorithms for the following optimisation problem.

Minimum Temporal Reachability Edge Deletion (Min TR Edge Deletion)
Input: A temporal graph (G, λ) and $h \in \mathbb{N}$.
Output: A set X of edges of *minimum* size such that the maximum temporal reachability of $(G, \lambda) \setminus X$ is at most h.

We begin with some more notation. If (G, λ) is a temporal graph and $v \in V(G)$, we say that T is a *reachable subtree for* v if T is a subtree of G, $v \in V(T)$ and, for all $u \in V(T) \setminus \{v\}$, there is a temporal path from v to u in (T, λ'), where λ' is the restriction of λ to the edges of T. We first observe that, if a temporal graph has maximum reachability more than h, we can efficiently find a minimal reachable subtree witnessing this fact.

Lemma 4.1. *Let (G, λ) be a temporal graph, and h a positive integer. There is an algorithm running in polynomial time which, on input $((G, \lambda), h)$,*

1. *if the maximum temporal reachability of (G, λ) is at most h, outputs "YES";*
2. *if the maximum temporal reachability of (G, λ) is greater than h, outputs a vertex $v \in V(G)$ and a reachable subtree T for v where T has exactly $h + 1$ vertices.*

Proof. For any vertex $v \in V(G)$, carrying out a version of Dijkstra's algorithm adapted to include temporal information starting from v for up to h steps will either identify a reachable subtree for v on $h+1$ vertices or determine that no such subtree exists. Thus, by repeating this process for each vertex $v \in V(G)$ we can either find some pair (v, T) such that T has $h+1$ vertices and is a reachable subtree for v, or determine that no vertex has a reachable subtree with $h+1$ vertices. In the latter case, it is clear that no vertex has a temporal reachability set of size more than h, so we can safely output "YES". □

Let h be a positive integer and $(G = (V, E), \lambda)$ be a temporal graph. We say that an edge set $E' \subseteq E$ is a *valid deletion* in $(G = (V, E), \lambda)$ with respect to h if the maximum temporal reachability of $(G = (V, E), \lambda) \backslash E'$ is at most h. Where h is clear from the context, we may not refer to it explicitly. We now make a simple observation about valid deletions.

Lemma 4.2. *Let (G, λ) be a temporal graph and h a positive integer. Suppose that T is a reachable subtree for some $v \in V(G)$ and that T has more than h vertices. If $E' \subseteq E(G)$ is a valid deletion in $(G = (V, E), \lambda)$ with respect to h, then $|E' \cap E(T)| \geq 1$.*

Proof. Suppose, for a contradiction, that E' does not contain any edge of T. Then T is a subgraph of $G'[V(T)]$ so, for every vertex $u \in V(T) \setminus \{v\}$, $G'[V(T)]$ and hence G' contains a temporal path from v to u. It follows that $V(T) \subseteq \text{reach}_{(G',\lambda')}(v)$ and hence $|\text{reach}_{(G',\lambda')}(v)| > h$, contradicting the assumption that E' is a valid deletion. □

Using these two observations, we now describe our first approximation algorithm.

Theorem 4.3. *There exists a polynomial-time algorithm to compute an h-approximation to* Min TR Edge Deletion*, where h denotes the maximum permitted reachability.*

Proof. Let $((G, \lambda), h)$ be an input instance of Min TR Edge Deletion, and let $E_{\text{opt}} \subseteq E$ be a minimum-cardinality edge set such that $(G, \lambda) \setminus E_{\text{opt}}$ has temporal reachability at most h. It suffices to demonstrate that we can find in polynomial time a set $E' \subseteq E$ such that $(G, \lambda) \setminus E'$ has temporal reachability at most h, and $|E'| \leq h|E_{\text{opt}}|$. We claim that the following algorithm achieves this.

1. Initialise $E' := \emptyset$.
2. While (G, λ) has reachability greater than h:
   (a) Find a pair (v, T) such that $v \in V(G)$, T is a reachable subtree for v and $|T| = h + 1$.
   (b) Add $E(T)$ to E', and update $(G, \lambda) \leftarrow (G, \lambda) \setminus E'$.
3. Return E'.

We begin by considering the running time of this algorithm. By Lemma 4.1 we can determine whether to execute the while loop and, if we do enter the loop, execute Step 2(a), all in polynomial time. Steps 1 and 2(b) can clearly both be carried out in linear time. Moreover, the total number of iterations of the while loop is bounded by the number of edges in G, so we see that the algorithm will terminate in polynomial time.

At every iteration, the algorithm removes exactly h edges, while the optimum deletion set E_{opt} must remove at least one of these h edges. Therefore, in total, we remove at most $h|E_{\text{opt}}|$ edges. To complete the proof, we observe that, by construction, the identified set E' is a valid deletion set. □

We now demonstrate that we can improve on this general approximation algorithm when the underlying graph has certain useful properties, in particular when the cutwidth is bounded.

The *cutwidth* of a graph $G = (V, E)$ is the minimum integer c such that the vertices of G can be arranged in a linear order $v_1, \ldots, v_n$, called a *layout*, such that for every $i, 1 \leq i < n$ the number of edges with one endpoint in $v_1, ..., v_i$ and one in $v_{i+1}, ..., v_n$ is at most c. Given a layout $v_1, v_2, \ldots, v_n$, we say that edges with one endpoint in $v_1, ..., v_i$ and one in $v_{i+1}, ..., v_n$ *span* v_i, v_{i+1}, and say that the maximum number of edges spanning a pair of consecutive vertices is the *cutwidth* of the layout. For any constant c, Thilikos et al. [51] give a linear-time algorithm to generate a layout of cutwidth at most c if one exists.

We can use a similar argument to that in Theorem 4.3 to give a polynomial-time algorithm to compute a c-approximation to Min TR Edge Deletion, where c is the cutwidth of the underlying graph of the

input temporal graph. In addition to Lemma 4.2, we will also make use of the following definition and observation:

Let $G = (V, E)$ be a graph. A *cut* (A, B) of G is a partition of V into two subsets A and B. The *cut-set* of cut (A, B) is the set $\{ab \in E \mid a \in A, b \in B\}$ of edges that have one endpoint in A and the other endpoint in B.

Lemma 4.4. *Let h be a positive integer, $(G = (V, E), \lambda)$ be a temporal graph, (A, B) be a cut of G, and E' be the cut-set of (A, B). If E'_A and E'_B are valid deletion sets for $(G[A], \lambda|_{E(G[A])})$, $(G[B], \lambda|_{E(G[B])})$, respectively, then $E'_A \cup E'_B \cup E'$ is a valid deletion set for $(G = (V, E), \lambda)$.*

We now describe a cutwidth approximation algorithm:

Theorem 4.5. *There exists a polynomial-time algorithm to compute a c-approximation to* Min TR Edge Deletion *provided that a layout of cutwidth c is given.*

Proof. Let $((G = (V, E), \lambda), h)$ be the input to Min TR Edge Deletion, and suppose that the layout $v_1, \ldots, v_n$ of V, with cutwidth c, is given. Consider the algorithm:

1. Initialise $E' := \emptyset$.
2. Initialise $i := 1$.
3. While (G, λ) has reachability greater than h:
   (a) Find the maximum $j \in \{i, \ldots, n\}$ such that the maximum reachability in the subgraph $(G[\{v_i, \ldots, v_j\}], \lambda|_{E(G[\{v_i,\ldots,v_j\}])})$ is at most h.
   (b) Add all edges that span v_j, v_{j+1} to E', and update $(G, \lambda) \leftarrow (G, \lambda) \setminus E'$.
   (c) Update $i \leftarrow j + 1$
4. Return E'.

First we consider the running time of this algorithm: within the loop, both $3(a)$ and $3(b)$ can be executed in polynomial time. The loop itself will execute at most $|V|$ times, and the logical condition can be evaluated in polynomial time. Thus the overall running time is polynomial.

At every iteration the algorithm detects a part $G[v_i, ..., v_{j+1}]$ of the graph, from which the optimum solution must delete at least one edge (by Lemma 4.2). In this case the algorithm deletes at most c edges, while guaranteeing that no further edge from within $G[v_i, ..., v_{j+1}]$ needs to be deleted in subsequent steps (by Lemma 4.4, and because the set deleted is the cut-set of cut $(\{v_i, \ldots, v_j\}, \{v_i, \ldots, v_n\} \setminus \{v_i, \ldots, v_j\})$ in $G[v_i, ..., v_n]$. Thus the algorithm provides a c-approximation of the optimum solution. □

Then, for any fixed cutwidth c, using the layout generation algorithm given by Thilikos et al. [51] and the algorithm described above, we can give a cutwidth-approximation to Min TR Edge Deletion for graphs with cutwidth c.

Corollary 4.6. *There exists a polynomial-time algorithm to compute a c-approximation to* Min TR Edge Deletion *whenever the cutwidth of the input graph is at most c.*

Note that as paths have cutwidth one, Corollary 4.6 gives us an exact polynomial-time algorithm for Min TR Edge Deletion on paths.

We conclude this section by demonstrating that there is unlikely to be a polynomial-time algorithm to compute any constant factor approximation to Min TR Edge Deletion in general, even for temporal graphs of lifetime two.

Theorem 4.7. *Unless $P = NP$, there exists a natural number c such that for every $\delta \in \left(0, \frac{1}{c+2}\right)$,* Min TR Edge Deletion *cannot be approximated in polynomial time to within a factor of $\delta \ln n$, where n is the number of vertices in the input temporal graph, even if the input temporal graph has lifetime two.*

Proof. To prove the theorem we provide a reduction from the Set Cover problem, which given a family $\mathcal{S}$ of subsets of a ground set U asks for a minimum size subfamily $\mathcal{S}' \subseteq \mathcal{S}$ that covers U, i.e. the union of the sets in $\mathcal{S}'$ equals the ground set. The result will be derived from the reduction and the fact that, unless $P = NP$, for every $\varepsilon \in (0, 1)$, Set Cover cannot be approximated to within factor of $(1 - \varepsilon) \ln N$ on instances with at most N^c sets, where N is the size of the ground set and c is some natural constant[1].

Let $(U, \mathcal{S})$ be an instance of Set Cover, where $U = \{1, 2, \ldots, N\}$, $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$, and $M \leq N^c$. Let f_i denote the frequency of element $i \in U$, i.e. $f_i = |\{S \ : \ S \in \mathcal{S} \text{ and } i \in S\}|$ and let $\ell = \max\{f_i \ : \ i \in U\}$. We define the Min TR Edge Deletion instance $((G, \lambda), h)$ as follows. The vertex set $V(G)$ of the underlying graph G is

$$U \cup \mathcal{S} \cup \{S'_1, S'_2, \ldots, S'_M\} \cup \{d^i_j \ : \ i \in U, j \in \{1, \ldots, 2(\ell - f_i) + N\}\}.$$

Thus, since $M \leq N^c$, the number n of vertices in the constructed graph G is

$$n \leq N + 2M + N(2(M - 1) + N) \leq N^c(3N + 2)$$

The edge set of G is such that

1. $S_i S'_i \in E(G)$ for every $i \in [M]$;
2. $q d^q_j \in E(G)$ for every $q \in [N]$ and $j \in [2(\ell - f_q) + N]$; and
3. $G[U \cup \mathcal{S}]$ is the bipartite element-set incidence graph of $\mathcal{S}$, i.e. the bipartite graph with parts U and $\mathcal{S}$ in which $i \in U$ is adjacent to $S \in \mathcal{S}$ if and only if $i \in S$.

Figure 4 illustrates the structure of (G, λ). The edges in $\{S_i S'_i \mid i \in [M]\}$ appear only in time step 2 and all the other edges appear only in time step 1. Finally, we set $h = 2\ell + N$.

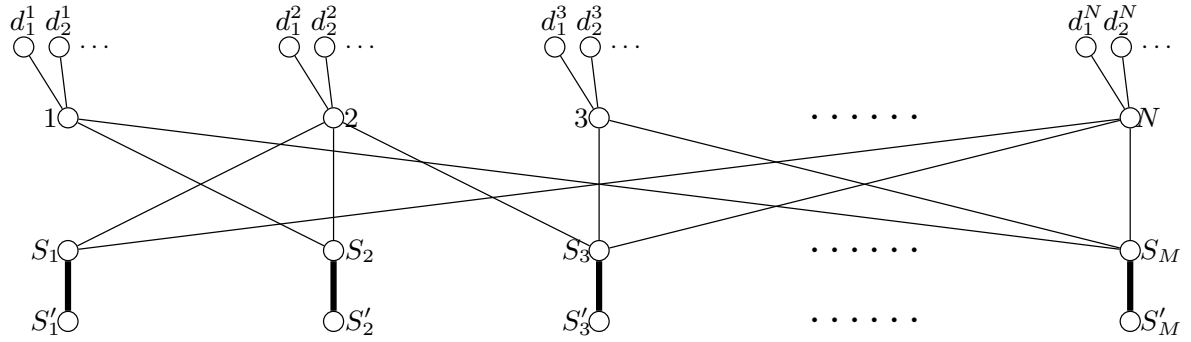

Figure 4: The temporal graph (G, λ) corresponding to an instance $(U, \mathcal{S})$ of Set Cover.

We claim that the size of a minimum subfamily of $\mathcal{S}$ that covers U is equal to the size of a minimum set X of edges such that the maximum temporal reachability of $(G, \lambda) \setminus X$ is at most h.

We start by observing that the temporal reachability of every vertex $i \in U$ is $2(\ell - f_i) + N + 2f_i + 1 = 2\ell + N + 1$, the temporal reachability of $S_j \in \mathcal{S}$ is $2 + |S_j| \leq 2 + N \leq 2\ell + N$, and the temporal reachability of any other vertex is 2. Therefore, in order to limit the temporal reachabilities of the vertices of (G, λ) to $h = 2\ell + N$, it is necessary and sufficient to reduce the temporal reachability of every vertex in U by one.

Next, we show that, if X is a feasible solution of Min TR Edge Deletion on $((G, \lambda), h)$, then there exists a feasible solution X' such that $|X'| \leq |X|$ and $X' \subseteq \{S_i S'_i \mid i \in [M]\}$. For this we notice that none of the edges incident with $i \in U$ affects the temporal reachability of $j \in U$ for any $j \in [N] \setminus \{i\}$. Therefore, if X contains an edge incident with i, excluding from X all such edges and adding to X an edge $S_p S'_p$ with $i \in S_p$ (if there is no such edge in X yet), preserves the feasibility of X. By repeating this procedure successively for every vertex in U, we obtain a set X' with the desired properties. Note that X' can be constructed from X in time linear in $|X|$.

[1]This fact follows from the reduction given by Moshkovitz [46] (see Definition 3 in [46]), which proves the $(1 - \varepsilon) \ln N$-approximation hardness of Set Cover assuming the projection games conjecture. This result was subsequently strengthened by Dinur and Steurer [20] by proving the same hardness result for Set Cover under the assumption that P≠NP.

To complete the proof of the claim, it remains to show that a set

$$X = \{S_{t_1}S'_{t_1}, S_{t_2}S'_{t_2}, \ldots, S_{t_k}S'_{t_k}\}$$

is a feasible solution of Min TR Edge Deletion on $((G,\lambda),h)$ if and only if $\{S_{t_1}, S_{t_2}, \ldots, S_{t_k}\}$ is a set cover of U, which immediately follows from the observation that the temporal reachability of $i \in U$ in $(G,\lambda)\setminus X$ is strictly smaller than that of i in (G,λ) if and only if i is contained in S_{t_j} for some $S_{t_j}S'_{t_j} \in X$.

Now, suppose there exists a polynomial-time algorithm $\mathcal{A}$ that for some $\delta \in \left(0, \frac{1}{c+2}\right)$ approximates Min TR Edge Deletion to within a factor of $\delta \ln n$, where n is the number of vertices in the input temporal graph. We will show that using $\mathcal{A}$ the Set Cover problem on instances with at most N^c sets can be efficiently approximated to within a factor of $(1-\varepsilon)\ln N$ for some $\varepsilon \in (0,1)$, which is not possible unless $P = NP$ [20]. Let $(U,\mathcal{S})$ be an arbitrary N-element Set Cover instance with $N \geq 4$. First, we construct in polynomial time, as in the reduction, the corresponding Min TR Edge Deletion instance $((G,\lambda),h)$. Then, using $\mathcal{A}$ we find a $\delta\ln n$-approximate solution X for $((G,\lambda),h)$. By the above discussion, in linear time we can construct from X a $\delta\ln n$-approximate set cover $\mathcal{S}'$ of U. Since, by construction, the number n of vertices in G is at most $N^c(3N+2)$, and $\delta\ln n \leq \delta(c+2)\ln N$ for every $N \geq 4$, we conclude that $\mathcal{S}'$ is a $(1-\varepsilon)\ln N$-approximate solution for $(U,\mathcal{S})$, where $\varepsilon = 1-\delta(c+2) \in (0,1)$. □

5 An exact FPT algorithm

Our results in the previous sections (see e.g. Theorem 3.2) imply that TR Edge Deletion is para-NP-hard, when simultaneously parameterised by h and Δ_G. In the current section we complement these results by showing that TR Edge Deletion admits an FPT algorithm, when simultaneously parameterised by h, Δ_G, and $\mathrm{tw}(G)$, where $\mathrm{tw}(G)$ is the treewidth of G.

Our results (see Theorem 5.4) illustrate how a celebrated theorem by Courcelle (see Theorem 5.2) can be applied to solve temporal graph problems, following a general framework that could potentially be applied to many other temporal problems as well: (i) we define a suitable family τ of relations (i.e. a suitable relational vocabulary) and a Monadic Second Order (MSO) formula ϕ (of length ℓ) that expresses our temporal graph problem at hand; (ii) we represent an arbitrary input temporal graph (G,λ) with an equivalent relational structure $\mathcal{A}$ (of treewidth at most t); (iii) we apply Courcelle's general theorem which solves our problem at hand in time linear to the size of the relational structure $\mathcal{A}$, whenever both ℓ and t are bounded; that is, in time $f(t,\ell)\cdot||\mathcal{A}||$.

Here, we apply this general framework to the particular problem TR Edge Deletion (by appropriately defining τ, ϕ, and $\mathcal{A}$) such that ℓ only depends on our parameter h, while t only depends on Δ_G and $\mathrm{tw}(G)$; this yields our FPT algorithm. Here, as it turns out, the size of $\mathcal{A}$ is quadratic on the size of the input temporal graph (G,λ). Before we present the main result of this section (see Section 5.2), we first present in Section 5.1 some necessary background on logic and on tree decompositions of graphs and relational structures. For any undefined notion in Section 5.1, we refer the reader to [28].

5.1 Preliminaries for the algorithm

Treewidth of graphs

Given any tree T, we will assume that it contains some distinguished vertex $r(T)$, which we will call the *root* of T. For any vertex $v \in V(T)\setminus\{r(T)\}$, the *parent* of v is the neighbour of v on the unique path from v to $r(T)$; the set of *children* of v is the set of all vertices $u \in V(T)$ such that v is the parent of u. The *leaves* of T are the vertices of T whose set of children is empty. We say that a vertex u is a *descendant* of the vertex v if v lies somewhere on the unique path from u to $r(T)$. In particular, a vertex is a descendant of itself, and every vertex is a descendant of the root. Additionally, for any vertex v, we will denote by T_v the subtree induced by the descendants of v.

We say that $(T,\mathcal{B})$ is a *tree decomposition* of G if T is a tree and $\mathcal{B} = \{\mathcal{B}_s : s \in V(T)\}$ is a collection of non-empty subsets of $V(G)$ (or *bags*), indexed by the nodes of T, satisfying:

(1) for all $v \in V(G)$, the set $\{s \in T : v \in \mathcal{B}_s\}$ is nonempty and induces a connected subgraph in T,

(2) for every $e = uv \in E(G)$, there exists $s \in V(T)$ such that $u, v \in \mathcal{B}_s$.

The *width* of the tree decomposition $(T, \mathcal{B})$ is defined to be $\max\{|\mathcal{B}_s| : s \in V(T)\} - 1$, and the *treewidth* of G is the minimum width over all tree decompositions of G.

Although it is NP-hard to determine the treewidth of an arbitrary graph [6], the problem of determining whether a graph has treewidth at most w (and constructing such a tree decomposition if it exists) can be solved in linear time for any constant w [8]; note that this running time depends exponentially on w.

Theorem 5.1 (Bodlaender [8])**.** *For each $w \in \mathbb{N}$, there exists a linear-time algorithm, that tests whether a given graph $G = (V, E)$ has treewidth at most w, and if so, outputs a tree decomposition of G with treewidth at most w.*

Relational structures and monadic second order logic

A *relational vocabulary* τ is a set of relation symbols. Each relation symbol R has an *arity*, denoted $\text{arity}(R) \geq 1$. A *structure* $\mathcal{A}$ of vocabulary τ, or τ-structure, consists of a set A, called the *universe*, and an interpretation $R^{\mathcal{A}} \subseteq A^{\text{arity(R)}}$ of each relation symbol $R \in \tau$. We write $\overline{a} \in R^{\mathcal{A}}$ or $R^{\mathcal{A}}(\overline{a})$ to denote that the tuple $\overline{a} \in A^{\text{arity}(R)}$ belongs to the relation $R^{\mathcal{A}}$.

We briefly recall the syntax and semantics of first-order logic. We fix a countably infinite set of (*individual*) *variables*, for which we use small letters. *Atomic formulas of vocabulary* τ are of the form:

1. $x = y$ or
2. $R(x_1 \dots x_r)$,

where $R \in \tau$ is r-ary and $x_1, \dots, x_r, x, y$ are variables. *First-order formulas* of vocabulary τ are built from the atomic formulas using the Boolean connectives $\neg, \wedge, \vee$ and existential and universal quantifiers $\exists, \forall$.

The difference between first-order and second-order logic is that the latter allows quantification not only over elements of the universe of a structure, but also over subsets of the universe, and even over relations on the universe. In addition to the individual variables of first-order logic, formulas of second-order logic may also contain *relation variables*, each of which has a prescribed arity. Unary relation variables are also called *set variables*. We use capital letters to denote relation variables. To obtain second-order logic, the syntax of first-order logic is enhanced by new atomic formulas of the form $X(x_1 \dots x_k)$, where X is k-ary relation variable. Quantification is allowed over both individual and relation variables. A second-order formula is *monadic* if it only contains unary relation variables. Monadic second-order logic is the restriction of second-order logic to monadic formulas. The class of all monadic second-order formulas is denoted by MSO.

A *free variable* of a formula ϕ is a variable x with an occurrence in ϕ that is not in the scope of a quantifier binding x. A *sentence* is a formula without free variables. Informally, we say that a structure $\mathcal{A}$ *satisfies* a formula ϕ if there exists an assignment of the free variables under which ϕ becomes a true statement about $\mathcal{A}$. In this case we will write $\mathcal{A} \models \phi$.

Treewidth of relational structures

The definition of tree decompositions and treewidth generalizes from graphs to arbitrary relational structures in a straightforward way. A *tree decomposition* of a τ-structure $\mathcal{A}$ is a pair $(T, \mathcal{B})$, where T is a tree and $\mathcal{B}$ a family of subsets of the universe A of $\mathcal{A}$ such that:

(1) for all $a \in A$, the set $\{s \in V(T) : a \in \mathcal{B}_s\}$ is nonempty and induces a connected subgraph (i.e. subtree) in T,

(2) for every relation symbol $R \in \tau$ and every tuple $(a_1, \dots, a_r) \in R^{\mathcal{A}}$, where $r := \text{arity}(R)$, there is a $s \in V(T)$ such that $a_1, \dots, a_r \in \mathcal{B}_s$.

The *width* of the tree decomposition $(T, \mathcal{B})$ is the number $\max\{|\mathcal{B}_s| : s \in V(T)\} - 1$. The *treewidth* $\text{tw}(\mathcal{A})$ of $\mathcal{A}$ is the minimum width over all tree decompositions of $\mathcal{A}$.

We will make use of the version of Courcelle's celebrated theorem for relational structures of bounded treewidth, which, informally, says that the optimisation problem definable by an MSO formula can be solved in FPT time with respect to the treewidth of a relational structure. The formal statement is an analogue of a similar theorem for the model-checking problem.

Theorem 5.2 (analogue of Theorem 9.21 in [19]). *Let ϕ be an MSO formula with a free set variable E, and let $\mathcal{A}$ be a relational structure on universe A, where $\mathrm{tw}(\mathcal{A}) \leq t$. Then, given a width-$t$ tree decomposition of $\mathcal{A}$, a minimum-cardinality set $E \subseteq A$ such that $\mathcal{A}$ satisfies $\phi(E)$ can be computed in time*

$$f(t, \ell) \cdot ||\mathcal{A}||,$$

where f is a computable function, ℓ is the length of ϕ, and $||\mathcal{A}||$ is the size of $\mathcal{A}$.

5.2 The FPT algorithm

In this section we present an FPT algorithm for TR Edge Deletion when parameterised simultaneously by three parameters: h, Δ_G, and $\mathrm{tw}(G)$. Our strategy is first, given an input temporal graph (G, λ), to construct a relational structure $\mathcal{A}_{G,\lambda}$ whose treewidth is bounded in terms of Δ_G and $\mathrm{tw}(G)$. Then we construct an MSO formula ϕ_h with a unique free set variable E, such that $\mathcal{A}_{G,\lambda}$ satisfies $\phi_h(E)$ for some $E \subseteq E(G)$ if and only if the maximum reachability of $(G, \lambda) \setminus E$ is at most h. Finally, we apply Theorem 5.2 to find the minimum cardinality of such a set $E \subseteq E(G)$. If the minimum cardinality is at most k, then $((G, \lambda), k, h)$ is a yes-instance of the problem, otherwise it is a no-instance.

We note that in the setting we consider in this paper, that is temporal graphs in which each edge is active at a single timestep, the construction below might be simplified slightly; however, in order to demonstrate the flexibility of this general framework, we choose to define a relational structure which would allow us to represent temporal graphs in which edges may be active at more than one timestep. Observe that Theorem 5.4 can immediately be adapted to this more general context if we replace Δ_G by the maximum temporal total degree of the input temporal graph (i.e. the maximum number of time-edges incident with any vertex).

Given a temporal graph (G, λ), we define a relational structure $\mathcal{A}_{G,\lambda}$ as follows. The ground set $A_{G,\lambda}$ consists of

- the set $V(G)$ of vertices in G,
- the set $E(G)$ of edges in G, and
- the set of all time-edges of (G, λ), i.e. the set $\Lambda(G, \lambda) = \{(e, t) \mid e \in E(G), t \in \lambda(e)\}$.

On this ground set $A_{G,\lambda}$, we define three binary relations $\mathcal{I}$, $\mathcal{R}$, and $\mathcal{L}$ as follows:

1. $(v, e) \in \mathcal{I}$ if and only if $v \in V(G)$, $e \in E(G)$, and v is incident to e.
2. $((e_1, t_1), (e_2, t_2)) \in \mathcal{R}$ if and only if the following conditions hold:
   (a) $(e_1, t_1), (e_2, t_2) \in \Lambda(G, \lambda)$;
   (b) e_1, e_2 share a vertex in G;
   (c) $t_1 < t_2$.
3. $(f, (e, t)) \in \mathcal{L}$ if and only if $f \in E(G)$, $(e, t) \in \Lambda(G, \lambda)$, and $f = e$.

First we show that the treewidth of $\mathcal{A}_{G,\lambda}$ is bounded by a function of Δ_G and $\mathrm{tw}(G)$.

Lemma 5.3. *The treewidth of $\mathcal{A}_{G,\lambda}$ is at most $(2\Delta_G + 1)(\mathrm{tw}(G) + 1) - 1$.*

Proof. To prove the lemma we show how to modify an optimal tree decomposition of G into a desired tree decomposition of $\mathcal{A}_{G,\lambda}$. Suppose that $(T, \mathcal{B})$ is a tree decomposition of G of width $\mathrm{tw}(G)$. The relational structure $\mathcal{A}_{G,\lambda}$ then has a tree decomposition $(T, \mathcal{B}')$, where, for every $s \in V(T)$,

$$\mathcal{B}'_s = \mathcal{B}_s \cup \bigcup_{v \in \mathcal{B}_s} \{e : e \in E(G), e \text{ is incident to } v\} \cup \bigcup_{v \in \mathcal{B}_s} \{(e, t) : (e, t) \in \Lambda(G, \lambda), e \text{ is incident to } v\}.$$

It is clear that

$$|\mathcal{B}'_s| \leq |\mathcal{B}_s| + 2\Delta_G|\mathcal{B}_s| \leq (2\Delta_G + 1)(\mathrm{tw}(G) + 1)$$

for all $s \in V(T)$, and it is easy to verify that $(T, \mathcal{B}')$ is indeed a tree decomposition for $\mathcal{A}_{G,\lambda}$. □

Using this, we now provide the main result of this section.

Theorem 5.4. TR Edge Deletion *admits an FPT algorithm with respect to the combined parameter of* h, Δ_G, *and* $\mathrm{tw}(G)$.

Proof. Note that the input to TR Edge Deletion is a temporal graph (G, λ). Note also that, by Theorem 5.1, we can compute a minimum tree decomposition of any (static) graph G by an FPT algorithm, parameterised by treewidth. Furthermore, it follows from the proof of Lemma 5.3, a tree decomposition of the underlying (static) graph G can be transformed in linear time (in the size of the temporal graph (G, λ)) into the tree decomposition of $\mathcal{A}_{G,\lambda}$. Therefore, since such a tree decomposition of $\mathcal{A}_{G,\lambda}$ can be computed in linear time overall, we assume here that such a decomposition is already computed.

By Lemma 4.1, if the temporal reachability of a vertex u is greater than h, then (G, λ) contains a reachable subtree for u with $h+1$ vertices. To express this property in first-order logic, we first introduce some auxiliary notation. Let $\mathcal{S}_h$ be a fixed set of rooted trees with vertex set $[h+1]$ such that $\mathcal{S}_h$ contains exactly one element from every isomorphism class of rooted trees on $h+1$ vertices. We assume that the edges of every tree $S \in \mathcal{S}_h$ are labelled by distinct numbers from $[h]$, and we denote the label of an edge $e \in E(S)$ by $\sigma(e)$. We denote by $a_i, b_i \in [h+1]$ the smallest and the largest endpoint of the edge $\sigma^{-1}(i)$, respectively. Given a rooted tree $S \in \mathcal{S}_h$, we define $\rho(S)$ to be the following set of pairs of edge labels

$$\Big\{\big(\sigma(e_1), \sigma(e_2)\big) : e_1, e_2 \in E(S), \exists v \in V(S) \text{ such that } e_1 \text{ lies on the path from } v \text{ to the root of } S, \text{ and } v \text{ is incident to } e_1, e_2\Big\}.$$

We now define a first-order formula expressing the property that there is some copy of S in G such that all vertices in this copy are temporally reachable in (G, λ) from the root via edges in S.

$$\theta(S) = \Big(\exists \text{ distinct } v_1, v_2, \ldots, v_{h+1} \in V(G)\Big)\Big(\exists (e_1, t_1), \ldots, (e_h, t_h) \in \Lambda(G, \lambda)\Big)\Big(\exists e'_1, \ldots, e'_h \in E(G)\Big)$$
$$\bigwedge_{i=1}^{h} \mathcal{L}(e'_i, (e_i, t_i)) \wedge \bigwedge_{i=1}^{h} \Big(\mathcal{I}(v_{a_i}, e'_i) \wedge \mathcal{I}(v_{b_i}, e'_i)\Big) \wedge \bigwedge_{(i,j)\in\rho(S)} \mathcal{R}((e_i, t_i), (e_j, t_j)).$$

In our modified temporal graph (that is, the graph obtained by deleting edges), the maximum temporal reachability is at most h if and only if there is no copy S of a rooted tree on $h+1$ vertices in G such that all vertices of S are temporally reachable in (G, λ) from the root via edges in S. We therefore define another formula, which captures the property that in any copy of such a tree, at least one edge must belong to the set E of removed edges:

$$\theta'(S, E) = \Big(\forall \text{ distinct } v_1, v_2, \ldots, v_{h+1} \in V(G)\Big)\Big(\forall (e_1, t_1), \ldots, (e_h, t_h) \in \Lambda(G, \lambda)\Big)\Big(\forall e'_1, \ldots, e'_h \in E(G)\Big)$$
$$\Bigg[\Bigg(\bigwedge_{i=1}^{h} \mathcal{L}(e'_i, (e_i, t_i)) \wedge \bigwedge_{i=1}^{h} \Big(\mathcal{I}(v_{a_i}, e'_i) \wedge \mathcal{I}(v_{b_i}, e'_i)\Big) \wedge \bigwedge_{(i,j)\in\rho(S)} \mathcal{R}((e_i, t_i), (e_j, t_j))\Bigg)$$
$$\implies \exists e \in E\Bigg(\bigvee_{i\in[h]} \mathcal{L}(e, (e_i, t_i))\Bigg)\Bigg].$$

We can now define an MSO formula which is true if and only if the deletion of a given set E of edges ensures that there is no "bad" subtree.

$$\phi_h(E) = \bigwedge_{S\in\mathcal{S}_h} \theta'(S, E).$$

Optimising to find the smallest possible set E satisfying $\phi_h(E)$ is then equivalent to solving TR Edge Deletion. Note that the length of the formula depends only on h. The result then follows from the application of Theorem 5.2 to the MSO formula ϕ_h. □

6 A "clocked" generalization of temporal reachability

In many applications, we might want to generalise our notion of temporal reachability: we might require that the time between arriving at and leaving any vertex on a temporal path falls within some fixed range. For example, in the context of disease transmission, an upper bound on the permitted time between entering and leaving a vertex might represent the time within which an infection would be detected and eliminated (thus ensuring no further transmission). On the other hand, a lower bound might represent the time before an individual becomes infectious or, in the case of vertices corresponding to multiple individuals in the same location (e.g. animals on a farm or humans in the same household) the minimum time individuals must spend together for there to be a non-trivial probability of disease transmission (so that one individual brings the infection to the location, but another transmits it onwards). Motivated by this, we now define a generalized notion of temporal reachability which allows for such "clocked" restrictions. For the rest of the section we fix two natural numbers α and β such that $\alpha \leq \beta$.

Definition. *Let (G, λ) be a temporal graph. An* (α, β)-temporal walk *from u to v in (G, λ) is a walk $u = v_0 \dots v_\ell = v$ in G such that, for each $1 \leq i \leq \ell - 1$, $\alpha \leq \lambda(v_i v_{i+1}) - \lambda(v_{i-1} v_i) \leq \beta$. An* (α, β)-temporal path *from u to v in (G, λ) is an (α, β)-temporal walk along which all vertices are distinct.*

Notice that, in this setting, the notion of reachability differs depending on whether we allow u to reach v via an (α, β)-temporal walk or only via an (α, β)-temporal path, since it is possible for there to exist an (α, β)-temporal walk from u to v but no (α, β)-temporal path; for an example, see Figure 5. The most appropriate choice of definition will depend upon the specific context. When considering the spread of a disease, if each vertex corresponds to an individual and an individual becomes immune to the disease upon recovery, then we should consider reachability via temporal paths only. On the other hand, if an individual can be infected with the same disease on multiple occasions, or if vertices in fact represent a group of individuals (e.g. the animals on a particular farm, or humans in the same household), then the notion of (α, β)-temporal walks provides a more realistic model.

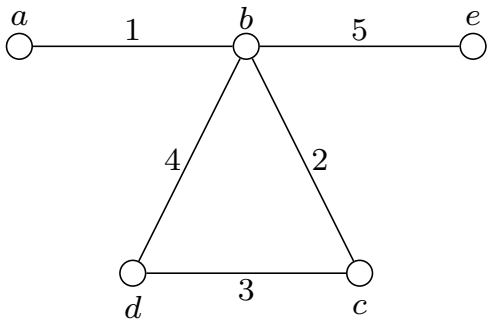

Figure 5: There exists a (1,1)-temporal walk from a to e, but there is no (1,1)-temporal path between the vertices. The (1,1)-reachability set of a contains all the vertices of the graph, and no edge can be removed without reducing the reachability set.

Here, we focus on the latter setting, for two reasons. Firstly, from an application perspective, we are particularly interested in the setting of livestock trade networks, in which it is to be expected that a single vertex (corresponding to a farm or similar) could be infected repeatedly as they restock with fresh animals. Secondly, even the problem of deciding the existence or otherwise of an (α, β)-temporal path between two vertices is known to be NP-complete [15], whereas (α, β)-temporal walks can be found efficiently [33]; therefore, there is much more hope of obtaining positive algorithmic results in the latter setting.

We therefore say that v is (α, β)-temporally reachable from u *starting at time* t_0 if there exists an (α, β)-temporal walk from u to v whose first edge e satisfies $t_0 + \alpha \leq \lambda(e) \leq t_0 + \beta$. The (α, β)-*temporal reachability set of u starting at time t_0* is defined in the obvious way, similarly to the classical temporal reachability (as defined in Section 2). The *maximum* (α, β)-*temporal reachability* of a temporal graph (G, λ) is the maximum cardinality of the (α, β)-temporal reachability set of v_0 starting at time t_0, taken over all pairs (v_0, t_0) with $v_0 \in V(G)$ and $t_0 \in \mathbb{N}$ where t_0 is at most the lifetime of the graph.

We now define the (α, β)-extension of TR Edge Deletion; note that this problem clearly belongs to NP since we can verify (α, β)-temporal reachability between each pair of vertices (starting at any given time t_0) in polynomial time [33].

(α, β)-Temporal Reachability Edge Deletion ((α, β)-TR Edge Deletion)
Input: A temporal graph (G, λ), and $k, h \in \mathbb{N}$.
Question: Is there a set $E' \subseteq E(G)$, with $\lvert E' \rvert \leq k$, such that the maximum (α, β)-temporal reachability of $(G, \lambda) \setminus E'$ is at most h?

All of our results – both algorithms and intractability results – can be generalised in a natural way to the more general setting of (α, β)-TR Edge Deletion, at the cost of a slightly worse approximation factor in some cases.

Theorem 6.1. *For any fixed $1 \leq \alpha < \beta$, (α, β)-TR Edge Deletion is W[1]-hard when parameterised by the maximum number k of edges that can be removed, even when the input temporal graph has lifetime $2\alpha + 1$.*

Proof. With a slight modification, the reduction of Theorem 3.1 works also for (α, β)-TR Edge Deletion. Indeed, given an instance of (α, β)-TR Edge Deletion, the reduced graph (G, λ) is constructed exactly as one in the proof of Theorem 3.1, with the only difference that every time label "1" is replaced by $\alpha + 1$ and every time label "2" is replaced by "$2\alpha + 1$". The proof then works verbatim for the generalized problem (α, β)-TR Edge Deletion (note that the maximum (α, β)-temporal reachability of the resulting graph will be the cardinality of the (α, β)-temporal reachability set of s starting at time 1). □

Exactly the same arguments as in the proof of Theorem 6.1 show that the following analogue of Theorem 3.2 holds.

Theorem 6.2. *(α, β)-TR Edge Deletion is NP-complete, even if the maximum temporal reachability h is at most 6, and the input temporal graph (G, λ) has:*

1. *maximum degree Δ_G at most 5, and*
2. *lifetime at most $2\alpha + 1$.*

To adapt Theorem 4.3 to the generalized setting, we need an (α, β)-analogue of Lemma 4.1; however, the minimal set of edges needed for a single vertex to reach h others may no longer form a tree (see Figure 5 for an example). Such an analogue is obtained as an easy adaptation of [33, Theorem 1].

Lemma 6.3 (Follows from [33])**.** *Let (G, λ) be a temporal graph, and h a positive integer. There is an algorithm running in polynomial time which, on input $((G, \lambda), h)$,*

1. *if the maximum (α, β)-temporal reachability of (G, λ) is at most h, outputs "YES";*
2. *if the maximum (α, β)-temporal reachability of (G, λ) is greater than h, outputs a vertex $v_0 \in V(G)$, a time t_0, and a subgraph H of G on exactly $h + 1$ vertices such that every vertex in H is (α, β)-temporally reachable in H from v_0 starting at time t_0.*

Since the subgraph H we find in the case of a no-instance is not necessarily a tree, when imitating the proof of Theorem 4.3 we might have to delete up to $\binom{h+1}{2}$ edges in this setting, resulting in a worse approximation factor.

Theorem 6.4. *There exists a polynomial-time algorithm to compute an $\left(\frac{h(h+1)}{2}\right)$-approximation to* Min *$(\alpha, \beta)$-TR Edge Deletion, where h denotes the maximum permitted reachability.*

Theorem 6.5. *There exists a polynomial-time algorithm to compute a c-approximation to* Min *(α, β)-TR Edge Deletion, provided that a layout of cutwidth c is given.*

Proof. The proof of Theorem 4.5 applies here; it suffices to notice that we can determine the maximum (α, β)-reachability of any subgraph efficiently. □

Theorem 6.6. *Unless $P = NP$, there exists a natural number c such that for every $\delta \in \left(0, \frac{1}{c+2}\right)$,* Min (α, β)-TR Edge Deletion *cannot be approximated in polynomial time to within a factor of $\delta \ln n$, where n is the number of vertices in the input temporal graph, even if the input temporal graph has lifetime $2\alpha + 1$.*

Proof. We adapt the proof of Theorem 4.7 by replacing every time label "1" with time label $\alpha + 1$ and every label "2" with "$2\alpha + 1$"; the result follows immediately. □

Theorem 6.7. (α, β)-TR Edge Deletion *admits an FPT algorithm with respect to the combined parameter of h, Δ_G, and* $\mathrm{tw}(G)$.

Proof. The proof follows the logic of the proof of Theorem 5.4, but requires changes to reflect the "clocked" restrictions and the fact that a minimum (α, β)-reachability subgraph is not necessarily a tree.

First, we replace the relation $\mathcal{R}$ by the relation $\mathcal{R}'$, where $((e_1, t_1), (e_2, t_2)) \in \mathcal{R}'$ if and only if $((e_1, t_1), (e_2, t_2)) \in \mathcal{R}$ and $\alpha \leq t_2 - t_1 \leq \beta$; and introduce one more binary relation $\mathcal{P}$, where $((e_1, t_1), (e_2, t_2)) \in \mathcal{P}$ if and only if e_1, e_2 share a vertex in G, and there exists a natural number t_0 such that $t_0 + \alpha \leq t_1 \leq t_0 + \beta$ and $t_0 + \alpha \leq t_2 \leq t_0 + \beta$.

Next, it follows by definition that a vertex u has a (α, β)-temporal reachability set of size at least $h + 1$ if and only if there exist a subgraph H of G on $h + 1$ vertices and a natural number t_0 such that $u \in V(H)$ and every vertex $v \in V(H) \setminus \{u\}$ is (α, β)-temporally reachable from u starting at time t_0 and using only edges of H. The latter means that for every vertex $v \in V(H) \setminus \{u\}$, there exists a walk $W_v = (x_0^v x_1^v \dots x_{\ell_v}^v)$ in H from u to v (with $x_0^v = u$ and $x_{\ell_v}^v = v$) such that

(1) $t_0 + \alpha \leq \lambda(x_0^v x_1^v) \leq t_0 + \beta$, and

(2) $\alpha \leq \lambda(x_i^v x_{i+1}^v) - \lambda(x_{i-1}^v x_i^v) \leq \beta$, for each $1 \leq i \leq \ell_v - 1$.

To express these conditions in first-order logic, we first introduce some auxiliary notation. Let H be a connected graph with vertex set $[h + 1]$ and m edges that are labeled by distinct numbers from $[m]$. We denote the label of an edge $e \in E(H)$ by $\sigma(e)$. Furthermore, we denote by $a_i, b_i \in [h + 1]$ the smallest and the largest endpoint of the edge $\sigma^{-1}(i)$, respectively. We assume that for every vertex $v \in V(H) \setminus \{u\}$, the graph H contains a walk $W_v = (x_0^v x_1^v \dots x_{\ell_v}^v)$ from u to v (with $x_0^v = u$ and $x_{\ell_v}^v = v$) such that the labels of the edges of the walk increase along the walk. We will call the triple $\big(H, u, \{W_v | v \in V(H) \setminus \{u\}\}\big)$ an (α, β)-*reachability template*, and denote by $\mathcal{D}_{h,m}$ the set of all such templates over graphs H with $h + 1$ vertices and m edges.

We now define the first-order formula expressing the property that there is some copy of H in G such that all vertices in this copy are (α, β)-temporally reachable in (G, λ) according to an (α, β)-reachability template $D = \big(H, u, \{W_v | v \in V(H) \setminus \{u\}\}\big) \in \mathcal{D}_{h,m}$.

$$\begin{aligned}\nu(D) = &\Big(\exists \text{ distinct } v_1, v_2, \dots, v_{h+1} \in V(G)\Big)\Big(\exists (e_1, t_1), \dots, (e_m, t_m) \in \Lambda(G, \lambda)\Big)\Big(\exists e'_1, \dots, e'_m \in E(G)\Big)\\
&\bigwedge_{i=1}^{m} \mathcal{L}(e'_i, (e_i, t_i)) \wedge \bigwedge_{i=1}^{m} \Big(\mathcal{I}(v_{a_i}, e'_i) \wedge \mathcal{I}(v_{b_i}, e'_i)\Big) \wedge\\
&\bigwedge_{v,w \in V(H)\setminus\{u\}} \mathcal{P}\left((e_{\sigma(x_0^v x_1^v)}, t_{\sigma(x_0^v x_1^v)}), (e_{\sigma(x_0^w x_1^w)}, t_{\sigma(x_0^w x_1^w)})\right) \wedge\\
&\bigwedge_{v \in V(H)\setminus\{u\}} \bigwedge_{i=1}^{\ell_v - 1} \mathcal{R}'\left((e_{\sigma(x_{i-1}^v x_i^v)}, t_{\sigma(x_{i-1}^v x_i^v)}), (e_{\sigma(x_i^v x_{i+1}^v)}, t_{\sigma(x_i^v x_{i+1}^v)})\right).\end{aligned}$$

As in the case of TR Edge Deletion the first part of the above formula,

$$\bigwedge_{i=1}^{m} \mathcal{L}(e'_i, (e_i, t_i)),$$

is included for technical reasons, so that we have access to edge variables corresponding to the associated time-edge pairs in the second part of the formula,

$$\bigwedge_{i=1}^{m} \Big(\mathcal{I}(v_{a_i}, e'_i) \wedge \mathcal{I}(v_{b_i}, e'_i)\Big),$$

which expresses the property that H is a subgraph of G. The third part of the formula

$$\bigwedge_{v,w \in V(H)\setminus\{u\}} \mathcal{P}\left((e_{\sigma(x_0^v x_1^v)}, t_{\sigma(x_0^v x_1^v)}), (e_{\sigma(x_0^w x_1^w)}, t_{\sigma(x_0^w x_1^w)})\right)$$

expresses the property that there exists a natural number t_0 such that for all first edges of the walks $W_v, v \in V(H) \setminus \{u\}$ their time labels belong to the interval $[t_0 + \alpha, t_0 + \beta]$. To see that this is indeed equivalent to the requirement, expressed by this part of the formula, that for *every pair* of vertices $v, w \in V(H) \setminus \{u\}$ there exists a natural number t_0 such that the time labels of the first edges of W_v and W_w are in the interval $[t_0 + \alpha, t_0 + \beta]$, consider the smallest and largest time label assigned to any first edge on a walk: if there exists t_0 such that both these labels belong to the interval $[t_0 + \alpha, t_0 + \beta]$ then the labels of all other first edges must also belong to this interval, as required. Finally, the fourth part of the formula

$$\bigwedge_{v \in V(H)\setminus\{u\}} \bigwedge_{i=1}^{\ell_v - 1} \mathcal{R}'\left((e_{\sigma(x_{i-1}^v x_i^v)}, t_{\sigma(x_{i-1}^v x_i^v)}), (e_{\sigma(x_i^v x_{i+1}^v)}, t_{\sigma(x_i^v x_{i+1}^v)})\right)$$

expresses the property that all walks from u to $v \in V(H) \setminus \{u\}$ are (α, β)-temporal walks.

Now, similarly to the formula for TR Edge Deletion in Section 5.2, we define a formula, which captures the property that in any such copy of H at least one edge must belong to the set E of removed edges

$$\begin{aligned}
\nu'(D,E) = &\Big(\forall \text{ distinct } v_1, v_2, \ldots, v_{h+1} \in V(G)\Big)\Big(\forall (e_1,t_1), \ldots, (e_m,t_m) \in \Lambda(G,\lambda)\Big)\Big(\forall e'_1, \ldots, e'_m \in E(G)\Big) \\
&\Bigg[\Bigg(\bigwedge_{i=1}^{m} \mathcal{L}(e'_i, (e_i,t_i)) \wedge \bigwedge_{i=1}^{m} \Big(\mathcal{I}(v_{a_i}, e'_i) \wedge \mathcal{I}(v_{b_i}, e'_i)\Big) \wedge \\
&\quad \bigwedge_{v,w \in V(H)\setminus\{u\}} \mathcal{P}\left((e_{\sigma(x_0^v x_1^v)}, t_{\sigma(x_0^v x_1^v)}), (e_{\sigma(x_0^w x_1^w)}, t_{\sigma(x_0^w x_1^w)})\right) \wedge \\
&\quad \bigwedge_{v \in V(H)\setminus\{u\}} \bigwedge_{i=1}^{\ell_v - 1} \mathcal{R}'\left((e_{\sigma(x_{i-1}^v x_i^v)}, t_{\sigma(x_{i-1}^v x_i^v)}), (e_{\sigma(x_i^v x_{i+1}^v)}, t_{\sigma(x_i^v x_{i+1}^v)})\right)\Bigg) \\
&\quad \implies \exists e \in E \left(\bigvee_{i \in [m]} \mathcal{L}(e, (e_i,t_i))\right)\Bigg].
\end{aligned}$$

Finally, we define an MSO formula which is true if and only if the deletion of a given set E of edges ensures that there is no "bad" subgraph.

$$\phi_h(E) = \bigwedge_{m=h}^{\binom{h+1}{2}} \bigwedge_{D \in \mathcal{D}_{h,m}} \nu'(D,E).$$

Similarly to the proof of Theorem 5.4, optimising to find the smallest possible set E satisfying $\phi_h(E)$ is then equivalent to solving (α, β)-TR Edge Deletion. Note that the length of the formula depends only on h. The result then follows from the application of Theorem 5.2 to the MSO formula ϕ_h. □

7 Conclusions and open problems

In this paper we studied the problem TR Edge Deletion of removing a small number of *edges* from a given *temporal graph* (i.e. a graph that changes over time) to ensure that every vertex has a temporal path to fewer than h other vertices. The main motivation for this problem comes from the need to limit the spread of disease over a network, for example in a livestock trade network in which farms are represented by vertices and the edges encode trades of animals between farms [21, 45]. Further motivation for the problem of removing edges to limit the temporal connectivity of a temporal graph comes from scenarios of sensitive information propagation through rumor-spreading. In practical applications, removing an

edge would correspond to completely prohibiting any contact between two entities, while removing an edge availability at time t would correspond to just temporally restricting their contact at that time point. Motivated by these applications, we also considered a "clocked" generalisation of the problem in which transmission of disease (or information) to a vertex's neighbours can only occur in some specified time-window after the vertex is itself infected.

We showed that both problems remain NP-complete even when strong restrictions are placed on the input. More specifically, the problems are para-NP-hard with respect to the combination of the three parameters h (the maximum permitted reachability), the maximum degree Δ_G of G, and lifetime of (G, λ); they are also W[1]-hard parameterised by the number k of permitted deletions. Moreover, with respect to this last parameterisation, we cannot improve significantly on a brute force approach unless the Exponential Time Hypothesis fails. On the positive side, we proved that these problems admit fixed-parameter tractable algorithms with respect to the combination of three parameters: the treewidth $\mathrm{tw}(G)$ of the underlying graph G, the maximum allowed temporal reachability h, and the maximum degree Δ_G of (G, λ). It remains open whether TR Edge Deletion is in FPT parameterised simultaneously by the treewidth and either one of Δ_G and h.

We also considered approximation algorithms for the optimisation version of TR Edge Deletion in which the goal is to find a minimum-size set of edges to delete. We demonstrated that an h-approximation can be found in polynomial time on arbitrary graphs, and a constant factor polynomial approximation is possible on graphs of bounded cutwidth. However, we also showed that there is unlikely to be a polynomial-time algorithm to compute any constant-factor approximation in general, even on temporal graphs of lifetime two. Nevertheless, a natural open question is whether we can improve the approximation ratio for general graphs.

Acknowledgements. The authors wish to thank Bruno Courcelle and Barnaby Martin for useful discussions and hints on monadic second order logic. We are grateful to the anonymous referees for their thorough reading of the paper and the insightful comments and suggestions, which considerably improved the presentation of the paper.

References

[1] Eric Aaron, Danny Krizanc, and Elliot Meyerson. DMVP: foremost waypoint coverage of time-varying graphs. In *Proceedings of the 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 8747 of *Lecture Notes in Computer Science*, pages 29–41. Springer, 2014.

[2] Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87:109–120, 2016.

[3] Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017.

[4] Eleni C. Akrida, George B. Mertzios, Sotiris E. Nikoletseas, Christoforos L. Raptopoulos, Paul G. Spirakis, and Viktor Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, (ICALP)*, volume 132 of *LIPIcs*, pages 131:1–131:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[5] Eleni C Akrida, George B Mertzios, Paul G Spirakis, and Viktor Zamaraev. Temporal vertex cover with a sliding time window. *Journal of Computer and System Sciences*, 107:108–123, 2020.

[6] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277—284, 1987.

[7] Alain Barrat, Marc Barthlemy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 2008.

[8] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 226–234, 1993.

[9] Alfredo Braunstein and Alessandro Ingrosso. Inference of causality in epidemics on temporal contact networks. *Scientific Reports*, 6:27538, 2016. URL: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4901330/`, `doi:10.1038/srep27538`.

[10] Dirk Brockmann and Dirk Helbing. The hidden geometry of complex, network-driven contagion phenomena. *Science*, 342(6164):1337–1342, 2013.

[11] Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

[12] Arnaud Casteigts and Paola Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013. URL: `https://hal.archives-ouvertes.fr/hal-00865762`.

[13] Arnaud Casteigts and Paola Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013. URL: `https://hal.archives-ouvertes.fr/hal-00865764`.

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

[15] Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. The computational complexity of finding temporal paths under waiting time constraints. *arXiv preprint arXiv:1909.06437*, 2019.

[16] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David Juedes, Iyad A Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005.

[17] Andrea E. F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 24(4):1694–1712, 2010.

[18] Vittoria Colizza, Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences*, 103(7):2015–2020, 2006.

[19] Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*. Cambridge University Press, 2012.

[20] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th annual ACM Symposium on Theory of Computing (STOC)*, pages 624–633. ACM, 2014.

[21] Jessica Enright and Kitty Meeks. Deleting edges to restrict the size of an epidemic: a new application for treewidth. *Algorithmica*, 80(6):1857–1889, 2018.

[22] Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. 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)*, volume 138 of *LIPIcs*, pages 57:1–57:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[23] Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning times to minimise reachability in temporal graphs, 2021. `doi:https://doi.org/10.1016/j.jcss.2020.08.001`.

[24] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 9134 of *Lecture Notes in Computer Science*, pages 444–455. Springer, 2015.

[25] Afonso Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5):24–29, 2004.

[26] Stephen Finbow and Gary MacGillivray. The firefighter problem: a survey of results, directions and questions. *Australasian J. Combinatorics*, 43:57–78, 2009.

[27] Paola Flocchini, Bernard Mans, and Nicola Santoro. Exploration of periodically varying graphs. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, volume 5878 of *Lecture Notes in Computer Science*, pages 534–543. Springer, 2009.

[28] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[29] Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 2019.

[30] George Giakkoupis, Thomas Sauerwald, and Alexandre Stauffer. Randomized rumor spreading in dynamic graphs. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 8573 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2014.

[31] Daniel T. Haydon, Rowland R. Kao, and Paul R. Kitching. The UK foot-and-mouth disease outbreak—the aftermath. *Nature Reviews Microbiology*, 2(8):675, 2004.

[32] Itai Himelboim, Marc A. Smith, Lee Rainie, Ben Shneiderman, and Camila Espina. Classifying twitter topic-networks using social network analysis. *Social Media + Society*, 3(1):2056305117691545, 2017.

[33] Anne-Sophie Himmel, Matthias Bentert, André Nichterlein, and Rolf Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. In Hocine Cherifi, Sabrina Gaito, José Fernendo Mendes, Esteban Moro, and Luis Mateus Rocha, editors, *Complex Networks and Their Applications VIII*, pages 494–506, Cham, 2020. Springer International Publishing.

[34] Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35:1–35:16, 2017.

[35] Petter Holme and Jari Saramäki, editors. *Temporal Networks*. Springer, 2013.

[36] David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.

[37] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[38] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[39] George B. Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 7966 of *Lecture Notes in Computer Science*, pages 657–668. Springer, 2013.

[40] George B Mertzios, Hendrik Molter, Rolf Niedermeier, Viktor Zamaraev, and Philipp Zschoche. Computing maximum matchings in temporal graphs. In *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[41] George B Mertzios, Hendrik Molter, and Viktor Zamaraev. Sliding window temporal graph coloring. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7667–7674, 2019.

[42] Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.

[43] Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, 2018.

[44] A. Mitchell, D. Bourn, J. Mawdsley, W. Wint, R. Clifton-Hadley, and M. Gilbert. Characteristics of cattle movements in Britain – an analysis of records from the cattle tracing system. *Animal Science*, 80(3):265–273, 2005.

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

[46] Dana Moshkovitz. The projection games conjecture and the NP-hardness of ln n-approximating set-cover. In *Proceedings of the 15th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-RANDOM 2012)*, pages 276–287, 2012.

[47] Maria Noremark and Stefan Widgren. EpiContactTrace: an R-package for contact tracing during livestock disease outbreaks and for risk-based surveillance. *BMC Veterinary Research*, 10(1), 2014.

[48] Piotr Sapiezynski, Arkadiusz Stopczynski, Radu Gatej, and Sune Lehmann. Tracking human mobility using wifi signals. *PloS One*, 10(7):e0130824, 2015.

[49] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag Berlin Heidelberg, 2003.

[50] John Kit Tang, Mirco Musolesi, Cecilia Mascolo, and Vito Latora. Characterising temporal distance and reachability in mobile and online social networks. *ACM Computer Communication Review*, 40(1):118–124, 2010.

[51] Dimitrios M. Thilikos, Maria Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms*, 56(1):1 – 24, 2005.

[52] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.

[53] Eugenio Valdano, Luca Ferreri, Chiara Poletto, and Vittoria Colizza. Analytical computation of the epidemic threshold on temporal networks. *Phys. Rev. X*, 5:021005, Apr 2015.

[54] Eugenio Valdano, Chiara Poletto, Armando Giovannini, Diana Palma, Lara Savini, and Vittoria Colizza. Predicting epidemic risk from past temporal contact data. *PLoS Computational Biology*, 11(3):e1004152, 03 2015.

[55] Jordan Viard, Matthieu Latapy, and Clémence Magnien. Revealing contact patterns among high-school students using maximal cliques in link streams. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1517–1522, 2015.

[56] Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

[57] Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020.