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

Jessica Enright
Global Academy of Agriculture and Food Security, University of Edinburgh, UK
jessica.enright@ed.ac.uk

Kitty Meeks
School of Computing Science, University of Glasgow, UK
kitty.meeks@glasgow.ac.uk

George B. Mertzios
Department of Computer Science, Durham University, UK
george.mertzios@durham.ac.uk

Viktor Zamaraev
Department of Computer Science, Durham University, UK
viktor.zamaraev@durham.ac.uk

Abstract
Spreading processes on graphs are a natural model for a wide variety of real-world phenomena, including information or behaviour spread over social networks, biological diseases spreading over contact or trade networks, and the potential flow of goods over logistical infrastructure. Often, the networks over which these processes spread are dynamic in nature, and can be modeled with graphs whose structure is subject to discrete changes over time, i.e. with temporal graphs. Here, we consider temporal graphs in which edges are available at specified timesteps, and 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 (i.e. a path, along which the times of the edge availabilities increase). We introduce a natural deletion problem for temporal graphs and we provide positive and negative results on its computational complexity, both in the traditional and the parameterised sense (subject to various natural parameters), as well as addressing the approximability of this problem.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Temporal networks, spreading processes, graph modification, parameterised complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.10

Funding Kitty Meeks: supported by a Royal Society of Edinburgh Personal Research Fellowship, funded by the Scottish Government.
George B. Mertzios: partially supported by the EPSRC Grants EP/P020372/1
Viktor Zamaraev: supported by the EPSRC Grant EP/P020372/1

Acknowledgements The authors wish to thank Bruno Courcelle and Barnaby Martin for useful discussions and hints on monadic second order logic.

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 change dynamically, transportation networks, and several physical systems are only a few examples of networks that change over time [31,38]. Due to
Deleting edges to restrict the size of an epidemic in temporal networks

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, 24, 44], evolving [11, 15, 22],
dynamic [14, 27], and graphs over time [33]; for a recent 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 optimization problems
that have been studied so far are based on the notion of temporal paths (see Definition 2
below) and other path-related notions, such as temporal analogues of distance, diameter,
reachability, exploration, and centrality [2–4, 19, 21, 35, 37]. Recently, non-path temporal graph
problems have also been addressed theoretically, including for example temporal variations
of coloring [36], vertex cover [5], and maximal cliques [30, 49, 50].

Inspired by the foundational work of Kempe et al. [32], 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.

**Definition 1** (temporal graph). A temporal graph is a pair \((G, \lambda)\), where \(G = (V, E)\) is an
underlying (static) graph and \(\lambda : E \to 2^\mathbb{N}\) is a time-labelling function which assigns to every
edge of \(G\) a set of discrete-time labels.

For every edge \(e \in E\) in the underlying graph \(G\) of a temporal graph \((G, \lambda)\), \(\lambda(e)\) denotes
the set of time slots at which \(e\) is active in \((G, \lambda)\).

Unless stated otherwise, to simplify the presentation of our results we restrict our
attention in this paper to temporal graphs in which each edge is assigned a singleton set by
the time-labelling function, that is, in which each edge is active at exactly one time.

Spreading processes on networks or graphs are a topic of significant research across
network science [7], and a variety of application areas [28, 29], as well as inspiring more
theoretical algorithmic work [23]. 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, and rumours or news (both fake and real) spreading
over information-passing networks. 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 [42], agricultural trade networks [39], and social networks [34]). 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 [16], and fine-grained recording of livestock
movements between farms in most European nations [40]. There is very strong evidence
that these networks play a critical role in large and damaging epidemics, including the 2009

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, 41, 47, 48].

Here, similarly to [20], we focus our attention on deleting edges from \((G, \lambda)\) in order
to limit the temporal connectivity of the remaining temporal subgraph. To this end, the
The following temporal extension of the notion of a path in a static graph is fundamental [32,35].

Definition 2 (temporal path). A temporal path from $u$ to $v$ in a temporal graph $(G, \lambda)$ 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 $0 \leq i < k$.

In many applications, it may be more realistic to generalise our notion of temporal paths so that the time between arriving at and leaving any vertex must fall 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 between individuals being exposed to an infection and becoming infectious themselves.

Our contribution

We consider a natural deletion problem for temporal graphs, namely Temporal Reachability Edge Deletion (or 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 $(G, \lambda)$ and two natural numbers $k, h$, the goal is to delete at most $k$ edges from $(G, \lambda)$ 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 very restricted classes of graphs. We give two different reductions. The first shows that, assuming the Exponential Time Hypothesis, it is unlikely that we can 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 $\Delta_G$, or lifetime of $(G, \lambda)$ (i.e. the maximum label assigned by $\lambda$ 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 an $h$-approximation to Min TR Edge Deletion on arbitrary temporal graphs, then show how similar techniques can be applied to compute a $c$-approximation on inputs in which the underlying graph has cutwidth $c$. We conclude our consideration of approximation algorithms by showing that in general there is unlikely to be a polynomial-time algorithm to compute any constant-factor approximation, even on temporal graphs of lifetime two.

In Section 5, we consider exact FPT algorithms. Our hardness results show that the problem remains intractable when parameterised by $h$ or $\Delta_G$ alone; here we obtain an FPT algorithm by parameterising simultaneously by $h$, $\Delta_G$ and the treewidth $\text{tw}(G)$ 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 14) can be applied to solve problems in temporal graphs.

We note that all of our results can be applied, with minor modifications to the proofs, to the setting of $(\alpha, \beta)$-temporal paths.
## 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 $G$. Given a temporal graph $(G, \lambda)$, where $G = (V, E)$, the maximum label assigned by $\lambda$ to an edge of $G$, called the lifetime of $(G, \lambda)$, is denoted by $T(G, \lambda)$, or simply by $T$ when no confusion arises. That is, $T(G, \lambda) = \max\{t \in \lambda(e) : e \in E\}$. Throughout the paper we consider temporal graphs with finite lifetime $T$.

Furthermore, we assume that the given labelling $\lambda$ is arbitrary, i.e. $(G, \lambda)$ is given with an explicit list of labels for every edge. Thus, the size of the input temporal graph $(G, \lambda)$ is $O\left(|V| + T + \sum_{i=1}^{T} |E_i|\right) = O(n + mT)$: when we are considering temporal graphs in which edges are active at a single timestep, it suffices to only consider the space required to represent the single time assigned to each edge, and thus the size of the temporal graph is $O(n + m \log T)$. We say that an edge $e \in E$ appears at time $t$ if $t \in \lambda(e)$, and in this case we call the pair $(e, t)$ a time-edge in $(G, \lambda)$. Given a subset $E' \subseteq E$, we denote by $(G, \lambda) \setminus E'$ the temporal graph $(G', \lambda')$, where $G' = (V, E \setminus E')$ and $\lambda'$ is the restriction of $\lambda$ to $E \setminus E'$.

We say that a vertex $v$ is temporally reachable from $u$ in $(G, \lambda)$ 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 $(G, \lambda)$, and $k, h \in \mathbb{N}$.

**Output:** Is there a set $E' \subseteq E(G)$, with $|E'| \leq k$, 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 [3, 32, 35]). 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 [26, 51].

## 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 $n^{O(k)}$, in which we consider all possible sets of $k$ edges to delete, cannot be significantly improved in general.

**Theorem 4.** 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. Moreover, assuming ETH, there is no $f(k)\cdot \tau^{o(k)}$ time algorithm for TR Edge Deletion, where $\tau$ is the size of the input temporal graph.

The $W[1]$-hardness reduction of Theorem 4 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 [43, 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 5.** TR Edge Deletion is NP-complete, even when the maximum temporal reachability $h$ is at most 7 and the input temporal graph $(G, \lambda)$ has:

1. maximum degree $\Delta_G$ 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 each of the parameters $h$, $\Delta_G$, 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 [46].

**3,4-SAT**

**Input:** A CNF formula $\Phi$ with exactly 3 variables per clause, such that each variable appears in at most 4 clauses.

**Output:** Does there exists a truth assignment satisfying $\Phi$?

Let $\Phi$ 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 $\Phi$. Indeed, if a variable $x_i$ appears only negated (resp. unnegated) in $\Phi$, then we can trivially set $x_i = 0$ (resp. $x_i = 1$) and then remove from $\Phi$ 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 $\Phi$ is satisfiable.

![Figure 1](image.png)

**Figure 1** 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 $(G, \lambda)$ as follows. For each variable $x_i$ we introduce in $G$ a copy of the subgraph shown in Figure 1, which we call an $x_i$-gadget. There are three special vertices in an $x_i$-gadget: $x_i$, $\overline{x_i}$, which we call literal vertices, and $v_{x_i}$, which we call the head vertex of the $x_i$-gadget. All the edges incident to $v_{x_i}$ appear in time step 1, the other two edges of
Deleting edges to restrict the size of an epidemic in temporal networks

$x_i$-gadget, which we call literal edges, appear in time step 2. Additionally, for every clause $C_s$ we introduce in $G$: 1) a clause vertex $C_s$ that is adjacent to the three literal vertices corresponding to the literals of $C_s$, and 2) one more vertex adjacent only to $C_s$, which we call the satellite vertex of $C_s$. All the new edges incident to $C_s$ appear in time step 1. See Figure 2 for an illustration. Finally, we set $k = n$ and $h = 7$.

First recall that, in $\Phi$, 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 $\Phi$, it follows that each of the two vertices corresponding to the literals $x_i, \overline{x_i}$ is connected to at most three clause gadgets. Therefore the degree of each vertex corresponding to a literal in the constructed temporal graph $(G, \lambda)$ (see Figure 2) is at most five. Moreover, it can be easily checked that the same also holds for every other vertex of $(G, \lambda)$, and thus $\Delta_{G,\lambda} \leq 5$.

We continue by observing temporal reachabilities of the vertices of $(G, \lambda)$. A literal vertex can temporally reach only the corresponding clause vertices, and the two neighbours in its gadget. Since every literal belongs to at most 4 clauses in $\Phi$, the temporal reachability of the literal vertex in $(G, \lambda)$ is at most 7 (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 $(G, \lambda)$ is 8. Any other vertex belonging to a gadget can temporally reach only its unique neighbour in $G$ and so has temporal reachability 2. Every clause vertex can reach only the corresponding literal vertices, their neighbours incident to the literal edges, and its own satellite vertex. Hence the temporal reachability of every clause vertex in $(G, \lambda)$ is 8. Finally, every satellite vertex reaches only its neighbour, and thus its temporal reachability is 2. 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 7, then $\Phi$ 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 $\Phi$.

To show the converse, given a truth assignment $(\alpha_1, \ldots, \alpha_n)$ satisfying $\Phi$ we construct a set $E$ of $n$ edges such that the maximum temporal reachability of $(G, \lambda) \setminus E$ is at most 7. 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 $\Phi$, 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 $(G, \lambda)$, we strictly decrease temporal reachability of every head and clause vertex. ♦
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 \((G, \lambda)\) 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 \((G, \lambda)\) 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 \((T, \lambda')\), where \(\lambda'\) is the restriction of \(\lambda\) 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 6.** Let \((G, \lambda)\) 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 \((G, \lambda)\) is at most \(h\), outputs “YES”;
2. if the maximum temporal reachability of \((G, \lambda)\) 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.

Let \(h\) be a positive integer and \((G = (V, E), \lambda)\) be a temporal graph. We say that edge set \(E' \subseteq E\) is a valid deletion of \((G = (V, E), \lambda)\) with respect to \(h\) if the maximum temporal reachability of \((G = (V, E), \lambda) \setminus E'\) 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 7.** Let \((G, \lambda)\) 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 with respect to \(h\), then \(|E' \cap E(T)| \geq 1\).

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

**Theorem 8.** There exists a polynomial-time algorithm to compute an \(h\)-approximation to Min TR Edge Deletion, where \(h\) denotes the maximum permitted reachability.
Deleting edges to restrict the size of an epidemic in temporal networks

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 \((G, \lambda)\) 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 6 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 \(E_{\text{opt}}\) 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 temporal 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 \(i\) with \(1 \leq i < n\) the number of edges with one endpoint in \(v_1, \ldots, v_i\) and one in \(v_{i+1}, \ldots, v_n\) is at most \(c\). Given a layout \(v_1, v_2, \ldots, v_n\), we say that edges with one endpoint in \(v_1, \ldots, v_i\) and one in \(v_{i+1}, \ldots, v_n\) 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. [45] 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 8 to give a polynomial-time algorithm to compute a \(c\)-approximation to Min TR Edge Deletion, where \(c\) is the cutwidth of the input temporal graph. In addition to Lemma 7, we will also make use of the following definition and observation:

Let \(G = (V, E)\) be a graph. We say that an edge set \(E_S \subseteq E\) is an edge separator that separates \(G\) into \(G_A = (V_A, E_A)\) and \(G_B = (V_B, E_B)\) if, in \(G_S = (V, E \setminus E_S)\) no vertex in \(V_A\) is reachable from \(V_B\).

Lemma 9. Let \(h\) be a positive integer and \((G = (V, E), \lambda)\) be a temporal graph with an edge separator \(E_S\) that separates \(G\) into \(G_A = (V_A, E_A)\) and \(G_B = (V_B, E_B)\). If, for the given \(h\), \(E_A'\) and \(E_B'\) are valid deletion sets for \((G_A, \lambda|_{E_A'})\), \((G_B, \lambda|_{E_B'})\), respectively, then \(E_A' \cup E_B' \cup E_S\) is a valid deletion set for \((G = (V, E), \lambda)\).

We now describe a cutwidth approximation algorithm:

Theorem 10. 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 (Sketch). Let \((G = (V, E), \lambda)\) 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. We claim that the following algorithm returns a \(c\)-approximation to Min TR Edge Deletion in polynomial time:
1. Initialise $E' := \emptyset$.
2. Initialise $i := 0$.
3. While $(G, \lambda)$ 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 and update $(G, \lambda) \leftarrow (G, \lambda) \setminus E'$.
   c. Update $i \leftarrow j + 1$
4. Return $E'$.

For any fixed cutwidth $c$, using the layout generation algorithm given by Thilikos et al. [45] and the algorithm described above, we can give a cutwidth-approximation to Min TR Edge Deletion for graphs with cutwidth $c$.

**Corollary 11.** There exists a polynomial-time algorithm to compute a $c$-approximation to Min TR Edge Deletion whenever the cutwidth of the input graph is bounded above by $c$.

Note that as paths have cutwidth one, Corollary 11 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 12.** Unless $P = NP$, Min TR Edge Deletion cannot be approximated in polynomial time to within a factor of $(1 - o(1)) \ln \log_2 \sqrt{n}$, where $n$ is the number of vertices in the input temporal graph, even if the input temporal graph has lifetime two.

### 5 An exact FPT algorithm

In this section we show that TR Edge Deletion admits an FPT algorithm, when simultaneously parameterised by $h$, $\Delta_G$, and $tw(G)$, where $\Delta_G$ is the maximum degree of $G$ and $tw(G)$ is the treewidth of $G$. It is worth noting that, although the parameters $h$ and $\Delta_G$ may at first seem to be large, parameterising only by these two parameters is not enough, as our results in the previous sections (see e.g. Theorem 5) imply that TR Edge Deletion is para-NP-hard, when simultaneously parameterised by $h$ and $\Delta_G$.

Our results in this section (see Theorem 16) illustrate how a celebrated theorem by Courcelle (see Theorem 14) 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 $\tau$ of relations (i.e. a suitable relational vocabulary) and a Monadic Second Order (MSO) formula $\phi$ (of length $\ell$) that expresses our temporal graph problem at hand; (ii) we represent an arbitrary input temporal graph $(G, \lambda)$ 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 $\ell$ and $t$ are bounded; that is, in time $f(\ell, t) \cdot |\mathcal{A}|$.

Here, we apply this general framework to the particular problem TR Edge Deletion (by appropriately defining $\tau$, $\phi$, and $\mathcal{A}$) such that $\ell$ only depends on our parameter $h$, while $t$ only depends on $tw(G)$ and $\Delta_G$; 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 $(G, \lambda)$. 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 [25].
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 $(T, B)$ is a tree decomposition of $G$ if $T$ is a tree and $B = \{B_s : s \in V(T)\}$ 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 $\{s \in T : v \in B_s\}$ 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 $u, v \in B_s$.

The width of the tree decomposition $(T, B)$ is defined to be $\max\{|B_s| : s \in V(T)\} - 1$, 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 $w$.

\textbf{Theorem 13} (Bodlaender [8]). For each $w \in N$, 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 $\tau$ is a set of relation symbols. Each relation symbol $R$ has an arity, denoted $\text{arity}(R) \geq 1$. A structure $A$ of vocabulary $\tau$, or $\tau$-structure, consists of a set $A$, called the universe, and an interpretation $R^A \subseteq A^{\text{arity}(R)}$ of each relation symbol $R \in \tau$. We write $\bar{a} \in R^A$ or $R^A(\bar{a})$ to denote that the tuple $\bar{a} \in A^{\text{arity}(R)}$ belongs to the relation $R^A$.

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 $\tau$ are of the form:

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

where $R \in \tau$ is $r$-ary and $x_1, \ldots, x_r, x, y$ are variables. First-order formulas of vocabulary $\tau$ 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 $X(x_1 \ldots x_k)$, 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 $\phi$ is a variable $x$ with an occurrence in $\phi$ 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 $A$ satisfies a formula $\phi$ if there exists an assignment of the free variables under which $\phi$ becomes a true statement about $A$. In this case we will write $A \models \phi$.

### 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 $\tau$-structure $A$ is a pair $(T, B)$, where $T$ is a tree and $B$ a family of subsets of the universe $A$ of $A$ such that:

1. For all $a \in A$, the set $\{s \in V(T) : a \in B_s\}$ is nonempty and induces a connected subgraph (i.e. subtree) in $T$.
2. For every relation symbol $R \in \tau$ and every tuple $(a_1, \ldots, a_r) \in R^A$, there is a $s \in V(T)$ such that $a_1, \ldots, a_r \in B_s$.

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

We will make use of the version of Courcelle's celebrated theorem for relational structures of bounded treewidth, which, informally, says that the optimization 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 adaptation of an analogous theorem (see Theorem 9.21 in [18]) for the model-checking problem [17].

**Theorem 14 ([18])**. Let $\phi$ be an MSO formula with a free set variable $E$, and let $A$ be a relational structure on universe $A$, where $\text{tw}(A) \leq t$. Then, given a width-$t$ tree decomposition of $A$, a minimum-cardinality set $E \subseteq A$ such that $A$ satisfies $\phi(E)$ can be computed in time $f(t, \ell) \cdot ||A||$,

where $f$ is a computable function, $\ell$ is the length of $\phi$, and $||A||$ is the size of $A$.

### 5.2 The FPT algorithm

In this section we present an FPT algorithm for TR Edge Deletion when parameterised simultaneously by three parameters: $h$, $\text{tw}(G)$ and $\Delta_G$. Our strategy is first, given an input temporal graph $(G, \lambda)$, to construct a relational structure $A_{G, \lambda}$ whose treewidth is bounded in terms of $\text{tw}(G)$ and $\Delta_G$. Then we construct an MSO formula $\phi_h$ with a unique free set variable $E$, such that $A_{G, \lambda}$ satisfies $\phi_h(E)$ for some $E \subseteq A$ if and only if the maximum reachability of $(G, \lambda) \setminus E$ is at most $h$. Finally, we apply Theorem 14 to find the minimum cardinality of such a set $E \subseteq A$. 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 case we consider here 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 16 can immediately be adapted to this more general context if we replace $\Delta_G$ 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 \((G, \lambda)\), we define a relational structure \(A_{G, \lambda}\) 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 \((G, \lambda)\), 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 two binary relations \(R\) and \(L\) as follows:

1. \(((e_1, t_1), (e_2, t_2)) \in R\) 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. \(t_1 < t_2\).
2. \((e, (e, t)) \in L\) if and only if \((e, t) \in \Lambda(G, \lambda)\).

First we show that the treewidth of \(A_{G, \lambda}\) is bounded by a function of \(tw(G)\) and \(\Delta_G\).

\(\blacktriangleright\) Lemma 15. The treewidth of \(A_{G, \lambda}\) is at most \((2\Delta_G + 1)(tw(G) + 1) - 1\).

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

\(\blacktriangleright\) Theorem 16. TR Edge Deletion admits an FPT algorithm with respect to the combined parameters \(h\), \(tw(G)\), and \(\Delta_G\).

6 Conclusions and open problems

In this paper we studied the problem 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 at most \(h\) other vertices. The main motivation for this problem comes from the need to limit spreading processes on dynamic graphs. Such a graph could, for example, capture potentially-infectious contacts between individuals, and removing an edge would correspond to restricting or prohibiting contact between two entities in order to limit the spread of an epidemic.

We show that our problem is \(W[1]\)-hard when parameterised by the maximum number \(k\) of edges that can be removed and, assuming the Exponential Time Hypothesis, we cannot significantly improve on the brute-force algorithm that considers all possible deletions sets of \(k\) edges. On the positive side, we prove that this problems admits a fixed-parameter tractable (FPT) algorithm with respect to the combination of three parameters: the treewidth \(tw(G)\) of the underlying graph \(G\), the maximum allowed temporal reachability \(h\), and the maximum degree \(\Delta_G\) of \((G, \lambda)\). Moreover, we show that the latter two parameters combined (i.e., without the treewidth \(tw(G)\)) are not enough for deriving an FPT algorithm as the problem is para-NP-complete with respect to both of these parameters. On the other hand, it remains open whether this problem is FPT, when parameterised by treewidth \(tw(G)\), combined with only one of the other two parameters \(h\) and \(\Delta_G\). We also consider the approximability of this problem, and give two polynomial-time approximation algorithms.

The first computes an \(h\)-approximation on an arbitrary input graph, where \(h\) denotes the maximum allowable temporal reachability, and the second computes a \(c\)-approximation on graphs of cutwidth \(c\). We complement these positive results by showing that no constant-factor approximation algorithm exists for general input graphs unless \(P = NP\). A natural open problem is whether we can improve these approximation algorithms. Our lower bound rules out a \((\log \log h)\)-factor approximation, but a significant improvement on our factor \(h\) approximation may be possible.
References


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


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


