Temporal Separators with Deadlines

Hovhannes A. Harutyunyan
Department of Computer Science and Software Engineering,
Concordia University, Montreal, Canada

Kamran Koupayi
Department of Computer Science and Software Engineering,
Concordia University, Montreal, Canada

Denis Pankratov
Department of Computer Science and Software Engineering,
Concordia University, Montreal, Canada

Abstract

We study temporal analogues of the Unrestricted Vertex Separator problem from the static world. An (s, z)-temporal separator is a set of vertices whose removal disconnects vertex s from vertex z for every time step in a temporal graph. The (s, z)-Temporal Separator problem asks to find the minimum size of an (s, z)-temporal separator for the given temporal graph. The (s, z)-Temporal Separator problem is known to be $\mathcal{NP}$-hard in general, although some special cases (such as bounded treewidth) admit efficient algorithms [15].

We introduce a generalization of this problem called the (s, z, t)-Temporal Separator problem, where the goal is to find a smallest subset of vertices whose removal eliminates all temporal paths from s to z which take less than t time steps. Let τ denote the number of time steps over which the temporal graph is defined (we consider discrete time steps). We characterize the set of parameters τ and t when the problem is $\mathcal{NP}$-hard and when it is polynomial time solvable. Then we present a τ-approximation algorithm for the (s, z)-Temporal Separator problem and convert it to a τ^2-approximation algorithm for the (s, z, t)-Temporal Separator problem. We also present an inapproximability lower bound of $\Omega(\ln(n) + \ln(\tau))$ for the (s, z, t)-Temporal Separator problem assuming that $\mathcal{NP} \not\subset \textsc{Dtime}(n^{\log \log n})$. Then we consider three special families of graphs: (1) graphs of branchwidth at most 2, (2) graphs G such that the removal of s and z leaves a tree, and (3) graphs of bounded pathwidth. We present polynomial-time algorithms to find a minimum (s, z, t)-temporal separator for (1) and (2). As for (3), we show a polynomial-time reduction from the Discrete Segment Covering problem with bounded-length segments to the (s, z, t)-Temporal Separator problem where the temporal graph has bounded pathwidth.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases Temporal graphs, dynamic graphs, vertex separator, vertex cut, separating set, deadlines, inapproximability, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2023.38

Related Version *Full Version*: `https://arxiv.org/abs/2309.14185` [17]

Funding The research presented in this work is supported by NSERC.

1 Introduction

Suppose that you have been given the task of deciding how robust a train system of a given city is with respect to station closures. For instance, is it possible to disconnect the two most visited places, e.g., the downtown and the beach, by shutting down 5 train stations in the city? Does an efficient algorithm even exist? If not, what can we say about special classes of graphs? These are central questions of interest in this work.

More formally, we model the scenario as a graph problem. An important component missing from the classical graph theory is the ability of the graph to vary with time. The trains run on a schedule (or at least they are supposed to – for simplicity, we assume a perfectly punctual train system). Thus, it is not accurate to say that there is an edge between station A and station B just because there are tracks connecting them. It would be more accurate to say that if you arrive at A at some specific time t then you could get to B at some other time $t' > t$, where t is when the train arrives at station A and t' is the time when this train reaches station B. In other words, we can consider the edge from A to B as being present at a particular time (or times) and absent otherwise. This is an important point for the robustness of train networks, since it could be that due to incompatibility of certain train schedules the train network could become disconnected by shutting down even fewer stations than we otherwise would have thought if we didn't take time schedules into account.

The notion of graphs evolving with time has several formal models in the research literature [3, 23]. First of all, there is an area of online algorithms [1] where the graph is revealed piece by piece (thus the only allowable changes are to add objects or relations to the graph) and we need to make irrevocable decisions towards some optimization goal as the graph is being revealed. Secondly, streaming and semi-streaming graph algorithms deal with graphs that are revealed one piece at a time similar to online algorithms, but the emphasis is on memory-limited algorithms [12, 11]. Thus, in streaming one does not have to make irrevocable decisions, but instead tries to minimize the memory size necessary to answer some queries at the end of the stream. Thirdly, there is a notion of dynamic graph algorithms where the emphasis is on designing efficient data structures to support certain queries when the graph is updated by either adding or removing vertices or edges [24]. The goal is to maintain the data structures and answer queries, such as "are nodes u and v connected?", in the presence of changes more efficiently than recomputing the answer from scratch on every query. It is evident that none of these models is a good fit for our question: the train system is known in advance and it is not frequently updated (some cities that shall remain unnamed take decades to add a single station to the system). Fortunately, there is yet another model of graphs changing with time that has recently gotten a lot of attention and it happens to capture our situation perfectly. The model is called a temporal graph. In this work, we focus on undirected temporal graphs that have a fixed node set but whose edge sets change in discrete time units, all of which are known in advance. Other temporal graph models where changes to nodes are allowed and where time is modelled with the continuous real line have been considered in the research literature but they are outside of the scope of this work. We typically use τ to indicate the total number of time steps over which a given temporal graph is defined. For example, if we model the train system as a temporal graph with one minute-granularity and the schedule repeats every 24 hours then the temporal graph would have $\tau = (24H) \times (60M/H) = 1440M$ time steps in total. For emphasis, when we need to talk about non-temporal graphs and bring attention to their unchanging nature we shall call them "static graphs."

We study temporal analogues of the Unrestricted Vertex Separator problem from the static world. An (s, z)-temporal separator is a set of vertices whose removal disconnects vertex s from vertex z for every time step in a temporal graph. The (s, z)-Temporal Separator problem asks to find the minimum size of an (s, z)-temporal separator for the given temporal graph. The (s, z)-Temporal Separator problem is known to be $\mathcal{NP}$-hard in general [28], although some special cases (such as bounded treewidth) admit efficient algorithms [15]. This question can be thought of as a mathematical abstraction of the robustness of the train network of a city question posed at the beginning of this section. The (s, z)-Temporal

Separator problem asks you to eliminate all temporal paths between s and z by removing some nodes. Observe that, practically speaking, in real life, one doesn't actually have to eliminate all temporal paths between s and z – one would have to remove only reasonable temporal paths between s and z. Which paths would be considered unreasonable? We consider paths taking too much time as unreasonable. For example, if normally it takes 30 minutes to get from downtown to the beach, then eliminating all routes that take at most 4 hours would surely detract most downtown dwellers from visiting the beach. Motivated by such considerations, we introduce a generalization of the (s, z)-Temporal Separator problem called (s, z, t)-Temporal Separator problem, where the goal is to find the smallest subset of vertices whose removal eliminates all temporal paths from s to z which takes less than t time steps. Observe that setting $t = \tau$ captures the (s, z)-Temporal Separator problem as a special case of the (s, z, t)-Temporal Separator problem. Our results can be summarized as follows:

In Section 4.1, we present a characterization of parameters t and τ when the problem is $\mathcal{NP}$-hard. We also present an inapproximability lower bound of $\Omega(\ln(n) + \ln(\tau))$ for the (s, z, t)-Temporal Separator problem assuming that $\mathcal{NP} \not\subset \textsc{Dtime}(n^{\log\log n})$. In Section 4.2, we present a τ-approximation algorithm for the (s, z)-Temporal Separator problem, and we convert it to a τ^2-approximation algorithm for (s, z, t)-Temporal Separator problem.

In Section 5.1, we present a polynomial-time algorithm to find a minimum (s, z, t)-temporal separator on temporal graphs whose underlying graph (see Section 2) has branchwidth at most 2. In Section 5.2, we present another polynomial-time algorithm for temporal graphs whose underlying graph becomes a tree after removal of s and z. In Section 5.3, we show a polynomial-time reduction from the Discrete Segment Covering problem with bounded-length segments to the (s, z, t)-Temporal Separator problem where the temporal graph has bounded pathwidth. Therefore, solving the (s, z, t)-Temporal Separator problem on a temporal graph whose underlying graph has bounded pathwidth is at least as difficult as solving the Discrete Segment Covering problem where lengths of all segments are bounded.

2 Preliminaries

Temporal graphs (also known as dynamic, evolving [13], or time-varying [14, 6] graphs) are graphs whose edges are active at certain points in time. A temporal graph $G = (V, E, \tau)$ contains a set of vertices V, and a set of edges $E \subseteq V \times V \times [\tau]$ [1]. So each edge[2]. $e \in E$ contains two vertices of V and a time label $t \in [\tau]$ indicating a time step at which the edge is active. A graph $G_{\downarrow} = (V, E')$ where E' contains every edge e that is active at least once in the temporal graph G is called the *underlying graph* (alternatively, the *footprint*) of the temporal graph G. A static graph representing active edges for a specific time t is called the layer of the temporal graph at that time and is denoted by G_t. Some other ways of modelling temporal graphs could be found in [20]. We refer to $V(G)$ and $E(G)$ as the set of vertices and edges, respectively, of a graph G (either temporal or static). Also for any subset $U \subseteq V(G)$ we refer to the set of all edges in the subgraph induced by U as $E(U)$, and for any node $v \in V$ we use $E(v)$ to denote the set of all edges incident on v. We also use $\tau(G)$ to refer to the number τ of time labels of the temporal graph G.

[1] Notation $[n]$ stands for $\{1, 2, \ldots, n\}$.

[2] We only consider undirected graphs in this work, i.e. no self-loops and $(u, v, t) \in E$ if and only if $(v, u, t) \in E$

A temporal path in a temporal graph is a sequence of edges such that (1) it is a valid path in the underlying graph, and (2) the corresponding sequence of times when the edges are active is non-decreasing. Formally, a sequence $P = [(u_1, v_1, t_1), (u_2, v_2, t_2), \ldots, (u_k, v_k, t_k)]$ of edges in a temporal graph G is called an (s, z)-*temporal path* if $s = u_1, v_1 = u_2, \ldots, v_{k-1} = u_k, v_k = z$ and $t_1 \leq t_2 \leq \cdots \leq t_k$. If the sequence of times is in strictly increasing order, the temporal path is called *strict*. *Travelling time* of P, denoted by $\text{ttime}(P)$, is defined as $\text{ttime}(P) = t_k - t_1 + 1$, i.e., the time it takes to travel from s to z. If $\text{ttime}(P) \leq t$ then we refer to P as an (s, z, t)-temporal path. A temporal graph G is *connected* if for any pair of vertices $s, z \in V(G)$ there is at least one temporal path from s to z. A temporal graph G is *continuously connected* if for every $i \in [\tau(G)]$ layer G_i is connected.

We distinguish between three types of temporal paths: (1) *shortest* (s, z)*-temporal path*: a temporal path from s to z that minimizes the number of edges; (2) *fastest* (s, z)*-temporal path*: a temporal path from s to z that minimizes the traveling time; (3) *foremost* (s, z)*-temporal path*: a temporal path from s to z that minimizes the arrival time at destination. *Temporal distance* from node s to node z is equal to the traveling time of the fastest (s, z)-temporal path.

A set $S \subseteq V - \{s, z\}$ is called a *(strict)* (s, z)*-temporal separator* if the removal of vertices in set S removes all (strict) temporal paths from s to z. The *(strict)* (s, z)*-Temporal Separator problem* asks to find the minimum size of a (strict) (s, z)-temporal separator in a given temporal graph G. This problem has been studied before (see Section 3). In this work, we propose a new problem that is based on the notion of (s, z, t)-temporal paths. We define a set of vertices S to be a *(strict)* (s, z, t)*-temporal separator* if every (strict) (s, z, t)-temporal path contains at least one vertex in S, i.e., removal of S removes all (strict) (s, z, t)-temporal paths. Thus, the new problem, which we refer to as the *(strict)* (s, z, t)*-Temporal Separator problem* is defined as follows: given a temporal graph G, a pair of vertices $s, z \in V(G)$, and a positive integer t, the goal is to compute the minimum size of a (s, z, t)-temporal separator in G.

▶ **Lemma 1.** *Given a temporal graph $G = (V, E, \tau)$ and two distinct vertices s and z as well as an integer t, it is decidable in time $O(|S||E|)$ if there is a (s, z, t)-temporal path in G where $S = \{t' \mid \exists u : (s, u, t') \in E\}$.*

Proof. [25] and [27] present an algorithm that computes fastest paths from a single source s to all of the vertices in $O(|S|(|V| + |E|))$. We could ignore isolated vertices, then we could compute a fastest path from s to z in G and check if its travelling time is at least t. ◀

Branch decomposition and branchwidth of a graph is defined as follows.

▶ **Definition 2** (Branch Decomposition [8])**.** *Given a graph $G = (V, E)$, a branch decomposition is a pair (T, β), such that*

- *T is a binary tree with $|E|$ leaves, and every inner node of T has two children.*
- *β is a mapping from $V(T)$ to 2^E satisfying the following conditions:*
  - *For each leaf $v \in V(T)$, there exists $e \in E(G)$ with $\beta(v) = \{e\}$, and there are no $v, u \in V(T), v \neq u$ such that $\beta(v) = \beta(u)$.*
  - *For every inner node $v \in V(T)$ with children v_l, v_r, $\beta(v) = \beta(v_l) \cup \beta(v_r)$;*

▶ **Definition 3** (Boundary [8])**.** *Given a graph $G = (V, E)$, for every set $F \subseteq E$, the boundary $\partial F = \{v | v$ is incident to edges in F and $E \backslash F\}$.*

▶ **Definition 4** (Width of a Branch Decomposition [8])**.** *Given a branch decomposition (T, β) of $G = (V, E)$, the width of this decomposition is $max\{|\partial\beta(v)| \mid v \in V(T)\}$.*

The branchwidth $bw(G)$ of G is defined as the minimum width of a branch decomposition of G [8]. We note that for any fixed k there is a linear time algorithm to check if a graph has branchwidth k, and if so, the algorithm outputs a branch decomposition of minimum width [5].

Path decomposition and pathwidth of a graph are defined as follows.

▶ **Definition 5** (Path Decomposition [22]). *Given a graph $G = (V, E)$, a path decomposition of G is a pair (P, β), such that*

- *P is a path with nodes $a_1, \dots a_m$.*
- *β is a mapping from $\{a_1, \dots, a_m\}$ to 2^E satisfying the following conditions:*
  - *For $e \in E(G)$ there exists a_i such that vertices of e appear in $\beta(a_i)$.*
  - *For every $v \in V(G)$ the set of a_i, such that v appears in $\beta(a_i)$, forms a subpath of P.*

The width of a decomposition (P, β) is $\max_{a \in V(P)} |\beta(a)| - 1$. The pathwidth of a graph G is the minimum width of a path decomposition of G.

3 Related Work

Enright et al. in [9] adopt a simple and natural model for time-varying networks which is given with time-labels on the edges of a graph, while the vertex set remains unchanged. This formalism originates in the foundational work of Kempe et al. [18]. There has already been a lot of work on temporal graphs, too much to give a full overview. Thus, in this section, we focus only on the results most relevant to our work.

The fastest temporal path is computable in polynomial time, see, e.g. [27, 26, 25]. A nice property of the foremost temporal path is that it can be computed efficiently. In particular, there is an algorithm that, given a source node $s \in V$ and a time t_{start}, computes for all $w \in V \setminus \{s\}$ a foremost (s, w)-temporal path from the time t_{start} [19]. The running time of the algorithm is $O(n\tau^3 + |E|)$. It is worth mentioning that this algorithm takes as input the whole temporal graph G. Such algorithms are known as offline algorithms in contrast to online algorithms in which the temporal graph is revealed on the fly. The algorithm is essentially a temporal translation of the breadth-first search (BFS) algorithm (see e.g. [7] page 531).

While the Unrestricted Vertex Separator problem is polynomial time solvable in the static graph world (by reducing to the Maximum Flow problem), the analogous problem in the temporal graph world, namely, the (s, z)-Temporal Separator problem, was shown to be $\mathcal{NP}$-hard by Kempe et al. [18]. Zschoche et al. [28] investigate the (s, z)-Temporal Separator and strict (s, z)-Temporal Separator problems on different types of temporal graphs. A central contribution in [28] is to prove that both $(s, z)-$Temporal Separator and Strict (s, z)-Temporal Separator are $\mathcal{NP}$-hard for all $\tau \geq 2$ and $\tau \geq 5$, respectively, strengthening a result by Kempe et al. [18] (they show $\mathcal{NP}$-hardness of both variants for all $\tau \geq 12$) [28].

Fluschnik et al. [15] show that (s, z)-Temporal Separator remains $\mathcal{NP}$-hard on many restricted temporal graph classes: temporal graphs whose underlying graph falls into a class of graphs containing complete-but-one graphs (that is, complete graphs where exactly one edge is missing), or line graphs, or temporal graphs where each layer contains only one edge. In contrast, the problem is tractable if the underlying graph has bounded treewidth, or if we require each layer to be a unit interval graph and impose suitable restrictions on how the intervals may change over time, or if one layer contains all others (grounded), or if all layers are identical (1-periodic or 0-steady), or if the number of periods is at least the number of vertices. It is not difficult to show that this problem is fixed-parameter tractable when parameterized by $k + l$, where k is the solution size and l is the maximum length of a temporal (s, z)-path.

Lastly, we note that the classical Vertex Separator problem from the static world is often stated as asking to find a vertex separator such that after its removal the graph is partitioned into two blocks (one containing s and one containing z) of roughly equal size[3]. This "balanced" separator restriction makes the problem $\mathcal{NP}$-hard. The temporal separator problems considered in this work do not have such a restriction, and as discussed they are hard problems due to the temporal component. There is a lot of research on the Vertex Separator problem, but since our versions do not have this "balancedness" restriction, we do not discuss it in detail. An interested reader is referred to [2] and references therein.

4 Temporal Separators with Deadlines on General Graphs

4.1 Hardness of Exact and Approximate Solutions

Zschoche et al. [28] show that the (s, z)-Temporal Separator problem is $\mathcal{NP}$-hard on a temporal graph $G = (V, E, \tau)$ if $\tau \geq 2$ (and it is in $\mathcal{P}$ if $\tau = 1$). So, it is obvious that the (s, z, t)-Temporal Separator problem is $\mathcal{NP}$-hard if $t \geq 2$. In this section we strengthen this result by showing that the problem remains $\mathcal{NP}$-hard even when restricted to inputs with $t = 1$ and $\tau \geq 2$.

Reduction from the minimum satisfiability problem with non-negative variables to $(s, z, 1)$-Temporal Separator could be made by adding a path from s to z in layer G_i, which contains all the variables in the i-th clause. So, $(s, z, 1)$-Temporal Separator on temporal graphs with a sufficient number of layers is $\mathcal{NP}$-hard. However, it is not easy to establish the complexity of (s, z, t)-Temporal Separator on temporal graphs with a small number of layers. Here we aim to show that $(s, z, 1)$-Temporal Separator remains $\mathcal{NP}$-hard on a temporal graph $G = (V, E, \tau)$ if τ is equal to 2. To do that, we construct a reduction from the Node Multiway Cut problem. In this problem, one is given a graph $G = (V, E)$ and a set of terminal vertices $Z = \{z_1, z_2, \ldots z_k\}$. A multiway cut $S \in V \backslash Z$ is a set of vertices whose removal from G disconnects all pairs of distinct terminals z_i and z_j. The goal is to find a multiway cut of minimum cardinality. The Node Multiway Cut problem is $\mathcal{NP}$-hard for $k \geq 3$ [16]. For the proof of the next theorem, please see the full version of the paper [17].

▶ **Theorem 6.** *For every $t_0 \geq 1$, the (s, z, t)-Temporal Separator problem is $\mathcal{NP}$-hard on a temporal graph $G = (V, E, \tau)$ when restricted to inputs with $t = t_0$ and $\tau \geq 2$.*

Since Strict (s, z)-Temporal Separator is $\mathcal{NP}$-hard on a temporal graph with $\tau \geq 5$ [28], it is clear that Strict (s, z, t)-Temporal Separator is $\mathcal{NP}$-hard even when restricted to inputs with $t \geq 5$ and $\tau \geq 5$. However, by a small change to the reduction presented by Zschoche et al. [28], which is inspired by [26], we can show that Strict (s, z, t)-Temporal Separator remains $\mathcal{NP}$-hard even when restricted to inputs with $t = 3$ and $\tau = 4$. For the proof of the next theorem, please see the full version of the paper [17].

▶ **Theorem 7.** *Finding a strict $(s, z, 3)$-temporal separator on a temporal graph $G = (V, E, \tau)$ is $\mathcal{NP}$-hard when restricted to inputs with $\tau = 4$.*

Since every temporal path from s to z contains more than two edges, then $\emptyset$ is a strict $(s, z, 1)$-temporal separator. Since every strict $(s, z, 2)$-temporal path is of the form $(s, v, t), (v, z, t + 1)$, the Strict $(s, z, 2)$-Temporal Separator problem could be solved in

[3] That is why earlier we referred to a static world problem of interest as the Unrestricted Vertex Separator problem to emphasize that there is no balancedness requirement.

polynomial time easily. The Strict (s, z, t)-Temporal Separator problem on a graph $G = (V, E, \tau)$ with $\tau = t$ is the same as the Strict (s, z)-Temporal Separator. Therefore, in case $\tau = t = 3$ this problem is equivalent to the Strict (s, z)-Temporal Separator problem with $\tau = 3$. Zschoche et al. [28] present a polynomial time algorithm for finding a minimum strict (s, z)-temporal separator on a temporal graph $G = (V, E, \tau)$ when $\tau < 5$. So, this case could be solved in polynomial time. Although we know that finding a strict (s, z, t)-temporal separator on a temporal graph $G = (V, E, 3)$ is polynomial-time solvable with the algorithm which is presented in [28], we describe another simple algorithm to solve this problem.

In the first step of the algorithm, we check if there is an edge between s and t. If so, it is clear that there are no separator sets because the direct path using this edge from s to z will remain with the removal of any node from the graph.

Next, for every temporal path from s to z of length two, such as $(s, x, t_1), (x, z\ , t_2)$ with $t_2 = t_1 + 1$, it is clear that we have to remove x if we want to remove this path from the graph. So, it is clear that $x \in S$.

In the last step, we know that the length of every temporal path in the graph is three. So, every path from s to z should be of the following form:

$$(s, x, 1), (x, y, 2), (y, z, 3).$$

Now, put every node x with existing edge (s, x) into the set X with time label 1. Also, put every node y that is a neighbor of z into the set Y with time label 3. Now, it is clear that $X \cap Y = \emptyset$, for otherwise there exists a node u with two existing edges $e_1 = (s, u, 1)$ and $e_2 = (u, z, 3)$, while this node should be removed in the previous step. Therefore, every strict temporal path from s to z should have a corresponding edge $(x, y, 2)$ where $x \in X$ and $y \in Y$. So, we should remove either x or y for every edge $(x, y, 2)$, where $x \in X$ and $y \in Y$. In order to do this we could use any known polynomial time algorithm for the Vertex Cover problem in bipartite graphs.

In the rest of this section we show $\Omega(\log n + \log(\tau))$-inapproximability (assuming $\mathcal{NP} \subset \text{DTIME}(n^{\log \log n})$) for the (s, z, t)-Temporal Separator problem. This is proved by a strict reduction from the Set Cover problem. Recall that in the Set Cover problem, one is given a collection $\mathcal{S}$ of subsets of a universe U that jointly cover the universe. The goal is to find a minimum size sub-collection of $\mathcal{S}$ that covers U.

▶ **Theorem 8.** *For every $t > 0$ there is a strict polynomial time reduction from the Set Cover problem to the $(s, z, t)-$Temporal Separator problem.*

Proof. Let $(U, \mathcal{S})$ be an instance of the Set Cover problem, where $U = \{1, 2, \dots n\}$ is the universe and $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ is a family of sets the union of which covers U. For each $i \in U$ define the family $\mathcal{F}_i$ as $\mathcal{F}_i = \{S \in \mathcal{S} \mid i \in S\}$, i.e., $\mathcal{F}_i$ consists of all sets from $\mathcal{S}$ that contain element i. Let $k_i = |\mathcal{F}_i|$ and order the elements of each $\mathcal{F}_i$ in the order of increasing indices, i.e., $\mathcal{F}_i = \{S_{i_1}, \dots, S_{i_{k_i}}\}$.

Our reduction outputs a temporal graph $f(U, \mathcal{S}) = (V \cup \{s, z\}, E)$ where:

- the vertex set is $V \cup \{s, z\} = \{v_i | i \in [m]\} \cup \{s, z\}$;
- the edge set is the union over i of the sets $E_i = \{(s, v_{i_1}, i{\cdot}t), (v_{i_1}, v_{i_2}, i{\cdot}t), \dots, (v_{i_{k_i-1}}, v_{i_{k_i}}, i{\cdot}t), (v_{i_{k_i}}, z, i \cdot t)\}$.

The main idea behind the proof is to map every element of U to a path from s to z in $f(U, \mathcal{S})$ bijectively, so by covering an element, we remove the corresponding path in $f(U, \mathcal{S})$ as well as by removing a path we cover the corresponding element.

We claim that $V' = \{v_{j_1}, \dots, v_{j_\ell}\} \subseteq V$ is a $(s, z, t)-$temporal separator for $f(U, \mathcal{S})$ if and only if $\mathcal{S}' = \{S_{j_1}, \dots, S_{j_\ell}\} \subseteq \mathcal{S}$ is a set cover for $(U, \mathcal{S})$.

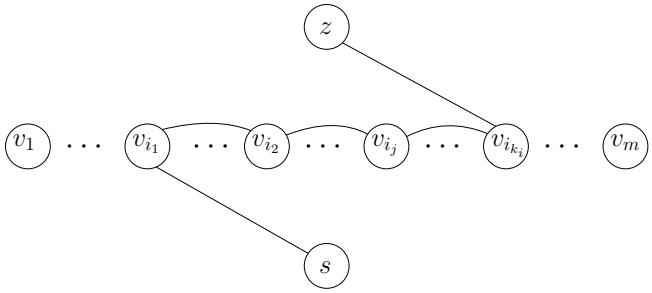

Figure 1 Layer $G_{i\cdot t}$ of the temporal graph used in the proof of Theorem 8.

Figure 1 represents the edges in the layer G_{it}, which contain all the edges in E_i. It illustrates that element i in the universe U corresponds to a path E_i, as well as the element i is covered by the set $S_{i_j} \in \mathcal{S}'$ if and only if a temporal path which is shown in Figure 1 is removed from the temporal graph by removing the vertex $v_{i_j} \in V'$.

$\rightarrow$ Suppose for contradiction that $\mathcal{S}'$ does not cover U. Pick an arbitrary item $i \in U$ that is not covered and consider the following path $P = [(s, v_{i_1}, i \cdot t), (v_{i_1}, v_{i_2}, i \cdot t), \ldots, (v_{i_{k_i-1}}, v_{i_{k_i}}, i \cdot t), (v_{i_{k_i}}, z, i \cdot t)]$, where the indices are according to the equation for $\mathcal{F}_i$. Since i is not covered, $\mathcal{F}_i \cap \mathcal{S}' = \emptyset$, so P is present in $f(U, \mathcal{S}) \setminus V'$ violating the assumption that V' is a $(s, z, t)-$temporal separator (note that $\text{ttime}(P) = 0$).

$\leftarrow$ Now, suppose for contradiction that V' is not a (s, z, t)-temporal separator. Thus, there is path P from s to z with $\text{ttime}(P) < t$. From the definition of $f(U, \mathcal{S})$ it is clear that P should be using edges only from E_j for some $j \in [n]$. Note that there is a unique (s, z)-temporal path that can be constructed from E_j, namely, $P = [(s, v_{j_1}, j \cdot t), (v_{j_1}, v_{j_2}, j \cdot t), \ldots, (v_{j_{k_j-1}}, v_{j_{k_j}}, j \cdot t), (v_{j_{k_j}}, z, j \cdot t)]$. This implies that element j is not covered by $\mathcal{S}'$, since otherwise, one of the v_{j_i} would be in V'.

Following the previous claim, every solution in (s, z, t)-Temporal Separator has a corresponding solution in Set Cover, and vice versa. Therefore, an optimal solution in (s, z, t)-Temporal Separator, has a corresponding optimal solution in Set Cover. As a result $\frac{|V'|}{|V_{opt}|} = \frac{|S'|}{|S_{opt}|}$. This implies that the reduction is strict. ◀

Due to the inapproximability of Set Cover (see [10]), we have the following:

▶ **Corollary 9.** *The (s, z, t)-Temporal Separator problem is not approximable to within $(1 - \epsilon)(\log n + \log(\tau))$ in polynomial time for any $\varepsilon > 0$, unless $\mathcal{NP} \subset \text{DTIME}(n^{\log \log n})$.*

4.2 Approximation Algorithms

In this section, we present an efficient τ^2-approximation for the (s, z, t)-Temporal Separator problem. We begin by establishing a τ-approximation for the (s, z)-Temporal Separator problem. The main tool used in this section is the *flattening*[4] of a temporal graph $G = (V, E, \tau)$ with respect to vertices s and z, denoted by $F(G, s, z) = (V', E')$. To ease the notation we omit the specification of s and z and denote the flattening of G by $F(G)$. The flattening $F(G)$ is a static directed graph defined as follows: the vertex set V' is the union of τ disjoint sets $V_1, V_2, \ldots, V_\tau$ and $\{s, z\}$, where each V_i is a disjoint copy of $V - \{s, z\}$. Denoting the

[4] The concept of flattening is not new, and it is similar to the static expansion of a temporal graph – see, for example, [19].

vertices of V by $v_1, v_2, \ldots, v_n$, we have $\forall i \in [\tau]\ \ V_i = \{v_{j,i} | v_j \in V - \{s, z\}\}$. The edge set E' of the flattening $F(G)$ is defined as follows:

- For each $(v_i, v_j, t') \in E$ with $v_i, v_j \notin \{s, z\}$ we add edges $(v_{i,t'}, v_{j,t'})$ and $(v_{j,t'}, v_{i,t'})$ to E'.
- For each $v_i \in V$ and each time $t' \in [\tau - 1]$ we add an edge $(v_{i,t'}, v_{i,t'+1})$ to E'.
- For each $(s, v_i, t') \in E$ we add an edge $(s, v_{i,t'})$ to E'.
- For each (z, v_i, t') we add an edge $(v_{i,t'}, z)$ to E'.

Clearly, $F(G)$ is defined to express temporal (s, z)-paths in G in terms of (s, z)-paths in $F(G)$. More specifically, if we have a temporal (s, z) path P in G then there is an analogous static (s, z) path P' in $F(G)$. If P begins with an edge (s, v_i, t_1) then P' begins with an edge (s, v_{i,t_1}). After that if the next edge in P is (v_i, v_j, t_2), we can simulate it in $F(G)$ by introducing a sequence of edges $(v_{i,t_1}, v_{i,t_1+1}), (v_{i,t_1+1}, v_{i,t_1+2}), \ldots, (v_{i,t_2-1}, v_{i,t_2})$ followed by an edge (v_{i,t_2}, v_{j,t_2}), and so on until the vertex z is reached. This correspondence works in reverse as well. If P' is a static (s, z) path in $F(G)$ then we can find an equivalent temporal (s, z) path in G as follows. If the first edge in P' is (s, v_{i,t_1}) then this corresponds to the first edge of P being (s, v_i, t_1). For the following edges of P', if the edge is of the form $(v_{i,t'}, v_{i,t'+1})$ then it is simply ignored for the purpose of constructing P (since it corresponds to the scenario where the agent travelling along the path is simply waiting an extra time unit at node v_i), and if the edge is of the form $(v_{i,t'}, v_{j,t'})$ then we add the edge (v_i, v_j, t') to P. This continues until z is reached. Thus, there is a temporal (s, z) path P in G if and only if there is a static (s, z) path P' in $F(G)$. Moreover, if S represents the internal nodes of the path P then we can find P' with internal nodes in $\bigcup_{t' \in [\tau]} \{v_{i,t'} : v_i \in S\}$. In the reverse direction, if P' uses internal nodes S' then we can find P with internal nodes in $\{v_i : \exists t'\ \ v_{i,t'} \in S'\}$.

Armed with these observations, we show that the sizes of (s, z)-temporal separators in G and (s, z)-separators (non-temporal) in $F(G)$ are related as follows.

▶ **Theorem 10.**

1. *If S is an (s, z)-temporal separator in G then there is an (s, z)-separator of size at most $\tau|S|$ in $F(G)$.*
2. *If S' is an (s, z)-separator in $F(G)$ then there is an (s, z)-temporal separator of size at most $|S'|$ in G.*

The proof of the above theorem, albeit rather simple, appears in the full version of the paper [17].

▶ **Corollary 11.** *The (s, z)-Temporal Separator problem on a temporal graph $G = (V, E, \tau)$ can be approximated within τ in $O((m + n\tau)n\tau)$ time, where $n = |V|$ and $m = |E|$.*

Proof. We can use any existing efficient algorithm to solve the (s, z) separator problem on $F(G)$ and return its answer, which will give τ-approximation by Theorem 10. For example, the stated runtime is achieved by applying Menger's theorem and the Ford-Fulkerson algorithm to compute the maximum number of vertex-disjoint paths in $F(G)$. Then the running time is $O(|E'||V'|)$. Observing that $|E'| \leq |E| + |V|\tau$ and $|V'| \leq |V|\tau$, finishes the proof of this corollary. ◀

Next, we describe how the (s, z, t)-Temporal Separator problem can be approximated using a slight extension of the above ideas. First, for a temporal graph $G = (V, E, \tau)$ and two integers $t_1 \leq t_2$ we define $E[t_1 : t_2] = \{(u, v, t) \in E : t_1 \leq t' \leq t_2\}$. We also define $G[t_1 : t_2] = (V, E[t_1 : t_2], t_2)$, which can be thought of as graph G restricted to time interval $[t_1, t_2]$. The idea behind approximating a minimum (s, z, t)-temporal separator is to combine (s, z)-temporal separators of $F(G[1 : t + 1]), F(G[2 : t + 2]), \ldots, F(G[\tau - t : \tau])$.

▶ **Theorem 12.** *The (s, z, t)-Temporal Separator problem on a temporal graph $G = (V, E, \tau)$ can be approximated within τ^2 in $O((m + n\tau)n\tau^2)$ time, where $n = |V|$ and $m = |E|$.*

Proof. The algorithm has essentially been described prior to the statement of the theorem, so the running time is clear. It is left to argue that it produces τ^2-approximation. This can be argued similarly to Theorem 10.

1. Let S be a (s, z, t)-temporal separator in G. Then for $G[i : i + t]$ we define S_i to consist of all nodes $v_{j,t'}$ with $v_j \in S$. Since S removes all paths from G of travelling time $\leq t$ and $G[i : i + t]$ only has paths of travelling time $\leq t$, then S_i is a (s, z)-separator in $G[i : i + t]$ of size $|S_i| = \tau|S|$. Thus, if there is an (s, z, t)-temporal separator of size $|S|$ in G then the combined size of all (s, z, t)-temporal separators of $G[1 : t + 1], G[1 : t + 2], \ldots, G[\tau - t, \tau]$ is at most $\tau^2|S|$.
2. Let S_i be a (s, z)-temporal separator in $G[i : i + t]$. Define $S = \{v_j : \exists i \exists t' \ \ v_{j,t'} \in S_i\}$. It is easy to see that S is a (s, z, t) temporal separator in G. Paths of travelling time at most t that begin with an edge (s, v_i, t_1) are present in $G[t_1, t_1 + t]$, and so removal of S_{t_1} removes such temporal paths in $G[t_1, t_1 + t]$. Since S_{t_1} is "projected" onto V and included in S, these paths are eliminated from G. ◀

5 Temporal Separators with Deadlines on Special Families of Graphs

5.1 Temporal Graphs with Branchwidth at most 2

The graphs with branchwidth 2 are graphs in which each biconnected component is a series-parallel graph [21]. In this section, we present an efficient algorithm to solve the (s, z, t)-Temporal Separator problem on temporal graphs whose underlying static graphs have branchwidth at most 2. In fact, our algorithm works for a more general class of problems, which we refer to as "restricted path (s, z)-Temporal Separator." The goal in this more general problem is to select a set of vertices S such that the removal of S from the given temporal graph G removes all (s, z) paths in a restricted family of paths. The (s, z, t)-Temporal Separator problem is seen as a special case of this, where paths are restricted to have travelling time less than t. Restricted family of paths could be any path family implicitly defined by a procedure $ExistsRestrictedPath(G, s, z)$ which takes as input a temporal graph G, a pair of nodes s and z, and returns true if and only if there exists a restricted temporal path between s and z in G. Due to Lemma 1, we know that such a procedure exists in the case of temporal paths restricted by travelling time, which is suitable for the (s, z, t)-Temporal Separator problem.

For the rest of this section, we assume that G is a temporal graph such that $bw(G_\downarrow) \leq 2$ unless stated otherwise. Furthermore, we assume that $G_\downarrow$ is connected, otherwise, if s and z belong to different connected components the answer to the problem is trivially $\emptyset$, and if they belong to the same connected component, the problem reduces to analyzing that connected component alone. We introduce some notation and make several observations about branch decomposition before we give full details of our algorithm. Recall from Section 2 that branch decomposition of G of width 2 can be computed in linear time. Thus, we assume that the algorithm has access to such a decomposition, which we denote by (T, β). We use ρ to denote the root of T and we define the function $top : V(G) \to V(T)$ as follows. For $v \in V(G)$ we let $top(v)$ be the furthest node $x \in V(T)$ from the root r which satisfies $E(v) \subseteq \beta(x)$. We also use x_l to denote the left child of x and x_r to denote the right child of x. For a node $x \in V(T)$ we define G_x^{in} to be the temporal graph obtained from G by keeping only those edges (u, v, t) with $(u, v) \in \beta(x)$ and removing all vertices of degree 0. We collect several useful observations about the introduced notions in the following lemma.

▶ **Lemma 13.**

1. *If* $v \in \partial\beta(x)$ *then* $v \in \partial\beta(x_\ell)$ *or* $v \in \partial\beta(x_r)$.
2. *If* $top(v) = x$ *then* $v \in \partial\beta(x_\ell)$ *and* $v \in \partial\beta(x_r)$.
3. *If* $v \in V(G_x^{in}) \setminus \partial\beta(x)$ *then all edges incident on* v *in* G *are present in* G_x^{in}.

Proof.

1. Since $v \in \partial\beta(x)$ it means that some but not all edges incident on v in G appear in $\beta(x)$. Since $\beta(x) = \beta(x_\ell) \cup \beta(x_r)$, it implies that some but not all edges incident on v must appear either in $\beta(x_\ell)$, or $\beta(x_r)$, or both.
2. If $top(v) = x$ then $E(v) \subseteq \beta(x)$. Suppose for contradiction that $v \notin \partial\beta(x_\ell)$. This can happen for two reasons: either (1) $E(v) \subseteq \beta(x_\ell)$, or (2) $E(v) \cap \beta(x_\ell) = \emptyset$. In case (1) we obtain a contradiction with the definition of $top(v)$ since x_ℓ is further from the root than x and it still contains all of $E(v)$. In case (2) observe that we must have $E(v) \subseteq \beta(x_r)$, thus obtaining a contradiction with the definition of $top(v)$ again since x_r is further from the root than x and it still contains all of $E(v)$.
3. Since $v \in V(G_x^{in}) \setminus \partial\beta(x)$ it means that there is at least one edge incident on v in $V(G_x^{in})$. Since v is not in the boundary of $\beta(x)$, it means that all edges incident on v in G must be present in $\beta(x)$. ◀

Algorithm 1 This algorithm finds a restricted $(s, z)-$temporal separator in a temporal graph G with $bw(G_\downarrow) \leq 2$.

```
Function RTS(G, s, z):
    if ExistsRestrictedPath(G, s, z) = false then
        return ∅;
    for v ∈ V(G) \ {s, z} do
        if ExistsRestrictedPath(G \ {v}, s, z) = false then
            return {v};
    if top(s) = top(z) then
        return RTS(G^in_{ρ_ℓ}, s, z) ∪ RTS(G^in_{ρ_r}, s, z);
    else if top(s), top(z) are not ancestors of each other then
        return ∂β(top(z));
    else
        /* assume top(z) is ancestor of top(s), otherwise swap s and z   */
        if z ∉ ∂β(top(s)) then
            return ∂β(top(s));
        else if ∂β(top(s)) = {z} then
            return RTS(G^in_{top(s)}, s, z);
        else
            /* ∂β(top(s)) = {z, q}, ∂β(top(s)_ℓ) = {s, z}, ∂β(top(s)_r) = {s, q}   */
            S ← RTS(G^in_{top(s)_ℓ}, s, z);
            if ExistsRestrictedPath(G \ S, s, z) then
                return S ∪ {q};
            else
                return S;
```

Now, we are ready to describe our algorithm, which is denoted by RTS. The algorithm starts by checking if there is a restricted temporal path from s to z in G, and if such a path does not exist then the algorithm immediately returns $\emptyset$. Then the algorithm checks if

there exists a restricted temporal separator of size 1 by testing whether there is a restricted temporal path in $G \setminus \{v\}$ for each $v \in V(G) \setminus \{s, z\}$. Then the algorithm computes $top(s)$ and $top(z)$ and the computation splits into three cases: (1) if $top(s) = top(z)$; (2) if $top(s)$ and $top(z)$ are not on the same root-to-leaf path in T (i.e., neither one is an ancestor of another); and (3) if one of $top(s), top(z)$ is an ancestor of another. We shall later see that case (1) implies that $top(s) = top(z) = \rho$. In this case, the algorithm invokes itself recursively on the two subtrees of T – the subtree rooted at the left child of ρ and the subtree rooted at the right child of ρ. The separators obtained on these two subtrees correspond to separators of $G^{in}_{\rho_\ell}$ and $G^{in}_{\rho_r}$ and their union is returned as the separator for G. In case (2) the algorithm returns the boundary of $\beta(top(z))$ (it could return the boundary of $\beta(top(s))$ instead – it does not make a difference) as the answer. In case (3), we assume without loss of generality that $top(z)$ is the ancestor of $top(s)$, and handling of this case depends on whether z belongs to the boundary of $\beta(top(s))$ or not. In fact, this case splits into three subcases: (3.1) if $z \notin \partial\beta(top(s))$ then the algorithm immediately returns $\partial\beta(top(s))$; (3.2) if $\partial\beta(top(s)) = \{z\}$ then the algorithm invokes itself recursively on $G^{in}_{top(s)}$; and (3.3) if $\partial\beta(top(s)) = \{z, q\}$ for some vertex $q \neq s, z$ then the algorithm first invokes itself recursively on $G^{in}_{top(s)_\ell}$ (assuming $\partial\beta(top(s)_\ell) = \{s, z\}$) and stores the answer in S. If S proves to be a separator in G then S is returned, otherwise, q is added to S and returned. The pseudocode is presented in Algorithm 1.

▶ **Theorem 14.** *Algorithm 1 correctly computes a minimum-sized restricted path (s, z)-temporal separator for a temporal graph G such that $bw(G_\downarrow) \leq 2$.*

Proof. The proof proceeds by the case analysis reflecting the structure of the algorithm. Clearly, the algorithm correctly identifies when there is a separator of size 0 or 1 since it performs brute-force checks for these special cases. Assuming that there is no separator of size ≤ 1, we discuss the correctness for the remaining three cases.

Case (1): $top(s) = top(z) = x \in V(T)$. Observe that Lemma 13, item 1, implies that $s, z \in \partial\beta(x_\ell)$ and $s, z \in \partial\beta(x_r)$. Since the branchwidth is 2, it implies that $\partial\beta(x_\ell) = \partial\beta(x_r) = \{s, z\}$. In addition, we know that $s, z \notin \partial\beta(x)$ by the definition of $top()$. And since every vertex in $\partial\beta(x)$ must appear in $\partial\beta(x_\ell)$ or $\partial\beta(x_r)$ (using Lemma 13, item 2), we conclude that $\partial\beta(x) = \emptyset$. By Lemma 13, item 3, every vertex in G^{in}_x has all its edges from G. Therefore G^{in}_x is disconnected from the rest of G. However, we assume that G is connected, so we must have $G^{in}_x = G$. This is true only when $x = \rho$. Thus, we must have in this case that $top(s) = top(z) = \rho$. Observe that if P is a restricted temporal path between s and z (that does not have s or z as intermediate nodes) then it cannot use edges from both $\beta(\rho_\ell)$ and $\beta(\rho_r)$. Suppose, for contradiction, that P uses both kinds of edges, then there must be a vertex v on this path incident on e_1 and e_2 such that $e_1 \in \beta(x_\ell)$ and $e_2 \in \beta(x_r)$. Since $\beta(x_\ell), \beta(x_r)$ partition all the edges, it implies that $e_2 \notin \beta(x_\ell)$. This means that $v \in \partial\beta(x_\ell) = \{s, z\}$, but $v \neq s, z$, giving a contradiction. Therefore, the minimum size restricted path temporal separator in G is the union of minimum size restricted path temporal separators in $G^{in}_{\rho_\ell}$ and $G^{in}_{\rho_r}$, which is precisely what our algorithm outputs.

Case (2): $top(s)$ and $top(z)$ do not lie on the same root-to-leaf path in T. One of the consequences of Lemma 13, item 3, is that removing $\partial\beta(x)$ from G separates all vertices in $V(G^{in}_x)$ from the rest of the graph. Therefore, removing $\partial\beta(top(z))$ separates all vertices in $G^{in}_{top(z)}$ from the rest of the graph. Observe that $z \in V(G^{in}_{top(z)})$ and $s \notin V(G^{in}_{top(z)})$ (by the condition of this case). Therefore removing $\partial\beta(top(z))$ separates s from z. We claim that this is the minimum separator in this case. This is because when this line is

reached we are guaranteed that there is no separator of size 1, and $|\partial\beta(top(z))| \leq 2$ (in fact, it must be then equal to 2). We only need to be careful that neither z nor s is in $\partial\beta(top(z))$, but it is clear from the definition of $top()$ and the case condition.

Case (3): $top(z)$ is an ancestor of $top(s)$ (if $top(s)$ is an ancestor of $top(z)$ then we can exchange the roles of s and z for the sake of the argument). This case has three subcases.

Subcase (3.1): $z \notin \partial\beta(top(s))$. This is similar to case (2) described above. The algorithm can return $\partial\beta(top(s))$ as a minimum size separator.

Subcase (3.2): $\partial\beta(top(s)) = \{z\}$. In this case, the structure of the graph is such that $G^{in}_{top(s)}$ is connected to the rest of the vertices in G via the node z, while vertex s lies in $G^{in}_{top(s)}$. Thus, to separate z from s, it is sufficient to separate them in $G^{in}_{top(s)}$, which is what the algorithm does.

Subcase (3.3): $\partial\beta(top(s)) = \{z, q\}$. By Lemma 13, item 2, it follows that $s \in \partial\beta(top(s)_\ell)$ and $s \in \partial\beta(top(s)_r)$. By Lemma 13, item 1, it follows that $z, q \in \beta(top(s)_\ell) \cup \beta(top(s)_r)$. Since branchwidth is at most 2, we have (without loss of generality) that $\partial\beta(top(s)_\ell) = \{s, z\}$ and $\partial\beta(top(s)_r) = \{s, q\}$. By an argument similar to the one in case (1), we can establish that any restricted (s, z) temporal path (that does not use s or z as intermediate nodes) must either consist entirely of edges in $\beta(top(s)_\ell)$ or entirely of edges in $\beta(top(s)_r)$. Thus, we can compute the two separators and take their union; however, we can simplify the calculation observing that the only separator we need to consider for the $G^{in}_{top(s)_r}$ is $\{q\}$, since $G^{in}_{top(s)_r}$ is connected to the rest of G only through q and s. ◀

▶ **Corollary 15.** *Given a temporal graph $G = (V, E, \tau)$ with $bw(G_\downarrow) \leq 2$, the problem (s, z, t)-Temporal Separator is solvable in time $O(|V||E||\mathcal{T}|)$ where $\mathcal{T} = \{t(e) : e \in E(s)\}$.*

5.2 Temporal Graphs with a "Tree-like" Underlying Graph

In this section, we present a polynomial time greedy algorithm (motivated by the point-cover interval problem) for computing a path restricted (s, z)-temporal separator (see Section 5.1) on a temporal graph G such that $G_\downarrow \setminus \{s, z\}$ is a tree if the existence of a restricted (s, z)-temporal path could be checked in polynomial time.

We assume that we are given a temporal graph G such that $G_\downarrow \setminus \{s, z\}$ is a tree, which we denote by T. For a pair of nodes (u, w), we let $P_{u,w}$ denote the unique shortest path in T between u and w. For a vertex $v \in V(T)$, we define a removal list of v, denoted by RL_v, to consist of all unordered pairs (u, w) such that $v \in V(P_{u,w})$ and there exists a restricted (s, z)-temporal path in G using the edges of $P_{u,w}$. For a pair $u, w \in V(T)$, we define two temporal graphs: (1) $G^1_{u,w}$ is G induced on the edges of $E(P_{u,w}) \cup \{(s, u), (v, z)\}$, and (2) $G^2_{u,w}$ is G induced on the edges of $E(P_{u,w}) \cup \{(s, v), (u, z)\}$. The removal lists for all vertices in $V(T)$ can be computed efficiently as follows. Initialize all removal lists to be empty. For each pair of vertices $u, w \in V(T)$ check if there is any restricted (s, z)-temporal path in $G^1_{u,w}$ or $G^2_{u,w}$, and if so, then add (u, w) to the removal lists of all nodes in $P_{u,w}$. Let $\mathcal{U} = \bigcup_{v \in V(T)} RL_v$ be the set of all pairs of nodes that appear in removal lists. The following observation is immediate from the definitions and shows that computing a minimum size restricted path (s, z)-temporal separator reduces to covering $\mathcal{U}$ with as few removal lists as possible.

▶ **Observation 16.** *A set of S is a restricted path (s, z)-temporal separator if and only if $\bigcup_{v \in S} RL_v = \mathcal{U}$.*

A vertex v is called topmost if there exists a pair $(u, w) \in RL_v$ such that $(u, w) \notin RL_{parent(v)}$. Our greedy algorithm, called $GreedyRTS$, starts out with an empty solution $S = \emptyset$, and then adds more vertices to S as follows. While there are non-empty removal lists,

the algorithm selects a topmost vertex v with maximum distance from the root of T, adds v to the set S, and removes all pairs in RL_v from the removing lists of all the other vertices. The pseudocode is given in Algorithm 2 (in Appendix B).

For the proof of the next theorem, please see the full version of the paper [17].

▶ **Theorem 17.** *Algorithm 2 computes a minimum-sized restricted path (s, z)-temporal separator in a temporal graph G with $G_{\downarrow} \setminus \{s, z\}$ being a tree.*

Based on Lemma 1, the existence of a (s, z, t)-temporal path can be solved in polynomial time. Thus, the following theorem follows from Theorem 17.

▶ **Theorem 18.** *The (s, z, t)-Temporal Separator problem is solvable in polynomial time on temporal graphs G where $G_{\downarrow} \setminus \{s, z\}$ is a tree.*

5.3 Temporal Graphs with Bounded Pathwidth

In this section, we present a reduction from the *Discrete Segment Covering (DISC-SC)* problem to the (s, z, t)-Temporal Separator problem on graphs with bounded pathwidth. In the DISC-SC problem, we are given a set Γ of n intervals (also called segments), on the rational line and a set $\mathcal{I}$ of unit-intervals on the rational line. We wish to find a subset of unit intervals $A \subseteq \mathcal{I}$ which covers all the segments in Γ. The objective is to minimize the size of A. An interval $I \in \mathcal{I}$ covers a segment $S \in \Gamma$ if at least one endpoint S lies in I. A segment $S \in \Gamma$ is covered by a set of intervals A if there is an interval $I \in A$ that covers S. We refer to the version of DISC-SC where all segments in Γ have length bounded by k as DISC-SC-k. DISC-SC problem is $\mathcal{NP}$-hard [4]. [4] also shows that the DISC-SC problem remains $\mathcal{NP}$-hard when the length of all segments in Γ are equal. DISC-SC-1 can be solved efficiently by a simple greedy algorithm [4]. However, the hardness of DISC-SC-k for general $k > 1$ remains open.

The following theorem serves as a warm-up, and it establishes a simple polynomial time reduction from DISC-SC to the (s, z, t)-Temporal Separator problem. For the proof of the next theorem, please see the full version of the paper [17].

▶ **Theorem 19.** *There is a polynomial-time reduction from the DISC-SC problem to the (s, z, t)-Temporal Separator problem.*

The issue with the above theorem is that it does not provide any structural guarantees about the temporal graph G used in the construction. In order to establish a reduction via a temporal graph G whose underlying graph has bounded pathwidth, we start with a restricted version of DISC-SC, namely, the DISC-SC-k problem. The following results can then be established.

▶ **Theorem 20.** *There is a polynomial-time reduction from the DISC-SC-k problem to the (s, z, t)-Temporal Separator in which the pathwidth of the underlying graph is bounded by $2k + 6$.*

Proof. Consider an instance $(\mathcal{I}, \Gamma)$ of the Discrete Segment Covering problem such that the length of all the segments in Γ is at most k. Consider intervals in $\mathcal{I} = (I_1, I_2, \dots\ I_n)$ in the non-decreasing order of their starting times. We choose a special set of intervals $SP \in \mathcal{I}$ by the following algorithm.

1. Let $SP = I_1$ and $index = 1$.
2. Let j be the largest index such that $s(I_j) < e(I_{index})$, if such j exists. Otherwise, let $j = index + 1$
3. Put I_j into the set SP, update the integer $index$ equal to j and if $j \leq n$ repeat the algorithm from step 2.

For the proof of the next lemma, please see the full version of the paper [17].

▶ **Lemma 21.** *A p is covered by $\mathcal{I}$ if SP covers it.*

The main idea of the proof is based on to the following features of the special set SP. Denote $SP = \{I_{m_1}, I_{m_2}, \dots I_{m_q}\}$. Based on the selection of interval $I_{m_{i+1}}$ it is clear that the starting point of $I_{m_{i+2}}$ is greater than the ending point of I_{m_i} which implies that $s(I_{m_{i+2}}) > s(I_{m_i})+1$. More generally, we have that $e(I_{m_{i+2k}}) > s(I_{m_i}) + k + 1$. Therefore, for any segment $C \in \Gamma$ and for any interval I_{m_i} and I_{m_j} such that $s(C) \in I_{m_i}$ and $e(C) \in I_{m_j}$, we could conclude that $j \leq i + 2k$. This feature for SP is the main idea used in constructing an instance of the (s, z, t)-Temporal Separator problem with low pathwidth.

Now we construct a temporal graph $G = (V, E, \tau)$ where $\tau = |\Gamma| \times t$. Let $V = \{u_i | i \in [n]\} \cup \{v_i | i \in [n]\} \cup \{s, z\}$. Now, for the i-th segment $C \in \Gamma$ we add a path from s to z at time $i \times t$. Let m_a and m_b be the indices of the first intervals in SP which cover points $s(C)$ and $e(C)$, respectively. Based on the Lemma 21 if m_a (or m_b) does not exist, then the point $s(C)$ (respectively, $e(C)$) will not be covered by any interval in $\mathcal{I}$. Therefore, we could treat C as a single point $e(C)$ (respectively, $s(C)$) and continue on with the algorithm. Let l_s be the index of the leftmost interval form $\mathcal{I}$ which covers $s(C)$, and let r_s be the index of the rightmost interval from $\mathcal{I}$ which covers $s(C)$. It is obvious that $s(C)$ is covered by all of the intervals between l_s and r_s in $\mathcal{I}$. Similarly, let l_e and r_e be the indices of the leftmost and the rightmost intervals which cover $e(C)$. If $l_e \leq r_s$ then consider $l_e = r_s + 1$ instead. Now, add the following (s, z, t)-temporal path to the temporal graph G. For simplicity, we denote $i \times t$ by θ.

$$
\begin{aligned}
&(s, u_{l_s}, \theta), (u_{l_s}, v_{l_s}, \theta), (v_{l_s}, v_{l_s+1}, \theta), \dots (v_{r_s-1}, v_{r_s}, \theta)\\
&(v_{r_s}, u_{r_s}, \theta), (u_{r_s}, u_{r_s-1}, \theta), \dots (u_{m_a+1}, u_{m_a}, \theta)\\
&(u_{m_a}, u_{m_b}, \theta)\\
&(u_{m_b}, u_{m_b-1}, \theta), \dots, (u_{l_e+1}, u_{l_e}, \theta), (u_{l_e}, v_{l_e}, \theta)\\
&(v_{l_e}, v_{l_e} + 1, \theta) \dots (v_{r_e-1}, v_{r_e}, \theta), (v_{r_e}, u_{r_e}, \theta), (u_{r_e}, z, \theta)
\end{aligned}
\tag{1}
$$

Figure 2 (in Appendix A) shows the above path in the graph layer $i \times t$. We claim that there exists $A \subseteq \mathcal{I}$ that covers Γ with $|A| \leq p$ if and only if there is a (s, z, t)-temporal separator $S \subseteq V$ such that $|S| \leq p$.

$\rightarrow$ Suppose that $A \subseteq \mathcal{I}$ covers all segments in Γ. Let $S = \{v_i | I_i \in A\}$. It is obvious that $|S| = |A|$. Now we prove that S is a (s, z, t)-temporal separator. Suppose that there is a temporal path P in G, based on the construction of G this temporal path should be of the form shown in equation 1 for some $i \in [n]$. This implies $I_j \notin A$ for all j such that $l_s < j < r_s$ or $l_e < j < r_e$ and results in the i-th segment not being covered by A. So, based on the contradiction we could conclude that S is a (s, z, t)-temporal separator.

$\leftarrow$ Suppose that $S \subseteq V$ is a (s, z, t)-temporal separator in a temporal graph G. Let $A = \{I_i | u_i \in S \text{ or } v_i \in S\}$, it is clear that $|A| \leq |S|$. Consider the i-th segment $C \in \Gamma$. There should be one vertex belonging to the temporal path P which is shown in equation 1 in S since S is a (s, z, t)-temporal separator. Therefore there is j where $l_s < j < r_s$ or $l_e < j < r_e$ and either u_i or v_i belong to S, which implies $C \in A$. Thus, A covers Γ.

Now we prove that the pathwidth of the underlying graph $G_\downarrow = (V, E')$ of the temporal graph $G(V, E, |\Gamma| \times t)$ is bounded by $2k + 6$. We refer to an edge $(u_{m_a}, u_{m_b}, \theta)$ in a path that is shown in equation 1 as a *crossing edge*. Figure 3 (in Appendix A) shows a graph G' of which $G_\downarrow$ is a subgraph. Now we give a path decomposition (P, β) for a graph $G_\downarrow$ in which the width of decomposition is at most $2k + 6$. Let $V(P) = \{a_1, a_2, \dots, a_m\}$ and

$E(P) = \{(a_1, a_2), \ldots, (a_{m-1}, a(m))\}$. Let $i \in [n]$ and $l(i)$ be the largest integer such that the starting point of the interval $I_{m_{l(i)}} \in SP$ is before the starting point of interval I_i. Now we define the $\beta(a_i)$ as follows: $\beta(a_i) = \{u_i, v_i, u_{i+1}, v_{i+1}, s, z\} \cup \{u_{m_l} | l \geq l(i) \text{ and } l \leq l(i) + 2k\}$.

▶ **Lemma 22.** *For any u_q and i, j, l such that $i < j < l$, if $u_q \in \beta(a_i)$ and $u_q \in \beta(a_l)$, we have $u_q \in \beta(a_j)$.*

Proof. If $I_q \notin SP$ then it is clear that u_q only appears in $\beta(a_{q-1})$ and $\beta(a_q)$. Now suppose that $I_1 \in SP$ and $q = m_p$. Since $u_{m_p} \in \beta(a_i)$ we have $m_p \leq l(i) + 2k$, also $l(l) \leq m_p$ since $m_p \in \beta(a_l)$. As a result we have $m_p \leq l(i) + 2k \leq l(j) + 2k$ and $l(j) \leq l(l) \leq m_p$ which implies that $u_q \in \beta(a_j)$. ◀

For any $v_i \in V$ it is clear that v_i just belongs to the two sets $\beta(a_{i-1})$ and $\beta(a_i)$. Also, s and z are present in all the sets. Therefore, by Lemma 1 we could say that the third property of path decomposition is satisfied. So, it is sufficient to show that for every edge $(u, v) \in E(G_\downarrow)$ there exists $i \in [n]$ such that $\{u, v\} \subseteq \beta(a_i)$. If the edges are not crossing edges, then there are three types of edges (u_i, v_i), (u_i, u_{i+1}), and (v_i, v_{i+1}) which satisfy the condition by the definition of $\beta(a_i)$. If $e = (u_i, u_j)$ is a crossing edge, then $I_i \in SP$ and $I_j \in SP$, so let $m_p = i$ and $m_q = j$. Since this edge corresponds to a segment C such that $s(C) \in I_{m_p}$ and $e(C) \in I_{m_q}$ we could conclude that $m_q \leq m_p + 2k$ which implies that $u_i, u_j \subseteq \beta(a_i)$. Also, the cardinality of all sets $\beta(a_i)$ is $2k + 7$ which implies that the width of (P, β) is $2k + 6$. Therefore the pathwidth of the underlying graph $G_\downarrow$ is at most $2k + 6$. ◀

▶ **Theorem 23.** *If the (s, z, t)-Temporal Separator problem on temporal graphs with bounded pathwidth is solvable in polynomial time then the DISC-SC-k problem is solvable in polynomial time.*

6 Conclusions

In this work, we defined the (s, z, t)-Temporal Separator problem, generalizing the (s, z)-Temporal Separator problem. We showed that (s, z)-Temporal Separator and (s, z, t)-Temporal Separator problems could be approximated within τ and τ^2 approximation ratio, respectively, in a graph with lifetime τ. We also presented a lower bound $\Omega(\log(n) + \log(\tau))$ for polynomial time approximability of (s, z, t)-Temporal Separator assuming that $\mathcal{NP} \not\subseteq \text{DTIME}(n^{\log \log n})$. Then we considered special classes of graphs. We presented two efficient algorithms: one for temporal graphs G with $bw(G_\downarrow) \leq 2$ and one for temporal graphs G with $G_\downarrow \setminus \{s, z\}$ being a tree. The question of whether there is a polynomial-time algorithm to compute a minimum (s, z, t)-temporal separator in a temporal graph of bounded treewidth remains an interesting open problem. However, we showed a reduction from the DISC-SC-k problem to (s, z, t)-Temporal Separator when the pathwidth of the underlying graph is bounded by a constant number. Therefore, designing efficient algorithms for bounded treewidth graphs encounters serious obstacles, such as making progress on the open problem of the hardness of DISC-SC-k. Another interesting direction of future research is to consider temporal separator problems with the additional restriction of "balancedness", as discussed at the end of Section 3.

References

1 Susanne Albers. Online algorithms: a survey. *Mathematical Programming*, 97(1-2):3–26, 2003.

2 Haeder Y. Althoby, Mohamed Didi Biha, and André Sesboüé. Exact and heuristic methods for the vertex separator problem. *Computers and Industrial Engineering*, 139:106135, 2020. doi:10.1016/j.cie.2019.106135.

3 Aris Anagnostopoulos, Ravi Kumar, Mohammad Mahdian, Eli Upfal, and Fabio Vandin. Algorithms on evolving graphs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 149–160, 2012.

4 Dan Bergren, Eduard Eiben, Robert Ganian, and Iyad Kanj. On covering segments with unit intervals. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

5 Hans L. Bodlaender and Dimitrios M. Thilikos. Constructive linear time algorithms for branchwidth. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming*, pages 627–637, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

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

7 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

8 Xiaojie Deng, Bingkai Lin, and Chihao Zhang. Multi-multiway cut problem on graphs of bounded branch width. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, pages 315–324. Springer, 2013.

9 Jessica Enright, Kitty Meeks, George B Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *arXiv preprint*, 2018. arXiv:1805.06836.

10 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

11 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *International Colloquium on Automata, Languages, and Programming*, pages 531–543. Springer, 2004.

12 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Departmental Papers (CIS)*, page 236, 2005.

13 Afonso Ferreira. Building a reference combinatorial model for manets. *IEEE network*, 18(5):24–29, 2004.

14 Paola Flocchini, Bernard Mans, and Nicola Santoro. Exploration of periodically varying graphs. In *International Symposium on Algorithms and Computation*, pages 534–543. Springer, 2009.

15 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020.

16 Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. Multiway cuts in directed and node weighted graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 487–498. Springer, 1994.

17 Hovhannes A. Harutyunyan, Kamran Koupayi, and Denis Pankratov. Temporal separators with deadlines, 2023. arXiv:2309.14185.

18 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.

19 George B Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Temporal network optimization subject to connectivity constraints. In *International Colloquium on Automata, Languages, and Programming*, pages 657–668. Springer, 2013.

20 Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

21 Neil Robertson and Paul D Seymour. Graph minors. x. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

22 Neil Robertson and P.D. Seymour. Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

23 Ryan A Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 667–676, 2013.

24 Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of computer and system sciences*, 26(3):362–391, 1983.

25 Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. Path problems in temporal graphs. *Proceedings of the VLDB Endowment*, 7(9):721–732, 2014.

26 Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016.

27 B Bui Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.

28 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.

A Figures

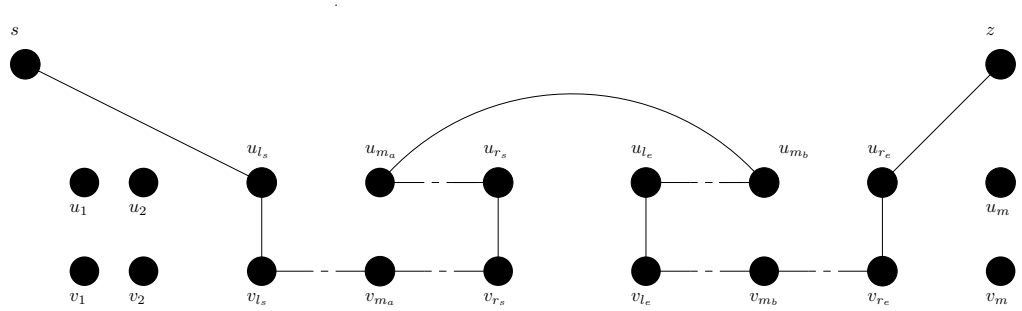

Figure 2 Demonstration of one step of the reduction in the proof of Theorem 20. The figure shows the (s, z, t)-temporal path in the layer $G_{j\times t}$. The time label for all edges is $j \times t$.

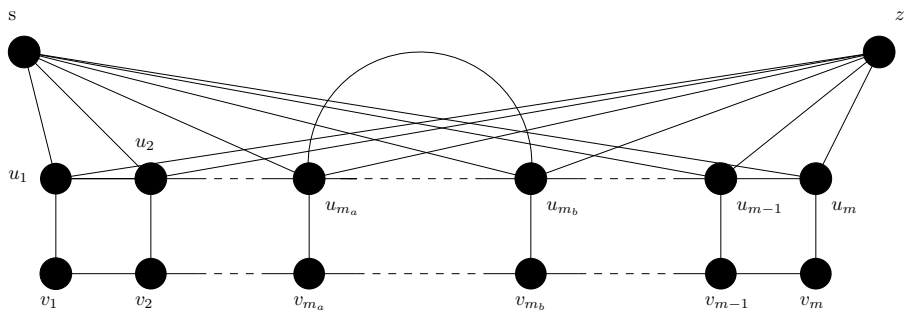

Figure 3 Illustration of the graph G' which is used to show that the output of the reduction from Theorem 20 has bounded pathwidth. The underlying graph $G_\downarrow$ is a subgraph of G'.

B Pseudocode

Algorithm 2 This algorithm computes a minimum sized restricted path (s, z)-temporal separator in a temporal graph G when $G_{\downarrow} \setminus \{s, z\}$ is a tree T.

Function `ComputeRLs`(G, s, z)**:**
  $\mathcal{U} \leftarrow \emptyset$;
  for $(u, w) \in V(T) \times V(T)$ **do**
    if `ExistsRestrictedPath`$(G^1_{u,w}, s, z)$ *or* `ExistsRestrictedPath`$(G^2_{u,w}, s, z)$ **then**
      $\mathcal{U} \leftarrow \mathcal{U} \cup \{(u, w)\}$;
      for $v \in V(P_{u,w})$ **do**
        $RL_v \leftarrow RL_v \cup \{(u, w)\}$;
Function `GreedyRTS`$(G, s, z, RL, \mathcal{U})$**:**
  $S \leftarrow \emptyset$;
  while $\mathcal{U} \neq \emptyset$ **do**
    $v \leftarrow$ furthest node from the root of T such that $\exists (u, w) \in RL_v \setminus RL_{parent(v)}$;
    $S \leftarrow S \cup \{v\}$;
    $\mathcal{U} \leftarrow \mathcal{U} \setminus RL_v$;
    for $w \in V(T)$ **do**
      $RL_w \leftarrow RL_w \setminus RL_v$;
  return S;