Novel Complexity Results for Temporal Separators with Deadlines

Riccardo Dondi
Università degli Studi di Bergamo, Italy

Manuel Lafond
Université de Sherbrooke, Canada

Abstract

We consider two variants, (s,z,ℓ)-Temporal Separator and (s,z,ℓ)-Temporal Cut, respectively, of the vertex separator and the edge cut problem in temporal graphs. The goal is to remove the minimum number of vertices (temporal edges, respectively) in order to delete all the temporal paths that have time travel at most ℓ between a source vertex s and target vertex z. First, we solve an open problem in the literature showing that (s,z,ℓ)-Temporal Separator is NP-hard even when the underlying graph has pathwidth bounded by four. We complement this result showing that (s,z,ℓ)-Temporal Separator can be solved in polynomial time for graphs of pathwidth bounded by three. Then we consider the approximability of (s,z,ℓ)-Temporal Separator and we show that it cannot be approximated within factor $2^{\Omega(\log^{1-\varepsilon}|V|)}$ for any constant $\varepsilon > 0$, unless $NP \subseteq ZPP$ (V is the vertex set of the input temporal graph) and that the strict version is approximable within factor $\ell - 1$ (we show also that it is unliklely that this factor can be improved). Then we consider the (s,z,ℓ)-Temporal Cut problem, we show that it is APX-hard and we present a $2\log_2(2\ell)$ approximation algorithm.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph theory

Keywords and phrases Temporal Graphs, Graph Algorithms, Graph Separators, Parameterized Complexity, Approximation Complexity

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.23

1 Introduction

A central problem in checking network robustness is finding the minimum number of vertices or edges that need to be removed in order to disconnect the network. In classic (static) graphs this is modeled by computing a minimum cut or a minimum vertex separator between a source vertex s and a target vertex z. The static graph model however does not consider how a network may change over time. In transportation networks, for example, the time schedule is a fundamental aspect that has to be taken into account for analyzing several properties, like connectedness and robustness. The need to incorporate time information of edge availability has led to the introduction of the *temporal graph* model [12, 9, 16, 10], where edges are assigned timestamps in a discrete set that define when each edge is available (thus when a specific transport is available in a transportation network).

The robustness problems (minimum cut and minimum vertex separator) have been considered also for temporal graphs, where the paths to be removed have to be *temporal*, that is they have to satisfy a time constraint. Given a source s and a target z, the (s,z)-Temporal Cut problem asks for the minimum number of temporal edges that have to be removed so that s and z are disconnected, while (s,z)-Temporal Separator asks for the minimum number of vertices that have to be removed so that s and z are disconnected. (s,z)-Temporal Cut is known to be in solvable in polynomial time [2], (s,z)-Temporal Separator is known NP-hard [12, 17], and its fixed-parameter tractability and approximability (and variants thereof) have been studied [7, 15, 8, 14, 11].

A variant of the (s,z)-Temporal Separator problem that has been considered in [8] to model the robustness of transportation system, defines s and z as separated if the time needed to move from s to z is above a time threshold. The motivation is that if the time to move from s to z is increased considerably, then this makes the possibility of moving from s to z unlikely. This is modeled in [8] by defining the (s,z,ℓ)-Temporal Separator problem, that asks for a smallest subset of vertices whose removal deletes each temporal path that takes at most time ℓ between s and z. Several results have been given for (s,z,ℓ)-Temporal Separator in [8]. The (s,z,ℓ)-Temporal Separator problem is NP-hard when $\ell = 1$ and the temporal graph is defined over (at least) two timestamps. On the other hand, the problem is solvable in polynomial time when the underlying graph is a tree (after deleting s and z), or it has branchwidth at most two. As for the approximation complexity, it is shown to be not approximable within factor $\Omega(\ln|V| + \ln\tau)$ assuming that $NP \not\subset DTIME(|V|^{\log\log|V|})$ (V is the set of vertices of the temporal graph, τ the number of timestamps); moreover, a τ^2-approximation algorithm is presented (also a τ-approximation algorithm for (s,z)-Temporal Separator). Finally, it is shown that solving (s,z,ℓ)-Temporal Separator when the underlying graph has bounded pathwidth is at least as difficult as solving a problem called Discrete Segment Covering, whose complexity is unsolved [1]. In Section 3, we settle the status of (s,z,ℓ)-Temporal Separator on graphs of bounded pathwidth: we show that the decision version of problem is NP-hard even with $\ell = 1$ and when the underlying graph has pathwidth at most 4, and we give a polynomial-time algorithm when the pathwidth is at most 3. Then in Section 4, we show that (s,z,ℓ)-Temporal Separator cannot be approximated within factor $2^{\Omega(\log^{1-\varepsilon}|V|)}$ for any constant $\varepsilon > 0$, unless $NP \subseteq ZPP$ and we present an $\ell - 1$-approximation algorithm for the strict variant of (s,z,ℓ)-Temporal Separator[1]. We show also that improving this factor is a challenging problem, since the strict variant of (s,z,ℓ)-Temporal Separator is hard to approximate as Vertex Cover k-Uniform Hypergraph, where $k = \ell - 1$. In Section 5, we consider the (s,z,ℓ)-Temporal Cut problem and we show that it is APX-hard, which contrasts with the polynomial-time solvability of the problem when deadlines are not considered, and we present a $2\log_2(2\ell)$-approximation algorithm. In Section 2 we give some definitions and we formally define the two problems we are interested into. Some of the proofs are omitted due to space constraint.

2 Preliminaries

For an integer n, we use the notation $[n] = \{1, 2, \ldots, n\}$. A *temporal graph* $G = (V, E, \tau)$ is defined over a set V of vertices and a set $E \subseteq V \times V \times [\tau]$ of temporal edges, where $\tau \in \mathbb{N}$. An *undirected* edge in a temporal graph is then a triple (u, v, t), where $u, v \in V$ and $t \in [\tau]$ is a timestamp[2]. Note that we denote by uv an edge in an undirected (static) graph and (u, v) an arc in a directed (static) graph.

We say that u, v are *neighbors* in G if they share some temporal edge, and denote the set of neighbors of u by $N_G(u)$ (we drop the subscript if it is clear from the context). Given a set $V' \subseteq V$, we denote by $G[V']$ the subgraph induced by V', which contains vertex set V' and every temporal edge whose two endpoints are in V'. We also denote by $G - V' = G[V \setminus V']$ the temporal graph obtained by removing each vertex in V'. Given a set $E' \subseteq E$, we denote by $G - E'$ the temporal graph obtained by removing each temporal edge in E'.

[1] In the strict variant a temporal path must uses temporal edges with increasing timestamps

[2] As in [8] we assume that (u, v, t) is a temporal edge of G if and only if (v, u, t) is a temporal edge of G.

An interval $[t_1, t_2]$, with $t_1, t_2 \in [\tau]$ and $t_1 \leq t_2$, is the sequence of consecutive timestamps between t_1 and t_2, including t_1 and t_2. Given interval $[t_1, t_2]$, $G([t_1, t_2])$ is the temporal subgraph of G that has temporal edges having timestamps in $[t_1, t_2]$.

A *temporal path* P in a temporal graph G is a sequence of temporal edges such that

$$P = [(u_1, v_1, t_1), (u_2, v_2, t_2), \ldots, (u_h, v_h, t_h)],$$

with $v_i = u_{i+1}$ and $t_i \leq t_{i+1}$, for each $i \in [h-1]$, and $u_i \neq u_j$, $v_i \neq v_j$, for each $i, j \in [h]$ with $i \neq j$; P is called an (u_1, v_1)-temporal path, since it starts from vertex u_1 and ends in vertex v_h. If $t_i < t_{i+1}$, with $i \in [h-1]$, then the temporal path is *strict*. Given two vertices s and z Given a temporal path $P = [(u_1, v_1, t_1), (u_2, v_2, t_2), \ldots, (u_h, v_h, t_h)]$, the *travelling time* $tt(P)$ of P is defined as $tt(P) = t_h - t_1 + 1$. The set of vertices of a temporal path P is denoted by $V(P)$. We may refer to a temporal path of travelling time *at most* ℓ as an ℓ-temporal path. Two temporal paths are *temporal edge disjoint* if they don't share any temporal edge.

Given a temporal graph $G = (V, E, \tau)$, we assume that there are two special vertices $s, z \in V$, which are the source and the target vertex of G. A set $V' \subseteq (V \setminus \{s, z\})$ is an (s, z)-temporal separator ((s, z)-strict temporal separator, respectively) in G if there is no temporal path (strict temporal path, respectively) in $G[V \setminus V']$ between s and z. Given $\ell \in [\tau]$, a set $V' \subseteq (V \setminus \{s, z\})$ is an (s, z, ℓ)-temporal separator ((s, z, ℓ)-strict temporal separator, respectively) in G if there is no temporal path (strict temporal path, respectively) between s and z in $G[V \setminus V']$ of travelling time at most ℓ.

A set $E' \subseteq E$ of temporal edges is an (s, z)-temporal cut ((s, z)-strict temporal cut, respectively) in G if there is no temporal path (strict temporal path, respectively) in $G - E'$ between s and z. Given $\ell \in [\tau]$ a set $E' \subseteq E$ is an (s, z, ℓ)-temporal cut ((s, z, ℓ)-strict temporal cut, respectively) in G if there is no temporal path (strict temporal path, respectively) in $G - E'$ between s and z of travelling time at most ℓ.

Now, we are ready to define the combinatorial problems we are interested in. The first, introduced in [8], is the following.

▶ **Problem 1** ((s,z,ℓ)-Temporal Separator).
Input: *a temporal graph $G = (V, E, \tau)$, two vertices $s, z \in V$, a positive integer $\ell \in [\tau]$.*
Output: *An (s,z,ℓ)-temporal separator $V' \subseteq (V \setminus \{s, z\})$ in G of minimum size.*

We denote by (s,z,ℓ)-Strict Temporal Separator the variant of (s,z,ℓ)-Temporal Separator where we look for an (s,z,ℓ)-strict temporal separator.

We consider now the second problem we are interested into.

▶ **Problem 2** ((s,z,ℓ)-Temporal Cut).
Input: *a temporal graph $G = (V, E, \tau)$, two vertices $s, z \in V$, a positive integer $\ell \in [\tau]$.*
Output: *An (s,z,ℓ)-temporal cut $E' \subseteq E$ in G of minimum size.*

Note that if $\ell = \tau$, then (s,z,ℓ)-Temporal Separator ((s,z,ℓ)-Temporal Cut, respectively) is exactly the (s,z)-Temporal Separator problem (the (s,z)-Temporal Cut) problem, respectively) where we look for an (s, z)- temporal separator ((s, z)-temporal cut, respectively) in G.

3 Graphs of bounded pathwidth

Let us first recall the notion of pathwidth. A *nice path decomposition* of a graph G is a sequence of set of vertices $(X_1, \ldots, X_q)$, with each $X_i \subseteq V(G)$ referred to as a *bag*, such that all of the following holds:

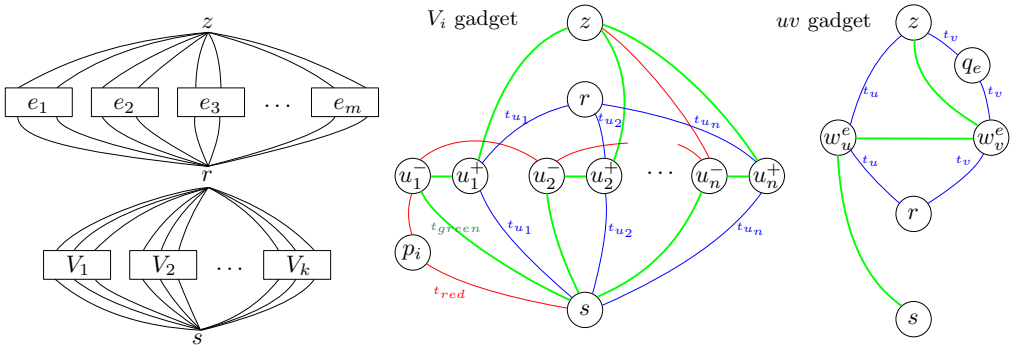

Figure 1 Top left: main structure of the reduction: there is a vertex-chossing phase leading to paths to r, followed by an edge vertification phase (note, not all edges are shown). Middle: a V_i gadget, with t_{green} edges in green, t_{red} edges in red, and t_u edges in blue. Right: an edge gadget for $e = uv$, with t_{green} edges in green and t_u, t_v edges in blue.

- $X_1 = X_q = \emptyset$;
- for $i \in \{2, \ldots, q\}$, either $X_i = X_{i-1} \cup \{v\}$, with $v \notin X_{i-1}$, in which case X_i is called an *introduce bag*; or $X_i = X_{i-1} \setminus \{v\}$, with $v \in X_{i-1}$, in which case X_i is a *forget bag*.
- for every pair of vertices u and v that share an edge in G, some bag X_i contains both u and v (regardless of the time of the edge).
- for every vertex $v \in V(G)$, there are $i, j \in [q-1]$ such that the set of bags that contain v is precisely $X_i, X_{i+1}, \ldots, X_j$.

The *width* of the nice path decomposition is $\max_{i \in [q-1]}(|X_i| - 1)$. The *pathwidth* of G is the minimum width of a nice path decomposition of G.

We show that (s,z,ℓ)-Temporal Separator is NP-hard even on graphs of pathwidth 4 and when $\ell = 1$. Our reduction is from the Multicolored Independent Set problem, where we are given a (static) graph G and a partition $\{V_1, \ldots, V_k\}$ of $V(G)$ into k sets. The V_i sets are called *color classes*. The question is whether there is an independent set I of G such that $|I \cap V_i| = 1$ for each $i \in [k]$, that is, we must choose one vertex per color class to form an independent set. Note that this problem is typically used to prove W[1]-hardness, but it is also NP-hard [6][3].

Let G be an instance of Multicolored Independent Set, with vertex partition $V_1, \ldots, V_k$. We assume, without loss of generality, that $|V_i| = n$ for every $i \in [k]$.

Let us construct from G an instance of the (s,z,ℓ)-Temporal Separator problem consisting of temporal graph H, vertices s and z to separate, and $\ell = 1$. In the construction, we assign to temporal edges of H a time in the set $T = \{t_{green}, t_{red}\} \cup \{t_u : u \in V(G)\}$, with the understanding that all elements of T correspond to a distinct integer. Since $\ell = 1$, one can view the temporal edges as being colored by an element of T and the problem as having to destroy all monochromatic paths – so the subscripts of the elements of T can be seen as colors. So from G, first add to $V(H)$ the vertices s, z, and a new vertex r. We then have vertex-choosing gadgets and edge-verification gadgets (see Figure 1 for an illustration).

[3] Note that the problem considered in [6], Multicolored Clique, is equivalent to Multicolored Independent Set if we consider complementary relations (edges and no edges) between vertices of different color classes of the input graph.

Vertex-choosing gadgets. For each $i \in [k]$, build a gadget for V_i as follows. For every $u \in V_i$, add to H two vertices u^- and u^+. Then add a temporal path at time t_{green} from s to z formed by the temporal edges $(s, u^-, t_{green}), (u^-, u^+, t_{green}), (u^+, z, t_{green})$. Also add a temporal path at time t_u from s to r formed by the temporal edges $(s, u^+, t_u), (u^+, r, t_u)$.

To complete the gadget, add a new vertex p_i, then add temporal edges so that there is a path with a time of t_{red} from s to z that first goes through p_i, then through all the u^- vertices exactly once. More precisely, denote $V_i = \{u_1, \ldots, u_n\}$, where the ordering is arbitrary. Then, add the temporal edges $(s, p_i, t_{red}), (p_i, u_1^-, t_{red})$ and (u_n^-, z, t_{red}), and for each $h \in [n-1]$, add the temporal edge $(u_h^-, u_{h+1}^-, t_{red})$.

Edge verification gadgets. Next, we construct gadgets to verify that the chosen vertices correspond to an independent set of G. For each edge $e = uv$ of G, where $u \in V_i$ and $v \in V_j$ such that $i < j$, add to H two vertices w_u^e and w_v^e. Add a temporal path at time t_{green} from s to z formed by the temporal edges $(s, w_u^e, t_{green}), (w_u^e, w_v^e, t_{green})$, and (w_v^e, z, t_{green}), enforcing the deletion of at least one of the two vertices. Next, add a temporal path at time t_u formed by the temporal edges $(r, w_u^e, t_u), (w_u^e, z, t_u)$. Finally, add a new vertex q_e and a temporal path at time t_v going from r to z formed by the temporal edges $(r, w_v^e, t_v), (w_v^e, q_e, t_v), (q_e, z, t_v)$.

▶ **Lemma 1.** *The graph H constructed from G as described above has pathwidth at most* 4.

The idea is that we can get a path decomposition by putting s, z, r in every bag. Then, the V_i and the uv gadgets are easy to construct using two extra vertices per bag. We next show that NP-hardness holds.

▶ **Lemma 2.** *The graph G has a multicolored independent set if and only if H has an $(s, z, 1)$-temporal separator of size $|V(G)| + |E(G)|$.*

Proof sketch. Let $I = \{u_1, \ldots, u_k\}$ be a multicolored independent set of G, with each $u_i \in V_i$. In H, in the V_i gadget, delete every u^+ vertex, *except* u_i^+, and delete u_i^- instead. This removes all the t_{green} and the t_{red} temporal paths, but s can reach r with a temporal path at time t_{u_i}. Then for each edge $e = uv$ of G, in the uv gadget, delete w_u^e if s reaches r with time t_u, and delete w_v^e otherwise. This removes the t_{green} temporal path, and since we delete at least one of u^+ or v^+, there remains no temporal path at time t_u or t_v going through the gadget.

Conversely, suppose there is an $(s, z, 1)$-temporal separator in H of size $|V(G)| + |E(G)|$. The t_{green} temporal paths enforce deleting, for each $u \in V(G)$, one of u^- or u^+, and also for each $e = uv \in E(G)$ one of w_u^e or w_v^e. There is no room for other deletions. Because of the t_{red} temporal path in the V_i gadget, some u^- is deleted and some u^+ is kept. Also, we cannot keep u^+ and v^+ from different V_i, V_j gadgets if $uv \in E(G)$, as otherwise there will be a t_u or t_v temporal path going through the uv gadget. Hence, the kept u^+ vertices correspond to a multicolored independent set. ◀

▶ **Theorem 3.** *The (s,z,ℓ)-Temporal Separator problem is NP-hard, even with $\ell = 1$ and on temporal graphs of pathwidth at most* 4*, and even if each edge is present in only one timestamp.*

We mention that it should be possible to modify the proof to prove a similar hardness for the *strict* variant of the problem, for larger values of ℓ, as it suffices to replace each label t_{green}, t_{red}, t_u with time intervals that are far enough from each other. This modification requires a large value of ℓ as the red path in our construction traverses $n + 1$ vertices, in addition to s and z. We leave the details for a future version, and the complexity of the strict variant for fixed pathwidth and ℓ remains open.

Graphs of pathwidth 3

Here we show that (s,z,ℓ)-Temporal Separator can be solved in polynomial time on graphs of pathwidth 3, for any ℓ, which shows that the above hardness is tight, assuming $P \neq NP$. Note that now, we allow temporal edges to have multiple activation times, and we allow them to be directed or not. Let G be a temporal graph. We first apply two reduction rules, which can easily seen to be safe, that is, a solution of size at most k exists in the temporal graph before the application of the rule if and only if it exists also after its application.

▶ **Rule 1.** *If G has a vertex v such that $(s, v, t)(v, z, t')$ is an ℓ-temporal path, then delete v.*

▶ **Rule 2.** *If $G - \{s, z\}$ has multiple connected components $C_1, C_2, \ldots, C_p$, then solve each subgraph $G[C_1 \cup \{s, z\}], \ldots, G[C_p \cup \{s, z\}]$ separately.*

We say that G is *clean* if none of the above rules is applicable to G. Note that for $n = |V(G)|$ and $m = |E(G)|$ (note that $E(G)$ is the set of temporal edges of G), one can determine in time $O(n + m)$ whether one of the rules applies, and each rule can be applied at most n times, and so a graph can be made clean in time $O(n^2 + nm)$. Note, for Rule 1 we assume that the time between the consecutive temporal arcs $(s, v, t), (v, z, t')$ can be compared in constant time, and Rule 2 does not use arc times, so there is no time dependency on τ.

We now turn to (clean) graphs of pathwidth at most 3. We assume that we have checked that G has pathwidth at most 3 and we can compute a decomposition of G in $O(n)$ time [3]. The idea is to first check whether there is an (s, z)-separator of size at most 3 (so, a separator that ignores edge times). If there is one, then there is an (s, z, ℓ)-temporal separator of size at most 3. In this case we can compute a solution by trying every combination of at most three vertices. If there is no such separator, we can rely on the structural lemma below, which is illustrated in Figure 2. First, we can show that there is a sequence of bags that contain both s and z; this allows to find B, consisting of the vertices introduced and forgotten within this sequence. This B induces a caterpillar[4], as removing s and z from these bags yields a subgraph of pathwidth at most 2. The vertices other than s or z in bags that precede this sequence form A, and the vertex s or z introduced last has at most one neighbor in A (e.g., in Figure 2, z has only y as a neighbor in A). Analogously, C contains vertices of bags that occur after the sequence.

▶ **Lemma 4.** *Suppose that G is clean, has pathwidth at most 3, and has no (s, z)-separator of size 3 or less. Then in $O(n^4 m)$ time we can compute a partition of $V(G) \setminus \{s, z\}$ into three non-empty sets A, B, C such that all of the following holds:*

- *either s has at most one neighbor in A (resp. C), or z has at most one neighbor in A (resp. C).*
- *the subgraph induced by B is a caterpillar with main path $w_1 - w_2 - \ldots - w_k$. Moreover, w_1 is the only vertex of $B \cup C$ whose neighborhood can intersect with A, and w_k is the only vertex of $B \cup A$ whose neighborhood can intersect with C.*

One can infer from this structure that "most" deletions occur on the caterpillar path.

[4] Recall that a *caterpillar* is a tree in which there is a *main path* $w_1 - \ldots - w_k$, and every vertex not on that path is a leaf adjacent to a vertex of the path.

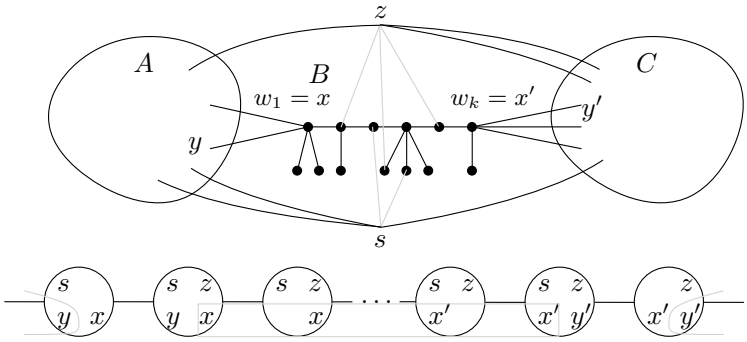

Figure 2 An illustration of the structure described by Lemma 4. The top part shows the temporal graph, the bottom part shows the bags that contain both s and z (the left and right areas represent A and C, and the middle box represents B).

▶ **Lemma 5.** *Suppose that G is clean and that $V(G) \setminus \{s, z\}$ can be partitioned into the sets A, B, C as in Lemma 4. Then there exists a minimum (s, z, ℓ)-temporal separator in which at most one vertex of A is deleted, at most one vertex of C is deleted, and all other deleted vertices are on the main path $w_1 - \ldots - w_k$ of the B caterpillar.*

Algorithm 1 Main algorithm.

```
1  function getTemporalSeparator(G, s, z, (A, B, C))
2      //We assume that G is clean
3      Let 𝒟 = {D ⊆ V(G) : D ∩ A ≤ 1, D ∩ C ≤ 1, and D ∩ B = ∅}
4      for each D ∈ 𝒟 do
5          D' = extendSeparator(G, s, z, (A, B, C), D)        // pass copy of D
6      end
7      return a smallest (s, z, ℓ)-separator D' found
8
9  function extendSeparator(G, s, z, (A, B, C), D)
10     if G − (D ∪ B) has an (s, z, ℓ)-temporal path then
11         return "impossible"
12     Let w_1 − ... − w_k be the main path of the caterpillar B
13     for i = 1, 2, ..., k − 1 do
14         Define W_i = A ∪ {s, z} ∪ (N[w_1, ..., w_i] \ {w_{i+1}})
15         if G[W_i] − D contains an (s, z, ℓ)-temporal path then
16             Add w_i to D
17     end
18     if G − D contains an (s, z, ℓ)-temporal path then
19         Add w_k to D
20     return D;
```

Our algorithm proceeds as follows. We first make G clean, possibly handling multiple connected components of $G - \{s, z\}$ separately, and construct the sets A, B, C from Lemma 4. We then run Algorithm 1, which first guesses every way that a solution could delete at most one vertex from A and at most one vertex from C, noting that such a solution exists by Lemma 5. There are $O(n^2)$ possible guesses, and for each of them, we solve a restricted version of the problem where we can only delete vertices from the caterpillar B.

This is done by the *extendSeparator* function, which attempts to extend the guess by finding the minimum number of deletions to do on the main B path $w_1 - \ldots - w_k$, traversing it from left to right. When at a specific w_i, we restrict the graph to $G[W_i] - D$, where D contains the deletions made so far and W_i contains A, w_i and the predecessors of w_i, and their neighbors. In a greedy manner, we delete w_i only if it creates a temporal path in this restricted graph, considering the deletions D applied so far. We can apply a greedy approach, since we can show that no vertex in $N(w_i)$ is removed, if needed w_i is removed instead. This ensures that there is no temporal path that goes through the w_i's in increasing order (possibly going in A as well), and we make one last check at the end to ensure that no temporal path goes through C and then on the path in the reverse order (if so, we delete w_k to prevent that).

▶ **Theorem 6.** *The (s,z,ℓ)-Temporal Separator problem can be solved in time $O(n^4 m)$ on graphs on pathwidth at most 3, where $n = |V(G)|$ and $m = |E(G)|$.*

Note that the above algorithm works for the strict variant of the problem. Indeed, the algorithm only needs to query whether a forbidden path exists, so it suffices to have access to a routine that determines whether a strict ℓ-temporal path exists. Again we leave the details for a future version.

4 Approximation of (s,z,ℓ)-Temporal Separator

4.1 Hardness of Approximation

In this section we strengthen the inapproximability of (s,z,ℓ)-Strict Temporal Separator (and later we extend the inapproximability to (s,z,ℓ)-Temporal Separator). We prove the result by giving an approximation preserving reduction from Directed Multicut to (s,z,ℓ)-Strict Temporal Separator. Directed Multicut is known to be inapproximable within factor $2^{\Omega(\log^{1-\varepsilon}|N|)}$, for any constant $\varepsilon > 0$, even for a directed acyclic graph with a set N of vertices, unless $NP \subseteq ZPP$ [4]. We recall here that Directed Multicut (we consider it is defined on a directed acyclic graph).

▶ **Problem 3** (Directed Multicut).
Input: *given a directed acyclic graph $D = (N, A)$, a set R of pairs $(s_1, z_1), \ldots, (s_h, z_h)$ of vertices, with $s_i, z_i \in N$, $i \in [h]$.*
Output: *asks for a minimum cardinality subset $A' \subseteq A$ so that each pair (s_i, z_i), $i \in [h]$, is disconnected in $D - A'$.*

Note that we assume that, for each $(s_i, z_i) \in R$, $i \in [h]$, it holds that $s_i \neq t_i$, otherwise the pair cannot be separated by edge removal. Given an instance (D, R) of Directed Multicut, the vertices that belong to a pair in R are called terminals; the set R_S (R_Z, respectively) contains those terminals v such that $v = s_i$ and $(s_i, z_j) \in R$ ($v = z_j$ and $(s_i, z_j = v) \in R$, respectively).

Consider an instance (D, R) of Directed Multicut, in the following we construct a corresponding instance of (s,z,ℓ)-Strict Temporal Separator. We first present the idea of the construction. Essentially G contains $|A| + 1$ copies of each vertex, plus one vertex for each arc in A, in addition to s and z. This ensures that only the vertices associated with arcs will be removed, as removing $|A| + 1$ copies of each vertex of D requires to delete too many vertices. As for the temporal edges of G, for each arc from u to v in D, there is a path of length two, consisting of a temporal edge from the vertex associated with u to vertex associated with arc (u, v) and a temporal edge from the vertex associated with

arc (u, v) to the vertex associated with v. Each vertex v_i is associated with a interval of ℓ timestamps $[\ell \cdot (2i-2), \ell \cdot (2i-1) - 1]$. The timestamps are assigned to temporal edges of G so that if there exists a path in D from v_i to v_j, then there exists a temporal path from two corresponding vertices of G, that is $x_{i,q}$ to $x_{j,r}$, $q, r \in [|A| + 1]$, with timestamps in the interval $[\ell \cdot (2i-2), \ell \cdot (2i-1) - 1]$.

Now, we present the formal definition of the instance of (s,z,ℓ)-Strict Temporal Separator. First, $\ell = 2|N|$. Given $N = \{v_1, \ldots, v_{|N|}\}$, since D is a DAG, we assume that the vertices of D are sorted according to a topological sorting of D, that is for $(v_i, v_j) \in A$, $i, j \in [|N|]$, it holds that $i < j$. We define the set V of vertices:

$$V = \{x_{i,q} : v_i \in N, q \in [|A| + 1]\} \cup \{y_{i,j} : (v_i, v_j) \in A, i < j\} \cup \{s, z\}.$$

Now, we define the set of temporal edges. For a vertex $v_i \in N$, $i \in [|N|]$, $Reach(D, v_i)$ denotes the set of vertices that can be reached with a path that starts from v_i in D.

$$\begin{aligned} E = &\{(s, x_{i,q}, \ell \cdot (2i-2)) : v_i \in R_S, q \in [|A| + 1]\} \cup \\ &\{(x_{j,q}, z, \ell \cdot (2i-1) - 1) : v_j \in R_Z, v_j \in Reach(D, v_i), \exists (v_i, v_j) \in R, q \in [|A| + 1]\} \cup \\ &\{(x_{i,q}, y_{i,j}, \ell \cdot (2h-2) + 2i - 1) : v_i \in N, v_i \in Reach(D, v_h), (v_i, v_j) \in A, h \in [i], q \in [|A| + 1]\} \cup \\ &\{(y_{i,j}, x_{j,q}, \ell \cdot (2h-2) + 2(j-1)) : v_j \in N, v_j \in Reach(D, v_h), (v_i, v_j) \in A, h \in [i], q \in [|A| + 1]\} \end{aligned}$$

Now, we prove the relations between Directed Multicut and (s,z,ℓ)-Strict Temporal Separator.

▶ **Lemma 7.** *Consider an instance (D, R) of Directed Multicut and a the instance (G, s, t, ℓ) of (s,z,ℓ)-Strict Temporal Separator, computed by the construction above. Then (1) given a solution A' of Directed Multicut on instance (D, R) consisting of k arcs we can compute in polynomial time a solution of (s,z,ℓ)-Strict Temporal Separator on instance (G, s, t, ℓ) consisting of k vertices; (2) given a solution of (s,z,ℓ)-Strict Temporal Separator on instance (G, s, z, ℓ) consisting of k vertices we can compute in polynomial time a solution A' of Directed Multicut on instance (D, R) consisting of most k arcs.*

Proof sketch. For (1), given a solution A' of Directed Multicut on instance (D, R) consisting of k arcs, we can compute in polynomial time a solution of (s,z,ℓ)-Strict Temporal Separator on the corresponding instance (G, s, t, ℓ) as follows: $V' = \{y_{i,j} : (v_i, v_j) \in A'\}$. Indeed, if $G[V \setminus V']$ contains a strict temporal path p it must pass through two vertices $x_{i,r}, x_{j,r}$ and there is a path between the corresponding vertices v_i, v_j of $D - A'$ such that $(v_i, v_j) \in R$. For (2), given a solution V' of (s,z,ℓ)-Strict Temporal Separator on instance (G, s, t, ℓ), where $|V'| = k$, first we can show that that there is a solution V^* of (s,z,ℓ)-Strict Temporal Separator on instance (G, s, t, ℓ), with $|V^*| \leq k$, that contains only vertices $y_{i,j}$. We can define a solution A' of Directed Multicut on instance (D, R) as follows $A' = \{(v_i, v_j) : y_{i,j} \in V^*\}$. ◀

It follows from Lemma 7 that we have designed an approximation preserving reduction from Directed Multicut to (s,z,ℓ)-Strict Temporal Separator. Since Directed Multicut is not approximable within factor $2^{\Omega(\log^{1-\varepsilon} |N|)}$, for any constant $\varepsilon > 0$, unless $NP \subseteq ZPP$ [4], we have the following result.

▶ **Theorem 8.** *The (s,z,ℓ)-Strict Temporal Separator problem is $2^{\Omega(\log^{1-\varepsilon} |V|)}$-hard to approximate for any constant $\varepsilon > 0$, unless $NP \subseteq ZPP$.*

The same result holds also for (s,z,ℓ)-Temporal Separator, we need to slightly modify the input temporal graph so that each ℓ-temporal path is forced to be strict.

▶ **Corollary 9.** *The (s,z,ℓ)-Temporal Separator problem is $2^{\Omega(\log^{1-\varepsilon}|V|)}$-hard to approximate for any constant $\varepsilon > 0$, unless $NP \subseteq ZPP$.*

4.2 Approximating (s,z,ℓ)-Strict Temporal Separator

We consider now the (s,z,ℓ)-Strict Temporal Separator problem and we present an $(\ell-1)$-approximation algorithm for it. First, we prove the following easy claim.

▷ Claim 10. A strict ℓ-temporal path between s and z contains at most $\ell-1$ vertices different from s and z.

Now, we present the approximation algorithm (Algorithm 2). The algorithm greedily looks for a strict ℓ-temporal path P between s and z. If there exists such a P, Algorithm 2 removes all the vertices of P from G, except for s and z, and adds its vertices to the vertex separator V'. If there exists no such path, then the algorithm stops and return V'.

Algorithm 2 The approximation algorithm for (s,z,ℓ)-Strict Temporal Separator.

```
1 V' ← ∅;
2 while there exists a strict ℓ-temporal path P, between s and z do
3 |   V' ← V' ∪ (V(P) \ {s, z});
4 |   Remove V(P) \ {s, z} from G;
5 end
6 return (V')
```

Algorithm 2 has approximation factor $\ell-1$ since the ℓ-temporal paths considered in each iteration are vertex-disjoint, thus any solution of (s,z,ℓ)-Strict Temporal Separator contains at least one vertex for each of these temporal path.

▶ **Theorem 11.** *Algorithm 2 is an $(\ell-1)$-approximation algorithm for (s,z,ℓ)-Strict Temporal Separator.*

Next, we prove that, using an approximation preserving reduction from Vertex Cover k-Uniform Hypergraph similar to the one presented in [8] from Set Cover, improving this approximation factor is a challenging problem. In particular, Vertex Cover k-Uniform Hypergraph is not approximable within factor $k-\varepsilon$, for any positive constant ε, assuming the Unique Game Conjecture [13].

▶ **Corollary 12.** *(s,z,ℓ)-Strict Temporal Separator is hard to approximate within factor $(\ell-1)-\varepsilon$, for any positive constant ε, assuming the Unique Game Conjecture.*

5 Tractability and approximation of (s,z,ℓ)-Temporal Cut

In this section we consider the (s,z,ℓ)-Temporal Cut problem. First, we prove the APX-hardness of the problem, then we present a $2\log_2(2\ell)$-approximation algorithm.

5.1 Hardness of (s,z,ℓ)-Temporal Cut

We show the APX-hardness of (s,z,ℓ)-Temporal Cut, by presenting an approximation preserving reduction from the Multiway Cut problem, which is APX-hard [5]. In this latter problem, the input is an undirected graph G with $k \geq 3$ distinguished vertices $v_1, \ldots, v_k$

called *terminals*. The goal is to remove the minimum number of edges from G to cut all paths between terminals, that is, compute $F \subseteq E(G)$ of minimum size such that in $G - F$ there is no path between v_i and v_j for every distinct $i, j \in [k]$.

Consider an instance $(G, v_1, \ldots, v_k)$ of Multiway Cut and construct a temporal graph H as follows. First define $\ell = k - 1$. We assume that some edges of H are *undeletable*. This can easily be achieved by replacing an undeletable edge (u, v, t) with a large number of parallel paths between u and v at time t. We construct H as follows. First, start with a copy of G in which every temporal edge is defined at time ℓ, and every such temporal edge is *deletable*: for each $uv \in E(G)$ we add (u, v, ℓ) to H. Then add vertices s and z, and add the following *undeletable* edges:

$$(s, v_1, 1), \quad (v_1, z, \ell + 1); \quad (s, v_2, 2), \quad (v_2, z, \ell + 2); \quad \ldots$$
$$(s, v_{k-1}, k - 1), \quad (v_{k-1}, z, \ell + k - 1); \quad (v_k, z, \ell).$$

In other words we add (s, v_i, i) and $(v_i, z, \ell + i)$ for $i \in [k - 1]$, and add (v_k, z, ℓ). Note that s is a neighbor of every v_i except v_k. Also observe that the edge $(v_{k-1}, z, \ell + k - 1)$ is useless, but we include it to preserve the pattern.

▶ **Theorem 13.** *The (s,z,ℓ)-Temporal Cut problem is APX-hard, even for $\ell = 2$.*

Proof sketch. Let $F \subseteq E(G)$ be a multiway cut of G, so that all $v_i - v_j$ paths are removed in $G - F$. In H, remove the set $F' = \{(u, v, \ell) : uv \in F\}$. Note that s needs to use two distinct v_i and v_j terminals to reach z, because it first needs to use some (s, v_i, i) temporal edges, but cannot use the temporal edge $(v_i, z, i + \ell)$. But F' cuts any path between two terminals, so F' is a temporal cut of H.

Conversely, any temporal cut $F' \subseteq E(H)$ contains only deletable edges which are copied from G. Consider $F = \{uv \in E(G) : (u, v, \ell) \in F'\}$. A case analysis shows that if $G - F$ has a path between v_i and v_j, then $H - F'$ has an ℓ-temporal path from s to z, going through v_i then v_j or vice-versa using that path. Thus F must be a multiway cut for G. ◀

Note for $\ell = 1$, (s,z,ℓ)-Temporal Cut is in P: the subgraphs that occur at different times can be treated independently, and thus $\ell = 1$ can be solved by computing a minimum edge cut for each possible time.

Also note that unlike most of our previous results, there is no obvious extension of the above reduction to the the strict variant, where one must delete a minimum number of edges to remove all strict temporal paths of travel time at most ℓ. We leave the complexity of latter as an open problem.

5.2 A $2\log_2(2\ell)$-Approximation for (s,z,ℓ)-Temporal Cut

We present a $2\log_2(2\ell)$-approximation algorithm for (s,z,ℓ)-Temporal Cut. We start by recalling a generalization of Menger's Theorem on temporal graphs [2].

▶ **Theorem 14.** *The maximum number of edge-disjoint (s, z)-temporal paths in a temporal graph G equals the size of a minimum temporal (s, z)-cut of G.*

Also given a temporal graph G, it is possible to compute in polynomial time a minimum temporal (s, z)-cut of G [2] and we denote such algorithm by `TempCut`(G). We present now a result that will be useful to prove the approximation factor.

▶ **Lemma 15.** *Given $t \in [2, \tau - 1]$, consider temporal graph $G([t, t + \ell - 1])$ and let E'_t be an (s, z)-cut of $G([t, t + \ell - 1])$. Consider a temporal path p of $G([t + i, t + \ell + i - 1]) - E'_t$, for some $i > 0$, and a temporal path p' of $G([t - j, t - j + \ell - 1]) - E'_t$, for some $j > 0$. Then p and p' are temporal edge disjoint.*

We describe now an algorithm that has approximation factor $\log_2 \tau$, then we show how to use it to achieve approximation factor $2\log_2(2\ell)$. We assume for simplicity that ℓ is even. The algorithm is recursive: given an interval $[a, b]$, at each step if the difference $b - a$ is at

Algorithm 3 $\log_2 \tau$-approximation algorithm for (s,z,ℓ)-Temporal Cut.

```
1 G' ← G,   E' ← ∅,   a ← 1, b ← τ;
2 Function ℓ − TempEdgeCut(G, r, l)
3 if b − a ≥ ℓ − 1 then
4 |   t ← ⌊(a+b)/2⌋;
5 |   E' ← E' ∪ TempCut(G([t − ℓ/2, t + ℓ/2 − 1)]));
6 |   ℓ − TempEdgeCut(G, a, t + ℓ/2 − 2);
7 |   ℓ − TempEdgeCut(G, t − ℓ/2 + 1, b);
8 end
```

least $\ell - 1$, thus $[a, b]$ contains at least ℓ timestamps, it defines a timestamp t that partitions in two (almost) equal parts the intervals of length ℓ contained in $[a, b]$. Then it computes a minimum temporal cut of $G[t - \ell/2, t + \ell/2 - 1]$ and recursively computes a solution on the first part of intervals and a solution on the second part of intervals (independently). Since the number of intervals of length ℓ is bounded by $\tau - \ell < \tau$, and each recursive call partitions in two equal parts this set of intervals, it follows that after $\log_2 \tau$ levels of recursion, the size of an interval is at most ℓ, thus there can be at most $\log_2 \tau$ levels of recursion.

Denote by h the number of recursion levels and by w_i the number of timestamps defined (as t in the pseudocode) at level i, $i \in [h]$. Denote by $t_{i,j}$ the j-th timestamp chosen by the algorithm at the i-th level of the recursion, $i \in [h]$ and $j \in [w_i]$. Now, consider an approximate solution E' and optimal solution Opt of (s,z,ℓ)-Temporal Cut. We partition E' and Opt as follows. Given a timestamp $t_{i,j}$, $i \in [h]$ and $j \in [w_i]$, we denote by $E'(t_{i,j})$ the set of temporal edges added to E' by the approximation algorithm when it defines $t = t_{i,j}$. By construction and by Lemma 15, the sets $E'(t_{i,j})$, $i \in [h]$, $j \in [w_i]$, define a partition of E'. Similarly, we define $Opt(t_{i,j})$, $i \in [h]$, $j \in [w_i]$, as the set of temporal edges $e \in Opt$ such that (1) there is an ℓ-temporal path in $G([t_{i,j} - \ell/2, t_{i,j} + \ell/2 - 1])$ from s to z that contains e and (2) e does not belong to any $Opt(t_{r,q})$, with $r < i$ for some $q \in [w_r]$. Note that by Lemma 15 this is indeed a partition, and in particular for each $i \in [h]$ it holds that $Opt(t_{i,j}) \cap Opt(t_{i,q}) = \emptyset$ for each $j, q \in [w_i]$ and $j \neq q$.

Since each temporal edge $e \in Opt$ belongs to exactly one set $Opt(t_{i,j})$ and since $Opt(t_{i,j}) \cap Opt(t_{i,q}) = \emptyset$, for each $j, q \in [w_i]$, we have that

$$Opt = \bigcup_{i \in [h], j \in [w_i]} Opt(t_{i,j})$$

and

$$|Opt| = \sum_{i \in [h], j \in [w_i]} |Opt(t_{i,j})|.$$

Now, we prove the following result.

▶ **Lemma 16.** *Consider $E'(t_{i,j})$, $i \in [h]$, $j \in [w_i]$, the set of temporal edges cut by Algorithm 3 when it defines $t = t_{i,j}$. For each level $i \in [h]$ of the recursion, it holds that*

$$\sum_{j \in [w_i]} |E'(t_{i,j})| \leq \sum_{j \in [w_i]} |Opt(t_{i,j})| + \sum_{r \in [h] : r < i, q \in [w_r]} |Opt(t_{r,q})|.$$

Based on the previous lemma, since the number of recursion levels is bounded by $\log_2 \tau$, we can prove the following.

▶ **Theorem 17.** *Algorithm 3 returns an (s, z, ℓ)-temporal cut of size at most $\log_2 \tau \cdot Opt$.*

Next we prove that the previous approximation algorithm can be used to obtain an approximation algorithm of factor $2\log_2(2\ell)$ for (s,z,ℓ)-Temporal Cut. We assume that τ is a multiple of 2ℓ (if not, increase τ to the next multiple of 2ℓ, with no temporal edges existing in the extra times appended). Consider the time domain $[1, \tau]$ and computes the following sets of disjoint intervals of $[1, \tau]$:

- $P_1 = \{[1, 2\ell], [2\ell + 1, 4\ell], ..., [\tau - 2\ell + 1, \tau]\}$;
- $P_2 = \{[\ell, 3\ell], [3\ell + 1, 5\ell], ..., [\tau - 3\ell, \tau - \ell]\}$.

Each interval of length ℓ is contained in at least one element of P_1 or P_2. and, moreover, two graphs defined on distinct intervals of a set P_i are temporal edge disjoint. Recall that $G(I)$ is the temporal graph defined on interval I.

▶ **Lemma 18.** *Consider the sets P_1 and P_2, then (1) for each $I = [t, t + \ell - 1]$ of length ℓ, there is an interval of P_1 or P_2 that contains it; (2) Given I and I' be two distinct interval of P_i, then a temporal path of $G(I)$ and a temporal path of $G(I')$ are temporal edge disjoint.*

Now, we run Algorithm 3 on each $G(I)$, with $I \in P_i$ and we compute a feasible solution since by Lemma 18 each interval I of length ℓ is contained in an interval of P_1 or P_2. The $2\log_2(2\ell)$-approximation factor is due to the fact that each interval I has length 2ℓ and we approximate each with a factor of $\log_2(2\ell)$. Then by Lemma 18 two intervals I and I' of P_i, are temporal edge disjoint, thus each temporal edge removed by an optimal solution belongs to at most one interval of P_1 and at most one interval of P_2. Because we remove edges from both the intervals in P_1 and P_2, our approximation factor is multiplied by two.

▶ **Corollary 19.** *(s,z,ℓ)-Temporal Cut admits a $2\log_2(2\ell)$-approximation algorithm.*

6 Conclusion

We have studied (s,z,ℓ)-Temporal Separator and (s,z,ℓ)-Temporal Cut, two variants of the vertex separator and the edge cut problem in temporal graphs. We have shown that (s,z,ℓ)-Temporal Separator is NP-hard even when the underlying graph has pathwidth bounded by four and it can be solved in polynomial time for graphs of pathwidth bounded by three. We have shown that (s,z,ℓ)-Temporal Separator cannot be approximated within factor $2^{\Omega(\log^{1-\varepsilon}|V|)}$ for any constant $\varepsilon > 0$, unless $NP \subseteq ZPP$, and that the strict version is approximable within factor $\ell - 1$; moreover, we have shown that it is unliklely that this factor can be improved. Finally, We have shown that (s,z,ℓ)-Temporal Cut is APX-hard and that it is approximable within factor $2\log_2(2\ell)$.

We conclude the paper with a few open problems:

- Is the (s,z,ℓ)-Temporal Separator problem NP-hard on graphs of treewidth at most three?
- Is there a $O(\ell)$-approximation for (s,z,ℓ)-Temporal Separator (non strict variant)?
- Is there a $O(1)$-approximation for the (s,z,ℓ)-Temporal Cut problem?
- Is the *strict* variant of the (s,z,ℓ)-Temporal Cut NP-hard?

References

1 Dan Bergren, Eduard Eiben, Robert Ganian, and Iyad Kanj. On covering segments with unit intervals. *SIAM J. Discret. Math.*, 36(2):1200–1230, 2022. doi:10.1137/20M1336412.

2 Kenneth A. Berman. Vulnerability of scheduled networks and a generalization of menger's theorem. *Networks*, 28(3):125–134, 1996. doi:10.1002/(SICI)1097-0037(199610)28:3\%3C125::AID-NET1\%3E3.0.CO;2-P.

3 Hans L Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996. doi:10.1006/JAGM.1996.0049.

4 Julia Chuzhoy and Sanjeev Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. *J. ACM*, 56(2):6:1–6:28, 2009. doi:10.1145/1502793.1502795.

5 Elias Dahlhaus, David S Johnson, Christos H Papadimitriou, Paul D Seymour, and Mihalis Yannakakis. The complexity of multiway cuts. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 241–251, 1992.

6 Michael R Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical computer science*, 410(1):53–61, 2009. doi:10.1016/J.TCS.2008.09.065.

7 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theor. Comput. Sci.*, 806:197–218, 2020. doi:10.1016/J.TCS.2019.03.031.

8 Hovhannes A. Harutyunyan, Kamran Koupayi, and Denis Pankratov. Temporal separators with deadlines. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation, ISAAC 2023, December 3-6, 2023, Kyoto, Japan*, volume 283 of *LIPIcs*, pages 38:1–38:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ISAAC.2023.38.

9 Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234, 2015.

10 Mohammad Mehdi Hosseinzadeh, Mario Cannataro, Pietro Hiram Guzzi, and Riccardo Dondi. Temporal networks in biology and medicine: a survey on models, algorithms, and tools. *Netw. Model. Anal. Health Informatics Bioinform.*, 12(1):10, 2023. doi:10.1007/S13721-022-00406-X.

11 Allen Ibiapina and Ana Silva. Mengerian graphs: Characterization and recognition. *J. Comput. Syst. Sci.*, 139:103467, 2024. doi:10.1016/J.JCSS.2023.103467.

12 David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *J. Comput. Syst. Sci.*, 64(4):820–842, 2002. doi:10.1006/jcss.2002.1829.

13 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

14 Nina Klobas, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Interference-free walks in time: temporally disjoint paths. *Auton. Agents Multi Agent Syst.*, 37(1):1, 2023. doi:10.1007/S10458-022-09583-5.

15 Nicolas Maack, Hendrik Molter, Rolf Niedermeier, and Malte Renken. On finding separators in temporal split and permutation graphs. *J. Comput. Syst. Sci.*, 135:1–14, 2023. doi:10.1016/J.JCSS.2023.01.004.

16 Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Math.*, 12(4):239–280, 2016. doi:10.1080/15427951.2016.1177801.

17 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. *J. Comput. Syst. Sci.*, 107:72–92, 2020. doi:10.1016/J.JCSS.2019.07.006.