Separation and Cover Problems in Temporal Graphs

Kamran Koupayi

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

December 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Kamran Koupayi

Entitled: Separation and Cover Problems in Temporal Graph

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Jaroslav Opatry

_____ Examiner
Dr. Thomas Fevens

_____ Examiner
Dr. Jaroslav Opatry

_____ Thesis Supervisor(s)
Dr. Hovhannes Harutyunyan

_____ Thesis Supervisor(s)
Dr. Denis Pankratov

Approved by _____
Dr. Hovhannes Harutyunyan Chair of Department or Graduate Program Director

_____
Dean Dr. Mourad Debbabi

Abstract

Separation and Cover Problems in Temporal Graphs

Kamran Koupayi

A graph that changes with time is called a temporal graph. In this work, we focus on temporal graphs whose vertex sets are fixed while edge sets change in discrete time steps. We use n to refer to the number of vertices in the graph and τ to refer to the total number of time steps over which a temporal graph is observed. We refer to non-temporal graphs as static graphs when we wish to emphasize their unchanging nature.

In this work, we study temporal analogues of the Vertex Separator and Vertex Cover problems from the static world with an emphasis on the Vertex Separator problem. 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 **NP**-hard in general, although some special cases (such as bounded treewidth) admit efficient algorithms [22].

We introduce a generalization of this 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 take less than t time steps. Observe that setting $t = \tau$ captures the (s, z)-Temporal Separator problem as a special case of (s, z, t)-Temporal Separator problem.

We present a τ-approximation algorithm for (s, z)-Temporal Separator problem, and we convert it to a τ^2-approximation algorithm for (s, z, t)-Temporal Separator problem. We also present an inapproximability lower bound of $\Omega(\ln(n) + \ln(\tau))$ for (s, z, t)-Temporal Separator problem assuming that $\mathbf{P} \neq \mathbf{NP}$.

We show a polynomial-time reduction from the Discrete Segment Covering problem with bounded-length segments to (s, z, t)-Temporal Separator where the temporal graph has bounded pathwidth. Therefore, solving (s, z, t)-Temporal Separator on

temporal graph whose underlying graph has bounded pathwidth is more difficult than solving Discrete Segment Covering problem where all segments' lengths are bounded.

Discrete segment cover is a set of unit-length intervals, which covers at least one of two endpoints of each input segment.

Lastly, we present a polynomial-time algorithm to find minimum (s, z, t)-temporal separator on temporal graphs whose underlying graph is a *series-parallel* graph or by removing the source and the terminal it is turned into a tree.

The second problem of interest is the Activity Timeline problem which is a generalization of a Vertex Cover to the temporal setting. An activity timeline is an assignment of time intervals to vertices. An edge between vertex u and vertex v is covered by an activity timeline φ at time t if one of the time intervals assigned to u or v includes the time t (this is required only if the edge is actually present at time t). In $MinTimeline$ the goal is to find an activity timeline that covers all edges while minimizing the total length of time intervals. This problem is known to be **NP**-hard. In another problem, called $MinTimeline_m$, we are asked to find an activity timeline subject to additional constraints specified by a set of prescribed times $\{m_v\}$, one for each vertex v. A valid activity timeline for $MinTimeline_m$ must guarantee that for every vertex v the interval corresponding to v contains m_v. The goal is again to minimize the total length of time intervals. Prior to this work, the best known approximation algorithm for $MinTimline_m$ problem was based on a rather complicated primal-dual linear programming approach and achieved 2 approximation [43]. In this work, we present a simple purely combinatorial 2-approximation algorithm for this problem.

The combinatorial algorithm is inspired by the famous Double Coverage algorithm for the k-Server problem on a line in the area of online algorithms [11]. This highlights an interesting cross-over between temporal graph algorithms and online algorithms that might require further investigation. Lastly, we present a polynomial-time algorithm for $MinTimeline$ on temporal graphs whose underlying graphs have bounded treewidth.

Acknowledgments

I would like to acknowledge everyone who played a role in my academic life. First of all, I would like to express my appreciation to my co-supervisors Dr. Harutyunyan and Dr. Pankratov, who guided me and advised me throughout the way. Next, I would like to thank my wife, parents, and sister for supporting me. I would like to dedicate this work to the souls of all 176 passengers of flight PS752 who were shot down shortly after takeoff on January 8th, 2020.

Contents

List of Figures

Chapter 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., downtown and the beach, by shutting down 5 train stations in the city? How would you go about solving this problem? Can you write an algorithm? Does an efficient algorithm even exist? These are central questions of interest in this thesis.

To answer these questions with precision we need a mathematical model. Perhaps the most natural choice is to model the train system of a city as a graph. After all, a graph is a mathematical structure used to abstract away a set of objects (also called vertices or nodes) and pairwise relations between them (also called edges or adjacency relations). Throughout this thesis, we typically use the variable n to refer to the number of vertices in a graph. We can represent the train system as a graph, in which vertices are train stations, and there is an edge between two train stations if and only if the two train stations are connected by train tracks. Graph theory has been immensely successful and influential from its early roots in the famous Seven Bridges of Königsberg problem (solved by Leonhard Euler in 1736 [16]) to modern applications. The list of applications of graph theory is too vast to even begin listing it here, thus we restrict ourselves by mentioning a couple of modern applications, such as studying social and physical networks brought about by the rapid growth of the Internet, and building complicated topologies for deep neural networks [27, 34]. For an introduction to graph theory, we refer an interested reader to [12, 47].

After modelling the train system in terms of classical graph theory, one quickly

realizes that there is an important component that is missing, namely, 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 the station A and the station B just because there are tracks connecting them. It would be more accurate to say that if you arrive to the station A at some specific time t then you could get to the station B at some other time $t' > t$, where t is when the train arrives at the station A and t' is the time when this train reaches the 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 questions, 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 [42, 3, 42]. First of all, there is an area of online algorithms [2] 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 similarly to online algorithms, but the emphasis is on memory-limited algorithms [19, 18]. 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 [46]. 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 a fourth 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 temporal graphs. In this work, we focus on 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 thesis. 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 Vertex Separator and Vertex Cover problems 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 **NP**-hard in general [51], although some special cases (such as bounded treewidth) admit efficient algorithms [22]. This question can be thought of as a mathematical abstraction of the robustness of 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 any 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 take less than t time steps. Observe that setting $t = \tau$ captures the (s, z)-Temporal Separator problem as a special case of (s, z, t)-Temporal Separator problem.

We present a τ-approximation algorithm for (s, z)-Temporal Separator problem,

and we convert it to a τ^2-approximation algorithm for (s, z, t)-Temporal Separator problem. We also present an inapproximability lower bound of $\Omega(\ln(n) + \ln(\tau))$ for (s, z, t)-Temporal Separator problem assuming that $\mathbf{P} \neq \mathbf{NP}$. We show a polynomial-time reduction from the Discrete Segment Covering problem with bounded-length segments to (s, z, t)-Temporal Separator where the temporal graph has bounded treewidth. Therefore, solving (s, z, t)-Temporal Separator on temporal graph whose underlying graph has bounded pathwidth is more difficult than to solve Discrete Segment Covering problem where lengths of all segments are bounded. Lastly, we present a polynomial-time algorithm to find minimum (s, z, t)-temporal separator on temporal graphs whose underlying graph is *series parallel* graph or by removing the source and the terminal it is turned into a tree.

The second problem of interest is the Activity Timeline problem which is a generalization of a Vertex Cover in the temporal setting. An activity timeline is an assignment of time intervals to vertices. An edge between vertex u and vertex v is covered by an activity timeline φ at time t if one of the time intervals assigned to u or v includes the time t (this is required only if the edge is actually present at time t). In $MinTimeline$ the goal is to find an activity timeline that covers all edges while minimizing the total length of time intervals. This problem is known to be **NP**-hard. In another problem, called $MinTimeline_m$, we are asked to find an activity timeline where each interval contains a point m_v for a given set of point $\{m_v\}_{v \in V}$ that minimizes the total length of time intervals. Adding 4 edges (x, y, m_v), (x, y, k), (x, v, m_v), and (y, v, m_v) where k is a sufficient large number, will force a time interval for vertex v to contain the time m_v. Therefore, this problem is much easier than $MinTimeline$ problem. Also, the setup with value of m_v is equal to 0 and all the edges present in time 1, will make the solution equal to the solution for Vertex Cover. So, this problem is more difficult than Vertex Cover. Prior to this work, the best known approximation algorithm for $MinTimline_m$ problem was based on a rather complicated primal-dual linear programming approach and achieved approximation ratio 2 [43]. In this work, we present a simple purely combinatorial 2-approximation algorithm for this problem. The combinatorial algorithm is inspired by the famous Double Coverage algorithm for the k-Server problem on a line in the area of online algorithms [11]. This highlights an interesting cross-over between temporal graph algorithms and online algorithms that might require further investigation. Lastly,

we present a polynomial-time algorithm for *MinTimeline* on temporal graphs whose underlying graphs have bounded treewidth.

Organization. The rest of this thesis is organized as follows. In the remainder of this chapter we present formal definitions necessary for the rest of the thesis. In Chapter 2, we survey related works on static and temporal graphs, in particular, works on problems concerning minimum paths, broadcasting, reducing reachability, and vertex separators. In Chapter 3, we introduce, define and study our new problem (s, z, t)-Temporal Separator. We present our main results in various sections of that chapter: an approximation algorithm, a lower bound, a connection between the Discrete Segment Covering problem and our problem, and a polynomial time algorithm for the graphs with branchwidth bounded by 2. In Chapter 4, we study the Activity Timeline problem, where we present our simplified purely combinatorial 2-approximation algorithm as well as our bounded treewidth algorithm. We finish this thesis with conclusions and discussion of future work in Chapter 5.

1.1 Formal Definitions

Temporal graphs (also known as dynamic, evolving [20], or time-varying [21, 10] graphs) are graphs whose edge is active on certain points of time. There are two general modeling forms of representation for the temporal graph.

In the first one, a graph $G = (V, E, \lambda)$ is presented by a vertex set V, an edge set E, and a time label function $\lambda : E \to 2^{\mathbb{N}}$ which assigns to every edge of G a set of natural numbers, and shows the time steps that each edge is active in.

The other model which is used mainly in this work, 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 $e \in E$ contains two vertices of V and a time label $t \in [\tau]$. A graph $G_{\downarrow} = (V, E')$ where E' contains every edge e that is active at least in one time in the temporal graph G is called the underlying graph [2] of the temporal graph G. A static graph representing active edges for a specific time is called the layer of the temporal graph at that time. So another representation model for the temporal graph is showing the graph with all of its layers.

There are so many problems defined on temporal graphs; since the temporal graph

[1] [n] is equal to the set of natural numbers lower than n *i.e.* $\{1, 2, \ldots, n\}$

[2] Also known as static graph or footprint.

is modeling the network that each edge is active at specific times, some problems like broadcasting, exploring, readability, covering problems, etc., were defined with the same meaning in a temporal graph. One of these problems is *Vertex Separator*. In graph theory, a subset S of vertices is vertex separator for non-adjacent vertices s and z if removing a set S from the graph disconnects two vertices s and z from each other. A Vertex Separator problem could be simply reduced to the max-flow Min-Cut problem, which is polynomial-time solvable in a static graph; however, the hardness of the equivalent problem in a temporal graph, called Temporal Separator, is shown by [30].

In this section, we mainly focus on basic definitions and notations for temporal graphs and problems introduced shortly in the introduction. However, we will also take a look at branch decomposition and tree decomposition on a static graph. Before going to the main part, let us define static graphs.

Definition 1 (Graph)**.** *A graph $G = (V, E)$ is a set of vertices V and a set of edges $E \subseteq V \times V$ that represent each link in the graph.*

1.1.1 Temporal Graphs

A graph that changes over time is known as a temporal graph. A formal definition for the temporal graph is represented below.

Definition 2 (Temporal graph)**.** *A temporal graph $G = (V, E, \tau)$ contains a set of vertices V, and a set of edges $E \subseteq V \times V \times [\tau]$ that represent a active link between 2 vertices in specific time.*

An edge $e \in E$ is a triple (u, v, t) such that $u, v \in V$ and $t \in [\tau]$ which shows the node u and v has link to each other in time t. We denote t as a time for edge (u, v, t). Figure 1 shows an example of a temporal graph. Some other modeling for temporal graphs could be found in [36]. In this work, we always use this model. A layer G_i such that $i \in [\tau]$ is a static graph representing all the active links in time i. Figure 2 shows three layers of the temporal graph that is shown in Figure 1. An underlying graph $G_{\downarrow}$ is a static graph representing all the links that are active at least at one time.

Definition 3 (Underlying graph)**.** *An underlying graph of temporal graph $G =$*

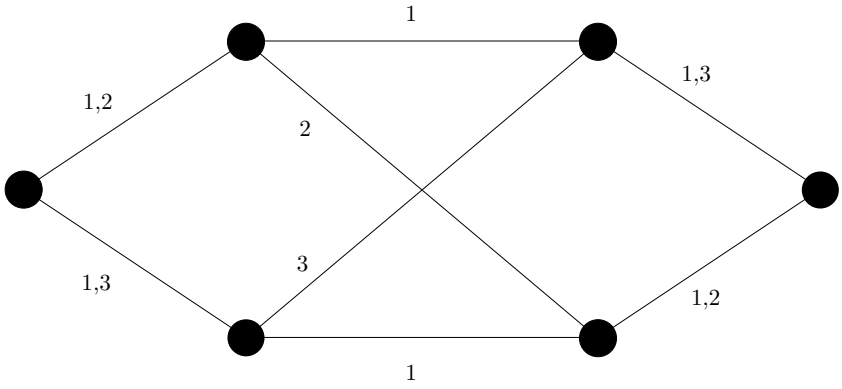

Figure 1: Example of a temporal graph.

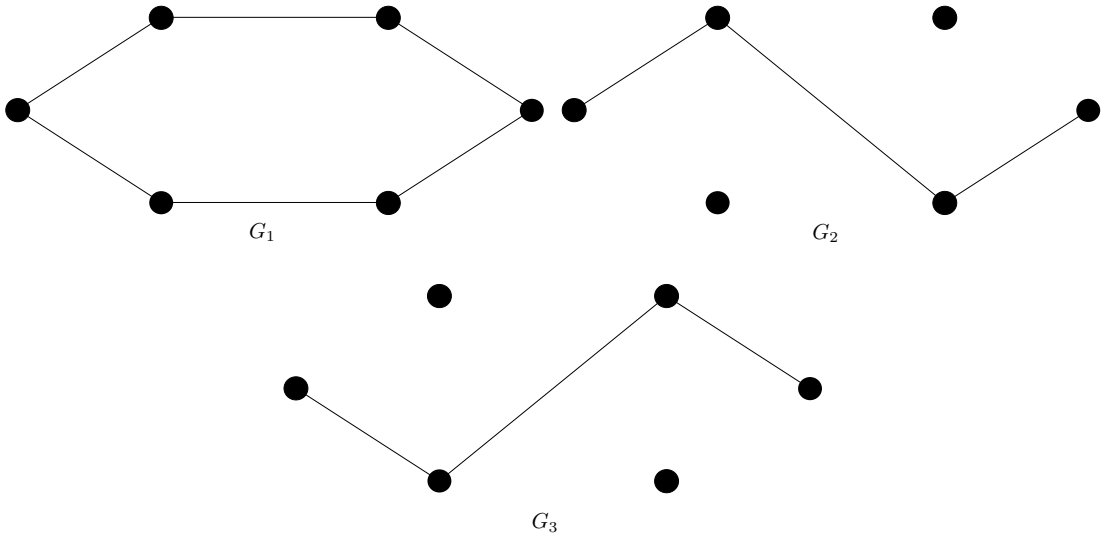

Figure 2: Example of layers of a temporal graph.

(V, E, τ) *is a static graph* $G = (V, E')$ *for which any* $(u, v) \in E'$ *if and only if exists a time* $t \in [\tau]$ *such that* $(u, v, t) \in E$.

In static graphs, a path P is a sequence of edges which joins a sequence of vertices. In the temporal graphs, a temporal path is a sequence of edges that creates a path in the underlying graph, and the sequence of time for edges is in non-decreasing order.

Definition 4 ((s, z)-temporal path)**.** *A sequence* $(u_1, v_1, t_1), (u_2, v_2, t_2), \ldots, (u_k, v_k, t_k)$ *of edges is called* (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 time is in strictly increasing order, the temporal path is called strict. So, a strict (s, z)-temporal path is a temporal path from source s to destination z such that the time of each edge in the sequence is strictly lower than the time for the next one.. *Traveling time* of (s, z)-temporal path P is the time that takes to travel from source s to the destination z.

In static graphs shortest path that number of edges is as short as possible. In addition to the shortest path in temporal graphs, more features are defined for the temporal path in the temporal graph. So, we have:

- ***Shortest*** (s, z)***-temporal path***: A temporal path from s to z that minimizes the number of edges.
- ***Fastest*** (s, z)***-temporal path***: A temporal path from s to z that minimizes the traveling time.
- ***Foremost*** (s, z)***-temporal path***: A temporal path from s to z that minimizes the arrival time of destination.

Temporal distance from node s to node z is equal to the traveling time of the fastest (s, z)-temporal path.

Like the static graph, we say that a temporal graph $G = (V, E, \tau)$ is connected if for any pair (s, z) of vertices there is at least one temporal path from vertex s to vertex z. Moreover, we say temporal graph $G = (V, E, \tau)$ is *continuously connected* if for every $i \in [\tau]$ layer G_i is connected. Temporal graphs that each edge will appear periodically is called a periodic temporal graph.

Definition 5 (Periodic temporal graph)**.** *A temporal graph $G = (V, E, \tau)$ is p-periodic if $p \in \mathbb{N}$ is the smallest number such that $G = G'^r$ for some $G'^r = (V, E'^r, p)$ and r is called the number of periods.*

1.1.2 Temporal Separator

Reducing reachability is one of the classics problems in graph theory. Graph Gut and Separator are both well-studied problems. Strict Temporal Separator and non-strict Temporal Separator are defined and studied. Here in this work, we will refer temporal separator to non-strict temporal separator. First, we look at those problems, and then we define a new problem.

Definition 6 ((s, z)-temporal separator)**.** *let $G = (V, E, \tau)$ be a temporal graph, and two vertices s and z. A set $S \subseteq V \backslash \{s, z\}$ is called (s, z)-temporal separators if removal of vertices in set S, remove all temporal path from vertex s to vertex z.*

Similarly, strict (s, z)-temporal separator is a set of vertices such that by removing them, all of the strict (s, z)-temporal path will remove from the graph. In the Temporal Separator problem, we want to find a set of the minimum temporal separator. The minimum separator's cardinality for the underlying graph of the temporal graph shown in Figure 3 is 3. However, the set $\{a, b\}$ is a temporal separator of the temporal graph.

Problem 1 ((s, z)-Temporal Separator)**.**

- ***Instance**: A temporal graph $G = (V, E, \tau)$ and a source $s \in V$ with the terminal $t \in V$.*
- ***Solution**: A set of (s, z)-temporal separator $S \in V \backslash \{s, z\}$*
- ***Measure**: Minimize cordiality of set S*

Similarly, a strict (s, z)-Temporal Separator problem will be defined. We could generalize this problem by applying any restriction to the (s, z)-temporal path. So we refer the *Restricted Path (s, z)-Temporal Separators* to a problem whose goal is to find a set of vertices that remove them will remove all the restricted (s, z)-temporal path. A natural one is restricted by the time that a path will take (i.e., the difference of arrival time of terminal and departure time of source).

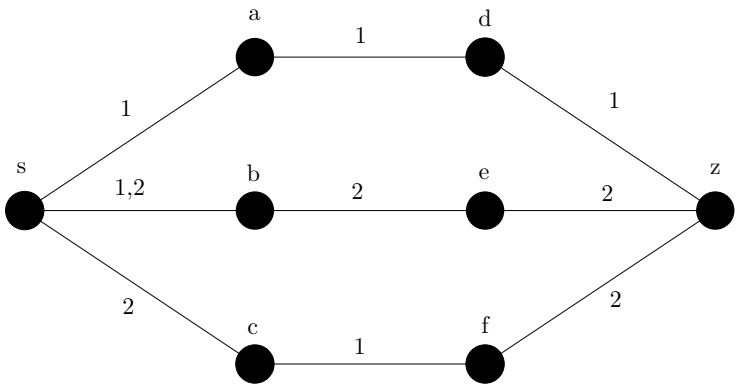

Figure 3: Three different types of separators.

Definition 7 ((s,z,t)-temporal path)**.** *$(s,z,t)-$temporal path is a (s,z)-temporal path such that the difference of arrival time of terminal t and source s is lower than t.*

(s,z,t)-temporal separator is a set of vertices S such that all (s,z,t)-temporal path contains on vertex from S. For instance, a set $\{a\}$ is a $(s,z,1)$-temporal separator for a temporal graph, which is shown in Figure 3, whereas the minimum size for any (s,z)-temporal separator is two.

Definition 8 ((s,z,t)-temporal separator)**.** *Let $G=(V,E,\tau)$ be a temporal graph and $s,z \in V$ two distinct vertices. A set $S \subseteq V \backslash \{s,z\}$ is a set of (s,z,t)-temporal separators if the temporal distance between node s and z in temporal graph $(V \backslash S, E)$ is greater than or equal to t.*

So, a new problem will be defined similarly.

Problem 2 ((s,z,t)-Temporal Separator)**.**

- ***Instance**: A temporal graph $G=(V,E,\tau)$ and a source $s \in V$ with the terminal $t \in V$.*
- ***Solution**: A set of (s,z,t)-temporal separator $S \subseteq V \backslash \{s,z\}$*
- ***Measure**: Minimize cardinality of set S*

1.1.3 Activity Timeline

Here in this section we go over the definition of problems in activity timeline. An activity timeline φ is a set of interval $\{I_v\}_{v \in V}$ which any intervals I_v is equal to $[s_v, e_v]$.

For any edge $e = (u, v, t) \in E$ we say that e is covered by activity timeline φ if $t \in I_v$ (i.e. $t \geq s_v$ and $t \leq e_v$) or $t \in I_u$, also we say that activity timeline φ covers graph G if and only if all the edge $e \in E$ are covered by φ. For activity timeline φ we define two measures total span and max span. We denote the total span of activity timeline by a function $\mathcal{S}$, the total span of activity timeline $\mathcal{S}(\varphi)$ is equal to:

$$\mathcal{S}(\varphi) = \sum_{u \in V} (e_u - s_u) \tag{1}$$

Respectively we denote max span of activity interval φ by a function $\Delta(\varphi)$, the max span of activity timeline $\Delta(\varphi)$ is equal to:

$$\Delta(\varphi) = \max_{u \in V} (e_u - s_u) \tag{2}$$

Two problems $MinTimeline$ and $MinTimeline_\infty$ are defined to find activity timeline φ where total span and max span of φ are minimized, respectively.

Problem 3 ($MinTimeline$)**.**

- ***Instance****: A temporal graph $G = (V, E, \tau)$*
- ***Solution****: An activity timeline φ such that cover the temporal graph G*
- ***Measure****: Minimize total span $\mathcal{S}(\varphi)$*

Problem 4 ($MinTimeline_\infty$)**.**

- ***Instance****: A temporal graph $G = (V, E, \tau)$*
- ***Solution****: An activity timeline φ such that cover the temporal graph G*
- ***Measure****: Minimize max span $\Delta(\varphi)$*

By applying an extra condition for each vertex by specifying a time that should be present for the vertex interval, another problem could be defined.

Problem 5 ($MinTimeline_m$)**.**

- ***Instance****: A temporal graph $G = (V, E, \tau)$ and set of time $\{m_u\}_{u \in V}$*
- ***Solution****: An activity timeline $\varphi = \{I_u\}_{u \in V}$ such that φ covers the temporal graph G and for each $u \in V$ satisfies the condition $m_v \in I_v$*
- ***Measure****: Minimize total span $\Delta(\varphi)$*

1.1.4 Tree Decomposition and Branch Decomposition

Most of the problems that are **NP**-Hard in graph theory could be solved in a graph with bounded treewidth. Here in this section, we go over the definition related to tree decomposition and branch decomposition.

Definition 9 (Tree Decomposition)**.** *A tree decomposition of a graph $G = (V, E)$ is a pair (T, β) consisting of a tree T and a family $\beta = \{\beta(i)\}_{i \in V(T)}$ of subsets of V and satisfying the following properties:*

- *The union of all sets $\beta(i)$ is equal to V. It means each vertex in the graph G will appear in at least one set.*
- *For every edge $(v, u) \in E$, there is subset $\beta(i)$ the contains both v and u. That is, vertices are adjacent in the graph only when the corresponding subtrees have a node in common.*
- *If $\beta(i)$ and $\beta(j)$ both contain vertex v then for all node $k \in V(T)$ in the unique path between node i and j, $v \in \beta(k)$. It can be stated equivalently that if i, j and k are nodes, and k is on the path from i to j, then $\beta(i) \cap \beta(j) \subseteq \beta(k)$.*

Definition 10 (Width of Tree Decomposition)**.** *Given a tree decomposition (T, β) of $G = (V, E)$, the width of this decomposition is maximum value of $\{|\beta(i)| : i \in V(T)\}$.*

A treewidth $tw(G)$ of G is defined as the minimum width of all tree decomposition (T, β) for G. Path decomposition for graph G is pair (P, β) consisting a path P and a family $\beta = \{\beta(i)\}_{i \in V(T)}$ of subsets of V such that satisfies all the condition for tree decomposition.

Tree decomposition could be turned to a nice tree decomposition in polynomial time [13].

Definition 11 (Nice Tree Decomposition)**.** *A tree decomposition $T = (T, \beta)$ of a graph $G = (V, E)$ is a nice tree decomposition if T is rooted, every node of the tree T has at most two children nodes, and for each node $i \in V(T)$ the following conditions are satisfied:*

- *If i has two children nodes $k, j \in V(T)$ in T, then $\beta(i) = \beta(k)\beta(j)$. Node i is called a join node.*

- *If i has one child node j, then one of the following conditions must hold:*
  - $\beta(i) = \beta(j) \cup \{v\}$. *Node i is called an introduce node of v.*
  - $\beta(i) = \beta(j) \backslash \{v\}$. *Node i is called a forget node of v.*
- *If i is a leaf in T, then $|\beta(i)| = 1$. Node i is called a leaf node.*

Branch decomposition and branchwidth of graph will be define as follows.

Definition 12 (Branch Decomposition)**.** [14] *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 $P(E)$ 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 13 (Boundary)**.** [14] *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 14 (Width of a Branch Decomposition)**.** [14] *Given a branch decomposition (T, β) of $G = (V, E)$, the width of this decomposition is $max\{|\partial\beta(v)| v \in V(T)\}$.*

The branchwidth $bw(G)$ of G is defined as the minimum width of all branch decomposition (T, β) for G [14].

Here in this thesis, we refer to treewidth, branchwidth, and pathwidth of a temporal graph as treewidth, branchwidth, and pathwidth, respectively, of its underlying graph.

Chapter 2

Literature Review

In this chapter we mainly discuss the different problems mentioned in the fields of temporal graphs and some related works and open problems in temporal separators. Furthermore, some theorems and lemmas have been provided which aid us in our results.

2.1 Temporal Path

A nice property of foremost temporal path is that they 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 $w \in V\{s\}$ a foremost (s, w)-temporal path from time t_{start} [35]. 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 D. Such algorithms are known as offline algorithms in contrast to online algorithms to 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. [12] page 531) with path length replaced by path arrival time. For every time t, the algorithm picks one after the other all nodes that have been already reached (initially only the source node s) and inspects all edges that are incident to that node at time t. If a time-edge (u, w, t) leads to a node w that has not yet been reached, then (u, w, t) is picked as an edge of a foremost temporal path from the source to w. This greedy algorithm is correct for the same reason that the BFS algorithm is correct. An immediate way to see this is by considering the static expansion of the temporal graph. The algorithm

begins from the upper copy (i.e. at level 0) of the source in the static expansion and essentially executes the following slight variation of BFS: at step $i+1$, given the set R of already reached nodes at level i, the algorithm first follows all vertical edges leaving R in order to reach in one step the $(i+1)$-th copy of each node in R, and then inspects all diagonal edges leaving R to discover new reachabilities. The algorithm outputs as a foremost temporal path to a node u, the directed path of time-edges by which it first reached the column of u (vertical edges are interpreted as waiting on the corresponding node). The above algorithm computes a shortest path to each column of the static expansion. Correctness follows from the fact that shortest paths to columns are equivalent to foremost temporal path to the nodes corresponding to the columns [36].

2.2 Broadcasting and Gathering of Information

A natural application domain of temporal graphs is that of *gossiping* and in general of *information dissemination*, mainly by a distributed set of entities (e.g. a group of people or a set of distributed processes). Two early such examples were the *telephone problem* [6] and the *minimum broadcast time problem* [40]. In both, the goal is to transmit some information to every participant of the system, while minimizing some measure of communication or time. A more modern setting, but in the same spirit, comes from the very young area of distributed computing in highly dynamic networks [38], [32], [33], [10], [37], [35].

There are n nodes. In this context, nodes represent distributed processes. Note, however, that most of the results that we will discuss, concern centralized algorithms (and in case of lower bounds, these immediately hold for distributed algorithms as well). The nodes communicate with other nodes in discrete rounds by interchanging messages. In every round, an adversary scheduler selects a set of edges between the nodes and every node may communicate with its current neighbors, as selected by the adversary, usually by broadcasting a single message to be delivered to all its neighbors. So, the dynamic topology behaves as a discrete temporal graph where the $i-th$ instance of the graph is the topology selected by the adversary in round i. The main difference, compared to the setting of the previous sections, is that now (in all results that we will discuss in this section, apart from the last one) the topology is

revealed to the algorithms in an online and totally unpredictable way. An interesting special case of temporal graphs consists of those temporal graphs that have *connected instances*.

Feasibility and reusability of solution for the problem *broadcast with termination detection* at the emitter, or TDB, with three metrics, was investigated on three types of the temporal graphs by [9]. TDB requires all nodes to receive a message with some information that initially was held by a single node x called source or emitter, and the source changes to the terminal state after all nodes have received the information, within a finite time. [9] discussed three metrics for TDB problem on the temporal graphs:

- TDB[*shortest*], where each node receives the information within a minimal number of hops from the emitter;
- TDB[*fastest*], where the overall duration between first global emission and last global reception is minimized;
- TDB[*foremost*], where each node receives the information at the earliest possible date following its creation at the emitter.

Feasibility and reliability for recurrent, recurrent bounded, and the periodic temporal graphs were investigated in [9].

2.3 Temporal Vertex Cover

Similar to Activity Timeline here we review another generalization of Vertex Cover problem. In spite of Activity Timeline, which we want to cover all the edges by selecting single intervals for each vertex, in this problem the goal is to cover all the edges by selecting appearance on vertices at specific points of time.

In [1] the complexity of *Temporal Vertex Cover (TVC)* has been investigated.

Let S be a temporal vertex subset of $G = (V, E, \tau)$. Let $e = (u, v) \in E'$ be an edge of the underlying graph $G_{\downarrow} = (V, E')$ and let (w, t) be a vertex appearance in S. We say that vertex w covers the edge e if $w \in \{u, v\}$, i.e. w is an endpoint of e; in that case, edge e is covered by vertex w. Furthermore we say that the vertex appearance (w, t) temporally covers the edge e if w covers e and $t \in \lambda(e)$, i.e. the edge e is active during the time slot t; in that case, edge e is temporally covered by

the vertex appearance (w, t). We now introduce the notion of a temporal vertex cover and the optimization problem Temporal Vertex Cover [1].

Definition 15. [1] *Let $G = (V, E, \tau)$ be a temporal graph. A temporal vertex cover of G is a temporal vertex subset $S \subseteq \{(v, t) : v \in V, t \in [\tau]\}$ of G such that every edge $e \in E$ is temporally covered by at least one vertex appearance (w, t) in S.*

In Theorem 1 and 2, [1] proved the hardness results for TVC on star temporal graphs (i.e. when the underlying graph G is a star), which is in wide contrast to the (trivial) solution of Vertex Cover on a static star graph. The hardness results are obtained via reductions to the problems Set Cover and Hitting Set, respectively. On the positive side they prove in [1] that, in general temporal graphs, TVC can be approximated within a factor of $H_{n-1} - \frac{1}{2} \approx \ln v$, via a reduction to Set Cover.

Theorem 1. [1] *TVC on star temporal graphs is **NP**-complete. Furthermore, for any $\epsilon > 0$, TVC on star temporal graphs does not admit any polynomial-time $(1 - \epsilon) \ln n$-approximation algorithm, unless **NP** has $n^{O(\log \log n)}$-time deterministic algorithms.*

Theorem 2. [1] *For every $\epsilon < 1$, TVC on star temporal graphs cannot be optimally solved in $O(2^{\epsilon T})$ time, unless the Strong Exponential Time Hypothesis (SETH) fails.*

2.4 Reducing Reachability in Temporal Graphs

In Temporal Separator problems the goal is disconnecting terminal from source. Whereas, another approach is to reduce the number of reachable vertices from the given source. Here we review a problem that wants to reduce the number of reachable vertices by removing edges in temporal graph.

Enright et al. in [15] 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. [30].

Given a temporal graph $G = (V, E, \tau)$ with underlying graph $G_{\downarrow} = (V, A)$. For a subset $A' \subseteq A$, it is denoted by $G \backslash A'$ the temporal graph G', where $G'_{\downarrow} = (V, A \backslash A')$. Similarly, given a subset $E \subseteq E'$ of time edges, it is denoted by $G \backslash E$ the temporal graph $G = (V, E \backslash E', \tau)$. Furthermore, a vertex v is temporally reachable from u in G if there exists a temporal path from u to v.

Definition 16. [15] *The temporal reachability set of a vertex u, denoted by $reach_{G,u}$, is the set of vertices which are temporally reachable from vertex u. The temporal reachability of u is the number of vertices in $reach_{G,u}$.*

Temporal Reachability Edge Deletion (TR Edge Deletion)
Input: A temporal graph $G = (V, E, \tau)$ where $G_{\downarrow} = (V, A)$ is the underlying graph of G, and $k, h \in \mathbb{N}$ Output: Is there a set $A' \subseteq A$, with $|A'| \leq k$, such that the maximum temporal reachability of $G \backslash A'$ is at most h?

Temporal Reachability Time-Edge Deletion (TR Time-Edge Deletion)
Input: A temporal graph $G = (V, E, \tau)$, and $k, h \in \mathbb{N}$. Output: Is there a set E' of time-edges, with $|E'| \leq k$, such that the maximum temporal reachability of $G \backslash E'$ is at most h?

Enright et al. [15], show that TR Edge Deletion and TR Time-Edge Deletion problems is **NP**-Complete, and also the TR Time-Edge Deletion problem is $W[1]$-hard when parameterized by the number of time edges that can be removed.

Theorem 3. [15] *TR Edge Deletion and TR Time-Edge Deletion are **NP**-complete, even when the maximum temporal reachability h is at most 7 and the input temporal graph $G = (V, E, \tau)$ has:*

1. *maximum temporal total degree Δ_G at most 5, and*
2. *lifetime at most 2.*

Theorem 4. [15] *TR Edge Deletion (resp. TR Time-Edge Deletion) is $W[1]$-hard when parameterized by the maximum number k of edges (resp. time-edges) that can be removed, even when the input temporal graph has the lifetime 2.*

Next, they show that both TR Edge Deletion and TR Time-Edge Deletion admit an FPT algorithm, when simultaneously parameterized by h, the maximum temporal total degree Δ_G of $G = (V, E, \tau)$, and the treewidth $tw(G_{\downarrow})$ of the underlying graph G.

Although it is **NP**-hard to determine the treewidth of an arbitrary graph [5], the problem of determining whether a graph has treewidth at most w (and constructing such a tree decomposition if it exists) can be solved in linear time for any constant w [8]; note that this running time depends exponentially on w [15].

2.5 Temporal Separators

Maximum Flow problems involve finding a feasible flow through a flow network that obtains the maximum possible flow rate.

The Maximum Flow problem was first formulated in 1954 by T. E. Harris and F. S. Ross as a simplified model of Soviet railway traffic flow [28, 44, 26].

In 1955, Lester R. Ford, Jr. and Delbert R. Fulkerson created the first known algorithm, the Ford–Fulkerson algorithm [23, 24]. In their 1955 paper [23], Ford and Fulkerson wrote that the problem of Harris and Ross is formulated as follows (see [44] p. 5):

Over the years, various improved solutions to the Maximum Flow problem were discovered, notably the shortest augmenting path algorithm of Edmonds and Karp and independently Dinitz; the blocking flow algorithm of Dinitz; the push-relabel algorithm of Goldberg and Tarjan; and the binary blocking flow algorithm of Goldberg and Rao. The algorithms of Sherman [45] and Kelner, Lee, Orecchia, and Sidford [29, 31] respectively, find an approximately optimal Maximum Flow but only work in undirected graphs. In 2013 James B. Orlin published a paper describing an $O(nm)$ algorithm for all values of n and m, where n is the number of vertices in the graph and m is the number of edges of the graph [39].

The max-flow min-cut theorem states that in a flow network, the maximum amount of flow passing from the source to the sink is equal to the total weight of the edges in a minimum cut, i.e., the smallest total weight of the edges, which if removed would disconnect the source from the sink.

In the Min-Cut problem, the goal is to remove some edges such that the source and the terminal are separated from each other (there is no path from the source to the terminal). This problem can be transformed into a new problem with the same goal. In this version, we intend to remove a subset of vertices such that the source and terminal are separated from one another. This new problem is known as Vertex Separator. In this section we review some results on Vertex Separator in temporal graphs.

One of the most important problems regarding temporal separators has been discussed in [51].

Another interesting thing is that reachability in graph G under journeys corresponds to (path) reachability in G'' so that we can use BFS on G'' to answer questions

about foremost journeys in G. Fortunately, the above important negative result concerning Menger's theorem has a turnaround. In particular, it was proved in [35] that if one reformulates Menger's theorem in a way that takes time into account then a very natural temporal analogue of Menger's theorem is obtained, which is valid for all (multi-labeled) temporal networks. The idea is to replace in the original formulation node-disjointness by node departure time disjointness (or out-disjointness) and node removals by node departure times removals. When it is said that the node departure time (u, t) is removed, we mean that we remove all edges leaving u at time t, i.e. we remove label t from all (u, v) edges $(\forall v \in V)$. So, when we ask "how many node departure times are needed to separate two nodes s and z?" we mean how many node departure times must be selected so that after the removal of all the corresponding time-edges the resulting temporal graph has no (s, z)-journey (note that this is a different question from how many time-edges must be removed and, in fact, the latter question does not result in a Menger's analogue). Two journeys are called out-disjoint if they never leave from the same node at the same time [36].

Theorem 5 (Menger's Temporal Analogue)**.** [35] *Take any temporal graph $\lambda(G)$, where $G = (V, E)$, with two distinguished nodes s and z. The maximum number of out-disjoint journeys from s to z is equal to the minimum number of node departure times needed to separate s from z.*

A central contribution in [51] is to prove that both $(s, z)-$Temporal Separator and Strict (s, z)-Temporal Separator are **NP**-complete for all $\tau \geq 2$ and $\tau \geq 5$, respectively, strengthening a result by Kempe et al. [30] (they show **NP**-hardness of both variants for all $\tau \geq 12$) [51].

Lemma 1. [51] *Let $G = (V, E, \tau)$ be an instance of (Strict) (s, z)-Temporal Separator. There is an algorithm which computes in $O(|E|)$ time an equivalent instance $(G' = (V, E', \tau')$ of Strict (s, z)-Temporal Separator, where $\tau' \leq |E'|$.*

Lemma 2. [51] *There is a linear-time computable many-one reduction from Strict (s, z)-Temporal Separator to (s, z)-Temporal Separator that maps any instance $G = (V, E, \tau)$ to an instance $G' = (V', E', \tau')$ with $\tau' = 2\tau$.*

Zschoche et al. [51] investigate the complexity of (s, z)-Temporal Separator for temporal graph $G = (V, E, \tau)$ where τ is a small number.

Theorem 6. [51] *(s, z)-Temporal Separator is **NP**-complete for every maximum label $\tau \geq 2$ and Strict (s, z)-Temporal Separator is **NP**-complete for every $\tau \geq 5$. Moreover, both problems are $W[1]$-hard when parameterized by the solution size k.*

Theorem 7. [51] *Strict (s, z)-Temporal Separator for maximum label $\tau \leq 4$ can be solved in $O(k|E|)$ time, where k is the solution size.*

Zschoche et al. [51] showed the following corollary using *Length Bounded* (s, z)-*Separators* on planar graphs.

Corollary 1. [51] *Both (s, z)-Temporal Separator and Strict (s, z)-Temporal Separator on planar temporal graphs are **NP**-complete.*

In their other work [22], they show that (s, z)-Temporal Separator remains **NP**-complete on many restricted temporal graph classes.

- (s, z)-Temporal Separator remains **NP**-complete on 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. However, if the underlying graph has bounded treewidth, then (s, z)-Temporal Separator becomes polynomial-time solvable.
- (s, z)-Temporal Separator remains **NP**-complete on temporal graphs where each layer contains only one edge. In contrast, if we require each layer to be a unit interval graph and impose suitable restrictions on how the intervals may change over time, then (s, z)-Temporal Separator becomes tractable.
- Regarding temporal graph classes defined solely by restrictions on how the edge sets of the layers may change over time, (s, z)-Temporal Separator becomes solvable in polynomial time on temporal graphs where one layer contains all others (grounded), on graphs where all layers are identical (1-periodic or 0-steady), or when the number of periods is at least the number of vertices. In all other considered cases (s, z)-Temporal Separator remains **NP**-complete

It's 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.

Lemma 3. [22] *Given a temporal graph $G = (V, E, \tau)$ and two distinct vertices s and z, a temporal (s, z)-path can be computed in $O(|E|)$ time.*

Lemma 4. [22] *(s, z)-Temporal Separator is solvable in $O(l^k|E|)$ time, and thus 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.*

They have investigated this problem on temporal graphs with bounded treewidth.

Theorem 8. [22] *For a given tree decomposition of the underlying graph, one can solve (s, z)-Temporal Separator in $O((\tau + 2)^{tw(G_\downarrow)+2}.tw(G_\downarrow).|V|.|E|)$ time, where τ is the maximum time label.*

Later on their paper, they considered restrictions on the layers and the underlying graph. They study temporal graph classes whose definitions do rely on the order of the layers. Herein, they discuss *monotone*, *periodic*, *consecutively connected*, and *steady* temporal graphs.

Intuitively, a temporal graph is p-monotone if it can be decomposed into p time intervals in each of which the layers are ordered by inclusion [22].

Definition 17. [22] *A temporal graph $G = (V, E, \tau)$ is p-monotone if $p \in N$ is the smallest number such that there are $1 = i_1 < i_2 < \cdots < i_{p+1} = \tau$ such that for all $l \in [p]$*

- *$E_j \subseteq E_{j+1}$ for all $i_l \leq j < i_{l+1}$, or*
- *$E_j \supseteq E_{j+1}$ for all $i_l \leq j < i_{l+1}$*

holds.

They present a set of interesting results for periodic temporal graphs as well.

Lemma 5. [22] *Let $G = G'^r$ be a p-periodic temporal graph such that the number of periods r is at least the distance to temporality from s to z in G'. Then (s, z)-Temporal Separator is solvable in $O(k|E|)$ time, where k is the solution size and $|E|$ the number of time-edges.*

Corollary 2. [22] *Let $G = (V, E, \tau)$ be a p-periodic temporal graph. If the number of periods $r \geq |V|$, then (s, z)-Temporal Separator is solvable in $O(k|E|)$ time, where k is the solution size and $|E|$ the number of time-edges.*

Fluschnik et al. in [22] studied temporal separator on T-interval connected temporal graphs.

Definition 18. [32], [22] *A temporal graph $G = (V, E, \tau)$ is T-interval connected for $T \geq 1$ if for every $t \in [\tau - T + 1]$ the static graph $G := (V, \bigcap_{i=t}^{t+T-1} E_i(G))$ is connected.*

Observation 1. [22] *There is a polynomial-time many-one reduction that maps any instance $(G = (V, E, \tau), s, z, k)$ of (s, z)-Temporal Separator to an equivalent instance $(G' = (V', E', \tau), s, z, k + 1)$ such that G' is T-interval connected for every $T \geq 1$.*

Next they move to steady temporal graphs. Steady temporal graphs present a class of graphs in which we do not expect very big changes over the time.

Definition 19. [22] *A temporal graph $G = (V, E, \tau)$ is λ-steady if $\lambda \in N$ is the smallest number such that for each point in time $t \in [\tau - 1]$ the size of the symmetric difference of two consecutive edge sets $|E_t \Delta E_{t+1}|$ is at most λ.*

Corollary 3. [22] *For any fixed λ we have that (s, z)-Temporal Separator on λ-steady temporal graphs is fixed-parameter tractable when parameterized by the maximum label τ.*

In next part we present the work of Fluschnik et al. at [22], section 6. They combine the two aspects studied in previous sections. To this end, they focus on temporal graphs where each layer is a unit interval graph and we further restrict how much the intervals may change over time. This is a layer-wise restriction with, additionally, a temporal restriction. Recall that (s, z)-Temporal Separator remains **NP**-complete on temporal graphs where each layer is a unit interval graph, even if the maximum label τ is a small constant.

In the following they show that if there is an ordering on the vertices that matches the relative positions of the intervals in all layers, then we can solve (s, z)-Temporal Separator in polynomial time [22].

A total ordering $<_V$ on a vertex set V is called compatible with a unit interval graph $G = (V, E)$ if there are unit intervals $[a_v, a_v + 1]$ with $a_v \in R$ for all vertices $v \in V$ that induce the graph G and for all $u, v \in V$ with $u <_V v$ we have that $a_u \leq a_v$. Note that for every unit interval graph there is a total ordering on the vertices that is compatible with it [22].

Definition 20. [22] *A temporal graph $G = (V, E, \tau)$ is an order-preserving temporal unit interval graph if G is a temporal unit interval graph and there is a total ordering $<_V$ on the vertex set V that is compatible with every layer G_i.*

Lemma 6. [22] *Order-preserving temporal unit interval graphs can be recognized in polynomial time and a compatible vertex ordering for a given order-preserving temporal unit interval graph can be computed in polynomial time.*

Lemma 7. [22] *Let $G = (V, E, \tau)$ be an order-preserving temporal unit interval graph with ordering $<_V$.*

(i) For all $1 \leq a \leq b \leq \tau$ and for all $S \subseteq V$ we have that $G_{[a:b]} - S$ is also an order-preserving temporal unit interval graph.

(ii) If for some $1 \leq i < j \leq n$ there is a temporal (v_i, v_j)-path P in G, then there is temporal (v_i, v_j)-path P' in G that visits its vertices in the order given by $<_V$.

(iii) Let $S \subseteq V$ be a temporal (v_i, v_j)-separator in G for some $1 \leq i < j \leq n$. Then $S' := S \backslash (V_{<i} \cup V_{>j})$ is also a temporal (v_i, v_j)-separator in G.

(iv) A temporal (v_i, v_j)-separator in G is also a temporal $(v_{i'}, v_{j'})$-separator in G for all $1 \leq i' \leq i < j \leq j' \leq n$.

(v) Let $S \subseteq V \backslash \{s, z\}$ such that v_i is the largest vertex reachable from s in $G - S$. Let t denote the first time v_i is reachable from s in $G - S$, and let $t \leq t' \leq \tau$. Then $N^{>}_{G_{t'}}(vi) \subseteq S$.

(vi) Let $S_1 \subseteq V \backslash \{s, z\}$ such that v_i is the largest vertex reachable from s in $G_{[1:t]} - S_1$ for some $t \in [\tau - 1]$. Let $S_2 \subseteq V \backslash \{s, z\}$ such that v_j is the largest vertex reachable from s in $G_{[t+1:\tau]} - S_2$. If $i \leq j$, then $S := S_1 \cup S_2$ is a temporal (s, z)-separator in G such that there is no vertex reachable from s in $G - S$ that is larger than v_j.

(vii) Let $S \subseteq V$ be an inclusion-wise minimal temporal (s, z)-separator in G with the property that a given v_i is the largest vertex that is reachable from s in $G - S$ and let v_j be the smallest vertex that is not in S such that S is also a temporal (s, v_j)-separator in G. Then for all $v_i <_V v <_V v_j$ with $v_i \neq v \neq v_j$ we have that $v \in S$, and we have that $S \cap V_{>j} = \varnothing$.

Theorem 9. [22] *(s, z)-Temporal Separator on order-preserving temporal unit interval graphs is solvable in $O(|V|^2.\tau^2)$ time.*

They also studied temporal separator on temporal unit interval graph.

Definition 21. [22] *(Shuffle Number). Given a temporal unit interval graph $G = (V, E, \tau)$, its shuffle number $\mathcal{K}$ is the smallest integer such that there are vertex orderings $<_V^1, <_V^2, \ldots, <_V^\tau$ with the property that $<_V^t$ is compatible with layer G_t for all $t \in [\tau]$, and the orderings of any two consecutive layers have Kendall tau distance at most $\mathcal{K}$, that is, for all $t \in [\tau - 1]$ we have that $K(<_V^t, <_V^{t+1}) \leq \mathcal{K}$. We say that the vertex orderings $<_V^1, <_V^2, \ldots, <_V^\tau$ witness the shuffle number of G.*

Theorem 10. [22] *Given the a temporal unit interval graph and a vertex orderings that witness its shuffle number $\mathcal{K}$, (s, z)-Temporal Separator is fixed-parameter tractable when parameterized by $\mathcal{K} + \tau$, where τ is the maximum label.*

2.6 Activity Timeline

In this section we discus the problem of determining when entities are active based on their interactions with each other. Rozenshtein et al. [43] have studied this problem in details. They consider a set of entities V and a sequence of time-stamped edges E among the entities. Each edge $(u, v, t) \in E$ denotes an interaction between entities u and v that takes place at time t.

Proposition 1. [43] *The decision version of the MinTimeline problem is **NP**-complete. Namely, given a temporal network $G = (V, E)$ and a budget l, it is **NP**-complete to decide whether there is timeline $\varphi = \{I_u\}_{u \in V}$ that covers G and has $S(\varphi) \leq l$.*

Proposition 2. [43] *Consider a maximal solution α_e to the dual program. Define a set of intervals $\varphi = \{I_v\}$ by $I_v = [min\ X_v,\ max\ X_v]$, where*

$$X_v = \{m_v\} \cup \{t \in T(v) | h(v, t, m_v) = |t - m_v|\}$$

Then φ is a 2-approximation solution for the problem $MinTimeline_m$.

Proposition 3. [43] *$MinTimeline_\infty$ can be solved in a polynomial time.*

Chapter 3

Temporal Separator

Although Vertex Separator could be solved in polynomial time, the Temporal Separator problem is one of the hardest problems. Zschoche et al. [51] investigate (s, z)-Temporal Separator and strict (s, z)-Temporal Separator on different types of temporal graph. Here in this work, we investigate the problem called (s, z, t)-Temporal Separator.

Checking if there is (s, z, t)-temporal path is solvable in polynomial time, since finding the fastest temporal path is solvable in polynomial time [50, 49, 48]. This will help us to give a polynomial-time algorithm for some family of graphs in the section 3.5.

Here in this Chapter, we refer to $V(G)$ or $E(G)$ as the set of vertices and the edges, respectively, of a temporal graph or a static graph G. Also for any subset $U \subseteq V(G)$ we refer $E(U)$ to the set of all edges in the subgraph induced by U, and for any node $v \in V$ we refer $E(v)$ to the set of all edges that has incident to the node v.

Lemma 8. *Given a temporal graph $G = (V, E, \tau)$ and two distinct vertices s and z as well as integer variable t, it is computable in time $O(|S||E|)$ to decide if there is (s, z, t)-temporal path in G where $S = \{t | \exists u : (s, u, t) \in E\}$.*

Proof. [48] and [50] present an algorithm that computes the fastest path from a single source s to all of the vertices in $O(|S|(|V|+|E|))$. We could ignore the isolated vertices so the factor of $|V|$ is in $O(|E|)$, then we could compute the fastest path from s to z in G and check that if it is greater than t or not. □

Here in this chapter first we will look at (s, z, t)-Temporal Separator on temporal graph $G = (V, E, \tau)$ for small t and τ. Then we will look at the approximation algorithm and lower bound for this problem. We also investigate (s, z, t)-Temporal Separator problem on temporal graphs that have bounded treewidth. Finally, we examine the problem on a temporal graph with restrictions on the underlying graph.

3.1 (s, z, t)-Temporal Separator with Small t

Zschoche et al. [51] shows that (s, z)-Temporal Separator problem is **NP**-Complete on a temporal graph $G = (V, E, \tau)$ if $\tau \geq 2$, and Strict (s, z)-Temporal Separator is **NP**-Complete on the temporal graph G if $\tau \geq 5$. So, it is obvious that the problem (s, z, t)-Temporal Separator is **NP**-Complete if $t \geq 2$, and Strict (s, z, t)-Temporal Separator is **NP**-Complete if $t \geq 5$.

Reduction from 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 variable in the i-th equation. So, $(s, z, 1)$-Temporal Separator on temporal graphs with a sufficient number of layers is **NP**-Complete. However, the solvability or complexity of (s, z, t)-Temporal Separator on temporal graphs with a small number of layers is not trivial. Here we show that $(s, z, 1)$-Temporal Separator remains **NP**-Complete on temporal graph $G = (V, E, \tau)$ if τ is equal to 2. To do that, we need to introduce a version of Separator in which the goal is to disconnect all the vertices for a set of terminals given as the input of the problem.

Problem 6 (Node Multiway Cut)**.**

- ***Instance****: A graph $G = (V, E,)$ and a set of terminal vertices $Z = \{z_1, z_2, \ldots z_k\}$.*
- ***Solution****: A set of multiway cut $S \in V \backslash Z$ which removal of set S from graph G disconnect all 2 distinct terminal z_i and z_j.*
- ***Measure****: Minimize cordiality of set S*

Node Multiway Cut problem is **NP**-Complete for $k \geq 3$ [25].

Theorem 11. *$(s, z, 1)$-Temporal Separator problem is **NP**-Complete on a temporal graph $G = (V, E, \tau)$ if $\tau \geq 2$.*

Proof. For a given graph H and three vertices z_1, z_2,and z_3 we construct a temporal graph $G = (V, E, 2)$. Let $V = (V(H) \backslash \{z_1, z_2, z_3\}) \cup \{s, z\}$ and for all the edges in H,

not incident to any terminal node equal to (u, v), add 2 edges $(u, v, 1)$ and $(u, v, 2)$ to set of edges E. For all the vertices u which is a neighbour of vertex z_1 add an edge $(s, u, 1)$ and for all vertices v which is neighbour of node z_2 or neighbour of node z_3 add an edge $(v, z, 1)$ to the set of edges E. Finally add $(s, u, 2)$ for all neighbours of vertex z_2, as well as $(v, z, 2)$ for all neighbours of vertex z_3 to the set of edges. We claim that a set of $S \in V \backslash \{s, z\}$ is a $(s, z, 1)$-temporal separator if and only if S is a set of multiway cut for H.

$\leftarrow$ Suppose that S is a set of multiway cut in the graph H and S is not a set of $(s, z, 1)$-temporal separator on the temporal graph G. So, there is a $(s, z, 1)$-temporal path P which $V(P) \in V \backslash S$. Based on the definition of $(s, z, 1)$-temporal path, either all the edges in the path belong to layer G_1 or all of them belong to layer G_2. Let's consider each case separately.

- **Case 1**. Consider a temporal path P where all the vertices belong to layer G_1. Suppose that path P starts with the edge $(s, u, 1)$, and ends with the edge $(v, z, 1)$. Based on the construction of graph G, it is clear that u is a neighbour of vertex z_1 and v is a neighbour of vertex z_2 or z_3. Due to the fact that all the edges in graph G that are not incidents to s and z, also appears in graph H, all the edges excepts starting and ending edge in path P appear in the graph H. Construct a new path P' by replacing vertex s with z_1 and vertex z with z_2 or z_3 that is adjacent to v. There is no vertex $x \in P$ such that $x \in S$, so all the vertices in path P' do not appear in S then $V(P') \subseteq V(H) \backslash S$ and this contradicts with the assumption of S being a multiway cut.
- **Case 2**. Consider a temporal path P where all the vertices belong to layer G_2. Similarly suppose that path P starts with the edge $(s, u, 2)$ ends with edge $(v, z, 2)$, the u is neighbour of vertex z_2 and v is neighbour of vertex z_3. Construct a path P' by replacing vertex s with z_2 and vertex z with z_3. So the path P' is in the graph H from vertex z_2 to z_3 and $V(P') \subseteq V(H) \backslash S$ which contradicts with the assumption.

$\rightarrow$ Suppose that S is a $(s, z, 1)$-temporal separator in G and S is not a set of multiway cut in the graph H. So there is a path P between two vertices of z_1, z_2, and z_3 in graph H which $V(P) \in V(H) \backslash S$. By replacing source vertex z_1 or z_2 with s and terminal vertex z_2 or z_3 with z we could construct a path P' in which $V(P) \in V \backslash S$. Now consider three cases for path source and terminal of P, since all the edges in P

except the first and last one does not incident to s or z, all edges in P appeared in both layer G_1 and G_2. Therefore for each case we could conclude that:

- **Case 1**. P is between z_1 and z_2. So, in this case P' is in layer G_1 and it contradicts with the assumption that S is $(s, z, 1)$-temporal separator.
- **Case 2**. P is between z_1 and z_3. So, in this case P' is in layer G_1 and it contradicts with the assumption that S is $(s, z, 1)$-temporal separator.
- **Case 3**. P is between z_2 and z_3. So, in this case P' is in layer G_2 and it contradicts with the assumption that S is $(s, z, 1)$-temporal separator.

□

$(s, z, 1)$-Temporal Separator problem is **NP**-Complete on temporal graphs G where the number of layers is greater than one, also (s, z, t)-Temporal Separator problem is **NP**-Complete if τ is greater than 2 since (s, z)-Temporal Separator is hard on the temporal graph with more than one layer.

Corollary 4. *Finding optimal (s, z, t)-Temporal Separator for temporal graph $G = (V, E, \tau)$ is **NP**-Complete if and only if $\tau \geq 2$.*

Strict (s, z)-Temporal Separator is **NP**-Complete on a temporal graph with more than four layers [51]. By restricting a temporal path to a strict temporal path with temporal distance lower than t, we could define another problem called Strict (s, z, t)-Temporal Separator. Since Strict (s, z)-Temporal Separator is **NP**-Complete on graphs with more than four layers, then it is clear that Strict (s, z, t)-Temporal Separator is **NP**-Complete on a temporal graph with more than four layers and $t \geq 5$. However, by a small change on the reduction presented by Zschoche et al. [51], which is inspired by [49], we show that Strict (s, z, t)-Temporal Separator is **NP**-Complete on a temporal graph with four layers and $t \geq 3$.

Theorem 12. *Given a temporal graph $G = (V, E, \tau)$ and two vertex s and z, finding strict (s, z, t)-temporal separator is **NP**-Complete if $\tau \geq 4$ and $t \geq 3$.*

Proof. We present a reduction from a vertex cover problem to an instance of Strict $(s, z, 3)$-Temporal Separator, which has four layers. Given a graph H, we construct a graph $G = (V, E, 4)$ and instance of input for Strict $(s, z, 3)$-Temporal Separator

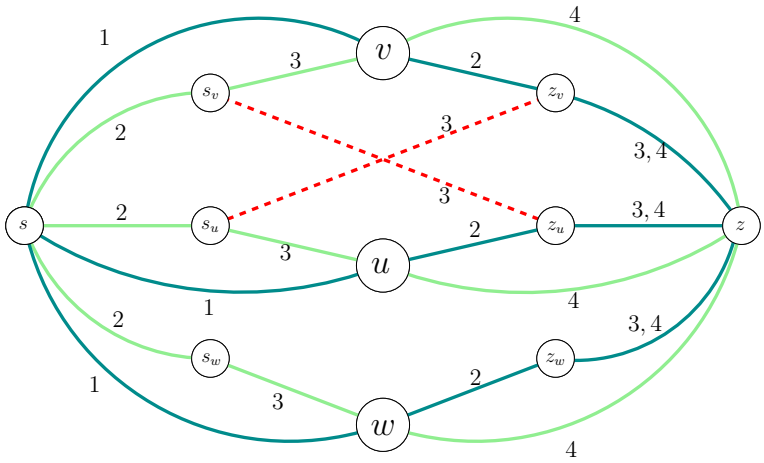

Figure 4: Instance of Strict $(s, z, 3)$-Temporal Separator problem with four layers that corresponds to a vertex cover problem instance.

problem. Let $V = \{s_v, v, z_v | v \in V(H)\} \cup \{s, z\}$ and edge set E equal to:

$$E := \{(s, s_v, 2), (s_v, v, 3), (v, z, 4), (s, v, 1), (v, z_v, 2), (z_v, z, 3), (z_v, z, 4) | v \in V(H)\} \cup$$
$$\{(s_u, z_v, 3), (s_v, z_u, 3) | (u, v) \in E(H)\}$$

Figure 4 shows the structure of the temporal graph G. Let $n = |V(H)|$; we claim that there is a vertex cover set in H with size k, if and only if there exists a set of strict $(s, z, 3)$-temporal separator in G with cardinality lower than or equal to $n + k$.

$\rightarrow$ Consider a set $C \in V(H)$, be a vertex cover with size k in H, then let $S = \{v | v \in V(H) \backslash C\} \cup \{s_v, z_v | v \in V(H)\}$. Assume that there is a strict $(s, z, 3)$-temporal path P in which all the vertices belong to $V \backslash S$. Since for every $v \in V(H)$ either $v \in S$ or $\{s_v, z_v\} \subseteq S$, temporal path P is like the following line:

$$P = (s, s_u, 2), (s_u, z_v, 3), (z_v, z, 4)$$

Which implies the existence of edge $(s_u, z_v, 3)$ in G, that results in $(u, v) \in E(H)$. On the other hand existence of vertex s_u and z_v in the path P implies that $\{u, v\} \subseteq V(H) \backslash C$ which contradicts with the fact of covering all the edges in $E(H)$ by vertex cover C. So, there is no $(s, z, 3)$-temporal path in induced temporal graph G by $V \backslash S$. The cardinality of set S which is a strict $(s, z, 3)$-temporal separator for temporal graph G is equal to $(n - k) + 2k$.

$\leftarrow$ Let $S \in V$ be a strict $(s, z, 3)$-temporal separator in which $|S| = n + k$. For any vertex $v \in V(H)$ we claim that either $v \in S$ or $\{s_v, z_v\} \subseteq S$, otherwise one of

the two strict $(s, z, 3)$-temporal path P_1 and P_2 which are shown in equation 3 and 4 respectively, will not be removed from graph G by removing set S.

$$P_1 = (s, s_v, 2), (s_v, v, 3), (v, z, 4) \tag{3}$$

$$P_2 = (s, v, 1), (v, z_v, 2), (z_v, z, 3) \tag{4}$$

Now we construct a set $C \in V(H)$ by the following choices for each vertex $v \in V(H)$.

- If more than one vertex of three vertices s_v, v, and z_v belong to S, then add v to C.
- If only one vertex from three vertex s_v, v, and z_v is belonged to S, then don't add v to C.

First, base on the fact that at least one of the vertex, the three vertices s_v, v, and z_v belong to S, it is clear that $|C| \leq k$. Second, if there is an edge in $(u, v) \in E(H)$ such that $\{u, v\} \subseteq V(H) \backslash C$, concerning the previous claims, it results in both path P_3 and P_4 which are shown in the equation,5 and 6 respectively will present in a temporal subgraph induced by $V \backslash S$. Therefore C is a vertex cover with cardinality lower than or equal to k.

$$P_3 = (s, s_v, 2), (s_v, z_u, 3), (z_u, z, 4) \tag{5}$$

$$P_4 = (s, s_u, 2), (s_u, z_v, 3), (z_v, z, 4) \tag{6}$$

□

Since every temporal path from s to z contains more than two edges, then $\varnothing$ is a strict $(s, z, 1)$-temporal separator. Also, all strict $(s, z, 2)$-temporal path is like $(s, v, t), (v, z, t + 1)$ and it is clear that $v \in S$ for all sets of S which is a strict strict $(s, z, 2)$-temporal separator. So, Strict $(s, z, 2)$-Temporal Separator problem could be solve in polynomial time easily. Strict (s, z, t)-Temporal Separator problem on graph $G = (V, E, \tau)$ such that $\tau = t$ is equal to the Strict (s, z)-Temporal Separator. Therefore, in case that $\tau = t = 3$ this problem is equal to Strict (s, z)-Temporal Separator which $\tau = 3$. Zschoche et al. [51] present a polynomial time algorithm to finding minimum strict (s, z)-temporal separator on temporal graph $G = (V, E, \tau)$ which $\tau < 5$. So, this case could be solve in polynomial time. Although we know that finding strict strict (s, z, t)-temporal separator on temporal graph $G = (V, E, 3)$ is polynomial time solvable with the algorithm which is represented in [51], we provide a simple algorithm to solve this problem.

In the first step of the algorithm, we check if there is an edge between s and t, then it is clear that there does not exist any separator sets, because with removing any nodes from the graph, the direct path with using this edge from s to z will remain.

Next, for every temporal path from s to z with length two, such as $(s, x, t_1), (x, z, t_2)$ with $t_1 < t_2$, it is clear that we have to remove node 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 like the following path.

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

Now, put every node x which there exists edge from s to x with time label one into the set X, also put every node y which there exists edge from y to z with time label 3 into set Y. Now, It is clear that $X \cap Y = \varnothing$. because if not, so there exists a node u, which there should exist two edges $e_1 = (s, u, 1)$ and $e_2 = (u, z, 3)$, while this node should be removed in the last step. Therefore every strict temporal path from s to z should have corresponding edge $(x, y, 2)$ which $x \in X$ and $y \in Y$. So, we should remove either x or y for every edge $(x, y, 2)$, which $x \in X$ and $y \in Y$. In order to do this we could use vertex cover problem in bipartite graph which is solvable in polynomial time.

Corollary 5. *Finding minimum strict (s, z, t)-temporal separator on graph $G = (V, E, \tau)$ is **NP**-Complete if and only if $\tau \geq 4$ and $t \geq 3$.*

3.2 Approximation of Temporal Separator Problems

Here in this section, we investigate an approximation algorithm to find optimal (s, z)-temporal separator and (s, z, t)-temporal separator.

Theorem 13. *let $G = (V, E, \tau)$ be a temporal graph. There exists a τ-approximation algorithm that finds optimal (s, z)-temporal separator in polynomial time.*

Proof. To prove this theorem, we introduce function F from an undirected temporal graph to a directed graph. Then we show that for any separator in $F(G)$, there is a

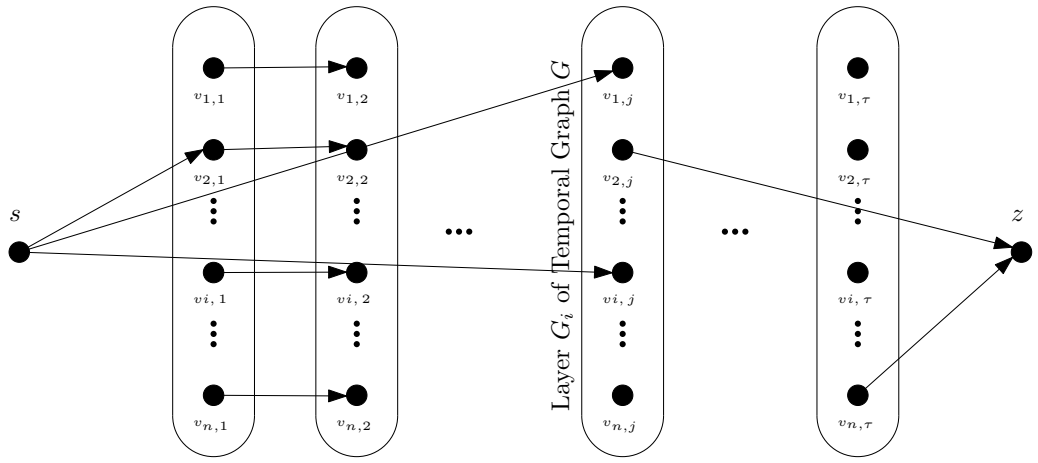

Figure 5: Example of a directed graph $F(G)$. For simplicity of presentation, edges in layer G_i are not drawn.

(s, z)-temporal separator with a cardinality of at most the separator set's cardinality. Also we show that for any set of (s, z)-temporal separator S, there exists a separator S' in $F(G)$ such that $|S'| \leq \tau|S|$.

For temporal graph $G = (V, E, \tau)$, the directed graph $H = F(G) = (V', E')$ is defined as follows. The set of vertex in H is the union of τ set of disjoint vertices $V_1, V_2, \ldots, V_\tau$ and $\{s, z\}$, where:

$$\forall i \in [\tau] : V_i = \{v_{j,i} | v_j \in V \backslash \{s, z\}\} \cup \{s, z\}$$

And the edges of H can be defined in four sets.

- For all edge $(v_i, v_j, t) \in E$ we add an edge $(v_{i,t}, v_{j,t})$ and $(v_{j,t}, v_{i,t})$ to the edge set E'.
- For all vertex v_i and time $t \in [\tau - 1]$ we add an edge $(v_{i,t}, v_{i,t+1})$ to edge set E'.
- For all edge (s, v_i, t) we add an edge $(s, v_{i,t})$ to edge set E'.
- For all edge (z, v_i, t) we add an edge $(v_{i,t}, z)$ to edge set E'.

Figure 5shows a sample for graph $F(G)$.

Lemma 9. *Let $G = (V, E)$ be a temporal graph and $H = (V', E')$ be a static graph that $H = F(G)$ and $V = V_1 \cup V_2 \cup \cdots \cup V_\tau \cup \{s, z\}$. For any (s, z)-temporal separator S in G, there exists a (s, z)-separator S' in H such that $|S'| \leq \tau|S|$.*

Proof. Suppose that S' is equal to $\bigcup_{v_i \in V} \bigcup_{j \in [\tau]} \{v_{i,j}\}$. It is clear that $|S'| \leq \tau |S|$, so to prove this lemma it is sufficient to show that S' is a set of (s, z)-temporal separator in H. Suppose that it is not, so there exists a path P' from s to z in subgraph of H induced by $V' \backslash S'$. It is clear that P' is as follows:

$$P' = (s, v_{i_1,t_1}), (v_{i_1,t_1}, v_{i_2,t_1}), \dots (v_{i_{k_1-1},t_1}, v_{i_{k_1},t_1}), (v_{i_{k_1},t_1}, v_{i_{k_1},t_2}),$$
$$(v_{i_{k_1},t_2}, v_{i_{k_1+1},t_2}) \dots (v_{i_{k_2-1},t_2}, v_{i_{k_2},t_2}), (v_{i_{k_2},t_2}, v_{i_{k_2},t_3}),$$
$$\dots$$
$$(v_{i_{k_{r-1}},t_r}, v_{i_{k_{r-1}+1},t_r}) \dots (v_{i_{k_r-1},t_r}, v_{i_{k_r},t_r}), (v_{i_{k_r},t_r}, z)$$

Where $t_1 < t_2 < \dots t_r$. So, consider the path P which is mentioned in the following:

$$P = (s, v_{i_1}, t_1), (v_{i_1}, v_{i_2}, t_1), \dots (v_{i_{k_1-1}}, v_{i_{k_1}}, t_1)$$
$$(v_{i_{k_1}}, v_{i_{k_1+1}}, t_2) \dots (v_{i_{k_2-1}}, v_{i_{k_2}}, t_2),$$
$$\dots$$
$$(v_{i_{k_{r-1}}}, v_{i_{k_{r-1}+1}}, t_r) \dots (v_{i_{k_r-1}}, v_{i_{k_r}}, t_r), (v_{i_{k_r}}, z, t_r)$$

Based on the fact that $V(P') \in V' \backslash S'$, it is clear that $V(P) \in V \backslash S$ and also it is clear that P is a temporal path. It contradicts that S is a set of (s, z)-temporal separator. Therefore there is no path such $V(P') \in V' \backslash S'$ and S' is a separator in directed graph H. □

Lemma 10. *Let $G = (V, E, \tau)$ be a temporal graph and $H = (V', E')$ be a static graph that $H = F(G)$ and $V = V_1 \cup V_2 \cup \dots \cup V_\tau \cup \{s, z\}$. For any (s, z)-separator S' in H, there exists a (s, z)-temporal separator S in G such that $|S| \leq |S'|$.*

Proof. Let S be equal to $\bigcup_{t \in [\tau]} \{v_i | v_{i,t} \in S'\}$. By the definition of S, it is clear that $|S| \leq |S'|$, so to prove the lemma it is sufficient to show that S is a set of (s, z)-temporal separator of G. Suppose that it is not, so there exists a temporal path P from s to z in subgraph of G induced by V S. It is clear that P is as follows.

$$P = (s, v_{i_1}, t_1), (v_{i_1}, v_{i_2}, t_1), \dots (v_{i_{k_1-1}}, v_{i_{k_1}}, t_1)$$
$$(v_{i_{k_1}}, v_{i_{k_1+1}}, t_2) \dots (v_{i_{k_2-1}}, v_{i_{k_2}}, t_2),$$
$$\dots$$
$$(v_{i_{k_{r-1}}}, v_{i_{k_{r-1}+1}}, t_r) \dots (v_{i_{k_r-1}}, v_{i_{k_r}}, t_r), (v_{i_{k_r}}, z, t_r)$$

Where $t_1 < t_2 < \ldots t_r$. So, consider the path P' which is mentioned in the following:

$$P' = (s, v_{i_1,t_1}), (v_{i_1,t_1}, v_{i_2,t_1}), \ldots (v_{i_{k_1-1},t_1}, v_{i_{k_1},t_1}), (v_{i_{k_1},t_1}, v_{i_{k_1},t_2}),$$
$$(v_{i_{k_1},t_2}, v_{i_{k_1+1},t_2}) \ldots (v_{i_{k_2-1},t_2}, v_{i_{k_2},t_2}), (v_{i_{k_2},t_2}, v_{i_{k_2},t_3}),$$
$$\ldots$$
$$(v_{i_{k_{r-1}},t_r}, v_{i_{k_{r-1}+1},t_r}) \ldots (v_{i_{k_r-1},t_r}, v_{i_{k_r},t_r}), (v_{i_{k_r},t_r}, z)$$

Based on the fact that $V(P) \in V \backslash S$, it is clear that $V(P') \in V' \backslash S'$ and also it is clear that P' is a path in subgraph of G induced by $V\ S$. So, its contradict that S' is a set of (s, z)-separator, therefore there is no path such $v(P) \in V \backslash S$ and S is a set of (s, z)-temporal separator in G. □

So, from the Lemma 9 we could conclude that if the minimum cardinality of (s, z)-temporal separator set in G is equal to k, there exists a separator with at most size $k \times \tau$ in the static graph $F(G)$. So, the minimum (s, z)-separator in $F(G)$ is at most with size of $k \times \tau$. therefore, due to the Lemma 10, we could find a (s, z)-temporal separator with a cardinality of at most $k \times \tau$ by finding a minimum separator in the static graph $F(G)$ which is solvable in polynomial time. □

Simply the above algorithm is not working for (s, z, t)-Temporal Separator problem. However, by a simple modification on a function F, we could give a polynomial-time τ^2-approximation by using an approximation algorithm from a problem called *Vertex k-Cut*.

Problem 7 (Vertex k-Cut)**.**

- ***Instance****: Graph $G = (V, E)$, a set $S = \{s_1, t_1, \ldots, s_k, t_k\}$ of special vertices, and a weight function $w : V \backslash S \to N$, and an integer k.*
- ***Solution****: A vertex k-cut, i.e., a subset $C \subseteq V - S$ of vertices such that their deletion from G disconnects each s_i from t_i for $1 \leq i \leq k$*
- ***Measure****: Minimize the sum of the weight of the vertices in the cut, i.e., $\sum_{v \in C} w(v)$.*

Theorem 14. *Finding minimum (s, z, t)-temporal separator is approximable within τ^2*

Proof. Similarly, define a function F from a temporal graph G, which is an instance of problem (s, z, t)-Temporal Separator to a static graph G and set of sources and terminals that is an instance of Vertex k-Cut.

Given a temporal graph $(G) = (V, E, \tau)$, we will construct a static graph $F(G) = H = (V', E')$ such that the set of vertex in H is the union of τ vertex sets $V_1, V_2, \ldots, V_\tau$ and set of sources S and terminals Z, where:

$$\forall i \in [\tau] : V_i = \{v_{j,i} | v_j \in V \backslash \{s, z\}\} \cup \{s, z\}$$
$$S = s_i | i \in [\tau] Z = z_i | i \in [\tau]$$

And the edges of H can be defined in four sets.

- For all edge $(v_i, v_j, t) \in E$ we add an edge $(v_{i,t}, v_{j,t})$ and $(v_{j,t}, v_{i,t})$ to the edge set E'.
- For all vertex v_i and time $t \in [\tau - 1]$ we add an edge $(v_{i,t}, v_{i,t+1})$ to edge set E'.
- For all edge (s, v_i, t) we add an edge $(s_t, v_{i,t})$ to edge set E'.
- For all edge (z, v_i, t) and all integer time $i - t < j \geq i$ such that $j > 0$ we add an edge $(v_{i,t}, z_j)$ to edge set E'.

And the set S and Z are the sources and terminal. So, in this input, k is equal to τ. Figure 6shows a sample for graph $F(G)$, similar to the Theorem 13 it is sufficient to prove the two lemmas with new construction.

Lemma 11. *Let $G = (V, E)$ be a temporal graph and $H = (V', E')$ be a static graph that $H = F(G)$ and $V = V_1, V_2, \ldots, V_\tau \cup S \cup Z$. For any (s, z, t)-temporal separator A in G, there exists a vertex k-cut A' in H such that $|A'| \leq \tau |A|$.*

Proof. Suppose that A' is equal to $\cup_{v_i \in V} \cup_{j=1}^{\tau} \{v_{i,j}\}$. It is clear that $|A'| \leq \tau |A|$, so to prove this lemma it is sufficient to show that A' is a set of vertex k-cut in H. Suppose that it is not, so there exists a path P' from s_i to z_i for specific integer i in subgraph

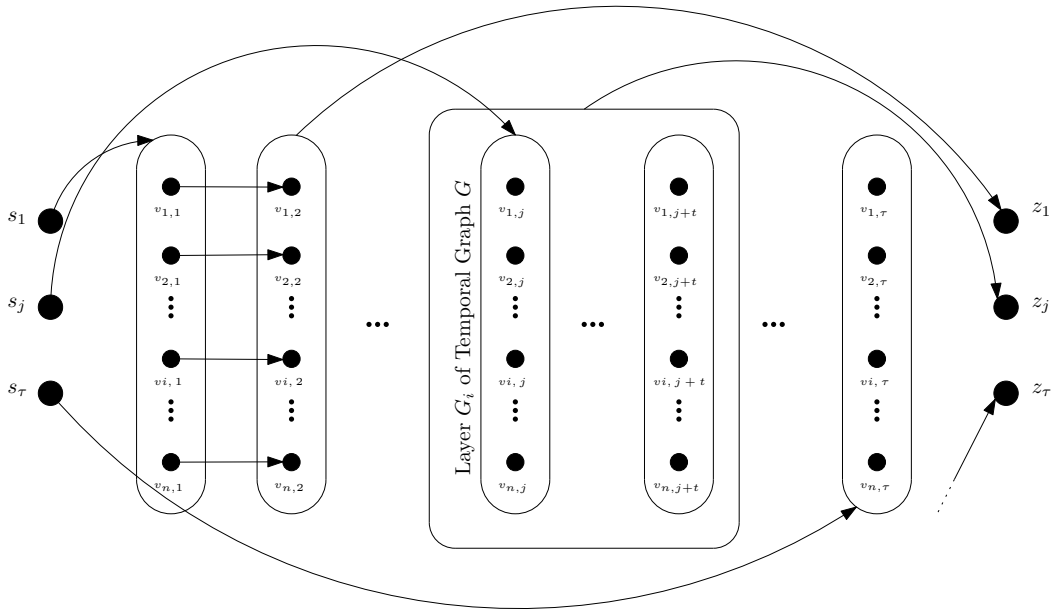

Figure 6: Example of a directed graph $F(G)$ an instance of vertex k-cut. Edges from(to) $s_j(z_j)$ to(from) sets shown by box implies that all the vertices v such that there is an edge from(to) $s_i(z_i)$ to(from) v is belongs to one of this sets, also in figure the edges in the layer G_i is not drawn.

of H induced by $V'\backslash A'$. Let $t_1 = i$, it is clear that P' is as follows:

$$P' = (s_{t_1}, v_{i_1,t_1}), (v_{i_1,t_1}, v_{i_2,t_1}), \dots (v_{i_{k_1-1},t_1}, v_{i_{k_1},t_1}), (v_{i_{k_1},t_1}, v_{i_{k_1},t_2}),$$
$$(v_{i_{k_1},t_2}, v_{i_{k_1+1},t_2}) \dots (v_{i_{k_2-1},t_2}, v_{i_{k_2},t_2}), (v_{i_{k_2},t_2}, v_{i_{k_2},t_3}),$$
$$\dots$$
$$(v_{i_{k_{r-1}},t_r}, v_{i_{k_{r-1}+1},t_r}) \dots (v_{i_{k_r-1},t_r}, v_{i_{k_r},t_r}), (v_{i_{k_r},t_r}, z_{t_1})$$

Where $t_1 < t_2 < \dots t_r$. So, consider the path P which is mentioned in the following:

$$P = (s, v_{i_1}, t_1), (v_{i_1}, v_{i_2}, t_1), \dots (v_{i_{k_1-1}}, v_{i_{k_1}}, t_1)$$
$$(v_{i_{k_1}}, v_{i_{k_1+1}}, t_2) \dots (v_{i_{k_2-1}}, v_{i_{k_2}}, t_2),$$
$$\dots$$
$$(v_{i_{k_{r-1}}}, v_{i_{k_{r-1}+1}}, t_r) \dots (v_{i_{k_r-1}}, v_{i_{k_r}}, t_r), (v_{i_{k_r}}, z, t_r)$$

Based on the fact that $V(P') \in V'\backslash A'$, it is clear that $V(P) \in V\backslash A$ and also it is clear

that P is a temporal path. Also base on the construction of graph H, there is edges from vertex $v_{i,j}$ to z_k only if $j - t < k \leq j$, therefore $t_r - t < t_1 \leq < t_r \Rightarrow t_r - t_1 < t$ results which path P is a (s, z, t)-temporal path. So, it contradicts that A is a set of (s, z, t)-temporal separator, therefore there is no path such $V(P') \in V' \backslash A'$ and A' is a separator in graph H. □

Lemma 12. *Let $G = (V, E, \tau)$ be a temporal graph and $H = (V', E')$ be a static graph that $H = F(G)$ and $V = V_1 \cup V_2 \cup \cdots \cup V_\tau \cup S \cup Z$ and set of sources S and terminals Z. For any set vertex k-cut A' in H, there exists a set of (s, z, t)-temporal separator A in G such that $|A| \leq |A'|$.*

Proof. Let A be equal to $\{v_i | \exists t : v_{i,t} \in S'\}$. By the definition of A, it is clear that $|A| \leq |A'|$, so to prove the lemma it is sufficient to show that A is a (s, z, t)-temporal separator of G. Suppose that it is not, so there exists a temporal path P from s to z in subgraph of G induced by $V\ S$. It is clear that P is as follows.

$$P = (s, v_{i_1}, t_1), (v_{i_1}, v_{i_2}, t_1), \ldots (v_{i_{k_1-1}}, v_{i_{k_1}}, t_1)$$
$$(v_{i_{k_1}}, v_{i_{k_1+1}}, t_2) \ldots (v_{i_{k_2-1}}, v_{i_{k_2}}, t_2),$$
$$\ldots$$
$$(v_{i_{k_{r-1}}}, v_{i_{k_{r-1}+1}}, t_r) \ldots (v_{i_{k_r-1}}, v_{i_{k_r}}, t_r), (v_{i_{k_r}}, z, t_r)$$

Where $t_1 < t_2 < \ldots t_r$ and $t_r - t_1 < t$. Therefore $t_r - t < t_1 \geq t_r$ which results $(v_{i_{k_r}, t_r}, z_{t_1}) \in E'$ So, consider the path P' which is mentioned in the following:

$$P' = (s_{t_1}, v_{i_1,t_1}), (v_{i_1,t_1}, v_{i_2,t_1}), \ldots (v_{i_{k_1-1},t_1}, v_{i_{k_1},t_1}), (v_{i_{k_1},t_1}, v_{i_{k_1},t_2}),$$
$$(v_{i_{k_1},t_2}, v_{i_{k_1+1},t_2}) \ldots (v_{i_{k_2-1},t_2}, v_{i_{k_2},t_2}), (v_{i_{k_2},t_2}, v_{i_{k_2},t_3}),$$
$$\ldots$$
$$(v_{i_{k_{r-1}},t_r}, v_{i_{k_{r-1}+1},t_r}) \ldots (v_{i_{k_r-1},t_r}, v_{i_{k_r},t_r}), (v_{i_{k_r},t_r}, z_{t_1})$$

Based on the fact that $V(P) \in V \backslash A$, it is clear that $V(P') \in V' \backslash A'$ and also it is clear that P' is a path in subgraph of G induced by $V\ S$ from s_{t_1} to z_{t_1}. So, its contradict that A' is a set of vertex k-cut, therefore there is no path such $v(P) \in V \backslash A$ and A is a (s, z, t)-temporal separator in G. □

In the instance of Vertex k-Cut on graph H, we have τ pair of separators. On the other hand cardinality of each set of vertex k-cut in H, which disconnects all the terminals from corresponding sources, is greater than the optimal separator's cardinality disconnects z_i from s_i. Therefore the union of all separators that disconnects z_i from s_i is at most τ times greater than optimal vertex k-cut in directed graph H. Moreover, by Lemma 11 and 12 we could conclude that cardinality of (s, z, t)-temporal separator that could be constructed by Lemma 11 from the union of all set of separators that disconnects z_i from s_i is at most τ^2 cardinality of optimal (s, z, t)-temporal separator. □

3.3 Inapproximability of Temporal Separator

To find an inapproximability result for (s, z, t)-Temporal Separator problem, we have tried to reduce from some innaproximable problem. One of them which give us **APX**-Hardness of (s, z, t)-Temporal Separator is *Feedback Vertex Set* and give us the idea of the main result in this section. Reduction from *Set Cover* to (s, z, t)-Temporal Separator show us a higher lower bound of $\Omega(\log(n) + \log(\tau))$ on approximation of this problem.

Problem 8 (Feedback Vertex Set)**.**

- ***Instance****: Directed graph $G = (U, A)$.*
- ***Solution****: A feedback vertex set, i.e., a subset $K \subseteq U$ such that K contains at least one vertex from every directed cycle in G.*
- ***Measure****: Minimum cardinality of the feedback vertex set, i.e., $|V'|$.*

Lemma 13. *For any $t > 0$ there is a strict reduction from Feedback Vertex Set problem to (s, z, t)-Temporal Separator problem.*

Proof. Let directed graph $H = (U, A)$ be an instance of Feedback Vertex Set such that $U = \{v_1, v_2, \dots v_n\}$. Now we define a corresponding instance of $G = (V, E, t \times n)$ for (s, z, t)-Temporal Separator problems such that:

$$V = \{v_i | i \in [n]\} \cup \{u_i | i \in [n]\} \cup \{s, z\}$$

And the the set of edges E is equal to union of four sets of edges:

- First set is equal to $\{(s, v_i, i \times t) | i \in [n]\}$
- The second one is $\{(u_i, z, i \times t) | i \in [n]\}$
- For any edge $e = (v_i, v_j) \in G$ we add n edges $(v_i, v_j, k \times t)$ for any $k \in [n]$.
- For any edge $e = (v_i, v_j) \in G$ we add n edges $(v_i, u_j, k \times t)$ for any $k \in [n]$.

Now we claim that there exists a set of feedback vertex set with a cardinality of k if and only if there exists a set of (s, z, t)-temporal separator with a cardinality of lower than or equal to k.

$\rightarrow$ Suppose that K is a feedback vertex set in H. Let S equal to:

$$S = \{v_i | v_i \in V\}$$

We claim that S is a set of (s, z, t)-temporal separator. Assume that there is a (s, z, t)-temporal path P from s to z.

$$P = (s, v_{i_1}, t_1), (v_{i_1}, v_{i_2}, t_2), \dots (v_{i_{k-1}}, v_{i_k}, t_k), (v_{i_k}, u_{i_{k+1}}, t_{k+1}), (u_{i_{k+1}}, z, t_{k+2}) \quad (7)$$

It is clear that (s, z, t)-temporal path P should be equal to the temporal path shown in equation 7, such that $t_{k+2} - t_1 < t$. Therefore, it is clear that $t_{k+2} = t_1$ and therefore $i_{k+1} = i_1$. Now it is clear that $(v_{i_1}, v_{i_1}, \dots v_{i_k})$ is the cycle and $\forall p \in [k] : v_{i_p} \in U \backslash K$ which contradict with the fact that K is a feedback vertex set. The contradiction implies that there is no (s, z, t)-temporal path P which $V(P) \subseteq V$ results in S is a set of(s, z, t)-temporal separator. Also base on the selection of set S, it is obvious that $|S| = |K| \leq |K|$.

$\leftarrow$ Suppose that set $S \subseteq V$ is a set of (s, z, t)-temporal separator. Let $K \subseteq U$ which define as follows:

$$K = \{v_i | (v_i \in S | u_i \in S)\}$$

First, it is clear that $|K| \leq |S|$. Now, assume that there is a cycle C in directed graph H such that $V(C) \subseteq U \backslash K$. Let assume that $C = (v_{i_1}, v_{i_2}, \dots, v_{i_k})$, so it is clear that $\forall p \in [k] : v_{i_p} \in V' \backslash S$ and $u_{i_p} \in V' \backslash S$, therefor consider the path P equal to:

$$P = (s, v_{i_1}, i_1 \times t), (v_{i_1}, v_{i_2}, i_1 \times t), \dots (v_{i_{k-1}}, u_{i_k} = u_{i_1}, i_1 \times t), (u_{i_1}, z, i_1 \times t) \quad (8)$$

Due to the fact that $\forall p \in [k] : v_{i_p} \in V' \backslash S$ and $u_{i_p} \in V' \backslash S$, all the vertices in (s, z, t)-temporal path P has been status in equation 8 belongs to $V \backslash S$ which contradict with

the fact that S is a (s, z, t)-temporal separator. The contradictions implies that there is no cycle C such that $V(C) \in U$ resulted in K is a feedback vertex set to G.

According to the previous claim, every solution in $(s, z, t)-$Temporal Separator, has a corresponding solution in Feedback Vertex Set, and vice versa. Therefore, an optimal solution in $(s, z, t)-$Temporal Separator, has a correspondent optimal solution in Feedback Vertex Set. As a result $\frac{|K'|}{|K_{opt}|} = \frac{|S'|}{|S_{opt}|}$. □

The main result of this section is $\Omega(\log n + \log(\tau))$-inapproximation for the (s, z, t)-Temporal Separator problem. This immediately follows from the theorem below, establishing a strict reduction from the Set Cover problem.

Problem 9 (Set Cover)**.**

- ***Instance****: Collection $\mathcal{S}$ of subsets of a universe U.*
- ***Solution****: A set cover for C, i.e., a subset $C \subseteq \mathcal{S}$ such that every element in U belongs to at least one member of C.*
- ***Measure****: Minimum cardinality of the set cover, i.e., $|C|$.*

In the following theorem, we present that any instance of set cover can be strictly reduced to (s, z, t)-Temporal Separator. First we show that any instance of the Set Cover where the universe $U = \{1, 2, \ldots, n\}$ and the family $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ such that $S_i \subseteq U$ can be mapped to an instance of the (s, z, t)-Temporal Separator problem. Finally we prove that for any solution $\mathcal{S}'$ to a Set Cover problem there exists a correspondent solution V' to the (s, z, t)-Temporal Separator problem, where $|\mathcal{S}'| = |V'|$.

Theorem 15. *For every $t > 0$ there is a strict 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 $U = \{1, 2, \ldots n\}$ is the universe and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is a family of sets the union of which covers U. For each $i \in U$ define the family $\mathcal{F}_i$ as follows:

$$\mathcal{F}_i = \{S \in \mathcal{S} \mid i \in S\},$$

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.,

$$\mathcal{F}_i = \{S_{i_1}, \ldots, S_{i_{k_i}}\}. \tag{9}$$

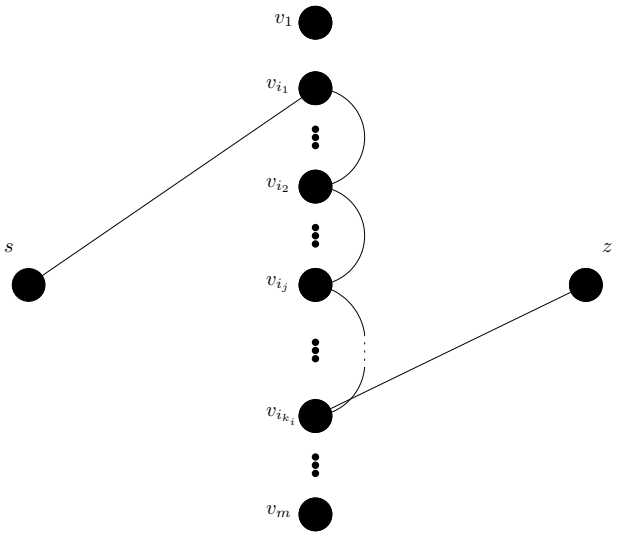

Figure 7: Layer $G_{i\times t}$ of temporal graph which is instance of (s, z, t)-Temporal Separator where $U = \{1, 2, \ldots, n\}$ and $\mathcal{S} = \{S_1, \ldots, S_m\}$ such that $i \in S_{i_1}, S_{i_2}, S_{i_{k_i}}$

Now, we are ready to define the output of our reduction. 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 $E = E_1 \cup E_2 \cup \cdots \cup E_n$, where

$$E_i = \{(s, v_{i_1}, i \cdot t), (v_{i_1}, v_{i_2}, i \cdot t), \ldots, (v_{i_{k_i-1}}, v_{i_{k_i}}, i \cdot t), (v_{i_{k_i}}, z, i \cdot t)\}.$$

The main idea behind the proof is to map every element of U to a path 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 $V' = \{v_{j_1}, \ldots, v_{j_\ell}\} \subseteq V$ is a $(s, z, t)-$temporal separator for $f(U, \mathcal{S})$ if and only if $\mathcal{S}' = \{S_{j_1}, \ldots, S_{j_\ell}\} \subseteq \mathcal{S}$ is a set cover for $(U, \mathcal{S})$.

Figure 7 represents the edges in the layer G_i, 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 7 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:

$$P = (s, v_{i_1}, i \cdot t), (v_{i_1}, v_{i_2}, i \cdot t), \ldots, (v_{i_{k_i-1}}, v_{i_{k_i}}, i \cdot t), (v_{i_{k_i}}, z, i \cdot t),$$

where the indices are according to (9) Since i is not covered, $\mathcal{F}_i \cap \mathcal{S}' = \varnothing$, 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 $(\Delta t)(P) = 0$).

$\leftarrow$ Now, suppose for contradiction that V' is not a $(s, z, t)-$-temporal separator. Thus, there is path P from s to z with $(\Delta t)(P) < t$. 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 $\mathcal{S}'$, for otherwise, one of the v_{j_i} would be in V'.

According to 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 correspondent optimal solution in Set Cover. As a result $\frac{|V'|}{|V_{opt}|} = \frac{|S'|}{|S_{opt}|}$. This implies that the reduction is strict. $\square$

According to the inapproximability result for Set Cover, which is given in [17], the following corollary could be resulted.

Corollary 6. *(s, z, t)-Temporal Separator problem is not approximable to within $(1-\epsilon)(\log n + \log(\tau))$ in polynomial time unless for any $\varepsilon > 0$, unless NP $\subset$ DTIME$(n^{\log \log n})$.*

3.4 Temporal Separator on Temporal Graphs with Bounded Pathwidth

A polynomial-time algorithm for (s, z)-Temporal Separator problem on temporal graphs with an underlying graph with bounded treewidth is given [22]. However, the algorithm does not work for (s, z, t)-Temporal Separator. The question about the hardness or polynomial-time solvability of (s, z, t)-Temporal Separator problem on temporal graphs with bounded treewidth of their underlying graph remains open.

However, in this section, we present a reduction from the *Discrete Segment Covering* problem on line interval.

Problem 10 (Discrete Segment Covering (DISC-SC))**.** [7]

- ***Instance****: A set* Γ *of* n *intervals (called segments from here on), on the rational line; a set* $\mathcal{I}$ *of unit-intervals on the rational line.*
- ***Solution****: A subset of unit interval* $A \subseteq \mathcal{I}$ *which covers all the segment in* Γ
- ***Measure****: Minimum cardinality of the set* 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. $DISC - SC$ problem is **NP**-Complete [7]. A simple polynomial-time reduction from the Discrete Segment Covering problem is presented as follows.

Theorem 16. *There is a polynomial-time reduction from the Discrete Segment Covering problem to* (s, z, t)*-Temporal Separator problem.*

Proof. We denote the starting and ending point of the interval I by $s(I)$ and $e(I)$ respectively. Also, we use this notation for all the segments. Consider $(I_1, I_2, \dots I_m)$ a non decreasing order of all intervals in $\mathcal{I}$, also consider $(C_1, C_2, \dots, C_n)$ an arbitrary order of segments in Γ. Based on the fact that size of all intervals in $\mathcal{I}$ is one, it could be concluded that for any point p and three indices $i < k < j$ if $p \in I_i$ and $p \in I_j$ the $p \in I_k$ since starting point of I_k is before the starting point of I_j and ending point of I_k is after the ending point of I_i.Now we construct a temporal graph $G = (V, E, t \times |\gamma|)$ such that $V = \{v_i | i \in [m]\}$. For any segment C_j we construct the layer $G_{j \times t}$ as following:

Let l_s and r_s the be the indices of first and last intervals which cover starting point $s(C_j)$. It is clear that a starting point of C_j is covered by all the interval between I_{l_s} and I_{r_s}. Similarly, L_e and r_e denote the index of first and last intervals which cover ending point $e(c_j)$. Since ending point $e(C_j)$ is after the starting point $s(C_j)$ the $l_s \leq l_e$ and $r_s \leq r_e$. So based on l_e and r_s we consider the two following case:

Case 1. $(l_e \leq r_s)$. In this case we add a following temporal path which creates the layer $G_{j \times t}$. Figure 8 shows this temporal path.

$$(s, v_{l_s}, j \times t), (v_{l_s}, v_{l_s+1}, j \times t), \dots, (v_{r_e-1}, v_{r_e}, j \times t), (v_{r_e}, z, j \times t) \tag{10}$$

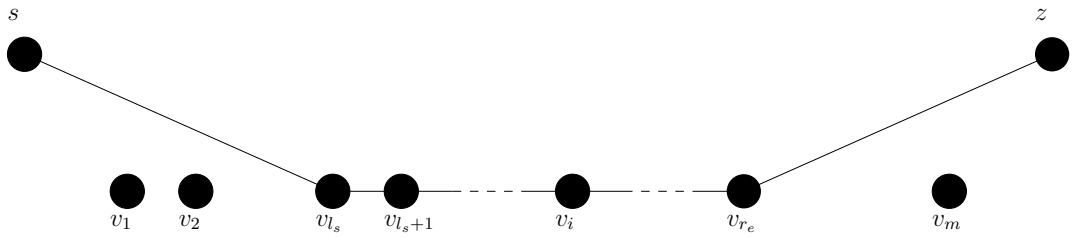

Figure 8: Layer $G_{j\times t}$ in case that $l_e \leq r_s$. The time label for all the edges is $j \times t$

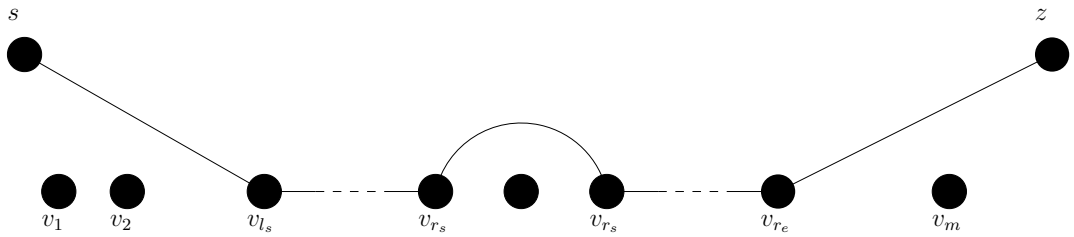

Figure 9: Layer $G_{j\times t}$ in case that $r_s < l_e$. The time label for all the edges is $j \times t$

Case 2. $(r_s < l_e)$. Similar to previous case we add a path from s to z which creates the layer $G_{j\times t}$. Figure 9 shows this temporal path.

$$\begin{aligned}(s, v_{l_s}, j \times t), (v_{l_s}, v_{l_s+1}, j \times t), \ldots, (v_{r_s-1}, v_{r_s}, j \times t),\\ (v_{r_s}, v_{l_e}, j \times t),\\ (v_{l_e}, v_{l_e+1}, j \times t), \ldots, (v_{r_e-1}, v_{r_e}, j \times t), (v_{r_e}, z, j \times t)\end{aligned} \tag{11}$$

Suppose that $A \in \mathcal{I}$, and let $S = \{v_i | I_i \in A\}$. We claim that A covers Γ if and only if set S is a (s, z, t)-temporal separator.

$\rightarrow$ A is a set of intervals which covers all segments C_j. If $l_e \leq r_s$ (Case 1) then there exists an interval $I_i \in A$ such that $l_s \leq i \leq r_e$ where the temporal path which is shown in equation 10 incidents with vertex v_i. On the other hand, if $r_s < l_e$ (Case 2) then there exists an interval I_i such that $l_s \leq i \leq r_s$ or $l_e \leq i \leq r_e$ where the temporal path which is shown in equation 3.4 incidents with vertex v_i. Therefore, every (s, z, t)-temporal path in the temporal graph G has incident with one vertex in S results which S is a (s, z, t)-temporal separator.

$\leftarrow$ S is a set of (s, z, t)-temporal separator. Then for any integer $j \in [n]$ a temporal path in time $j \times t$ should be incidents with one vertex in S. If $l_e \leq r_s$ (Case 1) then there exists $v_i \in S$ such that $l_s \leq i \leq r_e$ which implies that I_i which covers C_j belongs to A. If $r_s < l_e$ then there exists $v_i \in S$ such that $l_s \leq i \leq r_s$ or $l_e \leq i \leq r_e$ which

implies I_i that covers C_j belongs to A. Therefore all the segments are covers by an interval in A. □

A reduction that represents the results that (s, z, t)-Temporal Separator is **NP**-Complete, is already shown. For any tuple (i, j) such that $i \leq j$ and $i, j \in [m]$ we could put a segments $(e(I_i), s(I_j))$ into a set of segments Γ. As a result, the edge (v_i, v_j) will be present in the underlying graph of the temporal graph G, which means there is no bound on the underlying graph $G_{\downarrow}$. However, the presented reduction gave us a hint to preset the main result in this section.

It is shown in [7] that the Discrete Segment Covering problem remains **NP**-Complete when the length of all segments in Γ is equal; However, it does not bound the underlying graph $G_{\downarrow}$ which is used in the previous reduction. When the length of all segments is at most one, the problem could be solved by a simple greedy algorithm [7]. However, solvability or hardness of the problem when the length of all segments is bounded by a constant k remains open. Here in this section, we present a reduction $DISC-SC$ problem when the segments' length is bounded by k, to (s, z, t)-Temporal Separator where treewidth of the underlying graph is bounded by $2k + 6$.

Theorem 17. *There is a polynomial-time reduction from the Discrete Segment Covering problem. The length of all segments is bounded with size k to (s, z, t)-Temporal Separator in which treewidth of the underlying graph is bounded by size $2k + 6$.*

Proof. Consider $(\mathcal{I}, \Gamma)$ is an instance of Discrete Segment Covering problem such that the length of all the segments in Γ is lower than or equal to k. Consider intervals in $\mathcal{I} = I_1, I_2, \ldots\ I_n$ in non-decreasing order of their starting time. 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 does not exist, then 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 the step 2.

Lemma 14. *Any point p will be covered by $\mathcal{I}$ if SP covers it.*

Proof. We prove the lemma by induction. First, based on the algorithm, it is clear that $I_1 \in Sp$ and $I_n \in Sp$. Now we state the induction that for any $i \in [n]$ such that $I_i \in SP$, a point p will be covered by $\{I_1, I_2, \ldots, I_i\}$ if it is covered by $\{I_1, I_2, \ldots, I_i\} \cap SP$.

- **Base case**. For $i = 1$, it is clear that $I_1 \in SP$.
- **Induction step**. Suppose $j < i$ is the largest integer that $I_j \in SP$. Based on the assumption of induction any point p is covered by $\{I_1, I_2, \ldots, I_j\}$ if it is covered by $\{I_1, I_2, \ldots, I_j\} \cap SP$. On the other hand starting point of I_i is before the ending point of I_j, which results in point p being covered by $\{I_1, I_2, \ldots, I_i\}$ if it is covered by $\{I_1, I_2, \ldots, I_i\} \cap SP$.

Since $I_n \in SP$, point p is covered by $\{I_1, I_2, \ldots, I_n\} = \mathcal{I}$ if it is covered by $\{I_1, I_2, \ldots, I_i\} \cap SP = SP$. □

This proof's main idea is holding on to the following features for the special set SP. Denote $SP = \{I_{m_1}, I_{m_2}, \ldots I_{m_q}\}$, based on the selection of interval $I_{m_{i+1}}$ it is clear that starting point $I_{m_{i+2}}$ is greater than ending point I_{m_i} which implies that $s(I_{m_{i+2}}) > s(Im_i) + 1$ where results in $e(I_{m_{i+2k}}) > s(Im_i) + k + 1$. 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 a main idea to construct an instance of (s, z, t)-Temporal Separator problem.

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 in time $i \times t$. Let interval a and b be the indices of the first intervals in SP which cover point $s(C)$ and $e(C)$. Based on the Lemma 14 if such point $a(b)$ does not exist, then point $s(C)(e(C))$ will not be covered with any intervals in $\mathcal{I}$. Therefore, we could consider that segment c as a single point $e(C)(s(C))$ and continue on algorithm. Also let l_s be the index of the most left interval which covers $s(C)$, and let r_s be the index of the most right interval which covers $s(S)$. It is obvious that $s(C)$ is covered by all of the intervals between l_s and r_s. Similarly let l_e and r_e be the indices of the most left and the most right intervals which cover $e(C)$. It is possible that $l_e \leq r_s$ to prevent creating a loop instead of a path. If that is the case then consider $l_e = r_s + 1$. Now, add a following (s, z, t)-temporal path to the temporal graph G.For simplicity we denote $i \times t$ by θ.

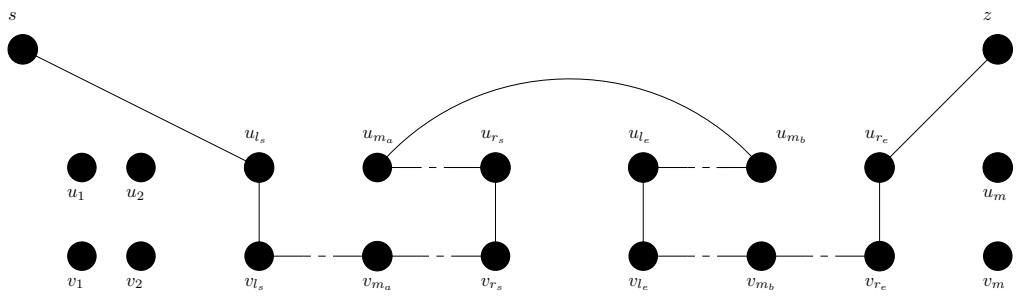

Figure 10: (s, z, t)-temporal path in the layer $G_{j\times t}$. The time label for all the edges is $j \times t$

$$
\begin{gathered}
(s, u_l, \theta), (u_l, v_l, \theta), (v_{l_s}, v_{l_s+1}, \theta), \dots (v_{r_s-1}, v_{r_s}, \theta) \\
(v_{r_s}, u_{r_s}, \theta), (u_{r_s}, u_{r_s-1}, \theta), \dots (u_{a+1}, u_a, \theta) \\
(u_a, u_b, \theta) \\
(u_b, u_{b-1}, \theta), \dots, (u_{l_e+1}, u_{l_e}, \theta), (u_{l_e}, v_{l_e}, \theta) \\
(v_{l_e}, v_{l_e}+1, \theta) \dots (v_{r_e-1}, v_{r_e}, \theta), (v_{r_e}, u_{r_e}, \theta), (u_{r_e}, z, \theta)
\end{gathered} \tag{12}
$$

Figure 10 shows the above path in the graph that is equal to the layer $i \times t$. We claim that there exists set $A \subseteq \mathcal{I}$ that covers Γ which $A \leq p$ if and only if there is a set of (s, z, t)-temporal separator $S \in V$ 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 that there is a temporal path P in G, base on the construction of G this temporal path should be equal to a temporal path that is shown in equation 12 for some $i \in [n]$, which is implies $I_j \notin A$ for all j such that $l_s < j < r_s$ or $l_e < j < r_e$ that 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 $S \in V$ is a (s, z, t)-temporal separator in a temporal graph G. Let $A = \{I_i | u_i \in S \text{OR} v_i \in A\}$, it is clear that $|A| \leq |S|$. Consider segment $C \in \Gamma$ the i-th segment in Γ. There should be one vertex belonging to the temporal path P which is shown in equation 12 in S , since S is a (s, z, t)-temporal separator. Therefore there is j where $l_s < j < r_s$ or $l_e < j < r_e$ that either u_i or v_i belong to S, which implies $C \in A$, that results in A covering Γ.

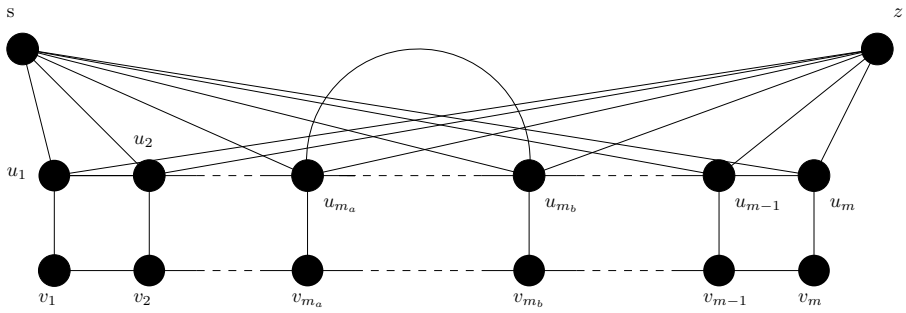

Figure 11: Graph G' is shown. The underlying graph $G_{\downarrow}$ is a subgraph of G'

Now we prove pathwidth of underlying graph $G_{\downarrow} = (V, E')$ of the temporal graph $G(V, E, |\gamma| \times t)$ is bounded by $2k+6$. We denote every edge (u_a, u_b, θ) in a path that is shown in equation 12 by crossing edge. Figure 11 shows that a graph G' which $G_{\downarrow}$ is a subgraph of G'. Now we give a path decomposition (P, β) for a graph $G_{\downarrow}$ in which the width of decomposition is lower than $2k+6$. Let $V(P) = \{a_1, a_2, \ldots, a_m\}$ and $E(P) = \{(a_1, a_2), \ldots, (a_{m-1}, a(m))\}$. Let $i \in [n]$ and $l(i)$ be the largest integers such that the starting point of 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:

$$\beta(a_i) = \{u_i, v_i, u_{i+1}, v_{i+1}, s, z\} \cup \{v_{m_l} | l \geq l(i) \text{ AND } l \leq l(i) + 2k\}$$

Lemma 15. *For any v_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 $\beta(q_i)$. 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 $u_q \in \beta(a_j)$. Note that if $m_p = i$ it will not impact the previous conditions. □

For any $v_i \in V$ is is clear that v_i just belongs to the two sets $\beta(a_{i-1})$ and $\beta(a_i)$. Also, s and z are presented in all the sets. Therefore, by Lemma 8 we could say that the third properties of path decomposition is satisfied. So, it is sufficient to show that for any 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 type of edges (u_i, v_i), (u_i, u_{i+1}), and (v_i, v_{i+1}) which satisfy the condition. If edge $e = (u_i, u_j)$ is a crossing edge, then $I_i \in SP$ and $I_j \in SP$, so let $m_p = i$ and $m_q = j$. Due to the fact that this edge is

corresponding 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 $v_i, v_j \subseteq \beta(a_i)$.

Also, the cardinality of all sets $\beta(a_i)$ is equal to $2k + 7$ which implies that the width of (P, β) is equal to $2k + 6$. Therefore the pathwidth of the underlying graph $G_{\downarrow}$ is lower than or equal to $2k + 6$. □

Corollary 7. *Discrete Segment Covering problem is solvable in polynomial time when a constant number bounds all segments' length, if (s, z, t)-Temporal Separator problem is solvable in polynomial time, in a temporal graph with bounded pathwidth by a constant number.*

3.5 Polynomial-time Algorithms for (s, z, t)-Temporal Separator

Here in this section, we will present polynomial algorithms for two types of temporal graph. We show that (s, z, t)-Temporal Separator is computable in polynomial time for a temporal graph $G = (V, E)$ if an underlying graph $G_{\downarrow}$ has branchwidth 2, or static graph $(V_{G_{\downarrow} \setminus \{s,z\}}, E_{G_{\downarrow}})$ is tree.

3.5.1 Temporal Separator on Graphs with Bounded Branchwidth

The graphs with branchwidth 2 are graphs in which each biconnected component is a series-parallel graph [41]. Here we present an algorithm to solve any version of restricted path (s, z)-Temporal Separator problem on a temporal graph with branchwidth, bounded by 2 on its underlying graph for which the existence of that path could be checked in polynomial time.

Theorem 18. *Given a temporal graph $G = (V, E)$, such that underlying graph $G_{\downarrow}$ has a maximum branchwidth 2, the problem restricted path (s, z)-Temporal Separator is solvable in time $O(|V|)f(G)$ where $f(G)$ is the time complexity of the algorithm to check the existence of restricted (s, z)-temporal path.*

Proof. Suppose that (T, β) is a branch decomposition of temporal graph $G = (V, E)$ which is rooted by r, we define the function $top : V \to V_T$ for every vertex of $v \in V$

equal to the furthest node $x \in V_T$ from root R which $E(v) \subseteq \beta(x)$. We also denote the x_l as the left child of x and x_r as the right child of x.

Lemma 16. *Let $G = (V, E)$ be a static graph with branch decomposition (T, β). For any vertex $v \in V$ such that $top(v) = x$, we have $v \in \partial\beta(x_l)$ and $v \in \partial\beta(x_r)$.*

Proof. Suppose that $v \notin \partial\beta(x_l)$ then it indicates $E(v) \in \beta(x_l)$ or for each edge e if $e \in E(v)$ the $e \notin \beta(x_l)$. If $E(v) \subseteq \beta(x_l)$, since x_l is further from root than x then there is a contradiction with the equality $top(v) = x$. On the contrary, if $E(v) \not\subseteq \beta(x_l)$ then due to the fact that $E(v) \subseteq \beta(X)$, we could conclude that $E(v) \subseteq \beta(x_r)$, since there is no edge in $e \in E(v)$ such that $e \in \beta(x_l)$. It also contradicts with the equality $top(v) = x$. Therefore, it can be derived that $v \in \partial\beta(x_l)$. Same could be apply to the other part. □

Given a temporal graph $G = (V, E)$ with underlying graph $G_\downarrow = (V, E')$ and branch decomposition (T, β) of $G_\downarrow$, we represent a polynomial time algorithm to solve a restricted path (s, z)-Temporal Separator problem on the temporal graph G. First we check weather there is restricted (s, z)-temporal path in G. If not, we return $\varnothing$ as a restricted path (s, z)-temporal separator. Otherwise we will continue on the algorithm. Suppose that $top(s) = x$ and $top(z) = y$. First consider the case that x and y are both equal to root of T, Algorithm 1 shows how to compute the restricted path (s, z)-temporal separator in this case:

Algorithm 1: Find restricted $(s, z)-$temporal separators from the root x

Function `FindSeparators`(T, x)**:**

$x_l \leftarrow$ left child x;

$x_r \leftarrow$ right child x;

if $\partial\beta(x_l) \cup \partial\beta(x_r) = \{s, z\}$ **then**

return `FindSeparators`$(T, x_l) \cup$ `FindSeparators`(T, x_r) ;

else

if *there is a restricted* (s, z)*-temporal path in* $\partial\beta(x)$ **then**

return $\partial\beta(x_l) \cup \partial\beta(x_r) \backslash \{s, z\}$;

else

return $\varnothing$;

end

end

Recall that boundary $\partial\beta(x)$ is equal to $\{v | v$ is incident to edges in $\beta(x)$ and $E \backslash \beta(x)\}$.

As shown in Algorithm 1 first we check if union of $\partial\beta(x_l)$ and $\partial\beta(x_r)$ is s and z, then we find separator on each temporal subgraph induced by $\beta(x_l)$ and $\beta(x_r)$ separately. Otherwise we check that if there is a restricted (s, z)-temporal path in the temporal graph induced by $\beta(x)$, we return $\partial\beta(x_l) \cup \partial\beta(x_r) \backslash \{s, z\}$ else we return $\varnothing$ as a separator.

Lemma 17. *Algorithm 1 finds a minimum restricted path* (s, z)*-temporal separator of temporal graph induced by* $\beta(v)$ *in* $O(|E|)f(G)$ *time if* $\partial\beta(v) \subseteq \{s, z\}$ *or* $top(s) = top(z) = v$.

Proof. Consider that $top(s) = top(z) = v$, base on Lemma 16 it is clear that $\partial\beta(v_l) = \{s, z\}$ and $\partial\beta(v_l) = \{s, z\}$, so we could conclude that every (s, z)-path P, $E(P) \in \beta(v_l)$ or $E(P) \in \beta(v_r)$. Because if the path contains edges from both $\beta(v_r)$ and $\beta(v_l)$ then there exists two consequential edge e_1 and e_2 in path P such that $e_1 \in \beta(v_l)$ and $e_2 \in \beta v_r$ which means that there is vertex p being incident to both edge sets $\beta(v_r)$ and $\beta(v_l)$. So $p \in \partial\beta(v_r)$ and $p \in \partial\beta(v_l)$ which contradicts with the assumption. So, then it is clear that the minimum restricted path (s, z)-temporal separator in graph induced by $\beta(v)$ is the union of tow minimum restricted path (s, z)-temporal

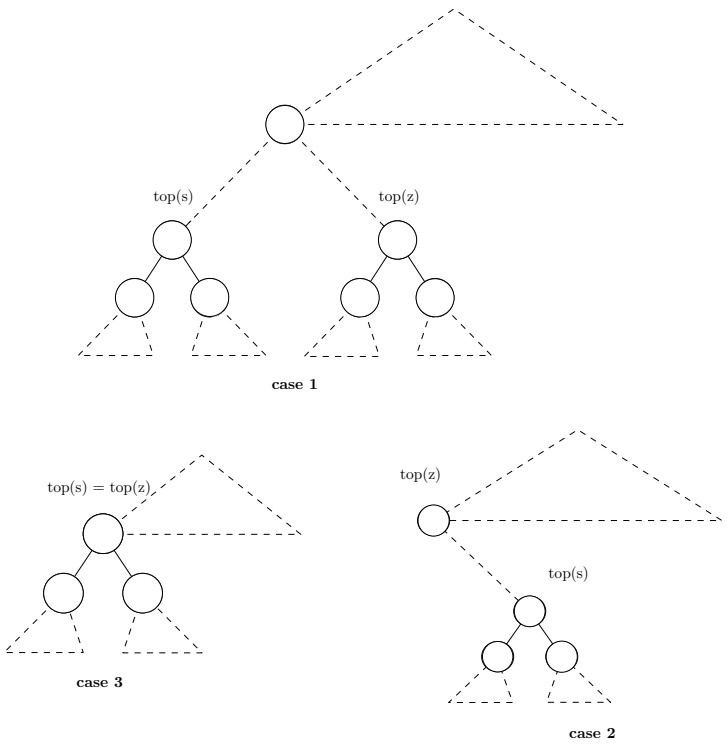

Figure 12: Three cases for node $top(s)$ and $top(z)$ in branch decomposition (T, β)

separator set of graph induced by $\beta(v_l)$ and $\beta(v_r)$, therefore algorithm 1 will find the minimum (s, z, t)-temporal separator properly.

If $\partial\beta(v) \subseteq \{s, z\}$ and $\partial\beta(v_l) \cup \partial\beta(v_r) \in \{s, z\}$, then based on the reasoning above, algorithm 1 computes the minimum restricted path (s, z)-temporal separator properly. Otherwise there is a vertex $q \in V$ such that $q \in \partial\beta(v_l) \cup \partial\beta(v_r)$. Due to the fact that $p \notin \partial\beta v$, it is clear that $top(q) = v$, so based on Lemma 16, $p \in \partial\beta(v_r)$ and $p \in \partial\beta(v_l)$. This means $\partial\beta(v_l) = \{s, q\}$ and $\partial\beta(v_r) = \{z, q\}$, or $\partial\beta(v_r) = \{s, q\}$ and $\partial\beta(v_l) = \{z, q\}$. This results in all the (s, z)-path in $\beta(v)$ having the node q. So, if there is any restricted (s, z)-temporal path in temporal graph induced by $\beta(v)$, then any restricted path (s, z)-temporal separator contains at least one vertex. Therefore, $\partial\beta(v_r) \cup \partial\beta(v_l) \backslash \{s, z\} = \{q\}$, a set of minimum (s, z, t)-temporal separator, will be returned from algorithm 1.

□

We consider three general cases for x and y and we will describe the algorithm case by case for all these three. Figure 12 Shows the different cases.

Case 1. $x \neq y$ and neither is x an ancestor of y, nor is y an ancestor of x. In this case first for all vertex $v \in V \backslash \{s, z\}$, we check if there is no restricted (s, z)-temporal path in G after removing vertex v. Then we return $\{v\}$ as the temporal separator. If such vertex does not exist, we return $\partial\beta(x)$.

Case 2. $x \neq y$, and one of them is the ancestor of the other one. Due to the symmetry, without loss of generality, we assume that y is the ancestor of x. Then we will consider three cases for the node y.

- **Case 2.1.** $z \notin \partial\beta(x)$. In this case for all vertex $v \in V \backslash \{s, z\}$, If there is no restricted (s, z)-temporal path in G when the vertex v is deleted, we return $\{v\}$ as the set of temporal separator. If such vertex does not exist then we will return $\partial\beta(x)$.

- **Case 2.2.** $\partial\beta(x) = \{z\}$. In this case we run the algorithm 1 from the node x in temporal subgraph induced by $\beta(x)$.

- **Case 2.3.** $\partial\beta(x) = \{z, q\}$ for some vertex $q \in V$. Here in this case p should be incident to only one of the two sets $\beta(x_l)$ and $\beta(x_r)$. Due to the symmetry, without loss of generality, assume that $\partial\beta(x_l) = \{q, z\}$, then $\partial\beta(x_r) = \{s, z\}$. So, we run the Algorithm 1 from the node y_l. Suppose that S is the result returned by Algorithm 1. We check that if there is any restricted (s, z)-temporal path in the temporal graph induced by $E \backslash \beta(x_r)$ then we will return $S \cup q$ as the temporal separator. Otherwise we return S as the temporal separator.

Case 3. $x = y$. In this case we consider two temporal subgraphs G_{in} and G_{out} that their underlying graph is equal to $G_{\downarrow in} = (V, \beta(x))$ and $G_{\downarrow out} = (V, E \backslash \beta(x))$ respectively. Then we find restricted path $(s, z)-$temporal separator on both temporal graph G_{in} and G_{out} separately. For G_{out} first we check that if there is no restricted (s, z)-temporal path then we return $\varnothing$ as a solution, otherwise we check if there is any vertex $v \in V$ such that by removing v from G_{out} there is no restricted (s, z)-temporal path in G_{out} then we will return $\{v\}$, and if there is no one we will return $\partial\beta(x)$ as a solution for restricted path (s, z)-temporal separator for graph G_{out}.

For finding restricted path (s, z)-temporal separator on G_{in} we run the Algorithm 1 on the node x.

We prove that the algorithm described above finds the minimum restricted path (s, z)-temporal separator. We prove the correctness of the algorithm for each case.

Lemma 18. *Given a branch decomposition (T, β). For any node q from T removing $\partial\beta(q)$ will disconnect every vertex that appears in one of the bags of nodes in the subtree, which is rooted by q from the rest of vertices.*

Proof. Based on the fact that $\partial\beta(q)$ contains all incident vertex of set $\beta(q)$ and $E\backslash\beta(q)$ then every path from one vertex that is incident to $\beta(q)$ to the vertices that are incident to $E\backslash\beta(q)$ will contain one vertex in $\beta(q)$, so removing the vertices in $\beta(q)$ will disconnect them. □

In **Case 1** because based on Lemma 18 removing $\partial\beta(x)$ will disconnect s from z in underlying graph G. So, it is clear that the smallest restricted path (s, z)-temporal separator has the cardinality at most 2. Because the algorithm checks all the possibilities of one node, then the algorithm finds the minimum restricted path (s, z)-temporal separator properly.

In **Case 2.1**, the same as case 1, as $z \notin \partial\beta(x)$ then it is not incident to any of the edges in set $\beta(x)$, so removing the vertices in $\partial\beta(x)$ will remove all the restricted path (s, z)-temporal separator in G.

In **Case 2.2** for every path from s to z all the edges in the path are in the set $\beta(v)$, and based on Lemma 17 algorithm 1 will finds the minimum restricted path (s, z)-temporal separator.

In **Case 2.3** for a (s, z, t)-temporal path in G, either all the edges are in $\beta(v)$ or all the edges are in $E\backslash\beta(v)$. Based on Lemma 17 S is the smallest (s, z, t)-temporal path in the graph induced by $\beta(v)$, and due to the fact that all other paths from s to v meet q, then $S \cup \{q\}$ is the smallest restricted path (s, z)-temporal separator if and only if there is any restricted (s, z)-temporal path on the temporal graph induced by $E\backslash\beta(v)$.

In **Case 3** all the restricted (s, z)-temporal paths are either in G_{in} or in G_{out}. Based on Lemma 17 algorithm finds the smallest set of restricted path (s, z)-temporal separator in G_in. Also it is obvious that all the (s, z, t)-temporal paths in G_{out} meet one of the vertices in $\partial\beta(x)$, so if there is a restricted (s, z)-temporal path and there is no single vertex in V such that removing it will remove all that path, then $\partial\beta(x)$ is one of the smallest restricted (s, z)-temporal path.

Time complexity. In all the cases based due to the fact that existence of restricted (s, z)-temporal path could be compute in time $f(G)$, it takes $O(|V|)f(G)$ of time steps to check for every node that whether deleting them removes all the (s, z, t)-temporal paths or not. Also, the algorithm 1 takes $O(|V|)f(G)$ of time steps, and because we just run the algorithm 1 one time, then the algorithm's time complexity is $O(|V|)f(G)$. □

Based on Lemma 8, existence of (s, z, t)-temporal path could be computed in polynomial time. So, by the Theorem 18, the following corollary could be concluded.

Corollary 8. *Given a temporal graph $G = (V, E)$, such that underlying graph $G_{\downarrow}$ has a maximum branchwidth 2, the problem (s, z, t)-Temporal Separator is solvable in time $O(|V||E||Ts|)$ where $Ts = \{t(e) : e \in E(s)\}$.*

3.5.2 Temporal Separators on Tree-Based Family of Graph

In this section we present a polynomial time greedy algorithm which is motivated by point-cover interval problem for path restricted (s, z)-temporal separators on the graph such that the graph $G_{\downarrow} = (V\backslash\{s, z\}, E(G_{\downarrow}))$ is tree, if the existence of restricted (s, z)-temporal path could be checked in polynomial time.

Theorem 19. *let $G = (V, E)$ be a temporal graph with underlying graph $G_{\downarrow}$, such that static graph $(V\backslash\{s, z\}, E(G_{\downarrow}))$ is a tree. Any version of restricted path (s, z)-temporal separators for which the existence of that path could be checked in polynomial time on a graph with path underlying graph, is computable in polynomial time on G.*

Proof. Given a temporal graph G with underlying graph $G_{\downarrow}$. Let rooted tree $T = (V\backslash\{s, z\}, E(G_{\downarrow}))$. For any node v in T consider removing list RL_v a list of all the node pairs on the graph where there is a restricted (s, z)-temporal path through the unique path between them. For each pair of node (u, v) where order does not matter (2 nodes could be the same) in tree T denote the unique path between u and v in T by $P_{u,v}$. First of all we compute the list RL_v for all node in T. In order to do that, for any pair of node (u, v) in T, we check that if there is any restricted (s, z)-temporal path on temporal graphs induced by set of edges $E(P_{u,v}) \cup \{(s, u), (v, z)\}$ or $E(P_{u,v}) \cup \{(s, v), (u, z)\}$, then we will add pair (u, v) to the remove list of all node in the unique path $P_{u,v}$.

Lemma 19. *A set of $S \in V$ is a restricted path (s, z)-temporal seprators if and only if for each pair (u, v) that exist on at least one removing list containing pair(u, v), there is at least one node $x \in S$ such that $(u, v) \in RL_x$.*

Proof. Let $S \in V$ be a set of vertices, here we prove both directions.
$\rightarrow$ Suppose that $S \in V \backslash \{s, z\}$ is a set of restricted (s, z)-temporal separators. For every restricted (s, z)-temporal path that starts with edge (s, u) and ends with edge (v, z), there is a node x in this path such that $x \in S$. Since there is only one path in T between node u and v, for every $x \in V_{P_{u,v}}$ either all of them contain pair (u, v) in their removing list or none of them contains pair (u, v) in their removing list.
$\leftarrow$ Suppose that $S \in V \backslash \{s, z\}$ is a set of vertices such that for each pair (u, v) that exist on at least one removing list containing pair(u, v), there is at least one node $x \in S$ such that $(u, v) \in RL_x$. Consider the restricted (s, z)-temporal path P in G. Let the first edge be (s, u) and the last edge be (v, z), then as the pair (u, v) exists only on the removing list of $P_{u,v}$ then there is a node $x \in S$ such exist in P. $\square$

Initiate the set S equal to $\varnothing$ and consider the rooted tree T. While there exists a none empty removing list, proceed to the following steps.

- In each step of algorithm select the node x with highest density in T for which there exists a pair $(u, v) \in RL_x$ such that $(u, v) \notin RL_{parent(v)}$.
- Add x to the set S and remove all pairs in RL_x from the removing lists of all the other nodes.

Algorithm 2: Find restricted $(s, z)-$temporal separators

```
Function FindSeparators(G = (V, E)):
    T ← (V_{G↓} − {s, z}, E_{G↓});
    S ← ∅;
    for (u, v) ∈ V_T × V_T do
        if exist a restricted (s, z)-temporal path starts with edge (s, u, t_1) and
          ends with (v, z, t_2) then
            for x ∈ V_{P_{u,v}} do
                RL_x ← RL_x ∪ {(u, v)};
            end
        end
    end
    Traverse((T, root));
    return S;
    Function Traverse(T, v):
        for u ∈ childs(v) do
            Traverse(T,u)
        end
        for (u, v) ∈ RL_v do
            if (u, v) ∈ RL_{parent(v)} then
                S ← S ∪ {v};
                remove (u, v) from all the removing list;
            end
        end
```

Here we provide proof for the algorithm. Suppose that the algorithm returns a set S. Based on Lemma 19, it is obvious that S is a valid restricted (s, z)-temporal path. So, it is sufficient to show that S is optimal. Suppose that S is not an optimal set of restricted (s, z)-temporal separators. Let S_{opt} be an optimal set of restricted (s, z)-temporal separators with the maximum common vertices selected in our algorithm at first steps. Suppose that $S_{\{opt\}}$ has k elements that are selected in the algorithm at k first steps. Let vertex x be the $(k + 1)$-th vertex that is selected in our algorithm. Based on the algorithm's selection, there is a pair of node (u, v) that appears in removing list RL_v and not in RL_{parent_v}. So, there should be a vertex $x' \in S_{opt}$ such

that $(u, v) \in RL_{x'}$. Then the following facts could be concluded.

- $(u, v) \notin RL_{parent(x)}$, therefore x' is in the subtree of x.
- There does not exist a pair $(u', v') \in RL_{x'}$ such that $(u', v') \in RL_x$. If there is such a pair then, there is a node y such in the sub-tree x such that $(u', v') \in RL_y$ and $(u', v') \notin RL_{parent_(y)}$, so the pair node y should be chosen before node x.

Therefore, by changing the vertex x' from the set S_{opt} to x, it will remain a valid set of restricted path (s, z)-temporal separators, and it contradicts with the selection of optimal set S_{opt}. It concludes that S is an optimal set of restricted path (s, z)-temporal separators. □

Based on Lemma 8, the existence of (s, z, t)-temporal path could be computed in polynomial time. So, by the Theorem 19, the following corollary could be concluded.

Corollary 9. *(s, z, t)-temporal separators problem is solvable in polynomial time on the temporal graph $G = (V, E, \tau)$ where the static graph $(V \backslash \{s, z\}, E(G_{\downarrow}))$ is a tree.*

Chapter 4

Activity Timeline

Activity timeline problems were investigated in [43]. $MinTimeline_{\infty}$ problem whose goal is to find an activity timeline with minimum max span is solvable in polynomial time [43]. However, neither approximation nor inapproximation has been presented for the $MinTimeline$ problem.

Observation 2. *An activity timeline φ with the minimum max span which can be computed by the polynomial-time algorithm of $MinTimeline_{\infty}$ problem is an n-approximation for $MinTimeline$ problem which n is a number of vertices.*

Proof. consider φ_{opt} an activity timeline an optimal solution for $MinTimeline$ problem and $\Delta = \Delta(\varphi)$. Since $\Delta(\varphi_{opt}) > \Delta$, there exists an interval $I_v \in \varphi_{opt}$ in which $|I_v| \geq \Delta$. Therefore, we could conclude that:

$$\mathcal{S}(\varphi_{opt}) \geq n|I_v| \geq n\Delta \rightarrow \mathcal{S}(\varphi_(opt)) \geq n|\Delta(\varphi)| \tag{13}$$

□

4.1 Approximation Algorithm for $MinTimeline_m$

The 2-approximation algorithm for $MinTimeline_m$ problem is implemented using the duality of its corresponding linear programming [43]. Here we introduce a pure combinatorics 2-approximation algorithm for this problem.

First we initialize each activity timeline by setting an activity interval $I_u = [m_u, m_u]$ for each vertex $u \in V$. Then we process all edges of the graph in an arbitrary order. In step k if edge $e_k = (u_k, v_k, t_k) \in V$ is not covered by I_{u_k} or I_{v_k}

then extend both activity intervals I_{u_k} and I_{v_k} equally until one of them covers the time t_k. Now we will present the pseudocode of algorithm.

Algorithm 3: 2-approximation algorithm for $MinTimeline_m$ problem

```
Function FindTimeline(G):
    Result: φ = {I_u}_{u∈V}
    I_v = [m_v, m_v] for all v ∈ V;
    Order edges e_1, e_2, ..., e_m arbitrarily;
    for i ← 1 to m do
        (u_i, v_i, t_i) ← e_i;
        if t_i ∉ I_{u_i} ∪ I_{v_i} then
            Extend I_{u_i} and I_{v_i} simultaneously toward t_i;
        end
    end
```

We indicate the interval for the vertex $u \in V$ which is selected by the algorithm at the end of step k^{th} by ALG_u^k, so the activity timeline which is chosen by the algorithm at the end is $\varphi_{ALG} = \{ALG_u^m\}_{u \in V}$.

Lemma 20. *For any $e_i = (v_i, u_i, t_i)$, we have:*

$$2|(\Delta ALG_{u_i}^i \cup \Delta ALG_{v_i}^i) \cap \varphi_{OPT}| \geq |(\Delta ALG_{u_i}^i \cup \Delta ALG_{v_i}^i)| \tag{14}$$

Proof. let $\varphi_{OPT} = \{I_u\}_{u \in V}$, φ_{OPT} covers e_i, so $[m_{v_i}, t_i] \subseteq I_{v_i}$ or $[m_{u_i}, t_i] \subseteq I_{u_i}$. In addition according to the algorithm it is clear that $\Delta ALG_{u_i}^k \subseteq [m_{u_i}, t_i]$ and $\Delta ALG_{v_i}^i \subseteq [m_{v_i}, t_i]$ as well as $|\Delta ALG_{u_i}^i| = |\Delta ALG_{v_i}^i|$. so:

$$\Delta ALG_{u_i}^i \subseteq \Delta(\Delta ALG_{u_i}^i \cap \varphi_{OPT}) \subseteq (\Delta ALG_{u_i}^i \cap \varphi_{OPT}) \cup (\Delta ALG_{v_i}^i \cap \varphi_{OPT})$$

or

$$\Delta ALG_{v_i}^i \subseteq \Delta(\Delta ALG_{v_i}^i \cap \varphi_{OPT}) \subseteq (\Delta ALG_{u_i}^i \cap \varphi_{OPT}) \cup (\Delta ALG_{v_i}^i \cap \varphi_{OPT})$$

then it is clear that,

$$2|(\Delta ALG_{u_i}^i \cup \Delta ALG_{v_i}^i) \cap \varphi_{OPT}| \geq |(\Delta ALG_{u_i}^i \cup \Delta ALG_{v_i}^i)|$$

□

Theorem 20. *Algorithm 3 is a 2-approximation for $MinTimeline_m$ problem.*

Proof. It is sufficient to prove that:

$$\mathcal{S}(\varphi_{ALG}) = \sum_{u \in V} |ALG_u^m| \leq \mathcal{S}(\varphi_{OPT}) \tag{15}$$

we have:

$$\begin{aligned}\mathcal{S}(\varphi_{ALG}) = \sum_{u \in V} |ALG_u^m| = \sum_{i \in [m]} \sum_{u \in V} |\Delta ALG_u^i| = \sum_{i \in [m]} |(\Delta ALG_{u_i}^i \cup \Delta ALG_{v_i}^i)| \\ \leq \sum_{i \in [m]} 2|(\Delta ALG_{u_i}^i \cup \Delta ALG_{v_i}^i) \cap \varphi_{OPT}|\end{aligned}$$

therefore we have:

$$\longrightarrow \mathcal{S}(\varphi_{ALG}) \leq 2 \sum_{i \in [m]} \sum_{u \in V} |\Delta ALG_{u^i} \cap \varphi_{OPT}| = 2\mathcal{S}(\varphi_{ALG} \cap \varphi_{OPT}) \leq 2\mathcal{S}(\varphi_{OPT})$$

Where the inequality is by Lemma 20. □

4.2 Finding Activity Timeline on Temporal Graph with Bounded Treewidth

Here in this section, we present a polynomial-time algorithm to find an activity timeline with minimum total span on temporal graph $G = (V, E, \tau)$ if treewidth of an underlying graph $G_\downarrow$ is bounded by a constant number.

Assume that nice tree decomposition of underlying graph $G_\downarrow$ of temporal graph $G = (V, E, \tau)$ is equal to (T, β) rooted by node R. For any node $X \in V(T)$ and timeline $\varphi = \{I_u\}_{u \in \beta(X)}$. We say φ covers X if for any edge $e = (u, v, t) \in E$ such that $u, v \in \beta(X)$, $t \in Iu$ or $t \in Iu$. For a node X with a timeline φ which covers X, we define $T(X, \varphi)$ equal to a total span φ' that covers all the edges $(u, v, t) \in G$ if there exists a node X' in subtree rooted by node X which $u, v \in \beta(X')$, such that $\varphi \subseteq \varphi'$ and minimize value $\mathcal{S}(\varphi' \backslash \varphi)$. We build a dynamic table D that $D(X, \varphi)$ is equal to minimum value $\mathcal{S}(\varphi' \backslash \varphi)$. Now to fill the dynamic table D, we consider four cases base on node type in nice tree decomposition of $G_\downarrow$ to fill the dynamic table D:

Algorithm 4: Computing dynamic table for a forget node X

Function `updateForgetNode`$((T, \beta, X))$**:**

$Y \leftarrow$ unique child of X ;
$\{v\} \leftarrow \beta(Y) \backslash \beta(X)$;
for $\varphi \leftarrow \{I_u\}_{u \in \beta(X)}$ *that* φ *covers* X **do**
  for I_v *an interval of node* v **do**
    if $D(Y, \varphi \cup \{I_v\}) + |I_v| < D(X, \varphi)$ **then**
      $D(X, \varphi) = D(Y, \varphi \cup \{I_v\}) + |I_v|$;
      $T(X, \varphi) = T(Y, \varphi \cup \{I_v\})$;
    end
  end
end

Algorithm 5: Computing dynamic table for an introduce node X

Function `updatIntroduceNode`$((T, \beta, X))$**:**

$Y \leftarrow$ unique child of X ;
$\{v\} \leftarrow \beta(X) \backslash \beta(Y)$;
for $\varphi \leftarrow \{I_u\}_{u \in \beta(X)}$ *that* φ *covers* B **do**
  $D(X, \varphi) = D(Y, \varphi \backslash I_v)$;
  $T(X, \varphi) = T(Y, \varphi \backslash I_v) \cup \{I_v\}$;
end

Algorithm 6: Computing dynamic table for a disjoint node X

Function `updateDisjointNode`$((T, \beta, X))$**:**

$X_l \leftarrow$ left child of X ;
$X_r \leftarrow$ right child of X ;
for $\varphi \leftarrow \{I_u\}_{u \in \beta(X)}$ *that* φ *covers* B **do**
  $D(X, \varphi) = D(X_l, \varphi) + D(X_r, \varphi)$;
  $T(X, \varphi) = T(X_l, \varphi) \cup T(X_r, \varphi)$;
end

- **Case 1**. For a leaf node $X = \{v\}$ and all timeline $\varphi = \{I_v\}$ we have:

$$D(X, \varphi) = 0$$

- **Case 2**. For a forget node X and timeline $\varphi = \{I_u\}_{u \in \beta(X)}$ that covers X with unique child Y, let v be a unique vertex such that $\{v\} = \beta(Y) \backslash \beta(X)$, then the value $D(X, \varphi)$ is equal to the minimum value of

$$|I_v| + D(Y, \varphi \cup \{I_v\}) \tag{16}$$

For all I_v such that $\varphi \cup Iv$ covers C. Algorithm 4 shows how to compute the dynamic table D for a forget node X.

- **Case 3**. For an introduce node X and timeline $\varphi = \{I_u\}_{u \in \beta(X)}$ that covers X with unique child Y, let v be a unique vertex such that $\{v\}\beta(X) \backslash \beta(Y)$. Then we the value of $D(X, \varphi)$ is equal to:

$$D(Y, \varphi \backslash \{I_v\})$$

Algorithm 5 shows how to compute the dynamic table D for an introduce node X.

- **Case 4**. For disjoint node X and activity timeline $\varphi = \{I_u\}_{u \in \beta(X)}$ with the children X_l and X_r the value of $D(X, \beta)$ is equal to:

$$D(X_l, \varphi) + D(X_r, \varphi)$$

Algorithm 6 shows how to compute the dynamic array D for a disjoint node X.

Algorithm 7: Polynomial-time algorithm for temporal graph which has underlying graph bounded treewidth

```
Function computeMinTimelineRecursively((T, β, X)):
    for Y ← child of X do
        computeMinTimelineRecursively((T, Y));
    end
    if X is a leaf node then
        {v} ← β(X) for φ ← {I_v} do
            D(X, φ) = 0;
            T(X, φ) = φ;
        end
    end
    else if X is a forget node then
        updateForgetNode(T, β, X);
    end
    else if X is an introduce node then
        updateIntroduceNode(T, β, X);
    end
    else if X is a disjoint node then
        updateDisjointNode(T, β, X);
    end
```

Algorithm 7 shows how to compute the dynamic table for every node $X \in V(T)$ and $\varphi = \{I_u\}_{u \in \beta(X)}$ that covers X.

Lemma 21. *Let (T, β) be a nice tree decomposition of $G_{\downarrow}$, underlying graph of temporal graph $G = (V, E, \tau)$. Let $X \in V(T)$ and $\varphi = \{I_u\}_{u \in \beta(X)}$ a timeline that covers X. Algorithm 7 will find timeline φ' such that minimize value $\mathcal{S}(\varphi' \backslash \varphi)$ and covers all the edges (u, v, t) where $u, v \in X'$ for some x' in a subtree of node X.*

Proof. We proof the lemma by induction base on the depth of node X in tree T. For the leaf X node, which is the base case of our induction, since there is only one vertex in $\beta(X)$, any activity timeline will cover all the edges.

Now we consider three cases of node X in the graph. Then for all the nodes X.

Algorithm 8: Polynomial-time algorithm for temporal graph which has underlying graph bounded treewidth

Function `FindTimeline(`G`):`

Declare dynamic table $D(A, \varphi)$ of numbers;
Declare table $T(A, \varphi)$ of activity timeline;
$(T, \beta) \leftarrow$ a nice tree decomposition of G;
$R \leftarrow$ root of T;
$minValue \leftarrow \infty$;
$T \leftarrow \varnothing$;
`computeMinTimelineRecursively(`(T, β, R)`);`
for $\varphi \leftarrow \{I_u\}_{u \in \beta(R)}$ *that* φ *covers* R **do**
  if $minValue > D(R, \varphi) + \mathcal{S}(\varphi)$ **then**
    $minValue > D(R, \varphi) + \mathcal{S}(\varphi)$;
  end
end
return $(T(R, \varphi))$;

- **Case 1**. If X is a forget node and Y is a child of X. Let $\{v\} = \beta(Y) \backslash \beta(x)$. Base on the assumption of induction for all φ and I_v such that φ covers X and $\varphi \cup \{I_v\}$ covers Y, $T(Y, \varphi \cup \{I_v\})$ will cover all the edges in subtree Y which implies that $T(X, \varphi)$ will cover all the edges in subtree X. On the other hand the algorithm computes $D(Y, \varphi \cup \{I_v\})$ correctly, which implies that the algorithm will compute $D(X, \varphi)$ correctly since we consider all the possible value for the interval of vertex v.

- **Case 2**. If X is an introduce node and Y is a child of X. Let $\{v\} = \beta(X) \backslash \beta(Y)$. Since φ covers X, $\varphi \backslash \{I_v\}$ covers Y. Therefore $D(X, \varphi) = D(Y, \varphi \backslash \{I_v\})$ and $T(X, \varphi) = T(Y, \varphi \backslash \{I_v\}) \cup \{I_v\}$ where $I_v \in \varphi$.

- **Case 3**. If X is a disjoint node with children X_l and X_r. Since $\beta(X) = \beta(X_l) = \beta(X_r)$, any vertex v in the subtree of X which $x \notin \beta(X)$ is only appears in on of the subtree of X_l or X_r. Therefore, $D(X, \varphi) = D(X_r, \varphi) + D(X_l, \varphi)$ and $T(X, \varphi) = T(X_r, \varphi) \cup T(X_l, \varphi)$.

□

To find the activity timeline that minimizes the total span and covers all the edges of the graph. First, we run the algorithm 7 from the root. Then we find the $\varphi = \{I_u\}_{u\in\beta(R)}$ that minimize the value of $D(R,\varphi) + \mathcal{S}(\varphi)$, and we return $T(R,\varphi)$ as a solution of algorithm. Algorithm 8 shows how to find the solution.

Theorem 21. *MinTimeline problem is solvable in time* $O(|E|^{2tw(G\downarrow)+2}|V|)$.

Proof. Let φ_{opt} an activity timeline that covers all the edges and minimize the total span. Suppose that φ^R_{opt} is equal to $\{I_v\}_{v\in\beta(R)}$ such tat for each $v \in \beta(R)$ an interval I_v belongs to φ. Base on the Lemma 21 algorithm 7 will finds the φ_{opt} and $D(R,\varphi^R_{opt})$ will be equal to $\mathcal{S}(\varphi_{opt}) - \mathcal{S}(\varphi^R_{opt})$. Let φ be the result of algorithm 8. Since we consider φ^R_{opt} as a timeline in algorithm 8:

$$\mathcal{S}(\varphi) \leq \mathcal{S}(\varphi_{opt}) - \mathcal{S}(\varphi^R_{opt}) + \mathcal{S}(\varphi^R_{opt})$$

On the other hand, from Lemma 21 , we could conclude that φ will cover all the edges in the subtree of R, which means φ will cover all the edges in the temporal graph.

It is sufficient to consider only intervals that there exists an edge in their starting point and ending point of time. So, for each vertex the number of intervals that we need to consider is at most $|E|^2$. Therefore, in Algorithm 8 the value of D and T is need to compute for $|V| \times |E|^{2tw(G_\downarrow)}$. To do that we consider all possibility for at most one interval which takes $O(|E|^2)$ times. Therefore the algorithm takes $O(|V||E|^{2tw(G_\downarrow)+2})$ time. □

Corollary 10. *Finding an activity timeline that minimizes total span is solvable in polynomial time for a temporal graph with bounded treewidth.*

Chapter 5

Conclusions and Future Work

In this work, we defined a (s, z, t)-Temporal Separator problem, generalizing the problem called (s, z)-Temporal Separator. 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 (s, z, t)-Temporal Separator. The reduction from Vertex Cover to (s, z)-Temporal Separator problem was presented by Zschoche et al. [51], but it does not imply any lower bound on the approximation ratio for (s, z)-Temporal Separator. This is because in the reduction solutions of size $n + k$ to (s, z)-Temporal Separator correspond to solutions of size k to Vertex Cover instances.

Many hard problems including Vertex Cover, Independent Set, and Dominating Set become polynomial time solvable when we restrict the input to graphs with bounded treewidth [4]. However, some problems, e.g., Temporal Vertex Cover, remain **NP**-complete even when we restrict the input to a tree or a star [1]. The question of whether there is any polynomial-time algorithm to compute a minimum (s, z, t)-temporal separator in a temporal graph of bounded treewidth remains open. However, we showed a reduction from the DISC-SC problem when all segments' lengths are bounded by a constant number to (s, z, t)-Temporal Separator when the pathwidth of the underlying graph is bounded by a constant number. Therefore, the question that arises is if both problems are polynomial time equivalent or not.

In addition, we show that the (s, z, t)-Temporal Separator problem can be solved in polynomial time when the underlying graph has branchwidth at most two, each biconnected component is a series-parallel graph, or the subgraph which contains

every node except source and terminal is a tree. Moreover, our algorithm will work if we want to solve a generalized problem of Temporal Separator when it could be computed if there is a restricted temporal path from s to z in polynomial time.

In this thesis, we investigated one generalization of a restriction on (s, z)-temporal paths. By restricting temporal paths in another way, the (s, z)-Temporal Separator problem could be generalized differently. For instance, another approach is to restrict temporal paths by their unweighted lengths. Meaning that the goal is to find a small set of vertices, removing which either disconnects s from z or leaves only those paths between s and z that have length greater than ℓ. This generalization of the Temporal Separator problem could be studied in future work.

The problem of finding an activity timeline that covers all edges and minimizes the total span is **NP**-Complete [43]. However, we showed that this problem is solvable in polynomial time on a temporal graph with bounded treewidth. Also, the question of whether there exists a polynomial-time approximation algorithm remains open. The 2-approximation algorithm for the problem when we bound every interval to contains a specific time was given by [43] using linear programming. In this thesis, we presented a purely combinatorial 2-approximation algorithm inspired by a double coverage algorithm for the k-Server problem on a line.

Bibliography

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

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

[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] S Arnborg and A Proskurowski. Linear time algorithms for np-hard problems on graph embedded in k-trees. *Discrete Applied Math*, 23(1):l–24, 1989.

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

[6] Brenda Baker. Gossips and telephones. *Discrete Mathematics*, 2(3):191–193, 1972.

[7] 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.

[8] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

[9] Arnaud Casteigts, Paola Flocchini, Bernard Mans, and Nicola Santoro. Shortest, fastest, and foremost broadcast in dynamic networks. *International Journal of Foundations of Computer Science*, 26(4):499–522, 2015.

[10] 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.

[11] Marek Chrobak, H Karloof, Tom Payne, and Sundar Vishwnathan. New ressults on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.

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

[13] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[14] 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.

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

[16] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

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

[18] 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.

[19] 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.

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

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

[22] 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.

[23] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[24] Lester Randolph Ford Jr and Delbert Ray Fulkerson. Flows in networks, princeton universitypress, princeton, nj. *FordFlows in Networks1962*, 1962.

[25] 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.

[26] Saul I Gass and Arjang A Assad. *An annotated timeline of operations research: An informal history*, volume 75. Springer Science & Business Media, 2005.

[27] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[28] T.E. Harris and F.S. Ross. Fundamentals of a method for evaluating rail net capacities. Technical report, RAND CORP SANTA MONICA CA, 1955.

[29] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 217–226. SIAM, 2014.

[30] 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.

[31] Helen Knight. New algorithm can dramatically streamline solutions to the 'max flow'problem. *MIT News*, 4:21–26, 2014.

[32] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522, 2010.

[33] Fabian Kuhn and Rotem Oshman. Dynamic networks: models and algorithms. *ACM SIGACT News*, 42(1):82–96, 2011.

[34] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[35] 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.

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

[37] Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *Journal of Parallel and Distributed Computing*, 74(1):2016–2026, 2014.

[38] Regina O'Dell and Rogert Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pages 104–110, 2005.

[39] James B Orlin. Max flows in o (nm) time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 765–774, 2013.

[40] Ramamoorthi Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 202–213. IEEE, 1994.

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

[42] 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.

[43] Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. The network-untangling problem: From interactions to activity timelines. *Data Mining and Knowledge Discovery*, pages 1–35, 2020.

[44] Alexander Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

[45] Jonah Sherman. Nearly maximum flows in nearly linear time. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 263–269. IEEE, 2013.

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

[47] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.

[48] 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.

[49] 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.

[50] 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.

[51] 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.