Technische Universität Berlin
Electrical Engineering and Computer Science
Institute of Software Engineering and Theoretical Computer Science
Algorithmics and Computational Complexity (AKT)

Feedback Sets in Temporal Graphs

Masterarbeit

von Roman Haag

zur Erlangung des Grades „Master of Science“ (M. Sc.)
im Studiengang Computer Science (Informatik)

Erstgutachter:	Prof. Dr. Rolf Niedermeier
Zweitgutachter:	Prof. Dr. Markus Brill
Betreuer:	Hendrik Molter, Malte Renken, Prof. Dr. Rolf Niedermeier

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbstständige und eigenständige Anfertigung versichert an Eides statt:

Ort, Datum	Unterschrift

Abstract

In this thesis, we define two temporal analogues to the well-known Feedback Edge Set problem and study their parameterized complexity. Our work is based on an established model called a *temporal graph* which consists of a set of vertices V, a lifetime $\tau \in \mathbb{N}$, and set of time-labeled edges $E \subseteq \binom{V}{2} \times [\tau]$. The *underlying graph* of a temporal graph is the static graph obtained by removing all time-labels and keeping only one edge from every multi-edge. A sequence of time-labeled edges that form a closed walk in the underlying graph is called a *temporal tour*, if its time-labels are strictly increasing (*strict* case) or non-decreasing (*non-strict* case). We distinguish between two variants to define feedback sets, i.e., sets whose deletion makes the graph tour-free: *time-edge deletion* removes an edge at one specific point in time and *connection deletion* removes all edges between a pair of vertices. We will call the corresponding problems of finding such deletion sets with minimum cardinality (Strict) Temporal Feedback Edge Set ((S)TFES) and (Strict) Temporal Feedback Connection Set ((S)TFES), respectively.

In contrast to the polynomial-time results known for the static case, we show that (S)TFES and (S)TFCS are NP-hard and presumably do not admit an XP algorithm when parameterized by the lifetime τ of the temporal graph. For the parameter solution size k we prove W[1]-hardness for all variants. Combining both parameters, we achieve a positive result and present a simple search tree algorithm solving the strict variants STFES and STFCS in $\mathcal{O}(\tau^k \cdot |V| \cdot |E|^2)$ time. Our main algorithmic contribution is a dynamic program which solves STFES in $\mathcal{O}(2^{2|V|^2} \cdot |V|^3 \cdot \tau)$ time and TFES in $\mathcal{O}(2^{3|V|^2} \cdot |V|^2 \cdot \tau)$ time, thus proving the fixed-parameter tractability of (S)TFES for the parameter $|V|$ (a result which is obtained trivially for (S)TFCS).

Zusammenfassung

In dieser Arbeit definieren wir zwei temporale Varianten des bekannten Feedback Edge Set Problems und studieren ihre parameterisierte Komplexität. Wir arbeiten mit dem als *temporaler Graph* bekannten Modell, welches aus einer Knotenmenge V, einer Lebensdauer $\tau \in \mathbb{N}$ und einer Zeitkantenmenge $E \subseteq \binom{V}{2} \times [\tau]$ besteht. Der *unterliegende Graph* eines temporalen Graphen ist der statische Graph, den man erhält, wenn man die Zeitstempel der Zeitkanten entfernt und von jeder entstandenen Mehrfachkante nur eine Kante behält. Eine Sequenz von Zeitkanten, die einen geschlossenen Weg im unterliegenden Graphen bildet, nennen wir eine *temporale Tour*, wenn ihre Zeitstempel strikt ansteigend (*strikter* Fall) oder nicht fallend (*nicht-strikter* Fall) sind. Um Feedback Sets, also Mengen deren Entfernung alle temporalen Touren zerstört, zu definieren, unterscheiden wir zwei Varianten der Kantenentfernung: *Zeitkantenentfernung* löscht eine einzelne Kante an einem bestimmten Zeitpunkt und *Verbindungsentfernung* löscht alle Kanten zwischen einem Knotenpaar. Wir nennen die Probleme, entsprechende Entfernungsmengen mit minimaler Größe zu finden, (Strict) Temporal Feedback Edge Set ((S)TFES) und (Strict) Temporal Feedback Connection Set ((S)TFES).

Entgegengesetzt zu den bekannten Polynomialzeitergebnissen für den statischen Fall zeigen wir, dass (S)TFES und (S)TFCS NP-schwer sind und vermutlich keinen XP-Algorithmus für den Parameter "Lebensdauer τ" besitzen. Für den Parameter "Lösungsgröße k" beweisen wir W[1]-Härte für alle Varianten. Für die Kombination aus diesen beiden Parametern erhalten wir ein positives Ergebnis und präsentieren einen einfachen Suchbaumalgorithmus, der die strikten Varianten STFES und STFCS in der Zeit $\mathcal{O}(\tau^k \cdot |V| \cdot |E|^2)$ löst. Der aus algorithmischer Sicht wichtigste Beitrag unserer Arbeit ist ein dynamisches Programm, das STFES in der Zeit $\mathcal{O}(2^{2|V|^2} \cdot |V|^3 \cdot \tau)$ und TFES in der Zeit $\mathcal{O}(2^{3|V|^2} \cdot |V|^2 \cdot \tau)$ löst, was beweist, dass (S)TFES "fixed-parameter tractable" für den Parameter $|V|$ ist (für (S)TFCS ist dasselbe Ergebnis trivial).

Contents

1 Introduction

Computer scientists and mathematicians model network-like data as *graphs* which consist of nodes, called vertices, and the connections between them, called edges. The countless applications of this model (e.g., analysis of transportation and social networks) have encouraged decades of research. However, many real-world networks are not static but evolve over time. This leads to the study of *temporal graphs* whose edges are only present at certain times, a property which occurs naturally in many applications such as time schedules of transportation networks or communication data with time stamps.

We use an established temporal graph model (see e.g., [AF16; KKK02; MMS19; MS16; Zsc+18]) where the vertex set V is fixed and each time-edge in the edge set E has a discrete time-label $t \in \{1, 2, \ldots, \tau\}$ where τ denotes the *lifetime* of the temporal graph. Figure 1.1 shows a small example with four time layers ($\tau = 4$). If we assume that this temporal graph represents a schedule for a public bus transportation system, then we can see that buses will be available, e.g., between stations a and b at 1 and 2 o'clock ($t = 1, 2$). Informally speaking, a *temporal tour* in such a network is a time-respecting sequence of edges starting and ending at the same vertex. For example, the graph in Figure 1.1 contains, among others, the temporal tours $a \to b \to a$ and $b \to a \to b$, both starting at $t = 1$ and ending at $t = 2$. Our main research topic is the search for small *feedback sets*, i.e., sets of vertices or edges whose removal from the temporal graph destroys all tours. In this work, we will consider the following three types of feedback sets.

Temporal feedback edge set. These sets contain time-edges, that is, connections between two specific vertices at a specific time. In the bus network example, this could represent a road that is temporarily blocked (i.e., for one time slot $t \in \tau$). The graph in Figure 1.1 shows such a set as dashed lines in the layer representation in the bottom row.

Temporal feedback connection set. These sets contain vertex pairs $\{v, w\}$ which define that all time-edges between v and w will be removed. In our example, this corresponds to a permanently blocked road (i.e., for the lifetime of the temporal graph).

Temporal feedback vertex set. These sets contain vertices to be removed along with all adjacent time-edges. In our example, such a set could represent bus stations that are out of service during the lifetime of the temporal graph.

One motivation for the study of temporal feedback sets is the following application in fraud detection: Consider the trading history of some product or trade good (e.g., stocks of one company) given as a temporal graph. Here, the vertices are agents in the market (i.e., buyers/sellers) and an edge between two agents with time label t represents a trade between them on day t. If a sequence of trades forms a tour, then this might indicate a common type of fraud: artificially increasing trade volume by buying your own goods via

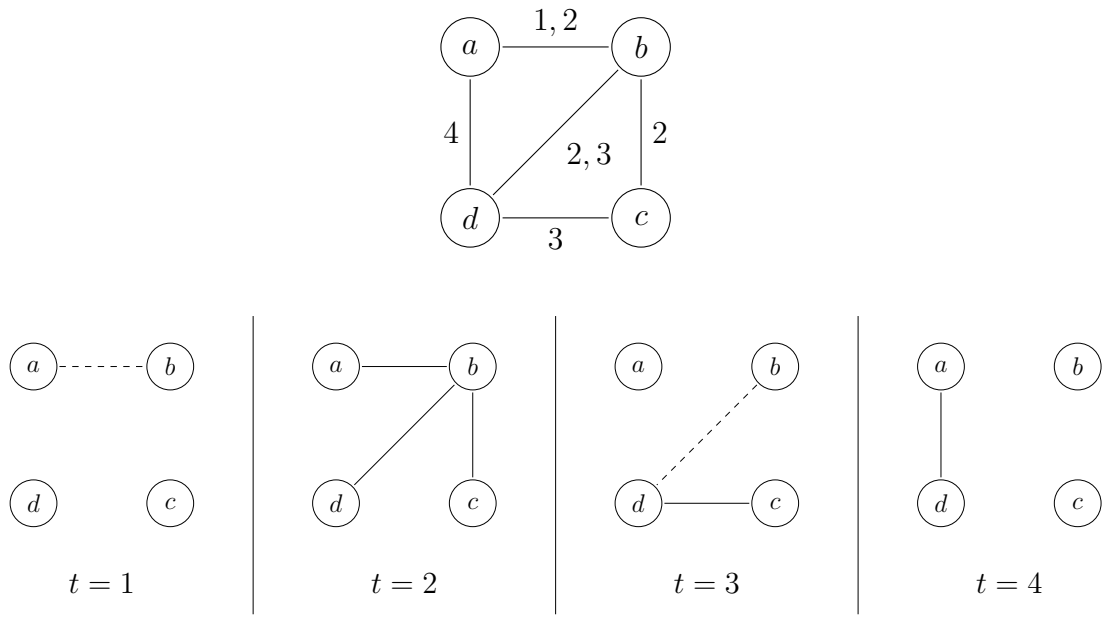

Figure 1.1: A temporal graph with time-labeled edges (top) and its four time layers (bottom). The two dashed edges form a strict temporal feedback edge set of minimal size.

intermediaries. From a computational perspective, occurrences of this can be detected rather easily in $\mathcal{O}(|E|^2 \cdot |V|)$ time (as will be shown in Lemma 3.2). A more difficult problem is to measure the amount of fraud, i.e., the number of transactions or agents that, if removed, leave a graph without tours. These form temporal feedback sets of the temporal graph.

In static graphs, feedback set problems have been extensively studied and have known applications in circuit design [Joh75], program verification [Sha79], deadlock prevention [WLS85], and Bayesian inference [BJG08]. Whenever the application contains a temporal component (e.g., resource access times in deadlock prevention), including this temporal data by considering the corresponding temporal feedback set problem might provide additional insight. Analogously to static graphs, the size of the smallest solution is also an interesting structural parameter of a given temporal graph. For time-edge deletion, we will call this parameter the *temporal feedback edge number*. When the application implies that temporal tours are unwanted (as in the fraud detection example above), the temporal feedback edge number gives us a measurement for the magnitude of the problem. On the other hand, if tours represent desired connections as, e.g., in a public transportation network, then the feedback edge number can serve as an indicator for the robustness or fault tolerance of network.

Defining feedback set problems in temporal graphs is not straight-forward and includes a set of decisions with multiple viable choices depending on the application. We now introduce and motivate these choices and the resulting problem variants (formal

definitions are presented in Chapter 2).

First, we note that we only consider *undirected* temporal graphs, that is, we assume that each time-edge represents a connection available in both directions. However, this assumption does not hold for many applications (e.g., for most transportation networks). While this simplification constitutes a limitation of our model, it is commonly used as starting point for research as the directed problem variants are often harder to solve. E.g., in static graphs, the (directed) Feedback Arc Set Problem is NP-hard [Kar72] while the undirected variant can be solved in polynomial time by computing a spanning tree.

Two temporal walk models. Temporal tours are a special case of the more general concept of *temporal walks* which are time-respecting connections between two vertices. Throughout this work, we will distinguish between the following two models for such time-respecting connections. *Strict* temporal walks have strictly increasing time labels on consecutive time-edges (an obvious requirement for, e.g., travel in a transportation network). *Non-strict* temporal walks have non-decreasing time labels on consecutive time-edges. This model can be used whenever the traversal time per edge is very short compared to the scale of the time dimension (e.g., sending messages in a communication network or propagation of electric current in circuits). More complex variants also exist, e.g., with individual traversal times for each time-edge and restrictions on the dwell time in each vertex [Him18], but are not considered in this work.

A special type of temporal walks, called a *temporal path*, is often considered in research (e.g., [AF16; KKK02; Zsc+18]). For paths, we have the additional requirement that all vertices must be pairwise distinct. We will call the corresponding special case of a temporal tour a *temporal cycle*. Comparing both models, we believe that temporal tours represent real-world applications more faithfully than cycles. For instance, when we consider a public transportation network, it is reasonable to assume that people travel through the network towards some destination and then take the same route back home (visiting the same vertices). In a market network, items might go back and forth between owners due to miscalculations or unexpected price changes (or to commit fraud as described above). However, most of our results also extend to temporal cycles and we will discuss the required changes at the end of the corresponding proofs.

Problem variants. We will focus on finding temporal feedback edge sets and temporal feedback connection sets (formalized in Chapter 2) of minimum cardinality, each using both the strict and non-strict temporal walk model. We call the corresponding problems (Strict) Temporal Feedback Edge Set and (Strict) Temporal Feedback Connection Set, respectively.

(Strict) Temporal Feedback Edge Set - (S)TFES
Input: A temporal graph $G = (V, E, \tau)$ and $k \in \mathbb{N}$.
Question: Is there a (strict) temporal feedback edge set E' of G with $|E'| \leq k$?

(Strict) Temporal Feedback Connection Set - (S)TFCS
Input: A temporal graph $G = (V, E, \tau)$ and $k \in \mathbb{N}$.
Question: Is there a (strict) temporal feedback connection set C' of G with $|C'| \leq k$?

We chose not to include a formal analysis of temporal feedback vertex sets as this problem is already NP-hard in static graphs [Kar72], but see Chapter 6 for a short overview on how our results might be extended to the (Strict) Temporal Feedback Vertex Set problem.

1.1 Related Work

To the best of our knowledge, the problems introduced above have not been studied before. Agrawal et al. [Agr+18] study feedback vertex sets in *α-edge-colored* graphs, a graph model where the edge set is partitioned into α color classes, and obtain an $\mathcal{O}^*(23^{\alpha k})$ time[1] algorithm for the NP-hard $\alpha-$Simultaneous Feedback Vertex Set problem. While both edge-colored and temporal graphs consist of multiple edge layers connecting a fixed vertex set, the ordering (over time) of these layers is only considered in temporal graphs. More precisely, the cycles considered by $\alpha-$Simultaneous Feedback Vertex Set must exist in a single layer while temporal tours/cycles can stretch over multiple layers.

Our work falls into the general context of analyzing temporal variants of well-studied static graph problems. Among the first works in this category is the research on temporal analogues of Menger's Theorem by Berman [Ber96]. For static graphs, this theorem states that, between any pair of vertices, the size of a minimum cut set is equal to the maximum number of disjoint paths and holds for both vertex and edge cut sets. Cut set problems are closely related to feedback set problems as both ask for optimal ways to destroy certain connections in a graph (making cut set problems good candidates to reduce from when attempting to prove computational hardness). Berman [Ber96] showed that the time-edge version of Menger's Theorem also holds in temporal graphs while the vertex version does not. Continuing the work on temporal analogues of Menger's Theorem, Kempe, Kleinberg, and Kumar [KKK02] showed that the vertex version of the theorem does hold, "in the spirit of Kuratowski's theorem", in temporal graphs that do not contain a subdivision of a specific temporal graph. For such graphs, they provided a polynomial-time algorithm computing the maximum number of time-respecting vertex-disjoint paths between two vertices. They further showed that the Temporal (s, z)-Separation problem, which asks for a minimum cardinality vertex separator between two vertices s and z, is NP-hard for $\tau \geq 12$. Zschoche et al. [Zsc+18] later completed the analysis of this boundary by showing NP-completeness for $\tau \geq 5$ and polynomial-time solvability for $\tau < 5$ for the strict problem variant and NP-completeness for $\tau \geq 2$ for the non-strict variant.

For static graphs, it is well known that removing a minimum size feedback edge set from the graph results in a spanning tree. Given this duality, it is natural to compare temporal feedback edge sets to the temporal analogue of a spanning tree. This analogue is known as the *minimum*[2] *temporally connected (sub)graph*, which is a graph containing

[1] The $\mathcal{O}^*$-notation suppresses polynomial time factors. The variable k denotes the size of the feedback vertex set (solution size).

[2] After removing any time-edge, the graph is not temporally connected anymore.

a time-respecting path from each vertex to every other vertex. The concept was first introduced by Kempe, Kleinberg, and Kumar [KKK02], together with an open question on the maximum density of time-edges that is possible in an n-vertex minimum temporally connected graph. The question was resolved by Axiotis and Fotakis [AF16] who showed that the maximum number of time-edges is in $\Omega(n^2)$. Additionally, both Axiotis and Fotakis [AF16] and Akrida et al. [Akr+15] provide proofs showing that, given a temporal graph, computing a minimum temporally connected subgraph is APX-hard (lower and uppers bound on the approximation ratio are presented in [AF16]).

Further examples for the analysis of classic graph problems in the temporal setting include works on the Traveling Salesman Problem by Michail and Spirakis [MS16], on the Clique Problem by Himmel et al. [Him+17] and Viard, Latapy, and Magnien [VLM16], an on the Vertex Cover Problem by Akrida et al. [Akr+18].

1.2 Our Contributions

The results presented in this thesis are divided into the two categories computational hardness results (Chapter 4) and algorithmic results (Chapter 5).

On the topic of computational hardness, we obtain multiple results based on a reduction from 3-SAT to all four problem variants (Section 4.1). While the reduction shows NP-hardness directly, the properties of the constructed temporal graph offer additional implications on the parameterized complexity of the problems. More precisely, the constructed graph uses $\tau = 8$ distinct time labels for the strict variants and $\tau = 3$ labels for the non-strict variants, showing that there can be no XP algorithm for the parameter lifetime τ of the graph, unless P = NP. The same argument holds if we consider the parameter "maximum number of time-edges between any pair of vertices", which has the value 1 in the constructed graph. This also implies that the problems remain NP-hard when restricted to *simple* temporal graphs, i.e., temporal graphs with at most one time-edge between any pair of vertices. Based on the Exponential Time Hypothesis, we derive an additional result stating that there is presumably no subexponential time algorithm solving (S)TFES or (S)TFCS. In Section 4.2, we show that all four problem variants are W[1]-hard when parameterized by the solution size k by reducing from the problem Directed Multicut in DAGs. A key idea of our proof is that DAGs can transformed into undirected temporal graphs with equivalent reachability. This is accomplished by using an *acyclic ordering* (also known as topological ordering) to add time-labels to the edges in such a way that the direction of original arcs is preserved.

On the positive side (algorithmic results), we are able to show in Section 5.1 that all problem variants are fixed-parameter tractable with respect to the combined parameter $L + k$ where L is the maximum length of a *minimal* temporal tour, i.e., a tour that does not contain any subtour. The result is based on a simple search tree algorithm: For any temporal tour in the graph, we have to take one of its at most L edges into the feedback edge/connection set, resulting in a search tree of size $\mathcal{O}(L^k)$. One obvious upper bound for L is the number of vertices. For the strict problem variants, it is also clear that $L \leq \tau$, giving us an FPT result for the combined parameter $\tau + k$. Searching

Table 1.1: Overview of our results for (Strict) Temporal Feedback Edge Set (marked with *) and (Strict) Temporal Feedback Connection Set (marked with **). Unmarked results apply to both variants. The parameters k and τ denote the solution size and the lifetime of the temporal graph, respectively. L denotes the maximum length of a minimal temporal tour.

Parameterization	Complexity	
	Strict variant	Non-strict variant
none	NP-hard [Thm. 4.1*/Cor. 4.2**]	NP-hard [Cor. 4.3]
k	W[1]-hard [Thm. 4.5]	W[1]-hard [Thm. 4.5]
τ	$\tau \geq 8$: NP-hard [Thm. 4.1*/Cor. 4.2**]	$\tau \geq 3$: NP-hard [Cor. 4.3]
$k + L$	$\mathcal{O}(L^k \cdot \lvert V\rvert \cdot \lvert E\rvert^2)$ [Proposition 5.1]	$\mathcal{O}(L^k \cdot \lvert V\rvert \cdot \lvert E\rvert^2)$ [Proposition 5.1]
$k + \tau$	$\mathcal{O}(\tau^k \cdot \lvert V\rvert \cdot \lvert E\rvert^2)$ [Cor. 5.3]	open
$\lvert V\rvert$	$\mathcal{O}(2^{2\lvert V\rvert^2} \cdot \lvert V\rvert^3 \cdot \tau)$* [Thm. 5.5*] $\mathcal{O}(2^{\frac{1}{2}(\lvert V\rvert^2-\lvert V\rvert)} \cdot \lvert V\rvert \cdot \lvert E\rvert^2)$**	$\mathcal{O}(2^{3\lvert V\rvert^2} \cdot \lvert V\rvert^2 \cdot \tau)$* [Thm. 5.5*] $\mathcal{O}(2^{\frac{1}{2}(\lvert V\rvert^2-\lvert V\rvert)} \cdot \lvert V\rvert \cdot \lvert E\rvert^2)$**

for additional upper bounds for L, we analyze the structure of the *underlying graph*, i.e., the static graph obtained by removing the time-labels and keeping only one edge from every multi-edge. We show that L can be upper-bounded by a function of the *vertex cover number* of the underlying graph, but not by a function of the *feedback vertex number*. In Section 5.2, we prove the fixed-parameter tractability of (S)TFES with respect to the number of vertices $|V|$ (for (S)TFCS, the same result is obtained trivially as there are $\frac{1}{2}(|V|^2 - |V|)$ vertex pairs to consider).

Our results are summarized in Table 1.1. Overall, we achieved similar results for all problem variants suggesting that the models are equally expressive/useful. However, there are a few noteworthy exceptions. Intuitively, the non-strict variants seem more complex as there are more options to construct temporal tours. This is supported by our result for the combined parameter $k+\tau$ for which the non-strict case remains unsolved as we cannot use τ to upper-bound the length of temporal tours. Comparing feedback edge sets to feedback connection sets, we note a disparity for the parameter number of vertices $|V|$. While both (S)TFES and (S)TFCS are fixed-parameter tractable with respect to $|V|$, proving this result for (S)TFES was much more involved, because the maximum number of time-edges depends on τ and $|V|$ whereas the number of "connections" only

depends on $|V|$.

For the parameter lifetime τ, it remains open whether there exists a polynomial-time algorithm for instances with $3 \leq \tau \leq 7$ in the strict case and $\tau = 2$ in the non-strict case.

2 Preliminaries

In this chapter, we introduce the notations and concepts used in this work.

We denote the set of natural numbers (not including zero) with $\mathbb{N}$. For $a \in \mathbb{N}$, we write $[a]$ to refer to the set $\{1, \ldots, a\}$.

2.1 Static and Temporal Graphs

For static graphs, we use established basic notations from graph theory [Die12]. Let $G = (V, E)$ be an undirected (static) graph. We denote the vertex set of G with $V(G)$ and the edge set with $E(G)$. For a vertex set $V' \subseteq V$, we write $G[V'] := \{V', \{\{v, w\} \in E \mid v, w \in V'\}\}$ to refer to the subgraph of G *induced* by V'. The graph without the vertices V' is denoted by $G - V' := G[V \setminus V']$. For an edge set $E' \subseteq E$, the graph without the edges in E' is denoted by $G - E' := (V, E \setminus E')$.

Let $D = (V, A)$ be a directed graph. A *cycle* in D is a sequence of arcs $C := (a_1, a_2, \ldots, a_\ell)$ with $a_i = (v_i, v_{i+1}) \in A$ for $i \in [\ell]$ in which all vertices are pairwise distinct except for $v_1 = v_{\ell+1}$. A directed graph that contains no cycle is called a *directed acyclic graph (DAG)*.

Temporal Graphs. Formally, we work with the following definition in which the vertex set does not change with time and each time-edge has a discrete time label.

Definition 2.1. (Temporal Graph, Underlying Graph). A (undirected) *temporal graph* $G = (V, E, \tau)$ is an ordered triple consisting of a set of vertices V, a set $E \subseteq \binom{V}{2} \times [\tau]$ of *time-edges*, and a lifetime $\tau \in \mathbb{N}$. The *underlying graph* $G_\downarrow$ is the static graph obtained by removing all time labels from G and keeping only one edge from every multi-edge. We call a temporal graph *simple* if each vertex pair is connected by at most one time-edge.

In the literature, temporal graphs are also known as time-varying [KRP19; LM17] and evolving graphs [Fer04], temporal networks [Hol18; KKK02; MMS19], edge-scheduled networks [Ber96], and link streams [LVM18; VLM16].

Let $G = (V, E, \tau)$ be a temporal graph. For $i \in [\tau]$, let $E_i(G) := \{\{v, w\} \mid (\{v, w\}, i) \in E\}$ be the set of edges with time label i. We call the static graph $G_i(G) = (V, E_i(G))$ *layer* i of G. For $t \in [\tau]$, we denote the temporal subgraph consisting of the first t layers of G as $G_{[t]}(G) := (V, \{(e, i) \mid i \in [t] \wedge e \in E_i(G)\}, t)$. We omit the function parameter G if it is clear from context.

Definition 2.2. (Temporal walk, temporal path). Given a temporal graph $G = (V, E, \tau)$, a *temporal walk* of length ℓ in G is a sequence $P = (e_1, e_2, \ldots, e_\ell)$ of time-edges $e_i = (\{v_i, v_{i+1}\}, t_i) \in E$ where $e_i \neq e_j$ for all $i, j \in [\ell]$ and $t_i \leq t_{i+1}$ for all $i \in [\ell - 1]$. If a

temporal walk P has the property $v_i \neq v_j$ for all $i, j \in [\ell+1]$, then P is a *temporal path*. A temporal walk or path is called *strict* if $t_i < t_{i+1}$ for all $i \in [\ell-1]$.

The following definitions and are based on temporal walks and paths and, thus, all have strict and non-strict versions. We write "(strict)" before edge sets and problem names to refer to both versions at once.

Definition 2.3. (Temporal tour, temporal cycle). Let $G = (V, E, \tau)$ be a temporal graph. A temporal tour in G is a temporal walk $P = (\{\{v_1, v_2\}, t_1), (\{v_2, v_3\}, t_2), \ldots, (\{v_\ell, v_{\ell+1}\}, t_\ell))$ with $v_1 = v_{\ell+1}$ and $\ell \geq 2$. A temporal tour is called *minimal* if it does not contain a subtour, i.e., there are no $i, j \in [\ell]$ such that $P' = (\{\{v_i, v_{i+1}\}, t_i), (\{v_{i+1}, v_{i+2}\}, t_{i+1}), \ldots, (\{v_j, v_{j+1}\}, t_j))$ is temporal tour. A *temporal cycle* is a minimal temporal tour of length $\ell \geq 3$.

Note that temporal walks may use each time-edge only once, and thus a single time-edge does not form a non-strict temporal tour. Also note that a minimal temporal tour has either length $\ell = 2$ (i.e., it has the form $v \to w \to v$) or all vertices except the first and last are pairwise distinct and it is a temporal cycle.

The definitions of (Strict) Temporal Feedback Edge Set and (Strict) Temporal Feedback Connection Set (see Chapter 1) are based on the following two sets (problem and set names are identical).

Definition 2.4. Let $G = (V, E, \tau)$ be a temporal graph. A time-edge set $E' \subseteq E$ is called a *(strict) temporal feedback edge set* of G if $G' = (V, E \setminus E', \tau)$ does not contain a (strict) temporal tour.

Definition 2.5. Let $G = (V, E, \tau)$ be a temporal graph with underlying graph $G_\downarrow = (V, E_\downarrow)$. An edge set $C' \subseteq E_\downarrow$ is a *(strict) temporal feedback connection set* of G if $G' = (V, E', \tau)$ with $E' = \{(\{v, u\}, t) \in E \mid \{v, u\} \notin C'\}$ does not contain a (strict) temporal tour.

The elements in a feedback connection set are known as *multi-edges* or *underlying edges* (edges of $G_\downarrow$). While we chose to deviate from this in the problem name "Temporal Feedback Connection Set", we will use the term underlying edges to refer to edges in C'.

2.2 Parameterized Complexity

We assume basic knowledge about complexity theory. In Chapter 4, we will prove that the problems introduced above are NP-hard. When dealing with the task of finding exact solutions to NP-hard problems, one of the options available in computer science is to perform a *parameterized* analysis. Here, we try to determine how certain properties of the problem input contribute to the computational hardness and search for efficient algorithms by exploiting knowledge about these properties.

Formally, a *parameterized (decision) problem* is defined as a language $L \subseteq \Sigma^* \times \Sigma^*$ where Σ^* is the set of all words over some finite alphabet Σ. Instances of L have the form

$(I, k) \in \Sigma^* \times \Sigma^*$ and we call the second component, k, the *parameter* of the problem. An instance $\mathcal{I} = (I, k)$ is a yes-instance if $\mathcal{I} \in P$ and a no-instance otherwise. An algorithm that decides whether $\mathcal{I}$ is a yes-instance in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some computable function f only depending on k is called an *FPT algorithm* for the problem L. If such an algorithm exists, then we say that L is *fixed-parameter tractable* and that it belongs to the corresponding complexity class FPT. Further, the complexity class XP is defined as the set of all parameterized problems which can be solved in $f(k) \cdot |I|^{g(k)}$ time with the functions f and g only depending on the parameter k. In between, we have a set of complexity classes called the W-hierarchy which we will not formally define here, but instead refer to the literature recommendations at the end of this section. Overall, we have the following hierarchy of parameterized complexity classes: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ XP. It is known that FPT $\subsetneq$ XP and conjectured that all other inclusions are also strict.

Analogously to the theory of NP-hardness, one can show parameterized hardness (i.e., presumably no FPT algorithm exists) via reduction from a problem assumed not to be in FPT. The technique is called *parameterized reduction* and we define it as follows. Let $L, L' \subseteq \Sigma^* \times \Sigma^*$ be two parameterized problems. A parameterized reduction from L to L' is a function $f : \Sigma^* \times \Sigma^* \to \Sigma^* \times \Sigma^* : (I, k) \mapsto (I', k')$ with the properties

1. $f(I, k)$ can be computed in $g(k) \cdot |I|^{\mathcal{O}(1)}$ time for some computable function g,
2. $k' \leq h(k)$ for some computable function h, and
3. $(I, k) \in L \Leftrightarrow (I', k') \in L'$.

For a more detailed introduction into this topic, we refer to the books by Downey and Fellows [DF13], Flum and Grohe [FG06], Cygan et al. [Cyg+15], and Niedermeier [Nie06].

3 Basic Observations

Before analyzing how temporal tours can be disconnected in Chapters 4 and 5, we describe a method to determine whether a given graph contains a temporal tour. In particular, we are interested in finding a shortest tour, that is, a tour of minimal length, which will be the basis for some of the algorithms shown Chapter 5. To this end, we will use the following result by Xuan, Ferreira, and Jarry [XFJ03] as a subroutine to find shortest temporal walks. Note that a shortest temporal walk is always a temporal path.

Theorem 3.1. [XFJ03] (Proposition 1). *Let $G = (V, E, \tau)$ be a temporal graph. For any vertex $v \in V$, we can compute a list of shortest temporal paths to all other vertices in $\mathcal{O}(|E| \cdot |V|)$ time.*

Remark. The original proposition in [XFJ03] states a running time of $\mathcal{O}(M \cdot D)$ where M is the number of edges in the underlying graph and D is the length of the longest temporal path (the "hop diameter"). Clearly, $M = |E_{\downarrow}| \leq |E|$ and $D \leq |V|$.

We now use Theorem 3.1 to find shortest temporal tours by fixing a starting vertex v and then computing the shortest path from each neighbor back to v.

Lemma 3.2. *Let $G = (V, E, \tau)$ be a temporal graph. In $\mathcal{O}(|E|^2 \cdot |V|)$ time, we can find a shortest (strict) temporal tour of G or confirm that G does not contain a (strict) temporal tour.*

Proof. For each vertex $v \in V$, we can find the shortest tour starting and ending at v as follows. For every edge $e = (\{v, w\}, t)$ incident to v, we fix e as first edge of a potential temporal tour and construct the temporal graph $G' := (G, E', \tau)$ with $E' = E \setminus \{\{e\} \cup \{(\{w, u\}, t') \in E \mid u \in V \wedge t' \leq t\}$ (we use $t' < t$ for the strict case). This ensures that no path in G' with w as first vertex starts before time t (the time label of our fixed time-edge e). If we can find a (strict) temporal path W from w to v in G' using Theorem 3.1, then $e + W$ is a temporal tour in G of length $|W| + 1$. We repeat this process for every vertex while storing the shortest tour. Overall, each edge will be considered twice (once for each endpoint), resulting in a running time of $\mathcal{O}(|E|^2 \cdot |V|)$. □

Note that a shortest temporal tour is always a minimal temporal tour (i.e., a tour not containing any subtour) but not necessarily a temporal cycle. The shortest possible tour has length two (visiting a neighbor of the starting vertex and then going back) while the shortest possible cycle has length three. However, we can observe that a shortest tour in a temporal graph has always length smaller than $|V|$.

Observation 3.3. *Let G be a temporal graph which is not tour-free and let C be a shortest temporal tour in G. Then, $|C| \leq |V|$.*

Proof. Let $C' = ((\{v_1, v_2\}, t_1), (\{v_2, v_3\}, t_2), \ldots, (\{v_\ell, v_1\}, t_\ell))$ be an arbitrary temporal tour in G. If $|C'| \leq |V|$, then we are done. In the other case, we know due to the pigeon hole principle that, for some $i, j \in [\ell]$ with $i < j$, we have $v_i = v_j$. Hence, $((\{v_i, v_{i+1}\}, t_i), (\{v_{i+1}, v_{i+2}\}, t_{i+1}), \ldots, (\{v_{j-1}, v_j\}, t_{j-1}))$ is a temporal tour of length $j - i < \ell$. The process can be repeated until the length of the resulting tour is at most $|V|$. □

In Section 4.1, we will show that (S)TFES and (S)TFCS are NP-hard for $\tau \geq 8$ (strict) and $\tau \geq 3$ (non-strict). For $\tau = 1$ however, all STFES and STFCS instances are yes-instances as a temporal graph with $\tau = 1$ contains no strict temporal tours. Furthermore, the non-strict variants TFES and TFCS are identical to the Feedback Edge Set Problem in static graphs which can easily be solved by computing a spanning tree.

Observation 3.4. *Let $G = (V, E, \tau)$ be a temporal graph with lifetime $\tau = 1$. Then, STFES and STFCS can be solved in constant time. Additionally, TFES and TFCS can be solved in $\mathcal{O}(|V| + |E|)$ time.*

For $\tau = 2$, we can solve STFES and STFCS easily due to the fact that strict temporal tours of length $\ell = 2$ must have the form $v \to w \to v$.

Observation 3.5. *Let $G = (V, E, \tau)$ be a temporal graph with lifetime $\tau = 2$. Then, STFES and STFCS can be solved in $\mathcal{O}(|V|^2)$ time.*

Proof. Clearly, all strict temporal tours in a temporal graph with $\tau = 2$ have the form $C = ((\{v, w\}, 1), (\{v, w\}, 2))$ for some $v, w \in V$. As both time-edges cannot be used in a tour other than C, taking either one them into an STFES solution is optimal. For STFCS, is is easy to see that the underlying edge $\{v, w\}$ must be in the solution. Assuming that we can find, for each pair of vertices (v, w), the at most two time-edges between v and w in constant time, we can construct the solution in $\mathcal{O}(|V|^2)$ time. □

As seen in the previous proof, any vertex pair which is connected by more than one time-edge, must be in the solution set of (S)TFCS. It is easy to see that, given an (S)TFCS instance, we can construct an equivalent instance in which no vertex pair is connected by more than one time-edge. In complexity theory, this concept is known as a *reduction rule*. However, we will not formalize this reduction rule as it will not be used in this work.

In contrast to static graphs, FPT results for the parameter $|V|$ are not always trivial in temporal graphs because the maximum number of time-edges $|E|$ also depends on the lifetime τ. However, the number of underlying edges depends only on $|V|$ which provides the following fixed-parameter tractability result for (S)TFCS.

Observation 3.6. *(S)TFCS can be solved in $\mathcal{O}(2^{\frac{1}{2}(|V|^2-|V|)} \cdot |V| \cdot |E|^2)$ time.*

Proof. Let G be a temporal graph with underlying graph $G_\downarrow = (V, E_\downarrow)$. As $G_\downarrow$ is a static graph, we have $|E_\downarrow| \leq \frac{1}{2}(|V|^2 - |V|)$. Thus, there are $2^{|E_\downarrow|} \leq 2^{\frac{1}{2}(|V|^2-|V|)}$ possible feedback connection sets, each of which can be tested in $\mathcal{O}(|E|^2 \cdot |V|)$ time (Lemma 3.2). □

4 Computational Hardness Results

In this chapter, we present our proofs for NP-hardness and parameterized hardness with respect to the parameters solution size k and lifetime τ of the temporal graph. We start with these results for two reasons. First, the parameters analyzed in this chapter appear directly in the problem definitions and can be assumed to be small for some real-world graphs. For instance, we can assume the vertex set to be small when we analyze the train schedule between cities with a population of, e.g., over one million. The lifetime of the graph mainly depends on the scale chosen for the time dimension and can be kept low by opting for coarse grained time labels (e.g., using days instead of hours). Second, the negative results for these parameters will motivate the analysis of larger (combined) parameters in Chapter 5.

4.1 NP-hardness and Parameterization by Lifetime of the Graph

In this section, we first prove the NP-hardness of (S)TFES and (S)TFCS by reducing from the well known 3-SAT problem. We then derive additional results by analyzing the properties of the temporal graph constructed for the reduction. Finally, we use the *Exponential Time Hypothesis* to show a lower bound result.

Theorem 4.1. Strict Temporal Feedback Edge Set (STFES) *is NP-hard even for simple temporal graphs with* $\tau \geq 8$.

Proof. We show NP-hardness via a polynomial-time many-one reduction from 3-SAT. For a boolean formula Φ in conjunctive normal form (CNF) with at most three variables per clause, 3-SAT asks if there is a satisfying truth assignment for Φ. 3-SAT is known to be NP-complete ([GJ79], [Kar72]). Let Φ be such a formula with variables $x_1, x_2, \ldots, x_n$ and clauses $c_1, c_2, \ldots, c_m$ of the form $c_j = (\ell_j^1 \vee \ell_j^2 \vee \ell_j^3)$. We construct an STFES instance with temporal graph $G(\Phi)$ and $k = n + 2m$ as follows.

For each variable x_i, we introduce a variable gadget (see Figure 4.1a) with vertices v_i, v_i^T, and v_i^F and edges $e_i^T := (\{v_i, v_i^T\}, 2)$, $e_i^F := (\{v_i, v_i^F\}, 3)$, and $e_i^h := (\{v_i^T, v_i^F\}, 1)$. As these three edges form the temporal tour (e_i^h, e_i^T, e_i^F), any solution for STFES must contain at least one of them. For each clause c_j, we introduce a clause gadget with four vertices, w_j, w_j^1, w_j^2, and w_j^3, and the edges $f_j^a = (\{w_j^1, w_j^2\}, 1)$, $f_j^b = (\{w_j^2, w_j^3\}, 2)$, $f_j^1 := (\{c_j, w_j^1\}, 7)$, $f_j^2 := (\{c_j, w_j^2\}, 6)$, $f_j^3 := (\{c_j, w_j^3\}, 5)$ (see Figure 4.1b). The clause gadget contains three tours which overlap in such a way that any solution has to contain at least two out of its five edges.

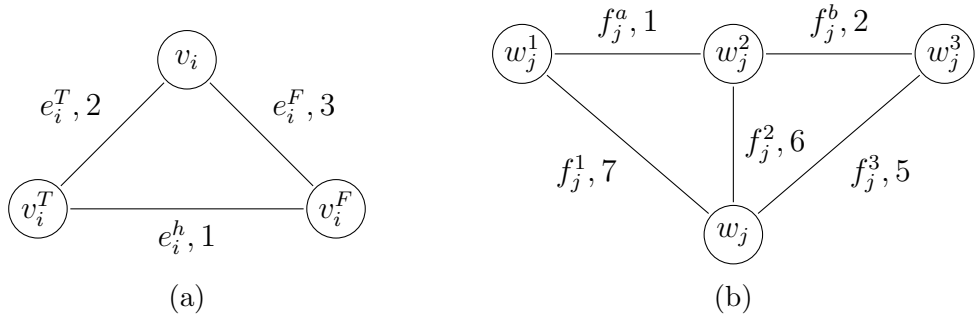

Figure 4.1: Variable gadget (a) and clause gadget (b) used in the proof of Theorem 4.1. Edges are labeled with their name and time label.

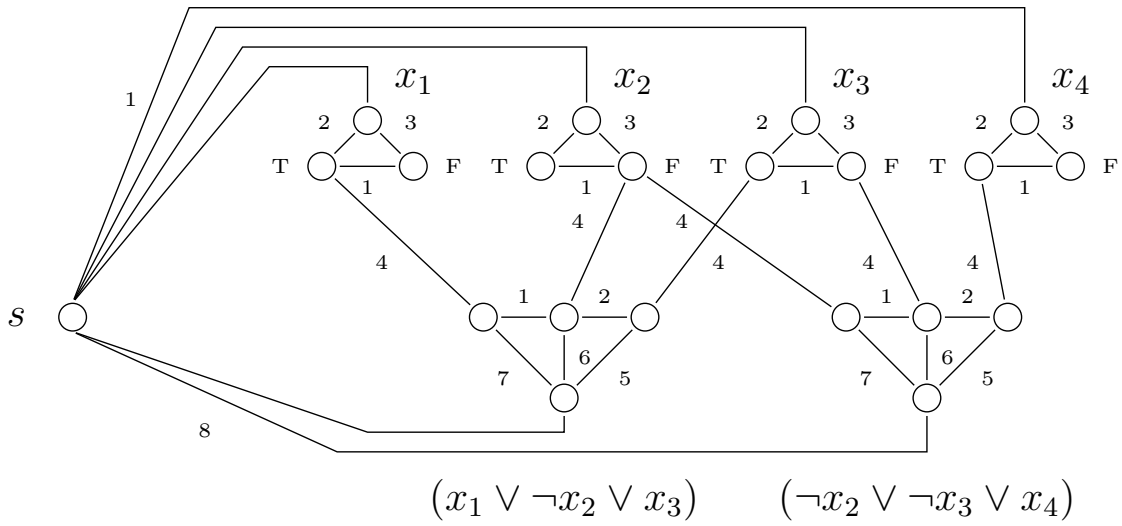

Figure 4.2: Example: Reduction from 3-SAT to STFES/STFCS.

We connect clauses to variables as follows (see Figure 4.2 for an example). Let $c_j = (\ell_j^1 \vee \ell_j^2 \vee \ell_j^3)$ be a clause of Φ. If $\ell_j^1 = x_i$, then we add the edge $(\{w_j^1, v_i^T\}, 4)$ and, if $\ell_j^1 = \neg x_i$, we add $(\{w_j^1, x_i^F\}, 4)$ (edges for ℓ_j^2 and ℓ_j^3 analogously). Further, we connect a new vertex s to all variable gadgets by $(\{s, v_i\}, 1)$ for all $i \in [n]$ and to all clause gadgets by $(\{s, w_j\}, 8)$ for all $j \in [m]$. This creates three additional tours per clause, each starting and ending in s. More precisely, if x_i ($\neg x_i$ is handled analogously) is the z-th literal of clause c_j, then $G(\Phi)$ contains the tour $C_{ij}^z = ((\{s, v_i\}, 1), (\{v_i, v_i^T\}, 2, (\{v_i^T, w_j^z\}, 4), (\{w_j^z, w_j\}, 8-z), (\{w_j, s\}, 8))$.

It is easy to see that $G(\Phi)$ can be computed in polynomial time. The general idea of this reduction is to use the solution size constraint to ensure that exactly one edge from each variable gadget and exactly two edges from each clause gadget are taken. Thus, out of the three tours starting in s and going through the clause gadget of c_j, only two can be disconnected by picking two edges from $\{f_j^1, f_j^2, f_j^3\}$. The remaining tour has to be disconnected inside its variable gadget by picking either e_i^T or e_i^F which "selects" the variable that will satisfy the clause and gives us its truth assignment. Now we show that

$(G(\Phi), k)$ is a yes-instance of STFES if and only if Φ is satisfiable.

($\Rightarrow$) : Let E' be a solution to the constructed STFES instance. Due to the size constraint $k \leq n+2m$ and the tours existing inside the gadgets, E' contains exactly one edge from each variable gadget and none of the edges adjacent to s or connecting variable and clause gadgets. We obtain the solution for the 3-SAT instance by setting x_i to true if $e_i^T \in E'$ and to false if $e_i^F \in E'$ or $e_i^h \in E'$. Assume towards contradiction that there is a clause $c_j = (\ell_j^1 \vee \ell_j^2 \vee \ell_j^3)$ which is not satisfied. Then, in all three variable gadgets connected to w_j, the edge needed to go from s to the corresponding literal vertex of c_j is present in $G(\Phi) \setminus E'$. As E' contains only two of the edges from the clause gadget, the path of one of the three literals can be extended to the vertex w_j and from there back to s, contradicting that $G(\Phi) \setminus E'$ is tour-free.

($\Leftarrow$) : For the other direction, suppose we have a satisfying truth assignment for Φ. We obtain a solution $E' = E_{\text{Var}} \cup E_{\text{Cl}}$ for the STFES instance $(G(\Phi) = (V, E, \tau = 8), k = n + 2m)$ as follows. For the variable gadgets, we use the variable assignment to add the feedback edges

$$E_{\text{Var}} = \{e_i^T \mid i \in [n], x_i = \text{true}\} \cup \{e_i^F \mid i \in [n], x_i = \text{false}\}.$$

For each clause $c_j = (\ell_j^1 \vee \ell_j^2 \vee \ell_j^3)$, let $z_j \in [3]$ be the number of one of the literals satisfying the clause, i.e., $\ell_j^{z_j} = \text{true}$. We add the edges between w_j and the other two literal vertices to the feedback edge set:

$$E_{\text{Cl}} = \{f_j^z \mid j \in [m], z \in [3], z \neq z_j\}.$$

Note that this breaks all tours inside the variable and clause gadgets and that $|E'| = |E_{\text{Var}}| + |E_{\text{Cl}}| = n + 2m$. Tours going through multiple gadgets but not starting in s are not possible as they would use at least two edges with time label 4. It remains to show that $G - E'$ does not contain any tour starting and ending in s. Assume towards contradiction that there is such a tour going through the variable gadget of x_i and the clause gadget of c_j. Further, assume that x_i was set to true (the other case is handled analogously) and that, therefore, $(\{v_i, v_i^F\}, 3) \in G \setminus E'$. Then, the tour begins with $(\{s, v_i\}, 1), (\{v_i, v_i^F\}, 3), (\{v_i^F, w_j^y\}, 4)$ for some $y \in [3]$. Note that the edges e_i^h, f_j^a, and f_j^b cannot be used due to the time labels. By construction of $G(\Phi)$, we know that if $(\{v_i^F, w_j^y\}, 4)$ exists, then ℓ_j^y is one of the literals satisfying the clause if $x_i = \text{false}$. Since we assumed that $x_i = \text{true}$, it holds that $y \neq z_j$ and, thus, $f_j^y \in E_{\text{Cl}}$. It follows that there is no edge which can be appended to the temporal walk and, in particular, no possibility of reaching s, thus contradicting the assumption. □

In the proof for Theorem 4.1, $G(\Phi)$ does not contain any pair of vertices which is connected by more than one time-edge. Hence, each underlying edge corresponds to a single time-edge and the same reduction can be applied in order to obtain the following corollary.

Corollary 4.2. Strict Temporal Feedback Connection Set (STFCS) *is NP-hard even for simple temporal graphs with* $\tau \geq 8$.

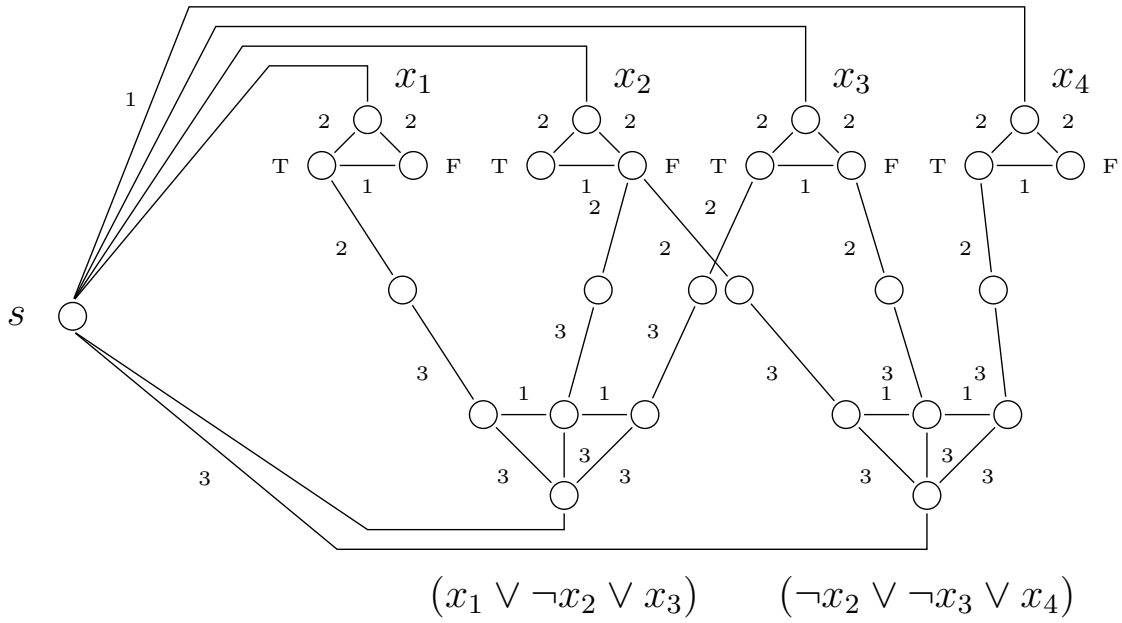

Figure 4.3: Example: Reduction from 3-SAT to TFES/TFCS.

As shown in Chapter 3, STFES and STFCS can be solved in polynomial time for $\tau \leq 2$ which leaves the question whether they admit polynomial-time algorithms for $3 \leq \tau \leq 7$ open.

A very similar reduction can also be used for problem variants using the non-strict temporal walk model. The changes are shown in Figure 4.3. Here, we have to subdivide the edges between variable and clause gadgets in order to avoid tours which go through multiple gadgets but not through s. In turn, only three different time labels are needed to create the required tours.

Corollary 4.3. Temporal Feedback Edge Set (TFES) *and* Temporal Feedback Connection Set (TFCS) *are both NP-hard even for simple temporal graphs with* $\tau \geq 3$.

In terms of parameterized complexity, the results in this section imply that for the **parameter lifetime** τ there can be no XP algorithm (and, thus, no FPT algorithm) for (S)TFES or (S)TFCS, unless P = NP. In order to verify this implication, consider that the running time of an XP algorithm ($f(\tau) \cdot |I|^{g(\tau)}$) is polynomial for any fixed value for τ. Thus, we could use such an algorithm to solve the NP-hard problem STFES in polynomial time. We can use the same argument when considering the parameter *maximum number of time-edges between two vertices* which has the value 1 in both reductions. It also follows that all problem variants remain NP-hard when restricted to simple temporal graphs.

Without formal proof, we observe that the temporal graphs constructed in this section do not contain any temporal tour which is not also a temporal cycle (as no vertex pair is

connected by multiple time-edges). Thus, all results in this section can also be extended to the problem variants based on temporal paths and cycles.

Finally, we show that (S)TFES and (S)TFCS presumably cannot be solved in subexponential time. We derive this result from the well accepted assumption known as the *Exponential Time Hypothesis (ETH)* which implies that 3-SAT cannot be solved in $2^{o(m+n)} \cdot (m+n)^{\mathcal{O}(1)}$ time [IP01; IPZ01].

Proposition 4.4. *Assuming the ETH, (S)TFES and (S)TFCS cannot be solved in* $2^{o(|E|+|V|)\cdot f(\tau)} \cdot (|E|+|V|)^{\mathcal{O}(1)}$ *time for any computable function* $f : \mathbb{N} \mapsto \mathbb{N}$.

Proof. Assume towards contradiction that we have an algorithm A solving (S)TFES or (S)TFCS in $2^{o(|E|+|V|)\cdot f(\tau)} \cdot (|E|+|V|)^{\mathcal{O}(1)}$ time. Given a 3-SAT instance with n variables and m clauses, we use the reductions described above to construct an equivalent instance $(G = (V, E, \tau), k)$ of (S)TFES or (S)TFCS in polynomial time. By construction, we have $|E| \leq 4n + 12m$ and $|V| \leq 3n + 7m + 1$. Additionally, we have $\tau = 8$ (strict) or $\tau = 3$ (non-strict), making $f(\tau)$ a constant, and it follows that $o(|E| + |V|) \cdot f(\tau) = o(m + n)$ and that $\mathcal{O}(|E| + |V|) = \mathcal{O}(m + n)$. Thus, algorithm A can be used to solve the 3-SAT instance in $2^{o(m+n)} \cdot (m+n)^{\mathcal{O}(1)}$ time, contradicting the ETH. □

4.2 Parameterization by Solution Size

In the previous section, we have shown that the problems we are interested in are NP-hard and presumably cannot be solved efficiently. Common next steps include the search for approximation algorithms and a parameterized analysis. We will focus on the latter for the remainder of this work, beginning with the parameter solution size in this section. This parameter is usually among the first to be considered as we are often only interested in small solutions (e.g., when limited by a budget). In this case, an FPT algorithm would allow us to find such small solutions efficiently, if they exist. However, for (S)TFES and (S)TFCS such an algorithm is presumably not possible as we will show by proving the next theorem.

Theorem 4.5. *(S)TFES and (S)TFCS, parameterized by the solution size k, are W[1]-hard.*

We prove W[1]-hardness with a parameterized reduction from Directed Multicut in DAGs parameterized by the solution size.

Directed Multicut in DAGs
Input: A DAG $D = (V, A)$, a set of terminal pairs $\mathcal{T} = \{(s_i, t_i) \mid i \in [r] \text{ and } s_i, t_i \in V\}$, and an integer k.
Question: Is there a set Z of at most k nonterminal vertices of G, such that for all $i \in [r]$ the terminal t_i is not reachable from s_i in $G \setminus Z$?

This problem was shown to be W[1]-hard when parameterized by the size k of the cut set by Kratsch et al. [Kra+15] who also provided the following lemma which will simplify our proof by further restricting the input instance.

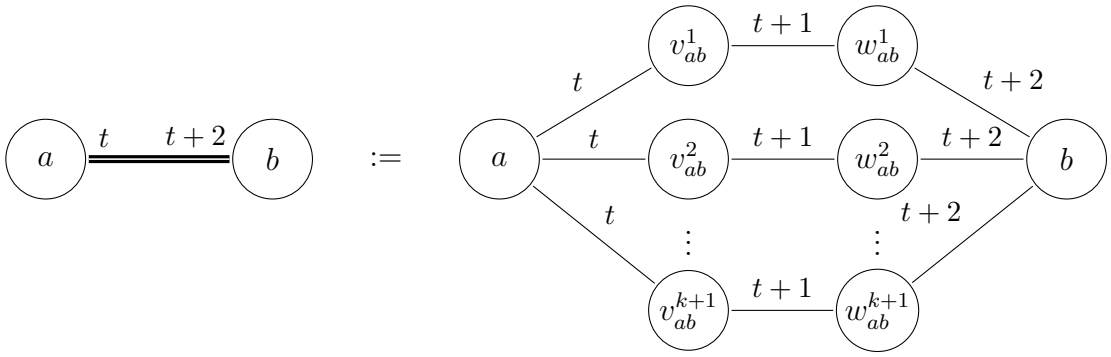

Figure 4.4: Heavy time-edge $\mathrm{h}(a, b, t)$.

Lemma 4.6. [Kra+15]. *There exists a polynomial-time algorithm that, given a* Directed Multicut in DAGs *instance* $(D, \mathcal{T}, k)$ *with* $D = (V, A)$*, computes an equivalent instance* $(D', \mathcal{T}', k')$ *with* $D' = (V', A')$ *such that*

1. $|\mathcal{T}| = |\mathcal{T}'|$ *and* $k = k'$*;*
2. $\mathcal{T}' = \{(s'_i, t'_i) \mid i \in [r]\}$ *and all terminals* s'_i *and* t'_i *are pairwise distinct;*
3. *for each* $v \in V$ *and* $i \in [r]$ *we have* $(v, s'_i) \notin A'$ *and* $(t'_i, v) \notin A'$*.*

We will from now on assume that we have an instance with these properties. The goal of our reduction will be to create one temporal tour for each terminal pair. Since there is a temporal walk from s_i to t_i (the pair can be ignored otherwise), we can create a tour by adding a *back edge* from t_i to s_i. To preserve the direction of the arcs in the (undirected) temporal graph, we will subdivide each arc into small paths with ascending time labels. As Multicut in DAGs asks for a vertex set, we also need to subdivide each nonterminal vertex v into two new vertices v_{in} and v_{out} which are connected by one edge. Then, the vertex v is in the cut set of the original problem if the edge between v_{in} and v_{out} is in the solution edge set of the STFES instance.

Before stating our reduction, we introduce two auxiliary concepts. First, when we want to exclude edges from (S)TFES/(S)TFCS solutions, we will employ a gadget we call *heavy time-edge* which connects two vertices using $k + 1$ parallel paths.

Definition 4.7. (Heavy time-edge). Let $\mathcal{I} = (G = (V, E, \tau), k)$ be an instance of (S)TFES or (S)TFCS. For $a, b \in V$ and $t \leq \tau - 2$, a heavy time-edge of G is a subgraph $\mathrm{h}(a, b, t) := (V_h, E_h, \tau_h = t + 2)$ connecting vertex a to vertex b with

$$V_h = \{a, b\} \cup \{v^i_{ab}, w^i_{ab} \mid i \in [k + 1]\} \quad \text{and}$$

$$E_h = \{(\{a, v^i_{ab}\}, t), (\{v^i_{ab}, w^i_{ab}\}, t + 1), (\{w^i_{ab}, b\}, t + 2) \mid i \in [k + 1]\}.$$

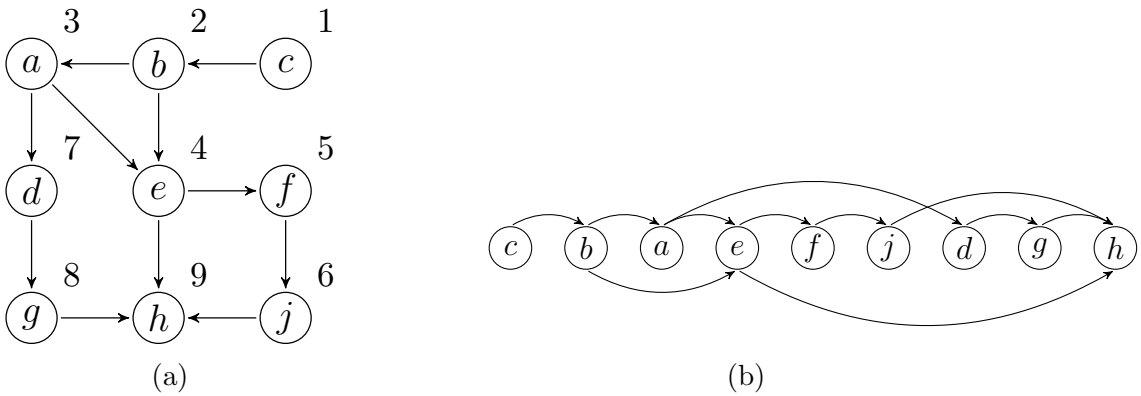

Figure 4.5: A DAG with values for $\pi(v)$ (derived from an acyclic ordering) for each vertex v (a) and with the vertices aligned on a line according to π (b).

The construction is shown in Figure 4.4. Let $e^h := \mathrm{h}(a, b, t)$ be a heavy time-edge. Due to the time labels, the gadget only connects a to b (and not b to a) which we will use to model directed arcs with (undirected) time-edges. For (S)TFES/(S)TFCS solutions, it is easy to see that if there is a temporal path from b to a, then the $k+1$ tours going through e^h cannot be disconnected by removing edges inside the gadget. Thus, we can assume w.l.o.g. that a given solution contains no edges from e^h. We also note that, for each specific temporal walk model (i.e., strict or non-strict), it is possible to design a smaller gadget with identical properties, but we opted to use one which works for both models simultaneously.

Second, in order to assign time labels while preserving all paths of the input graph D, we will use an *acyclic ordering* (also known as topological ordering) of D. For a directed graph $D = (V, A)$, an acyclic ordering $<$ is a linear ordering of the vertices with the property $(v, w) \in A \Rightarrow v < w$. In other words, if we place the vertices on a line in the order given by $<$, then all arcs point in one direction (see Figure 4.5 for an example). If D is a DAG, then such an ordering exists and can be computed in linear time [BJG08, Theorem 4.2.1]. For convenience, we represent this ordering as a function $\pi : V \to \mathbb{N}$ which maps each vertex to its position in the ordering. We now have all the ingredients to prove the theorem.

Proof of Theorem 4.5. Let $\mathcal{I} = (D, \mathcal{T}, k)$ be an instance of Directed Multicut in DAGs with terminal vertices $V_{\mathcal{T}} := \{s_i, t_i \mid (s_i, t_i) \in \mathcal{T}\}$. We construct an instance $\mathcal{I}' = (G, k' = k)$ of (S)TFES as follows.

1. We compute an acyclic ordering of the vertices $V := V(D)$ and store it as a function $\pi : V \to \mathbb{N}$ (see Figure 4.5). We use π to transform D into an equivalent temporal graph $G^1 = (V', E, \tau = 4|V|)$ by replacing each arc $(v, w) \in A$ with the heavy time-edge $e^{vw} := \mathrm{h}(v, w, 4\pi(v)+1)$. It is easy to verify that, for two vertices $s, t \in V$, the graph D contains a path from vertex s to vertex t if and only if G contains a temporal walk from s to t. Note that starting each heavy time-edge

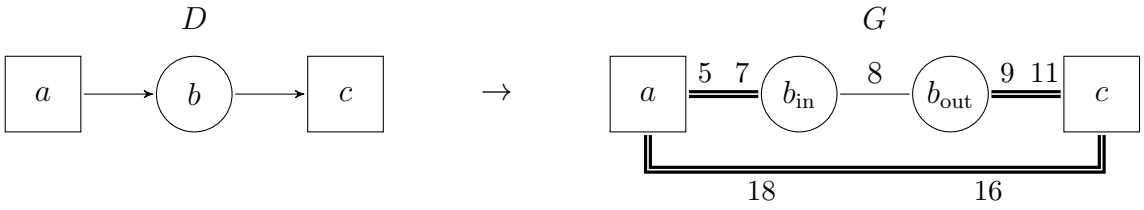

Figure 4.6: Example: Reduction from Directed Multicut in DAGs with input digraph D and one terminal pair (a, c) to TFES. Double lines represent heavy time-edges (Definition 4.7).

with time label $4\pi(v) + 1$ leaves layer $4\pi(v)$ empty which we will use in the next step.

2. In G^1, we replace each nonterminal vertex $v \in V \setminus V_{\mathcal{T}}$ with two new vertices v_{in} and v_{out} connected by time-edge $e^v = (\{v_{\text{in}}, v_{\text{out}}\}, 4\pi(v))$ and update the edges adjacent to v as follows. For each (incoming) edge of the form $e_{uv} := (\{u, v\}, t)$ with $t < 4\pi(v)$, replace e_{uv} with $(\{u, v_{\text{in}}\}, t)$. For each (outgoing) edge of the form $e_{vw} := (\{v, w\}, 4\pi(v) + 1)$, replace e_{vw} with $(\{v_{\text{out}}, w\}, 4\pi(v) + 1)$. Let G^2 denote the resulting graph. Clearly, for two vertices $s, t \in V$, removing v in G^1 disconnects all (s, t)-paths if and only if removing e^v in G^2 disconnects all (s, t)-paths.

3. We obtain $G^3 = G$ by adding a back edge $\mathrm{h}(t_i, s_i, \tau_{\text{be}})$ with $\tau_{\text{be}} = 4|V + 1|$ for each terminal pair (s_i, t_i). Since there is a temporal path from s_i to t_i, this creates at least one tour for each terminal pair.

Figure 4.6 shows a small example. It is easy to see that the construction can be done in polynomial time. Now we show that $\mathcal{I} = (D, \mathcal{T}, k)$ is a yes-instance of Directed Multicut in DAGs if and only if $\mathcal{I}' = (G, k' = k)$ is a yes-instance of (S)TFES.

($\Rightarrow$) : Let Z be a solution of $\mathcal{I}$. We claim that $E' = \{(\{v_{\text{in}}, v_{\text{out}}\}, 4\pi(v)) \mid v \in Z\}$ is a solution of $\mathcal{I}'$. We first show that, for any terminal pair (s_i, t_i), the graph $G - E'$ contains no temporal walk from s_i to t_i. For G^2, this is easily verified as $D - Z$ contains no (s_i, t_i)-path. In $G = G^3$, this claim holds if no temporal walk from s_i to t_i contains a back edge $e^{t_j s_j} = \mathrm{h}(t_j, s_j, \tau_{\text{be}})$ added in step 3. Due to the starting time label of the back edges, no walk can contain more than one back edge and, if it does, this back edge must be at its end. Clearly, the walk cannot end at both t_i and s_j, unless $t_i = s_j$ which we excluded by applying Lemma 4.6. Now, assume towards contradiction that $G - E'$ contains a tour C. Since G^2 was tour-free, C must use some back edge $e^{t_i s_i}$ introduced in step 3 and, as reasoned above, this back edge must be the last edge of C. However, there is no temporal walk from s_i to t_i in $G - E'$ and, thus, C cannot be a temporal tour.

($\Leftarrow$) : Let E' be a solution of $\mathcal{I}'$, i.e., $G - E'$ contains no tours. Recall that $V_{\mathcal{T}} := \{s_i, t_i \mid (s_i, t_i) \in \mathcal{T}\}$ is the set of terminal vertices of $\mathcal{I}$. As observed above, E' does not contain any edges from heavy time-edges, thus we have $E' \subseteq \{e^v \mid v \in V \setminus V_{\mathcal{T}}\}$ and define the solution for $\mathcal{I}$ as $Z = \{v \mid e^v \in E'\}$. Assume towards contradiction that $D - Z$

contains a (s_i, t_i)-path for some terminal pair (s_i, t_i). This path induces a temporal walk from s_i to t_i in $G - E'$ which we can extend back to s_i by appending the back edge $e^{t_i s_i}$ to obtain a tour in $G - E'$ and, thus, a contradiction.

For both directions, we have $|Z| = |E'| \leq k = k'$ meeting the requirements for the solution size.

As the constructed temporal graph G contains no pair of vertices connected by more than one time-edge, we can easily transform a minimal feedback edge set of G into a minimal feedback connection set. Thus, the arguments presented in this proof also hold for (S)TFCS. □

Finally, we remark that Theorem 4.5 also extends to the problem variants based on temporal cycles since, in the proof of the theorem, we used a graph in which every temporal tour is a temporal cycle.

5 Algorithmic Results

After showing computational hardness for the parameters solution size k and lifetime τ in Chapter 4, we now consider larger/combined parameters and present two FPT algorithms for (S)TFES and (S)TFCS.

5.1 Parameterization by Solution Size and Parameters Related to Tour Length

In this section, we show fixed-parameter tractability for multiple parameter pairs, each consisting of the solution size k and another parameter related to the tour length, using the following proposition. Recall that a minimal temporal tour does not contain any (shorter) subtours (Definition 2.3).

Proposition 5.1. *Let $G = (V, E, \tau)$ be a temporal graph and $L \in \mathbb{N}$ the length of the longest minimal temporal tour in G. Then, (S)TFES and (S)TFCS can be solved in $\mathcal{O}(L^k \cdot |V| \cdot |E|^2)$ time.*

Proof. We can construct a simple search tree based on the fact that at least one edge from each tour has to be in the solution. According to Lemma 3.2, we can confirm that G is tour-free or find some shortest tour C in $\mathcal{O}(|E|^2 \cdot |V|)$ time. If we find a tour C, then we branch over all of its $|C| \leq L$ time-edges and recursively solve the instance $\mathcal{I}'$ remaining after removing this time-edge (underlying edge for (S)TFCS) and lowering k by one. Clearly, removing any time-edge cannot create a new temporal tour and, thus, L is also an upper-bound for the length of a minimal temporal tour in $\mathcal{I}'$. The size of the resulting search tree is bounded by L^k. □

Due to Observation 3.3, we know that the length of a minimal tour cannot exceed the number of vertices, i.e., $L \leq |V|$. Thus, we immediately obtain the following corollary.

Corollary 5.2. *(S)TFES and (S)TFCS can be solved in $\mathcal{O}(|V|^{k+1} \cdot |E|^2)$ time.*

For the strict temporal walk model, it is easy to see that the length of any walk or tour is limited by the life time τ of the temporal graph. We can use this to derive another corollary showing that the strict problem variants are fixed-parameter tractable with respect to the combined parameter $\tau + k$.

Corollary 5.3. *STFES and STFCS can be solved in $\mathcal{O}(\tau^k \cdot |V| \cdot |E|^2)$ time.*

Next, we consider two structural parameters of the underlying graph as potential upper bounds for the length L of the longest minimal temporal tour. Given a static graph $G = (V, E)$ and an integer k, the Vertex Cover problem asks whether there is a vertex set $V' \subseteq V$ of size at most k such that $G - V'$ does not contain any edges.

Corollary 5.4. *Let G be a temporal graph with underlying graph $G_\downarrow$. Further, let X be a minimum cardinality vertex cover of $G_\downarrow$. Then, (S)TFES and (S)TFCS can be solved in $\mathcal{O}(|X|^k \cdot 2^k \cdot |V| \cdot |E|^2)$ time.*

Proof. By definition of the Vertex Cover problem, the vertices in $I = V \setminus X$ do not share an edge in $G_\downarrow$ and, therefore, also do not share a time-edge in G. Thus, no temporal walk can consecutively visit two vertices in I. This limits the length L of the longest minimal temporal tour to $L \leq 2|X|$. Combined with Proposition 5.1, we obtain the claimed running time. □

As a second example for parameters of the underlying graph, consider the *feedback vertex number* of $G_\downarrow$ (i.e., the size of a minimum cardinality feedback vertex set of $G_\downarrow$). For this parameter, we cannot obtain an FPT result using Proposition 5.1 due to the following counter example. In a simple temporal graph that consist of a single temporal cycle, the feedback vertex number of the underlying graph is one, but the length of the temporal cycle can be arbitrarily large. Note however that this does not prove parameterized hardness for the feedback vertex number of the underlying graph.

5.2 Parameterization by Number of Vertices

As we have shown in Observation 3.6, (S)TFCS is trivially fixed-parameter tractable with respect to the number of vertices $|V|$. For (S)TFES however, the same result is more difficult to show as the size of the search space is upper-bounded by $2^{\tau(|V|^2-|V|)}$. Here, the dependence on τ prevents us from using the (brute force) approach that worked for (S)TFCS. In this section, we confirm that (S)TFES is fixed-parameter tractable with respect to the number of vertices $|V|$ by proving the following theorem.

Theorem 5.5. *STFES can be solved in $\mathcal{O}(2^{2|V|^2} \cdot |V|^3 \cdot \tau)$ time and TFES can be solved in $\mathcal{O}(2^{3|V|^2} \cdot |V|^2 \cdot \tau)$ time, both requiring $\mathcal{O}(2^{|V|^2})$ space.*

We will prove Theorem 5.5 using a dynamic program which computes the minimum number of time-edges which have to be removed to achieve a specified connectivity at a specified point in time. The key idea is that in order to compute optimal solutions for time t we only need to know which temporal walks were possible at time $t - 1$. In particular, we will only need the start and end points of these temporal walks and, thus, avoid storing the time-edges that were removed before t which would cause problems when aiming at an FPT result for the parameter $|V|$. With the dynamic programming table available, we can solve (S)TFES by looking up the entry at time τ for a connectivity specification that excludes temporal tours.

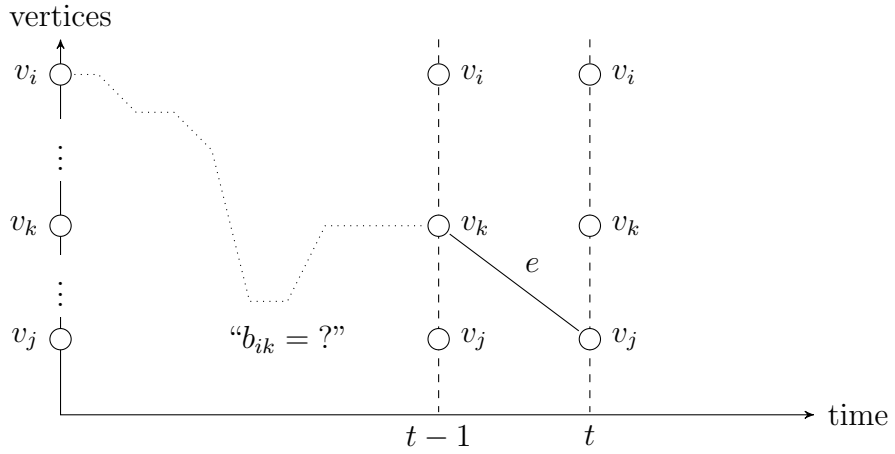

Figure 5.1: Illustration of the subproblem solved by srd/nrd(G, B, A). If we want to make sure that no temporal walk from v_i to v_j exists at time t (that is, we have $a_{ij} = 0$), then the time-edge e has to be removed from the graph because there might be a temporal walk from v_i to v_k ($b_{ik} = ?$, dotted line).

Before formally defining the dynamic program, we need to introduce some notations and intermediate results. Let $G = (V, E, \tau)$ be a temporal graph with $V = \{v_1, v_2, \ldots, v_n\}$. The first dimension of the dynamic programming table will be the connectivity between the vertices of G. We will store this as a *connectivity matrix* $A \in \{0, ?\}^{n \times n}$ encoding the connectivity relationships

$$\begin{aligned} a_{ij} = 0 \quad &\Rightarrow \quad \text{there is no non-trivial temporal walk from vertex } v_i \text{ to } v_j \text{ and} \\ a_{ij} = ? \quad &\Rightarrow \quad \text{there might be a temporal walk from } v_i \text{ to } v_j. \end{aligned}$$

We will use $a_{ii} = 0$ to specify that there must not be a temporal walk starting and ending at vertex v_i. Excluding trivial temporal walks, i.e., walks of length 0, these are exactly all possible temporal tours.

Next, we define two functions, srd(G, B, A) (**s**trict **r**equired **d**eletions) and nrd(G, B, A) (**n**on-strict **r**equired **d**eletions), which return the solution to the following subproblem. Given connectivity B (**b**efore) at time $t-1$, what is the minimum number of edge deletions required in G_t to ensure connectivity A (**a**fter) at time t? Figure 5.1 illustrates this problem for two vertices v_i and v_j. If $a_{ij} = 0$ and there is some vertex v_k which might be reachable from v_i (i.e., $b_{ik} = ?$ represented by the dotted path), then we must remove the edge between v_k and v_j. In order to guarantee correctness, we have to assume that every "?" in B represents an existing path. Additionally, if A and B encode incompatible connectivity, then the function value is defined as ∞. For strict temporal walks, we formalize this as follows.

Definition 5.6. Let $G = (V, E)$ be a static graph with $|V| = n$ and let $A, B \in \{0, ?\}^{n \times n}$ be two connectivity matrices. The function srd(G, B, A) is defined as follows.

If $\exists i, j \in [n] : b_{ij} = ? \wedge a_{ij} = 0$, then srd$(G, B, A) = \infty$.

Otherwise, srd$(G, B, A) = |\{\{v_k, v_j\} \in E \mid \exists i \in [n] : a_{ij} = 0 \wedge (b_{ik} = ? \vee i = k)\}|$.

Algorithm 1 Algorithm computing $\mathrm{srd}(G, B, A)$ (for strict temporal walks)

Parameters
G: a static graph
A, B: connectivity matrices
Output
$\mathrm{srd}(G, B, A)$

```
1:  function StrictRequiredDeletions(G, B, A)
2:      E' ← {}
3:      for i, j ∈ [n] do
4:          if b_ij =? ∧ a_ij = 0 then
5:              return ∞
6:          else if b_ij = 0 ∧ a_ij = 0 then
7:              for all e := {v_k, v_j} ∈ E(G) with b_ik =? do
8:                  E' ← E' ∪ {e}
9:              end for
10:         end if
11:     end for
12:     return |E'|
13: end function
```

There is one special case considered in Definition 5.6. Note that $b_{ik} =?$ is used to determine if v_k was reachable from v_i in the past. However, for $i = k$, the vertex $v_k = v_i$ is always reachable from v_i (even if $b_{ik} = 0$) by a trivial temporal walk. Next, we show that $\mathrm{srd}(G, B, A)$ can be computed in polynomial time.

Lemma 5.7. *Algorithm 1 computes the function* $\mathrm{srd}(G, B, A)$ *in* $\mathcal{O}(|V|^3)$ *time.*

Proof. In order to show correctness, we first point out that lines 4 and 5 check for the case where $\mathrm{srd}(G, B, A) = \infty$ in a straightforward manner. For the other case of Definition 5.6, it can be easily verified that the loops and conditions in lines 3 and 6 to 9 will count an edge $\{v_k, v_j\} \in E$ if $\exists i, j, k \in [n] : b_{ij} = 0 \wedge a_{ij} = 0 \wedge b_{ik} = ?$. The running time is determined by the loops in line 3 ($\mathcal{O}(|V|^2)$) and line 7 ($\mathcal{O}(|V|)$ as we have $|\{\{v_k, v_j\} \in E(G_t)\}| \leq |V|$ for any fixed vertex v_j). Using a hashset as data structure for E', we can add edges in constant time for an overall running time of $\mathcal{O}(|V|^3)$. □

Since, in the non-strict case, a temporal walk can successively use multiple edges from G_t, it is not possible to consider each entry $a_{ij} = 0$ separately (a single edge might be part of multiple unwanted temporal walks). Instead, we have to find an optimal edge cut disconnecting all "problematic" pairs (v_k, v_j) in G_t where $\exists i \in [n] : a_{ij} = 0 \wedge (b_{ik} = ? \vee i = k)\}$. This problem is known as the Multicut problem.

Multicut (Optimization variant)

Input: An undirected, static graph $G = (V, E)$ and a set of r terminal pairs $\mathcal{T} = \{(s_i, t_i) \mid i \in [r] \text{ and } s_i, t_i \in V\}$.

Output: A minimum cardinality edge set $E' \subseteq E$ whose removal disconnects all terminal pairs in $\mathcal{T}$.

We use this problem to define the second function, $\text{nrd}(G, B, A)$.

Definition 5.8. Let $G = (V, E)$ be a static graph with $|V| = n$ and let $A, B \in \{0, ?\}^{n \times n}$ be two connectivity matrices. The function $\text{nrd}(G, B, A)$ is defined as follows.

If $\exists i, j \in [n] : b_{ij} =? \wedge a_{ij} = 0$, then $\text{nrd}(G, B, A) = \infty$.

Otherwise, let E' be a solution to Multicut $(G, \mathcal{T})$ with $\mathcal{T} = \{(v_k, v_j) \mid \exists i \in [n] : a_{ij} = 0 \wedge (b_{ik} = ? \vee i = k)\}$. Then, $\text{nrd}(G, B, A) = |E'|$.

In order to compute $\text{nrd}(G, B, A)$, we have to solve Multicut which was shown to be APX-hard [Dah+94]. While there exist FPT algorithms [Guo+08; MR14] for the parameter solution size, our best upper bound for the solution size is $|V|^2 - |V|$ which results in a running time worse than the brute force approach we will use to prove the next lemma.

Lemma 5.9. *The function* $\text{nrd}(G, B, A)$ *can be computed in* $\mathcal{O}(2^{|V|^2} \cdot |V|^2)$ *time.*

Proof. The number of subsets of E is at most $2^{|V|^2 - |V|}$ and we can verify if a subset is a Multicut solution by performing a breadth-first search in $\mathcal{O}(|V| + |E|) = \mathcal{O}(|V|^2)$ time. Thus, $\text{nrd}(G, B, A)$ can be computed in $\mathcal{O}(2^{|V|^2} \cdot |V|^2)$ time. □

We can now define the dynamic program which we will use to prove Theorem 5.5. Let $A \in \{0, ?\}^{n \times n}$ be a connectivity matrix. The table entry $T(A, t) \in \mathbb{N}$ contains the minimum number of time-edges which have to be removed from $G_{[t]}$ in order to achieve the connectivity specified by A. We define T as follows.

$$T(A, 0) = 0 \quad \forall A \in \{0, ?\}^{n \times n} \tag{5.1}$$

$$\text{strict walks: } T(A, t) = \min_{B \in \{0,?\}^{n \times n}} T(B, t-1) + \text{srd}(G_t, B, A) \tag{5.2a}$$

$$\text{non-strict walks: } T(A, t) = \min_{B \in \{0,?\}^{n \times n}} T(B, t-1) + \text{nrd}(G_t, B, A) \tag{5.2b}$$

Lemma 5.10. *Let* $G = (V, E, \tau)$ *be a temporal graph with* $|V| = n$ *and let* $A \in \{0, ?\}^{n \times n}$ *be a connectivity matrix. Then, there exists no* $E' \subseteq E$ *with* $|E'| < T(A, t)$ *for which* $(G - E')_{[t]}$ *possesses the connectivity specified by* A*.*

Proof. We prove the lemma via induction over t. Recall that the connectivity matrix A can only encode that certain temporal walks must not exist. Thus, the correctness of the initialization $T(A, 0) = 0$ is easy to see since no temporal walks exist at time $t = 0$. For the correctness of the update step (Equations (5.2a) and (5.2b)), we note that, by minimizing over all possible $B \in \{0, ?\}^{n \times n}$, we always find the optimal state B for the

time $t-1$. With some B fixed, it remains to show that $T(B, t-1) + \mathrm{srd/nrd}(G_t, B, A)$ is minimal for achieving both connectivity B at time $t-1$ and connectivity A at time t. By induction hypothesis, we know that $T(B, t-1)$ is minimal. To show correctness and minimality of $\mathrm{srd}(G_t, B, A)$ and $\mathrm{nrd}(G_t, B, A)$, we analyze how the time-edges of layer G_t influence the possible temporal walks up to time t and which changes (i.e., time-edge deletions) are required to achieve connectivity A. For any two vertices $v_i, v_j \in V$, we compare the connectivity for time $t-1$ given by b_{ij} to the target connectivity given by a_{ij} and identify the following four cases.

(Case 1) $b_{ij} = ? \wedge a_{ij} = ?$: Here, we do not care if v_j is reachable from v_i and, thus, we do not need to remove any edges.

(Case 2) $b_{ij} = ? \wedge a_{ij} = 0$: We cannot disconnect a temporal walk that already exists at time $t-1$ by removing edges in layer t and, therefore, cannot guarantee $a_{ij} = 0$. In this case, there is no solution for the input parameters. Both functions are defined (see Definitions 5.6 and 5.8) to return ∞ in this case.

(Case 3) $b_{ij} = 0 \wedge a_{ij} = ?$: Identical to Case 1.

(Case 4) $b_{ij} = 0 \wedge a_{ij} = 0$: We have to ensure that v_j is not reachable from v_i in $G_{[t]}$. Both $\mathrm{srd}(G_t, B, A)$ and $\mathrm{nrd}(G_t, B, A)$ are defined based on the following concept. If there is a vertex v_k which was reachable from v_i in the past, then we cannot keep an edge (strict case) or any path (non-strict case) connecting v_k and v_j in layer t. Assuming that Case 2 is already excluded for all vertex pairs, it can be easily verified that Definition 5.6 uses exactly the set of such edges $\{v_k, v_j\}$. For the non-strict case, the set $\mathcal{T}$ of terminal pairs in Definition 5.8 is defined to exactly contain all such pairs (v_k, v_j). The minimality and correctness of $\mathrm{nrd}(G_t, B, A)$ follows from the definition of Multicut.

Finally, we must show that assuming "$b_{ij} = ? \Rightarrow$ there is a temporal walk from v_i to v_j" at time $t-1$ for the functions $\mathrm{srd/nrd}(G_t, B, A)$ did not result in unnecessary time-edge deletions. To this end, assume $b_{ij} = ?$ and that there is no temporal walk from v_i to v_j at time $t-1$. Let B' be a connectivity matrix identical to B except for $b'_{ij} = 0$. As the walk from v_i to v_j does not exist, we have $T(B, t-1) = T(B', t-1)$. If the entry $b_{ij} = ?$ resulted in an unnecessary time-edge deletion, i.e., $\mathrm{srd/nrd}(G_t, B, A) > \mathrm{srd/nrd}(G_t, B', A)$, then the minimum function in Equations (5.2a) and (5.2b) will not chose the value computed using B. □

Since we are only interested in specifying that certain walks (tours) must not exist, we chose to use "?"-entries in the connectivity matrices to represent entries we do not care about. The advantage is evident in Cases 1 and 3 of the previous proof. We now have all required ingredients to prove the main theorem.

Proof of Theorem 5.5. Let (G, k) be an instance of (S)TFES. Further, let A^* be an $n \times n$ connectivity matrix with $a^*_{ij} = 0$ if $i = j$ and $a^*_{ij} = ?$ otherwise. As "?"-entries cannot require more time-edge deletions than "0"-entries, A^* is the cheapest connectivity specification that does not allow any temporal tour to exist. Thus, it follows from Lemma 5.10 that (G, k) is a yes-instance if $T(A^*, \tau) \leq k$, and a no-instance otherwise.

For the running time, we first note that a connectivity matrix has size $|V|^2$ with two possible choices for each entry resulting in $2^{|V|^2}$ possible connectivity matrices. Thus,

the table size of the dynamic program is $2^{|V|^2} \cdot \tau$. To compute each table entry, we have to compute srd/nrd(G_t, B, A) for each of the $2^{|V|^2}$ possible choices for B. Together with Lemmas 5.7 and 5.9, we obtain the running times stated in the theorem. The computation requires $\mathcal{O}(2^{|V|^2})$ space as we only need the table entries for time $t-1$ in order to compute the entries for time t. Thus, it is not necessary to store more than two columns of the table, each of size $2^{|V|^2}$. □

We note that our dynamic program solves the optimization variant of (S)TFES. That is, given a temporal graph G, it finds the smallest k for which (G, k) is a yes-instance of the decision variant stated defined in Chapter 1. As shown in the previous proof, we can easily use the result to solve any instance (G, k') of the decision variant by comparing k' to k.

Aiming for a theoretical result, we did not store the actual solution, that is, the feedback edge set of size $T(A, t)$. However, the functions srd(G_t, B, A) and nrd(G_t, B, A) can easily be changed to return the solution edge sets for each layer t. Using linked lists, which can be concatenated in constant time, it is possible to include the solutions sets in the dynamic programming table without (asymptotically) changing the running time.

6 Conclusion

This work provides a first study on the (parameterized) complexity of feedback set problems in temporal graphs. We focused on the two edge-deletion variants (S)TFES and (S)TFCS while leaving the analysis of the vertex-deletion variant for future research (see below). We showed that all considered problem variants are NP-hard and, presumably, not fixed-parameter tractable when parameterized by either the solution size k or the lifetime of the temporal graph τ. As the Feedback Edge Set problem in static graphs can be solved in polynomial time, this supports the intuitive assumption that temporal problems are generally harder to solve than their static counterparts.

On the positive side, we described an algorithm solving (S)TFES and (S)TFCS in $\mathcal{O}(L^k \cdot |V| \cdot |E|^2)$ time where L is the length of the longest minimal temporal tour. We obtained additional FPT results, by showing that $L \leq |V|$, $L \leq \tau$ (strict case only), and $L \leq |X|$ where X is a minimum cardinality vertex cover. While (S)TFCS is trivially fixed-parameter tractable for the parameter $|V|$, showing the same result for (S)TFES was surprisingly difficult. The developed dynamic program takes $\mathcal{O}^*(2^{2|V|^2})$ time[1] (strict) or $\mathcal{O}^*(2^{3|V|^2})$ time (non-strict) to compute which is likely not feasible for practical applications. If the vertex set is small, then the tour length based algorithm with running time $\mathcal{O}(|V|^{k+1} \cdot |E|^2)$ is presumably a better candidate for implementation.

Future work. We conclude this work by describing future research opportunities and open questions.

Approximation algorithms. In the introduction, we suggested that minimum solution sizes for the different feedback set problems are interesting structural parameters of temporal graphs. However, these parameters are only useful if they can be computed reasonably fast. In static graphs, for instance, the Feedback Vertex Set Problem is both fixed-parameter tractable (with respect to the solution size) [Che+08] and can be approximated up to a constant factor of 2 [BBF99; BG96]. As we have shown W[1]-hardness for (S)TFES and (S)TFCS, finding an answer to the question whether they admit a constant-factor approximation is crucial to determine their usefulness both as a structural parameter and in practical applications.

Unsolved parameterizations. One promising parameter that was not covered in this work is the size of the *temporal core* [Zsc+18] which consists of all vertices that are incident to edges which are not permanently present. Both hardness proofs in Chapter 4 use temporal graphs in which the temporal core contains all vertices of the graph which leads us to believe that this parameter is generally large in hard instances, and thus likely to yield an FPT result.

[1] The $\mathcal{O}^*$-notation suppresses polynomial time factors.

For the parameter lifetime τ, it remains open whether there exists a polynomial-time algorithm for instances with $3 \leq \tau \leq 7$ in the strict case and $\tau = 2$ in the non-strict case. We believe that, for the strict case, the 3-SAT reduction shown in Section 4.1 can be changed to use only seven time-labels by removing the vertex s and connecting the clause gadgets directly to the variable gadgets. Similar to the results obtained for Temporal (s, z)-Separation by Zschoche et al. [Zsc+18], we expect that there is some $3 \leq \tau_p \leq 6$ for which STFES and STFCS are polynomial-time solvable for $\tau < \tau_p$ and NP-hard for $\tau \geq \tau_p$.

Another similarity to the work of Zschoche et al. [Zsc+18], is that we could not resolve the question whether the non-strict variants are fixed-parameter tractable for the combined parameter $\tau + k$. Proving parameterized hardness for this parameter would confirm our assumption that the non-strict variants are harder to solve than the strict ones.

In Section 5.1, we introduced two parameters based on the structure of the underlying graph. This approach offers a large number of additional, potentially interesting parameters. An overview can be found online [Rid] and in an unpublished work by Sorge and Weller [SW]. A related research opportunity is the study of (S)TFES and (S)TFCS versions that are restricted to specific temporal graph classes (see the work by Fluschnik et al. [Flu+19] for an overview).

Temporal feedback vertex set. While we did not include the vertex-deletion variant of in the main part of this work, we want give a short outlook on the problem.

(Strict) Temporal Feedback Vertex Set - (S)TFVS
Input: A temporal graph $G = (V, E, \tau)$ and $k \in \mathbb{N}$.
Question: Is there a vertex set $V' \subseteq V$ with $|V'| \leq k$ such that $G[V \setminus V']$ does not contain a (strict) temporal tour?

The non-strict variant TFVS is clearly NP-hard as it includes the NP-hard Feedback Vertex Set Problem in static graphs as special case for $\tau = 1$. Based on our proofs in Chapter 4, we conjecture that the strict variants also (presumably) not admit an XP algorithm for the parameter τ and that all variants are W[1]-hard when parameterized by the solution size k. The 3-SAT reduction in Section 4.1 is probably adaptable to vertex-deletion by removing the vertex s and tweaking the variable and clause gadgets. For W[1]-hardness, recall that we reduced from Directed Multicut in DAGs in Section 4.2, a problem which ask for cut set consisting of vertices. In the reduction for the edge-deletion variants, this required an additional step where we subdivided the non-terminal vertices into two vertices. For (S)TFVS, this step is not necessary; instead, we need to find a gadget which allows us to distinguish between terminal and non-terminal vertices.

Literature

[AF16] K. Axiotis and D. Fotakis. "On the Size and the Approximability of Minimum Temporally Connected Subgraphs". In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 2016, 149:1–149:14 (cit. on pp. 11, 13, 15).

[Agr+18] A. Agrawal, D. Lokshtanov, A. E. Mouawad, and S. Saurabh. "Simultaneous feedback vertex set: A parameterized perspective". In: *ACM Transactions on Computation Theory* 10.4 (2018), p. 18 (cit. on p. 14).

[Akr+15] E. C. Akrida, L. Gasieniec, G. B. Mertzios, and P. G. Spirakis. "On Temporally Connected Graphs of Small Cost". In: *Approximation and Online Algorithms - 13th International Workshop, WAOA 2015, Patras, Greece, September 17-18, 2015. Revised Selected Papers*. 2015, pp. 84–96 (cit. on p. 15).

[Akr+18] E. C. Akrida, G. B. Mertzios, P. G. Spirakis, and V. Zamaraev. "Temporal Vertex Cover with a Sliding Time Window". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. 2018, 148:1–148:14 (cit. on p. 15).

[BBF99] V. Bafna, P. Berman, and T. Fujito. "A 2-Approximation Algorithm for the Undirected Feedback Vertex Set Problem". In: *SIAM Journal on Discrete Mathematics* 12.3 (1999), pp. 289–297 (cit. on p. 43).

[Ber96] K. A. Berman. "Vulnerability of scheduled networks and a generalization of Menger's Theorem". In: *Networks* 28.3 (1996), pp. 125–134 (cit. on pp. 14, 19).

[BG96] A. Becker and D. Geiger. "Optimization of Pearl's Method of Conditioning and Greedy-Like Approximation Algorithms for the Vertex Feedback Set Problem". In: *Artificial Intelligence* 83.1 (1996), pp. 167–188 (cit. on p. 43).

[BJG08] J. Bang-Jensen and G. Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Science & Business Media, 2008 (cit. on pp. 12, 31).

[Che+08] J. Chen, Y. Liu, S. Lu, B. O'sullivan, and I. Razgon. "A fixed-parameter algorithm for the directed feedback vertex set problem". In: *Journal of the ACM* 55.5 (2008), p. 21 (cit. on p. 43).

[Cyg+15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015 (cit. on p. 21).

[Dah+94] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. "The Complexity of Multiterminal Cuts". In: *SIAM Journal on Computing* 23.4 (1994), pp. 864–894 (cit. on p. 39).

[DF13] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013 (cit. on p. 21).

[Die12] R. Diestel. *Graph Theory, 4th Edition*. Vol. 173. Graduate Texts in Mathematics. Springer, 2012 (cit. on p. 19).

[Fer04] A. Ferreira. "Building a reference combinatorial model for MANETs". In: *IEEE Network* 18.5 (2004), pp. 24–29 (cit. on p. 19).

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006 (cit. on p. 21).

[Flu+19] T. Fluschnik, H. Molter, R. Niedermeier, M. Renken, and P. Zschoche. "Temporal graph classes: A view through temporal separators". In: *Theoretical Computer Science* (2019) (cit. on p. 44).

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979 (cit. on p. 25).

[Guo+08] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier, and J. Uhlmann. "Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs". In: *European Journal of Operational Research* 186.2 (2008), pp. 542–553 (cit. on p. 39).

[Him+17] A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge. "Adapting the Bron–Kerbosch algorithm for enumerating maximal cliques in temporal graphs". In: *Social Network Analysis and Mining* 7.1 (2017), p. 35 (cit. on p. 15).

[Him18] A.-S. Himmel. "Algorithmic Investigations into Temporal Paths". MasterThesis. TU Berlin, 2018 (cit. on p. 13).

[Hol18] P. Holme. "Temporal Networks". In: *Encyclopedia of Social Network Analysis and Mining, 2nd Edition*. 2018 (cit. on p. 19).

[IP01] R. Impagliazzo and R. Paturi. "On the complexity of k-SAT". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 367–375 (cit. on p. 29).

[IPZ01] R. Impagliazzo, R. Paturi, and F. Zane. "Which problems have strongly exponential complexity?" In: *Journal of Computer and System Sciences* 63.4 (2001), pp. 512–530 (cit. on p. 29).

[Joh75] D. B. Johnson. "Finding all the elementary circuits of a directed graph". In: *SIAM Journal on Computing* 4.1 (1975), pp. 77–84 (cit. on p. 12).

[Kar72] R. M. Karp. "Reducibility Among Combinatorial Problems". In: *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. 1972, pp. 85–103 (cit. on pp. 13, 14, 25).

[KKK02] D. Kempe, J. Kleinberg, and A. Kumar. "Connectivity and inference problems for temporal networks". In: *Journal of Computer and System Sciences* 64.4 (2002), pp. 820–842 (cit. on pp. 11, 13–15, 19).

[Kra+15] S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström. "Fixed-parameter Tractability of Multicut in Directed Acyclic Graphs". In: *SIAM Journal on Discrete Mathematics* 29.1 (2015), pp. 122–144 (cit. on pp. 29, 30).

[KRP19] S Khodayifar, M. Raayatpanah, and P. Pardalos. "A polynomial time algorithm for the minimum flow problem in time-varying networks". In: *Annals of Operations Research* 272.1-2 (2019), pp. 29–39 (cit. on p. 19).

[LM17] Q. Liang and E. Modiano. "Survivability in Time-Varying Networks". In: *IEEE Transactions on Mobile Computing* 16.9 (2017), pp. 2668–2681 (cit. on p. 19).

[LVM18] M. Latapy, T. Viard, and C. Magnien. "Stream graphs and link streams for the modeling of interactions over time". In: *Social Network Analysis and Mining* 8.1 (2018), p. 61 (cit. on p. 19).

[MMS19] G. B. Mertzios, O. Michail, and P. G. Spirakis. "Temporal Network Optimization Subject to Connectivity Constraints". In: *Algorithmica* 81.4 (2019), pp. 1416–1449 (cit. on pp. 11, 19).

[MR14] D. Marx and I. Razgon. "Fixed-parameter tractability of multicut parameterized by the size of the cutset". In: *SIAM Journal on Computing* 43.2 (2014), pp. 355–388 (cit. on p. 39).

[MS16] O. Michail and P. G. Spirakis. "Traveling salesman problems in temporal graphs". In: *Theoretical Computer Science* 634 (2016), pp. 1–23 (cit. on pp. 11, 15).

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006 (cit. on p. 21).

[Rid] H. N. de Ridder. *Information System on Graph Classes and their Inclusions (ISGCI)*. URL: http://www.graphclasses.org (visited on 05/15/2019) (cit. on p. 44).

[Sha79] A. Shamir. "A linear time algorithm for finding minimum cutsets in reducible graphs". In: *SIAM Journal on Computing* 8.4 (1979), pp. 645–655 (cit. on p. 12).

[SW] M. Sorge and M. Weller. "The graph parameter hierarchy". Unpublished Manuscript (cit. on p. 44).

[VLM16] T. Viard, M. Latapy, and C. Magnien. "Computing maximal cliques in link streams". In: *Theoretical Computer Science* 609 (2016), pp. 245–252 (cit. on pp. 15, 19).

[WLS85] C.-C. Wang, E. L. Lloyd, and M. L. Soffa. "Feedback Vertex Sets and Cyclically Reducible Graphs". In: *Journal of the ACM* 32.2 (1985), pp. 296–313 (cit. on p. 12).

[XFJ03] B. B. Xuan, A. Ferreira, and A. Jarry. "Computing shortest, fastest, and foremost journeys in dynamic networks". In: *International Journal of Foundations of Computer Science* 14.02 (2003), pp. 267–285 (cit. on p. 23).

[Zsc+18] P. Zschoche, T. Fluschnik, H. Molter, and R. Niedermeier. "The Complexity of Finding Small Separators in Temporal Graphs". In: *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*. 2018, 45:1–45:17 (cit. on pp. 11, 13, 14, 43, 44).