Parameterised Complexity of Spreading Problems on Temporal Graphs

Samuel Hand

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow

December 2024

Abstract

Spreading problems are a class of decision problems on graphs that can be used to model the behaviour of a spread of a contagion over a network. Real world networks are often dynamic in nature, with the connections between the members of the network changing over time. Take for example the contact network of a population of people: individuals may come into contact for a period of time, and then move apart and contact other individuals. Graphs do not model this dynamic behaviour, and as such in this thesis we consider spreading problems on temporal graphs, which overcome this shortfall by augmenting a graph with temporal information. In particular we consider the problems of Temporal Firefighter and Temporal Graph Burning, extensions of the Firefighter and Graph Burning problems to temporal graphs. Temporal Firefighter asks how to best prevent the vertices of a temporal graph from "burning" when a fire is spreading over the graph, and Temporal Graph Burning asks how best to burn the vertices of a temporal graph to spread a fire as fast as possible. Both of these problems are **NP**-complete, and unlikely to yield efficient algorithms in general. Parameterised complexity theory provides tools for obtaining efficient algorithms for **NP**-complete problems, and in this thesis we consider various parameters for temporal graphs. We find that both Temporal Firefighter and Temporal Graph Burning are in **FPT** when parameterised by vertex-interval-membership-width and by temporal neighbourhood diversity. Furthermore, we prove a meta-theorem for showing that a temporal graph problem is in **FPT** when parameterised by vertex-interval-membership-width. Overall we demonstrate the usefulness of temporal graph parameters, suggesting future work in applying these to more problems on temporal graphs.

Contents

List of Figures

List of Algorithms

Acknowledgements

Many thanks to my supervisors Jessica Enright and Kitty Meeks, for their continued advice and support throughout my PhD. Thank you to my parents, Russell and Rachael Hand for their love and encouragement. Thank you to both Duncan and Lynn, and Mel and Chris, for their repeated hospitality, and to my friends for providing (not too much) fun and distraction.

Finally thank you to Izzi. I would not be here without you.

Declaration

With the exception of Chapter 1 which contains introductory material, all work in this thesis was carried out by the author unless otherwise explicitly stated.

Chapter 2 and Chapter 3 contains work that appears in [44, 43]. Chapter 4 contains work that appears in [29].

To Izzi.

Chapter 1

Introduction

Spreading occurs in a variety of real world scenarios, including the spread of epidemics [55], the spread of information and rumours over a social network [68], and the spread of political idealogies and opinions [6]. We may desire to compute details about how a given process spreads over such networks. For example, if we knew how an epidemic would spread, we could use this information to decide who best to target with a vaccine to limit the spread. We might alternatively be interested in promoting spreading over a network: if we wanted to inform a population about a certain topic and knew how information spreads on the social network of the population, we could use this to inform decisions about where best to target our campaign. We refer to a computational problem as a spreading problem if it asks how a process propagates over a given network.

In this thesis we consider algorithms for solving spreading problems. Unfortunately, most of these problems are hard to solve, in that as the size of the input network increases, the time taken by an algorithm to compute an answer likely becomes prohibitively large. We describe how the time taken by an algorithm to solve a problem varies with respect to the size of the network using the language of complexity theory. We then look for ways to achieve efficient runtimes for our algorithms as the network size increases, by exploiting structural features in the network. We search for parameterised algorithms, which can solve problems efficiently provided some measurement of the network, called a parameter, remains small.

Real world networks are dynamic. In population networks, the contacts between people change, as the people travel around. However, many existing spreading problems take place on mathematical objects known as graphs. These model networks, but only model the existence, or lack thereof, of a connection, not the times at which this connection exists. Temporal graphs overcome this limitation, by augmenting graphs with temporal information, specifying not just whether a connection exists, but when it exists. In this thesis we define two spreading problems on temporal graphs: Temporal Firefighter, and Temporal Graph Burning. These are both extensions to temporal graphs of

existing graph spreading problems, Firefighter and Graph Burning respectively. We explore the complexity of these new problems, and in doing so develop algorithmic techniques which we believe will have wide applicability to other problems on temporal graphs.

We define Temporal Firefighter and Temporal Graph Burning in Chapter 2, and determine their difficulty using the language of complexity theory. We then give further complexity results for Temporal Firefighter, inspired by those that already exist for Firefighter.

In Chapter 3 we consider the temporal graph parameters vertex-interval-membership-width and edge-interval-membership-width. Both of these parameters are small only when the temporal network has few connections on each timestep. The former measures, for any timestep, the maximum number of members of the network that are in contact or have already been in contact with another member, and are in contact or will later be in contact with another member. The latter is defined in a similar but measures numbers of connections instead of network members. We find using these parameters to be a good avenue for approaching these problems, and give an algorithm for Temporal Firefighter that achieves a fast runtime when one of these parameters known as the vertex-interval-membership-width is small. It is possible that the edge-interval-membership-with is small even when the vertex-interval-membership-width is large, and we show that with a small edge-interval-membership-width Temporal Firefighter remains hard.

In Chapter 4 we explore new temporal parameters that are small even when the temporal network has many connections on each timestep. These parameters are defined in joint work with Enright et al. [29]. We consider two parameters, temporal neighbourhood diversity and temporal modular width. The former of these measures the number of groups of members of the network where members in the same group cannot be distinguished by their connections. We show that Temporal Graph Burning and Temporal Firefighter can be solved quickly when the temporal neighbourhood diversity is small. The temporal modular width generalises temporal neighbourhood diversity and may be small even when when the temporal neighbourhood diversity is large, and we show that even with a small temporal modular width both Temporal Graph Burning and Temporal Firefighter remain hard.

Finally, in Chapter 5, we again turn our attention to the parameter of vertex-interval-membership-width, and consider how to more generally apply this parameter to more problems. We define a framework for expressing problems, and refer to problems expressible in this framework as locally temporally uniform. We then prove a meta-theorem showing that any locally temporally uniform problem is solvable efficiently when the vertex-interval-membership-width is small. We give examples of locally temporally uniform problems, including Temporal Firefighter, thus recreating the result from Chapter 3, as well

as the problems of Temporal Graph Burning, Temporal Dominating Set, and Temporal Hamiltonian Path, thus determining that all of these problems can be solved efficiently when the vertex-interval-membership-width is small.

1.1 Background and Preliminaries

We now give an introduction to, and review literature in, the topics of graph theory, temporal graphs, complexity theory and parameterised complexity. We also give some prerequisite definitions, including those of the existing problems of Firefighter and Graph Burning.

1.1.1 Graphs

A graph models the connections between a set of items, referred to as vertices, such as people, computers, or physical places. Connections between vertices are referred to as edges. Examples include: physical contacts or relationships between people, physical connections between computers, and transport links between places.

Definition 1 (Graph)**.** *A graph G is a pair (V, E), where V is a set of vertices, and E is a set of edges: unordered pairs of vertices.*

Given a graph $G = (V, E)$ we say that two vertices $v, u \in V$ are adjacent if $\{v, u\} \in E$, and that an edge $e \in E$ is incident at a vertex $v \in V$ if $v \in e$.

When considering spreading over the vertices of a graph, the concept of reachability is often relevant. We say that one vertex is reachable from another if there exists a *path* between the two vertices. If one vertex is reachable from another via the edges of the graph then a contagion may spread between these vertices.

Definition 2 (Path)**.** *A path on a graph $G = (V, E)$ is a sequence of edges $P = e_1, ...e_\ell$ such that there exists a sequence of vertices $P_V = v_1, ..., v_{\ell+1}$, where every vertex appears in P_V at most once, and $e_i = \{v_i, v_{i+1}\}$. The path P is said to be between the vertices v_1 and $v_{\ell+1}$. A vertex is said to be reachable from another if there exists a path between the two.*

Other structures present in graphs are also relevant to spreading, such as groups of vertices where every vertex is connected to every other, known as cliques. If a contagion spreads to just one vertex of a clique then it will be able to spread from this vertex to all other vertices in the clique, as they are all connected.

Definition 3 (Clique)**.** *A clique on a graph $G = (V, E)$ is a subset of the vertices $K \subseteq V$ such that for any pair of vertices u and v in K, there exists an edge between u and v: $\{u, v\} \in E$.*

Many other structural properties of graphs are defined in Bondy and Murty's textbook [17], to which we refer the reader for a detailed introduction to the field of graph theory.

Spreading on graphs is often studied from the perspective of a game, in which the players seek to influence a spreading process. Graph burning and firefighter are both examples of one player games where the player aims to encourage or restrict the spread of a fire over the vertices of a graph [13, 46]. Graph flooding games are an extension to arbitrary graphs of a common puzzle game in which a grid is to be coloured with a single colour by repeatedly flooding, or filling areas of the grid [4]. Closely related to spreading games are the two player pursuit evasion games, in which both players move agents over the vertices of a graph. One player takes the role of the pursuer, and attempts to move their agents to the same vertices as that of the other player, who aims to escape. Cops-and-robbers [70] is a typical example of this class of game.

In this thesis we focus specifically on algorithms that operate on graphs, and analyse these algorithms using the language of complexity theory, which we now discuss.

1.1.2 Complexity Theory

Complexity theory is the study of how hard a given problem is to solve algorithmically. We classify problems into complexity classes according to their difficulty. Two of the most widely studied complexity classes are **P** and **NP**. Intuitively, a problem in **P** is "easy" or (relatively) fast to solve, because an algorithm for the problem will terminate in time asymptotically bounded by a polynomial of the size of the input. Note this polynomial may be superlinear, so the runtime of the algorithm can still grow faster than the input size, but exponential growth, for example, is not possible. Problems in **NP** have no such guarantees on the runtimes of the algorithms that solve them, but it is in the same sense "easy" to verify solutions to problems in **NP**. In a landmark paper published in 1972, Karp showed 21 problems across combinatorics to be **NP**-complete: at least as hard as any other problems in **NP** [52]. Several of these now archetypal **NP**-Complete problems take place on graphs, including Vertex Cover, Max Cut, Graph Colouring, and Hamiltonian Cycle. For definitions of these problems and other **NP**-Hard problems on graphs, see Chapter 16 of the textbook by Skiena [71]. This story of hardness has continued until the present day, with many non-trivial graph problems being **NP**-Complete, including Cops and Robbers [37], Flood-It [4], and the two problems that we consider in this thesis: Firefighter [63] and Graph Burning [10].

Formally, we phrase problems as decision problems, in which we ask whether a given input belongs to a language (a set of sequences of symbols drawn from an alphabet, known a strings).

Definition 4 (Decision Problem)**.** *A decision problem is a language L over a finite alphabet Σ.*

An algorithm is a rigorous set of instructions, and is said to solve a decision problem if it can answer whether a given input word $w \in \Sigma^*$ is in L. The time complexity of an algorithm tells us asymptotically how many steps are taken by the algorithm before it terminates. This is given as a function of the size of the input. Formally, a problem is in **P** if the number of steps taken by an algorithm for solving the problem is asymptotic to a polynomial function of the size of the input. Similarly, a problem is in **NP** if the number of steps taken by an algorithm for verifying a solution to the problem is asymptotic to a polynomial function of the size of the input.

Every problem in **P** is also in **NP**, but it is widely conjectured that the converse is not true, that is **P** $\neq$ **NP**, and there exist problems where it is easy to check the validity of solutions, but it is not easy to find a solution in the first place. If we wish to show that a problem is unlikely to be in **P**, we usually do so by proving that it is **NP**-hard, meaning that it is at least as hard as any problem in **NP**. We can prove that a problem is **NP**-hard by a reduction from some known **NP**-hard problem, where we show that any algorithm for solving our new problem would also be capable of solving the known **NP**-hard problem. As a result of the previously mentioned conjecture that **P** $\neq$**NP**, any **NP**-hard problem is unlikely to be in **P**. The hardest problems in **NP**, that is problems in **NP** that are also **NP**-hard, are said to be **NP**-complete. For a complete treatment of complexity theory, including further complexity classes and the concept of space complexity, see the textbook by Arora and Barak [3].

Under the assumption that **P** $\neq$ **NP**, the best possible runtime of an algorithm for a **NP**-complete problem is superpolynomial, and the required time to execute the algorithm grows prohibitively large as the input size grows. Various techniques exist for solving **NP**-complete problems that cope with this intractability. In this thesis we primarily use techniques from the field of parameterised complexity, which we discuss in Section 1.1.6.

Another approach is that of approximation algorithms, which can be applied to optimisation problems. These are problems where instead of answering yes or no, an algorithm for the problem should produce a solution that either minimises or maximises some objective. Rather than finding an optimal solution, an approximation algorithm finds a solution within a constant factor of the true maximum or minimum. By not requiring that an algorithm finds a truly optimal solution, feasible runtimes can be achieved, even for **NP**-complete problems. For a detailed treatment of approximation algorithms see the book by Williamson and Shmoys [72], from which we take the following definition.

Definition 5 (α-approximation algorithm)**.** *An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution.*

The constant α is referred to as the approximation ratio, and it is convention that $\alpha < 1$ for maximisation problems, and $\alpha > 1$ for minimisation problems.

Approximation algorithms can be generalised by considering algorithms that do not have a fixed approximation ratio, but instead take the approximation ratio as part of the input, and run in time polynomial in the size of the input, for any fixed approximation ratio. Such an algorithm is called a polynomial time approximation scheme, or PTAS. The definitions below are adapted from work by Ausiello et al. [5].

Definition 6 (Approximation scheme)**.** *An approximation scheme for an optimisation problem is an algorithm that for any instance of the problem, and* α *such that* $0 < \alpha < 1$ *for a maximisation problem or* $0 < \frac{1}{\alpha} < 1$ *for a minimisation problem, returns a solution within a factor of* α *of the optimal solution.*

Definition 7 (PTAS)**.** *A polynomial time approximation scheme is an approximation scheme with time complexity* $O(n^{f(\alpha)}$*, where* f *is any computable function.*

Optimisation problems can be expressed in terms of linear programs, a system of variables and linear inequalities between the variables, and an objective function to maximise or minimise. When the variables can take any real value, it is possible to solve a linear program in polynomial time. When phrased as a linear program, many discrete problems restrict the variables to only take integer values, producing what is known as an integer linear program, or ILP. Solving integer linear programs is **NP**-Complete [52], however it is sometimes possible to obtain an approximate solution by solving the linear program that results from removing the constraint that the variables be integers, known as the LP relaxation. For a detailed treatment of linear programming and integer linear programming, including algorithms for solving linear programs, and discussion about LP relaxations, see the book by Wolsey and Nemhauser [73].

The majority of this thesis is concerned with variants of two **NP**-complete problems, Firefighter, and Graph Burning, which we now define.

1.1.3 Firefighter

Firefighter is a one player game first presented by Bert Hartnell in 1995 [46]. At the start of the game a specified set of vertices are burning. On each turn the player deploys a certain number of firefighters to defend chosen vertices, before the fire spreads, and all unburning and undefended vertices adjacent to the fire begin burning. Once a vertex is either burning or defended it remains so for the rest of the game.

On finite graphs, the primary algorithmic question is the Firefighter problem, which asks whether it is possible to save a certain number of vertices from the fire in the following process, where the player may defend one vertex a turn, given a graph with a single burning vertex, known as the root:

Definition 8 (Firefighter Process)**.**

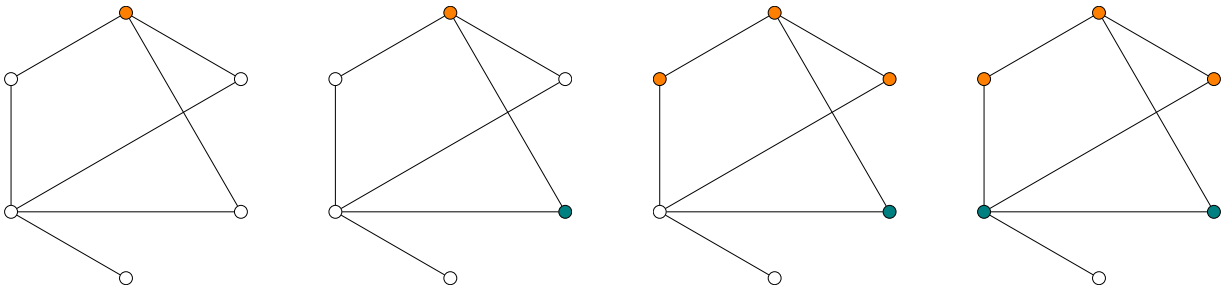

Figure 1.1: An example game of FIREFIGHTER, the orange vertices are burning, and the green are defended.

1. *At time $t = 0$, the root r is labeled as burning.*

2. *At all times $t \geq 1$, a chosen vertex is labeled as defended, and the fire then spreads to all undefended vertices adjacent to the fire.*

3. *This process ends once the fire can no longer spread.*

For an example of the process described in Definition 8, see Figure 1.2. A vertex v is valid to defend on timestep i if and only if v is not burning or already defended on timestep i. We refer to a sequence of such valid defences for a given instance of FIREFIGHTER as a strategy.

Definition 9 (FIREFIGHTER Strategy)**.** *Given a rooted graph (G, r), a strategy $S = v_1, v_2, ..., v_\ell$ is a sequence of distinct vertices from $V(G)$, such that if the fire begins at r on timestep 0, spreads according to Definition 8, and each vertex is defended in turn, each v_i is unburning and undefended on timestep i, and the fire stops spreading on or before timestep ℓ.*

We say a vertex is saved if it is not burning once the process ends. The decision problem then asks if a certain number of vertices can be saved on a given graph:

FIREFIGHTER
Input: A rooted graph (G, r) and an integer k.
Output: Does there exist a strategy that saves at least k vertices on G when the fire starts at vertex r?

The complexity of FIREFIGHTER has been well studied, and it was first shown to be **NP**-complete in 2003 by MacGillivray and Wang [63].

Theorem 1 (MacGillivray and Wang [63])**.** *FIREFIGHTER is **NP**-Complete, even on bipartite graphs.*

Finbow et al. [34] showed that FIREFIGHTER remains **NP**-complete even when restricted to trees of maximum degree three, by reduction from a restricted variant of 3-SAT:

the problem of determining if a boolean formula with 3 literals per clause is satisfiable. King and MacGillvray [54] used a similar reduction to show that the problem is also **NP**-complete on cubic graphs: graphs where every vertex has degree exactly 3. Despite the existence of hardness results for FIREFIGHTER even on very restrictive graph classes, polynomial-time algorithms have been identified for a few classes of graph. Finbow et al. [34] showed that FIREFIGHTER can be solved in polynomial-time on graphs of maximum degree 3, provided the root has degree 2. On such a graph, the maximum number of vertices are saved when the fire is restricted to spreading along a single path, as on each timestep the one vertex adjacent to the fire but not on the path can be defended. The algorithm then identifies the shortest path from the root, at the end of which the fire can be contained. The vertices that burn will be all the vertices on this path, and all other vertices will be saved. More recently, Fomin et al. [38] showed that FIREFIGHTER could be solved in polynomial-time for several graph classes: interval graphs, permutation graphs, P_k-free graphs for $k > 5$, split graphs, and cographs.

FIREFIGHTER has received significant attention when restricted to trees, due to its difficulty in this case. MacGillivray and Wang showed that it is always optimal to defend vertices adjacent to the fire on a tree [63]. If any vertex non-adjacent to the fire is defended, defending the ancestor of this vertex that is adjacent to the fire will save at least as many vertices. This observation suggests a greedy strategy for FIREFIGHTER on trees; at each timestep defend the vertex adjacent to the fire with the greatest number of descendants. Hartnell and Li [47] showed that the greedy strategy saves at least half of the number vertices that can be saved by an optimal strategy. This greedy strategy provides a $\frac{1}{2}$-approximation for the optimisation variant of FIREFIGHTER, where the goal is to maximise the number of vertices saved.

In their 2003 paper, MacGillvray and Wang [63] also presented a formulation of the optimisation variant of FIREFIGHTER as an integer linear program. Cai et al. [21] considered the LP relaxation of this program, and used it to produce a $(1 - \frac{1}{e})$-approximation for FIREFIGHTER on trees, improving on the greedy approximation. In 2006, Hartke [45] modified MacGillvray and Wang's integer linear program to reduce the difference between the solution to the LP relaxation and ILP solution - known as the integrality gap. In 2016, Adjiashvili et al. [1] utilised these existing LPs to construct a PTAS for FIREFIGHTER.

The game has been studied on infinite graphs, where the problem shifts to determining if and how the fire can be contained such that it is unable to spread further in a finite number of timesteps. Wang and Moeller showed that at least 8 timesteps are required when using 2 defences per turn to contain a fire that starts burning at a single vertex in a 2 dimensional square grid [67]. In the strategy presented by Wang and Moeller, 18 vertices burn, and Develin and Hartke used integer programming to prove that this is a minimum [28]. In the same paper, Develin and Hartke also considered the problem of containing

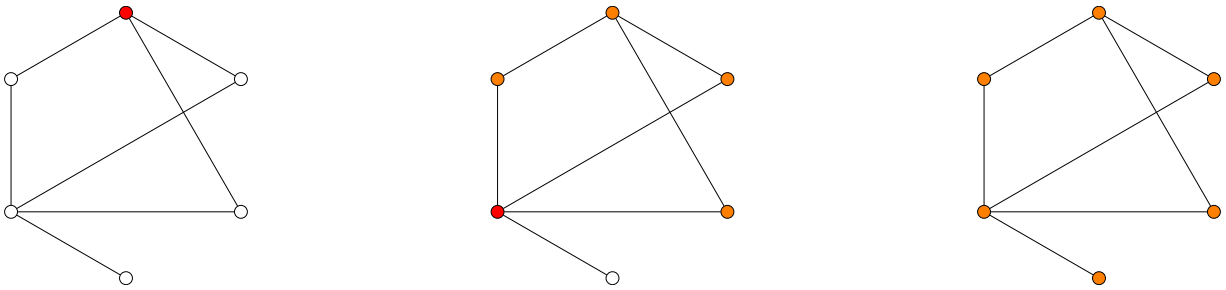

Figure 1.2: An example game of Graph Burning, the red vertices are newly placed fires, and all other burning vertices are in orange.

the fire on multi-dimensional grids, and showed that at least $2d-1$ defences per turn are required to contain a fire that starts burning at a single vertex in a square lattice of d dimensions [28].

Other decision problems related to Firefighter have also received attention in the literature. King and MacGillivray showed that the S-Fire problem is **NP**-complete, S-Fire asks whether it is possible to save a set S of vertices from the fire, rather than a given number of vertices. The number of defenses made per turn can also be altered, Bazgan et al. studied the (S,b)-Fire problem, which asks if it is possible to save the set S when b defences are made per turn. They showed that (S,b)-Fire is **NP**-complete, even on trees of maximum degree $b+2$.

1.1.4 Graph Burning

Similar to Firefighter, graph burning is also a one player game, but now the player burns the vertices of a graph to encourage the spread of a fire. The game was introduced by Bonato et al. in 2014 [13], and subsequently has received significant attention in the literature. On each turn in the graph burning game, the fire spreads to all vertices that it is adjacent to, and then the player chooses an additional vertex to burn:

1. At time $t=0$, all vertices are unburnt.
2. At all times $t \geq 1$, the fire spreads, burning all vertices adjacent to an already burning vertex. Then, a fire is placed at a chosen unburnt vertex.
3. This process ends once all vertices are burning.

An example graph burning game can be seen in Section 1.1.4.

A strategy for the graph burning game is then a sequence of distinct vertices at which fires are placed:

Definition 10 (Graph Burning Strategy)**.** *A strategy for a graph G is a sequence of vertices $S = v_1, v_2, ..., v_\ell$ where if fires are placed at each vertex in turn, vertex v_i is unburnt on timestep i, and every vertex in the graph is burnt on or before timestep $\ell+1$.*

The burning number of a graph refers to the length of the shortest strategy that burns every vertex in the graph. The algorithmic problem of determining if a graph has a given burning number is known as the Graph Burning problem.

Graph Burning
Input: A graph G and an integer ℓ.
Output: Does there exist a burning strategy for G of length less than or equal to ℓ?

Bessy et al. showed that Graph Burning is **NP**-Complete, even when on acyclic graphs of maximum degree 3 [10]. This was shown by reduction from Distinct-3-Partition, a problem that asks for a set X of $3n$ integers to be partitioned into subsets of size 3 such that the elements in each subset sum to the same target number B. They also gave a separate reduction from Distinct-3-Partition, showing that Graph Burning is **NP**-complete when restricted to spider graphs and path forests. This reduction operates by constructing a path forest of $3n$ paths all of length $2B-3$, along with paths Q_i of length $2i-1$ for every $i < \max(X)$ and $i \notin X$. ℓ is then set equal to $\max(X)+1$. If X is a yes-instance of Distinct-3-Partition then each of the paths of length $2B-3$ can be partitioned into 3 sub-paths with radii corresponding to 3 integers from a set in the partition of X. Then, a burning strategy that places fires at the center of each of these sub-paths and fires at the center of each path Q_i in a particular order will burn the graph and be of length less than ℓ. Conversely, if the constructed path forest is a yes-instance of Graph Burning the vertices of each path of length $2B-3$ at which a burning sequence of length less than ℓ places a fire can be used to determine a partition for X, implying that X is a yes-instance.

Theorem 2 (Bessy et al. [10])**.** *Graph Burning is **NP**-Complete, even when restricted to acyclic graphs with maximum degree three, spider graphs, and path forests.*

There are some cases where tractable algorithms for Graph Burning exist. Bessy et al. presented polynomial-time algorithms for Graph Burning on spider graphs and path forests where the number of arms and components is fixed [10]. Kare and Reddy found polynomial-time algorithms for cographs and split graphs [51], in fact, both of these algorithms are linear in the number of vertices and edges in the graph.

As with Firefighter, the approximability of finding an optimal strategy for the graph burning game has been considered. Bessy et al. provide a polynomial-time approximation for general graphs that produces a strategy at most 3 times longer than the optimum [10]. Bonato and Lidbetter considered the approximability for path forests, and provide a $\frac{3}{2}$-approximation algorithm [16]. Bonato and Kamali present two algorithms: a 2-approximation for trees, and a PTAS for path forests [15].

Graph burning is also often studied from a combinatorial perspective, where general bounds are sought on the burning number. It can be seen that the burning number is

$\lceil \sqrt{n} \rceil$ on any n vertex path. An optimal strategy for burning a path P places fires such that on each timestep t a fire is placed at the center of a ball with radius bg(P)-t+1, where bg(P) is the burning number for the path, such that any overlap between any two balls is minimised. Then $\Sigma_{i=1}^{\text{bg}(P)}(2i-1) \geq n$, and therefore $\text{bg}(P)^2 \geq n$, and $\text{bg}(P) \geq \lceil \sqrt{n} \rceil$. In their original paper, Bonato et al. conjectured that $\lceil \sqrt{n} \rceil$ is the maximum burning number on any connected graph [13]. This conjecture is referred to as the burning number conjecture.

Some progress has been made on proving the burning number conjecture on specific graph classes. Bonato and Lidbetter showed that the conjecture holds for spider graphs: trees with exactly one vertex of degree at least 3 [16]. The conjecture was proven for caterpillars, that is trees where every vertex either lies on a central path, known as the stalk, or is adjacent to a vertex on the stalk [62]. Hiller et al. also proved the conjecture for p-caterpillars with a sufficient number of leaves [48]. A p-caterpillar is a generalisation of a caterpillar where every vertex is at most distance p from the stalk.

Recent work has also found increasingly improving bounds on the burning number in general. Bonato et al. showed that the burning number was at most $2\lceil \sqrt{n} \rceil - 1$ on any connected graph of n vertices [13], and this has subsequently been repeatedly improved [9, 60, 14]. Most recently, Norin and Turcotte showed that the burning number holds asymptotically, with the burning number being at most $(1 + o(1))\sqrt{n}$ on any connected graph of n vertices [69].

1.1.5 Temporal Graphs

Real world networks are often dynamic in nature. Take for example the contact network for a population of people: a graph is only capable of modelling whether two individuals are ever in contact or not, and includes no information as to when or how long a contact occurs for. A pair of individuals may come into contact through sharing an office, mode of public transport, or by meeting each other socially. All of these connections occur at a certain time, and last for a certain length of time before the contact between them ends again. As further examples, transport networks, social networks, and various types of computer network can also exhibit dynamic behaviour [50]. Dynamic or time varying networks have been studied in a wide range of recent work, across a diverse range of fields, and under several different names including dynamic networks [22], graphs over time [61], and evolving graphs [18].

Using static graphs to model the connections between individuals in a population or network fails to capture any dynamic behaviour. In this thesis, we study temporal graphs, which remedy this limitation by augmenting a graph with temporal information that assigns times to the vertices or edges at which they are said to be active. Most existing algorithmic work uses a definition for temporal graphs similar to that given by Kempe et

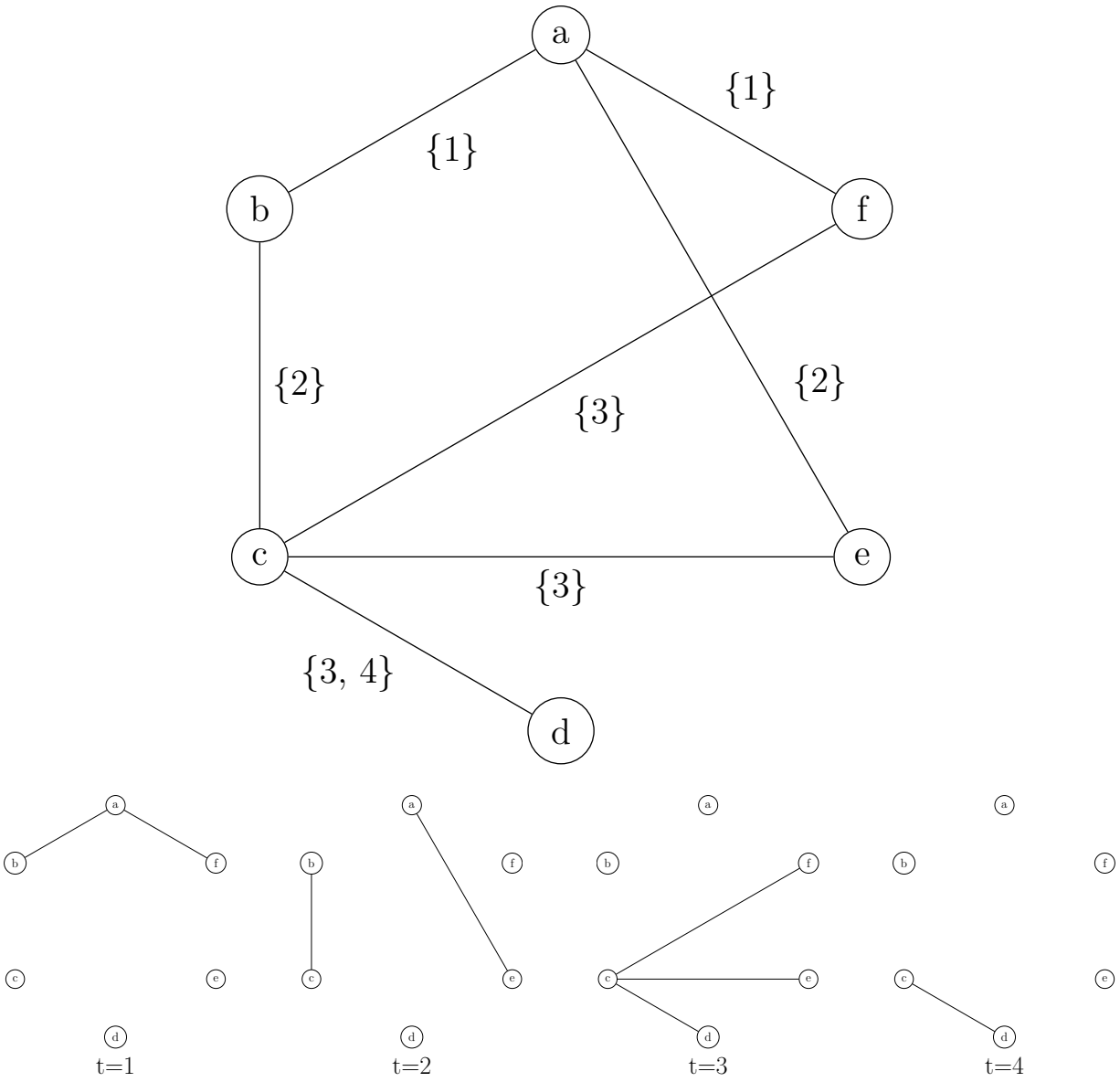

Figure 1.3: An example temporal graph, and the snapshots of the graph on each timestep.

al. [53], in which a graph is augmented with a time-labeling function that assigns to each edge a set of discrete times at which it is active.

Definition 11 (Temporal Graph)**.** *A temporal graph $\mathcal{G}$ is a pair (G, λ) where G is a static graph (V, E), and $\lambda : E \to 2^{\mathbb{N}}$ is a time-labeling function. We say that an edge $e \in E$ is active on timestep t if and only if $t \in \lambda(e)$.*

We refer to the maximum time on which any edge is active as the lifetime $\Lambda = \max\{\max \lambda(e) : e \in E(G)\}$ of the temporal graph. For example, the temporal graph in Figure 1.3 has lifetime 4.

For a temporal graph $\mathcal{G} = (G, \lambda)$ we use $\mathcal{G}\downarrow$ to refer to G, and say that this is the underlying graph of the temporal graph. Additionally, we use $V(\mathcal{G})$ as shorthand for the vertex-set of G, and similarly $E(\mathcal{G})$ for the edge-set. Throughout this thesis we assume

that temporal graphs are represented as a sequence of static graphs, with one graph per timestep containing all the edges active on that timestep. We say that these graphs are the *snapshots* of the temporal graph, and given a temporal graph $\mathcal{G}$ we let $\mathcal{G}_t$ denote the snapshot of $\mathcal{G}$ on timestep t. Figure 1.3 shows an example temporal graph, along with its snapshots. If we represent these graphs as adjacency matrices the size of a temporal graph is $n^2\Lambda$ where n is the number of vertices, and we can check in constant time whether a given pair of vertices are adjacent on a given timestep.

This definition introduces new notions of adjacency and paths, which we define below.

Definition 12 (Temporal Adjacency)**.** *Two vertices v_1 and v_2 in the temporal graph $\mathcal{G} = (G, \lambda)$ are adjacent at time t if they are adjacent in the underlying graph G, and $t \in \lambda(\{v_1, v_2\})$.*

For example, vertices c and d in Figure 1.3 are adjacent in the underlying graph, but only adjacent at timesteps 3 and 4 in the temporal graph.

As the edges of a temporal graph are only active at certain times, new notions of paths are introduced. Two vertices may have a sequence of edges connecting them, but the existence of such a path no longer ensures that either vertex is reachable from the other. As such, a path on a temporal graph is said to be *temporal* if its edges are active on an increasing sequence of timesteps, so each edge can be traversed one after the other. In this thesis we require that the sequence of timesteps on the edges is strictly increasing, and define a temporal path as a sequence of edge appearances, but note there is not one settled on definition for paths on temporal graphs in the literature.

Definition 13 (Temporal Path)**.** *A temporal path originating at u and arriving at v on the temporal graph $\mathcal{G} = (G, \lambda)$ is a sequence $P = (e_1, t_1)..., (e_\ell, t_\ell)$ of edge-appearances: pairs of edges and timesteps such that the sequence $e_1, ..., e_\ell$ is a path on G between u and v, and the sequence of times $t_1, ...t_\ell$ is strictly increasing, and $t_i \in \lambda(e_i)$ for every i.*

We say that a vertex v is temporally reachable from u if there is a temporal path originating at u and ending at v. Note that reachability is not symmetric, unlike in static undirected graphs, where vertex v is reachable from u if and only if u is reachable from v. For example, in Figure 1.3 there is a temporal path from vertex a to vertex d, but no temporal path from vertex d to vertex a.

When considering a temporal path, the length of the path, that is the number of edges that it contains, does not necessarily relate to the length of time taken to traverse the path. It is entirely possible that a path contains very few edges, but still takes a long time to traverse, due to the times at which its edges are active. We say that the arrival time of a temporal path is the final timestep on the path. We may then ask for a path from one vertex to another such that the arrival time is minimised, which is referred to as the foremost path, or such that the total time taken to traverse the path is as low as

possible, referred to as a fastest path [18]. Many path-related problems have been explored on temporal graphs by recent work, which we now discuss. Michail and Spirakis defined multiple temporal analogues of the traveling salesman problem, proving inapproximability for the problem of exploring the vertices of a temporal graph as soon as possible, but providing approximation algorithms for the problem of finding a minimum cost traveling salesman tour where every edge is given a weight of 1 or 2. Bumpus and Meeks showed that the problem of finding Eulerian circuits on a temporal graph is **NP**-hard, unlike the problem on static graphs [20]. Mertzios et al. provided polynomial-time algorithms for finding foremost and shortest paths on temporal graphs, and also proved an analogue of Menger's theorem. They also considered two optimisation problems in which times must be assigned to the edges of a graph in a manner minimal in the number of active timesteps per edge, or total number of timesteps used, such that a certain degree of connectivity is maintained [64]. Enright et al. studied the problem of temporally ordering the edges of a graph, such that the reachability is minimised [32], and showed that this problem is **NP**-hard. A related problem of minimising reachability by deleting edges of an existing temporal graph has also been studied, and is also **NP**-hard [31]. Additionally, some non-path related problems have been considered on temporal graphs, including finding temporal cliques [49], matchings [65], colourings [66], and covers [2].

1.1.6 Parameterised Complexity

Perhaps unsurprsingly, problems that are hard on static graphs remain hard when considered on temporal graphs, since a static graph can be seen as a special case of a temporal graph with all edges active on every timestep. Furthermore, many problems get even harder when studied on temporal graphs. Kempe et al. [53] showed that Menger's theorem, an important result for static graphs, does not hold in its original formulation on temporal graphs, and thus several problems concerned with the connectivity that are in **P** for static graphs are **NP**-Complete when considered on temporal graphs.

The field of parameterised complexity theory provides tools for coping with the hardness of problems. It considers not just the size of the input, but also some other measure or parameter. By doing so, algorithms are able to exploit structural properties of the input provided this parameter is small, and runtimes can be obtained that are only polynomial in the size of the input [26]. We use the definitions of parameterised and fixed-parameter tractable problems provided by Flum and Grohe, who give a detailed treatment on the field of parameterised complexity theory [35].

Definition 14 (Parameterised Problem)**.** *A parameterised problem is a pair* (L, κ) *where* L *is a language over an alphabet* Σ *and* $\kappa : \Sigma^* \to \mathbb{N}$ *is a polynomial time computable function, referred to as a parameterisation of* Σ^**.*

Definition 15 (Fixed-parameter tractable)**.** *A parameterised problem (L, κ) is fixed-parameter tractable or in **FPT** if it can be decided if an input x belongs to the language L in time $O(f(\kappa(x))|x|^{O(1)})$ where f is any computable function that depends only on the value of κ.*

Note that whilst the above definition requires that the parameterisation be polynomial time computable, this restriction is often relaxed. For example, the parameter of treewidth, which we discuss below, cannot be computed in polynomial time, but the problem of determining if a graph has treewidth at most k is fixed-parameter tractable with respect to the parameter k.

Treewidth is arguably the typical example of a parameter. Informally, this parameter measures how "tree-like" a graph is. Many problems are in **P** when restricted to trees, but **NP**-complete for general graphs. Such problems often also admit tractable algorithms on sufficiently tree-like graphs, and so are **FPT** with respect to treewidth. For example, vertex cover, a problem that is **NP**-Complete on general graphs but in **P** on trees, is in **FPT** when parameterised by treewidth [12]. A key meta-theorem of Courcelle states that any problem that can be expressed in monadic second order logic (MSO) is in **FPT** with respect to the aforementioned parameter of treewidth [24]. Such meta-theorems have also been found for other pairs of parameters and logics. For instance, it is the case that any problem expressible in first order logic is in **FPT** with respect to neighbourhood diversity, a parameter that measures the number of classes of vertices with identical neighbourhoods [58]. Furthermore there is a corresponding result for cliquewidth, a parameter that generalises both neighbourhood diversity and treewidth, which states that any problem expressible in a particular restriction of MSO is in **FPT** [25].

The parameterised complexity of both the Firefighter and Graph Burning problems has been studied. Cai et al. showed that Firefighter on trees is in **FPT** when parameterised by the number of the vertices to be saved [21]. Bazgan et al. then went on to show that Firefighter is in **FPT** when parameterised by a combination of the number of vertices to be saved and the treewidth, and also showed that it is in **FPT** when parameterised by a combination of the number of vertices to be saved and the vertex cover number [7]. The runtimes in this work were later improved by Cygan et al. [27]. For graph burning, Kare and Reddy showed that the parameters of distance to cluster and neighborhood diversity yield fixed-parameter tractable algorithms [51]. This work was continued by Kobayashi and Otachi, who showed that the problem is in **FPT** when parameterised by cliquewidth and the maximum diameter [56].

Some problems on temporal graphs admit fixed-parameter tractable algorithms when parameterised by classical static graph parameters. Casteigts et al. showed that the problem of finding time-respecting paths with a maximum waiting time is in **FPT** when parameterised by the vertex cover number, treedepth, and feedback edge number of the

underlying static graph [23]. Furthermore, they find that whilst this problem remains hard when parameterised by the static parameter feedback vertex number, it is in **FPT** when parameterised by the parameter timed feedback vertex number: a temporal version of feedback vertex number. Enright et al. find that the problem of deleting edges to minimise the maximum reachability of a temporal graph is in **FPT** when parameterised by a combination of the maximum degree and treewidth of the underlying graph, along with the reachability to be achieved [31]. This result is arrived at by use of Courcelle's meta-theorem: they provide a method for representing a temporal graph by a relational structure of bounded treewidth provided the underlying graph has both bounded treewidth and maximum degree. Thus any temporal graph problem that can be expressed by an MSO formula of bounded length on such a relational structure is in **FPT** when parameterised by the sum of treewidth and maximum degree.

Moreover, the lifetime of a temporal graph (the maximum time label on any edge) often serves as a useful parameter, either by itself or in combination with a classical static graph parameter. Erlebach and Spooner find that the problem of deciding whether there exists a temporal walk that visits every vertex of a graph is in **FPT** when parameterising by lifetime, providing that the sequence of time-edges traversed by the walk is allowed to be non-strictly increasing [33]. The problem of finding separators between vertices in a temporal graph, and the problem of counting temporal paths are both in **FPT** when parameterised by both the lifetime and the treewidth of the underlying graph [36, 30]. Similarly, Kutner and Larios-Jones study problems related to finding temporal reachability dominating sets, that is sets of vertices from which all others in the graph can be reached. By utilitising Courcelle's metatheorems they show the existence of fixed parameter tractable algorithms, again using the parameters of lifetime and treewidth, and also the size of the dominating sets to be achieved [57]. Finally, Bocci et al. study the problem of modifying a temporal graph to producd a disjoint union of temporal cliques, and find it to be in **FPT** with respect to both the lifetime and number of allowed modifications [11].

Recent work has shown it to be fruitful to define new parameters that measure details of the temporal structure of the graph. Zschoche et al. define the temporal graph parameter temporal core, and find that a particular variant of the problem of finding separators between two vertices is in **FPT** [74]. Bumpus and Meeks define the temporal parameters interval membership width and vertex interval membership width, presenting fixed parameter tractable algorithms for the problems of determining if a graph is temporally Eulerian, and finding a set of vertices that if deleted minimise the temporal reachability of the graph respectively [20]. The parameter of vertex interval membership width is also used to produce a fixed parameter tractable algorithm for counting temporal paths in the aforementioned work by Enright et al. [30]. Whilst the aforementioned work has shown the parameter of vertex interval membership width to be useful, some temporal graph prob-

lems require that the input graph be temporally connected, meaning that every vertex can be reached from every other vertex. It is worth noting that in these cases parameterising by the vertex interval membership width is not meaningful, as the parameter becomes equal to the number of vertices in the graph.

1.2 Table of Results

Parameter	Temporal Firefighter	Temporal Graph Burning
Graphs of maximum degree 3 with a root of degree 2	**P** (Theorem 9)	
Cliques	**NP**-Complete (Theorem 6)	
Vertex-interval-membership-width	**FPT** (Theorem 17)	**FPT** (Corollary 7)
Edge-interval-membership-width	**NP**-Complete (Theorem 19)	
Temporal neighbourhood diversity	**FPT** (Theorem 32)	**FPT** (Theorem 39)
Temporal modular width		**NP**-Complete (Theorem 40)

Table 1.1: The complexity classes to which the problems of Temporal Firefighter and Temporal Graph Burning belong when parameterised by the parameters considered in this thesis.

Chapter 2

Temporal Firefighting and Burning Games

In this chapter we define the problems Temporal Firefighter and Temporal Graph Burning. These are both natural extensions of the Firefighter and Graph Burning problems to temporal graphs. We show that Temporal Firefighter and Temporal Graph Burning are **NP**-Complete, by reducing from Firefighter and Graph Burning respectively. Both of these reductions preserve the underlying graph, in fact, in both cases we simulate the static input instance by assigning times to the edges of the input graph, making no other modifications. In this way the static problems can be seen as special cases of our newly defined temporal extensions.

As a consequence of these results, for any class $\mathscr{C}$ of static graphs for which it is known that Firefighter is **NP**-Complete, Temporal Firefighter is **NP**-Complete on the class of temporal graphs with underlying graphs belonging to $\mathscr{C}$, and the same is true of Graph Burning and Temporal Graph Burning. There are several classes of graph for which Firefighter is known to be in **P**, and in the final section of the chapter we explore the complexity of Temporal Firefighter when the underlying graph belongs to these classes. We show that Temporal Firefighter is **NP**-Complete when the underlying graph is a clique, again by reduction from Firefighter. This allows us to show that for all but one of the classes for which Firefighter is known to be in **P**, Temporal Firefighter remains **NP**-Complete, as all of these classes can contain cliques of unbounded size. The sole remaining class is graphs of maximum degree 3 where the fire starts at a vertex of degree 2. In this case we provide an algorithm for Temporal Firefighter with linear runtime, which operates in an analogous way to the known algorithm for static Firefighter.

2.1 Problem Definitions and Hardness

As stated in Sections 1.1.3 and 1.1.4, in both Firefighter and Graph Burning a fire spreads over a graph by spreading to adjacent vertices on each timestep. We now extend both of these problems to temporal graphs, such that the fire spreads to temporally adjacent vertices on each timestep.

2.1.1 Temporal Firefighter

We now define Temporal Firefighter, an extension of Firefighter to temporal graphs. In Temporal Firefighter, just as in Firefighter, the fire begins burning at a root vertex r, and on each timestep a single vertex can be defended before the fire spreads. Unlike Firefighter the fire does not spread to all adjacent vertices in the underlying graph, but only to vertices to which it is adjacent at that timestep. Thus, we consider the following process.

1. At time $t = 0$, the root r is labeled as burning.
2. At all times $t \geq 1$, a chosen vertex is labeled as defended, and the fire then spreads to all undefended vertices adjacent to the fire on timestep t.
3. This process ends once there is no undefended and unburning vertex adjacent to the fire on the current timestep or any subsequent timestep.

Note that there is a subtle difference here in when the process will end. In Firefighter, if there is a timestep on which the fire is unable to spread, the process will end on that timestep. In Temporal Firefighter, it is possible that the fire will be unable to spread on a timestep, but may still spread again in the future, and thus the process will not end until the fire is unable to ever spread again. This will certainly be the case once the lifetime of the graph is exceeded, as no edges are ever active after this time.

We define a strategy for Temporal Firefighter equivalently to Definition 9 for Firefighter.

Definition 16 (Temporal Firefighter Strategy)**.** *Given a rooted temporal graph $(\mathcal{G}, r)$, a strategy $S = v_1, v_2, ..., v_\ell$ is a sequence of vertices from $\mathcal{G}$, such that if the fire begins at r on timestep 0 and each vertex in S is defended in turn, each v_i is unburning and undefended on timestep i, and the fire stops spreading on or before timestep ℓ.*

We now define the decision problem equivalently to Firefighter.

Temporal Firefighter
Input: A rooted temporal graph $(\mathcal{G}, r)$ and an integer k.
Output: Does there exist a strategy that saves at least k vertices on $\mathcal{G}$ when the fire starts at vertex r?

In some ways modifying Firefighter to take place on a temporal graph actually makes the task of defending the graph easier. For any vertex v to burn in Firefighter on a rooted graph (G, r), it is a necessary condition that there is a path from the root r to v. For the same vertex v to burn in Temporal Firefighter on a rooted temporal graph $(\mathcal{G}, r)$ with $\mathcal{G}\downarrow = G$, there must exist a temporal path from r to v. We observe that every temporal path on $\mathcal{G}$ has a corresponding path on G. Thus, there is no vertex reachable by the fire in Temporal Firefighter that is not reachable by the fire in Firefighter on the underlying graph.

Observation 1. *For every temporal path P on $\mathcal{G}$ there exists a path P' on $\mathcal{G}\downarrow$ that traverses the same vertices as P.*

Furthermore, in Firefighter on a static graph the fire spreads along paths at a rate of exactly one vertex per timestep. In Temporal Firefighter the fire cannot spread along a temporal path at a rate of greater than one vertex per timestep, and in fact the fire may spread at a slower rate if it has to wait at a vertex on the path for the next edge to become active. Thus the additions of times on the edges of a path will never increase the speed at which the path can be traversed, and may in fact slow this speed.

Observation 2. *The arrival time of a temporal path is at least the number of edges on the path.*

As a result, if the same defences are made in the same order, the fire cannot reach anywhere in Temporal Firefighter on a rooted temporal graph $(\mathcal{G}, r)$ that it would not be able to reach in Firefighter on the underlying static graph $(\mathcal{G}\downarrow, r)$. This gives us the following observation.

Observation 3. *Any strategy $S = v_1, ..., v_\ell$ for Firefighter on a rooted graph (G, r) is also a strategy for Temporal Firefighter on any rooted temporal graph $(\mathcal{G}, r)$ with $\mathcal{G}\downarrow = G$. Furthermore any vertex saved by S in Firefighter is also saved by S in Temporal Firefighter.*

As stated, the times on the edges in Temporal Firefighter may actually cause the fire to spread at a slower rate than in Firefighter on the underlying graph. Thus, there may exist strategies for Temporal Firefighter that save more vertices than it is possible to save in Firefighter on the underlying graph. It is therefore worthwhile to consider algorithms specific for Temporal Firefighter that produce strategies that take advantage of the temporal nature of the graph.

Unfortunately, such algorithms are unlikely to have polynomial-time runtimes. We now give a reduction from Firefighter to Temporal Firefighter, showing that the decision problem remains hard. Given an instance $((G, r), k)$ of Firefighter, the reduction produces an instance of Temporal Firefighter $((\mathcal{G}, r), k)$ with lifetime $|V(G)|-1$,

$\mathcal{G}\downarrow = G$, and every edge active on every timestep. Temporal Firefighter on $(\mathcal{G}, r)$ then behaves exactly as Firefighter on (G, r), due to the following observation.

Observation 4. *In both Firefighter and Temporal Firefighter the process must end by timestep $|V(G)| - 1$, as one vertex is defended per timestep, so every vertex other than the root will have been defended, and the fire will not be able to spread.*

Theorem 3. *Temporal Firefighter is **NP**-Complete.*

Proof. First see that Temporal Firefighter is in **NP**, with a strategy acting as a certificate that can be checked in polynomial time by simulating Temporal Firefighter. We now continue by reduction from Firefighter. Given an instance $((G, r), k)$ of Firefighter, we construct an instance $((\mathcal{G}, r), k)$ of Temporal Firefighter where $\mathcal{G}\downarrow = G$, and $\lambda(e) = \{1, ..., |V(G)| - 1\}$ for every edge $e \in E(G)$. Any strategy S that saves at least k vertices in Firefighter on (G, r), will also save at least k vertices in Temporal Firefighter on $(\mathcal{G}, r)$ as an immediate result of Observation 3. Furthermore, any strategy S that saves k vertices in Temporal Firefighter on $(\mathcal{G}, r)$ will also save at least k vertices in Firefighter on (G, r), as on every timestep t on which the fire may be able to spread, two vertices are adjacent on timestep t in $\mathcal{G}$ if and only if they are adjacent in G. □

We can thus view Firefighter as a special case of Temporal Firefighter. We then have the following corollary, as the above reduction preserves the underlying graph class:

Corollary 1. *For every class $\mathscr{C}$ of graphs for which Firefighter is **NP**-complete, Temporal Firefighter is **NP**-complete on the class of temporal graphs with the graphs of $\mathscr{C}$ as the underlying graphs.*

2.1.2 Temporal Graph Burning

We now define Temporal Graph Burning, an extension of Graph Burning to temporal graphs. Just as in Graph Burning, the player chooses a vertex to burn on each timestep, but as in Temporal Firefighter the fire only spreads to temporally adjacent vertices, rather than all adjacent vertices, as described in the following process:

1. At time $t = 0$, all vertices are unburnt.
2. At all times $t \geq 1$, the fire spreads, burning all vertices adjacent to an already burning vertex on timestep t. Then, a fire is placed at a chosen unburning vertex.
3. This process ends once all vertices are burning.

We define a strategy for Temporal Graph Burning equivalently to Definition 10 for Graph Burning.

Definition 17. *A strategy for a temporal graph $\mathcal{G}$ is a sequence of vertices $S = s_1, s_2, ..., s_\ell$ where if fires are placed at each vertex in turn, vertex s_i is unburnt on timestep i, and every vertex in the graph is burnt on or before timestep $\ell + 1$.*

We now define the decision problem equivalently to Graph Burning.

Temporal Graph Burning
Input: A temporal graph $\mathcal{G}$ and an integer h.
Output: Does there exist a successful burning strategy for $\mathcal{G}$ of length less than or equal to h?

Due to Observation 1 and Observation 2, the addition of times on the edges can only serve to limit the spread of the fire, just as in Temporal Firefighter. Therefore, any strategy for Temporal Graph Burning on a temporal graph $\mathcal{G}$ can be used to produce a strategy of no greater length for Graph Burning on $\mathcal{G}\downarrow$.

Observation 5. *Given a strategy S for Temporal Graph Burning on $\mathcal{G}$ there exists a strategy for Graph Burning on $\mathcal{G}\downarrow$ that burns the same vertices as S, but skips any turns that attempt to burn an already burning vertex.*

Any strategy must burn the graph by timestep $|V(G)|$ in both Graph Burning and Temporal Graph Burning:

Observation 6. *In both Graph Burning and Temporal Graph Burning the process will end by timestep $|V(G)|$, as a fire is placed at one vertex on every timestep, and thus if the process lasts for $|V(G)|$ timesteps a fire will have been placed at every vertex.*

Similarly to the previous reduction for Temporal Firefighter, we show that Temporal Graph Burning is **NP**-Complete by reduction from Graph Burning.

Theorem 4. *Temporal Graph Burning is **NP**-Complete.*

Proof. First see that Temporal Graph Burning is in **NP**, with a strategy acting as a certificate that can be checked in polynomial time by simulating Temporal Graph Burning. We now continue by reduction from Graph Burning. Given an instance (G, k) of Graph Burning, we construct an instance $(\mathcal{G}, k)$ of Temporal Graph Burning where $\mathcal{G}\downarrow = G$, and $\lambda(e) = \{1, ..., |V(G)|\}$ for every edge $e \in E(G)$. Any strategy S of length less than k for Temporal Graph Burning on $\mathcal{G}$, will also be a strategy for Graph Burning on G as an immediate result of Observation 5. Then consider a strategy S of length less than k for Graph Burning on G. From Observation 6 we have that $k \leq |V(G)|$, and if the fire spreads from any vertex u to an adjacent vertex v on

timestep t when S is played in Graph Burning on G, we have that $t \leq k \leq |V(G)|$ and u and v are adjacent on timestep t on $\mathcal{G}$. Therefore the fire will spread from vertex u to vertex v on timestep t when S is played in Temporal Graph Burning on $\mathcal{G}$, and S is a strategy for Temporal Graph Burning on $\mathcal{G}$ of length less than k. □

2.2 Restricting the Underlying Graph for Firefighter

We showed that for every class $\mathscr{C}$ of graphs for which Firefighter is **NP**-complete, Temporal Firefighter is **NP**-complete on the class of temporal graphs with the graphs of $\mathscr{C}$ as the underlying graphs. Therefore, any class of underlying graph for which Temporal Firefighter has a polynomial-time algorithm must also be a class for which Firefighter has a polynomial-time algorithm. Thus, we now determine the complexity of Temporal Firefighter on underlying graph classes for which Firefighter is known to be solvable in polynomial time. These are: interval graphs, permutation graphs, P_k-free graphs for $k > 5$, split graphs, cographs, and graphs of maximum degree 3 providing the root is of degree 2 [34, 38]. We prove that Temporal Firefighter is **NP**-Complete when we restrict the underlying graph to belong to all of these classes except the last, in which case we find there is a polynomial time algorithm. Additionally, we establish that it is **NP**-Complete when the underlying graph is an AT-free graph, a class for which the complexity of Firefighter has not been determined.

All of these hardness results follow from the fact that Temporal Firefighter is **NP**-hard when the underlying graph is a clique. This can be shown by reduction from Firefighter. We assign times to the edges in a static graph G so that they will be active at all times up until $|V(G)|-1$, at which point the fire can always no longer spread. We then add further edges to make the graph a clique, and have them only active from time $|V(G)|-1$ onwards such that they will not affect the spread of the fire. Temporal Firefighter on such a temporal clique will then simulate Firefighter on G. A sketch of this construction can be seen in Fig. 2.1.

We first show that given a yes-instance for Temporal Firefighter we may add edges to the input temporal graph in such a way that the resulting temporal graph still forms a yes-instance.

Lemma 5. *Suppose there is a strategy $S = v_1, ..., v_\ell$ for Temporal Firefighter on the rooted temporal graph $\mathcal{G} = (((V, E), \lambda), r)$ that saves k vertices. Let F be any set of edges disjoint from E, and $\lambda' : E \cup F \to \mathbb{N}$ be a labelling function such that $\lambda'(e) = \lambda(e)$ for every $e \in E$, and furthermore $\min(\lambda'(f)) \geq |V| - 1$ for all $f \in F$. Let S' be the strategy consisting of all the defences in S followed by defending every remaining undefended vertex in an arbitrary order. Then, S' will save k vertices in Temporal Firefighter on $(((V, E \cup F), \lambda'), r)$.*

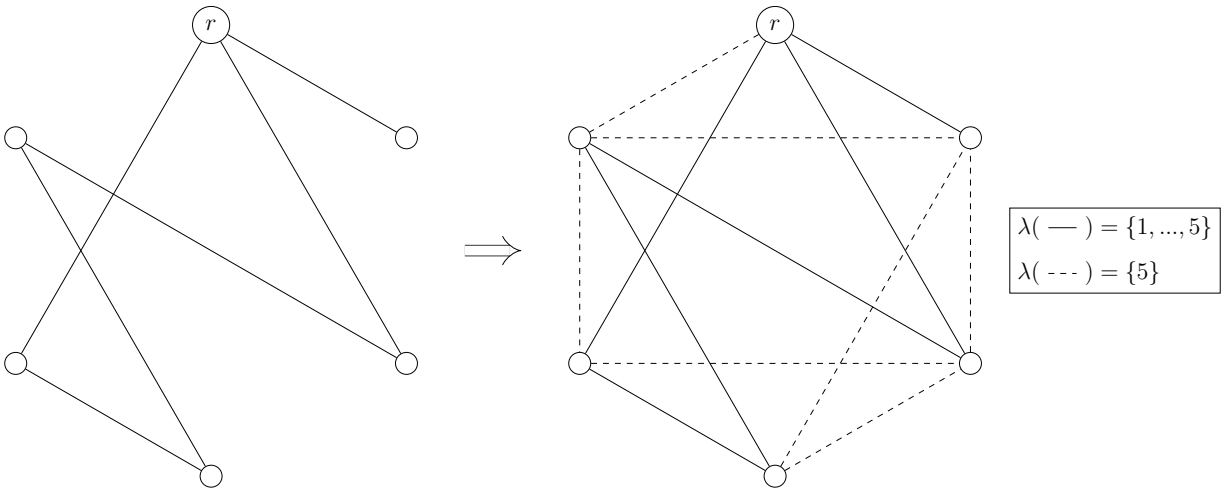

Figure 2.1: An example of the reduction for Temporal Firefighter that produces a clique.

Proof. We show by induction on the timestep t that any vertex unburnt on timestep t when the first t defences in S are played on $\mathcal{G}$ is also unburnt when the first t defences in S are played on $(((V, E \cup F), \lambda'), r)$. This allows us to inductively assume all the defences in S can be made in order on $(((V, E \cup F), \lambda'), r)$, as in particular the inductive hypothesis will imply that for any vertex $v_{t'}$ defended by S on a timestep $t' \leq t$ will not burn before the end of timestep $t' - 1$ on $(((V, E \cup F), \lambda'), r)$.

If $t = 0$ then the only vertex to have burnt in both instances is r. Otherwise, consider a vertex v unburnt on timestep t when S is played on $\mathcal{G}$, and see that as v is unburnt on timestep t, each of the temporal paths from r to v in $\mathcal{G}$ either contains a defended vertex or has an arrival time greater than t. Now consider the temporal paths from r to v in $(((V, E \cup F), \lambda'), r)$. Each of these paths is either also a path in $(((V, E), \lambda), r)$, or contains an edge from F and thus has an arrival time of at least $|V| - 1 > t$. In either case, the fire cannot have burnt along the path to v on timestep t, and thus v is unburnt on this timestep when the defences from S are played on $(((V, E \cup F), \lambda'), r)$.

Finally see that all vertices saved but not defended by S on $\mathcal{G}$ can be defended after the defences from S are played on $(((V, E \cup F), \lambda'), r)$. Consider any such vertex v, and see that any path from r to v in $(((V, E \cup F), \lambda'), r)$ either traverses a defended vertex or contains an edge from F. Thus any path from r to v that does not traverse a defended vertex has arrival time of at least $|V| - 1$, and v will not burn before this time. By Observation 4 every vertex in the graph must have been defended by this time, and thus v will never burn. □

We are now ready to give the reduction. This result allows us to determine that Temporal Firefighter is **NP**-complete on the class of temporal graphs $\{((G, \lambda), r) : (G, r) \in \mathscr{C}\}$ for all but one of the classes $\mathscr{C}$ for which it is currently known that Fire-

FIGHTER has a polynomial time solution.

Theorem 6. *TEMPORAL FIREFIGHTER is **NP**-complete on the class of temporal graphs where the underlying graph is a clique.*

Proof. We give a reduction from FIREFIGHTER. Given an instance (G, r, k) of FIREFIGHTER, we construct an instance $((G', \lambda), r', k')$ of TEMPORAL FIREFIGHTER which is a yes-instance if and only if (G, r, k) is a yes-instance of FIREFIGHTER.

We construct $((G', \lambda), r', k')$ as follows. Let G' be the vertex-edge pair (V, E') such that $V = V(G)$, and E' contains edges connecting every pair of distinct vertices. Then we define λ as follows:

$$\lambda(e) = \begin{cases} \{1, 2, ..., n-2\} & \text{if } e \in E(G) \\ \{n-1\} & \text{otherwise} \end{cases}$$

Finally, let $r' = r$ and $k' = k$.

We now show that if (G, r, k) is a yes-instance of FIREFIGHTER, then $((G', \lambda), r', k')$ is a yes-instance of TEMPORAL FIREFIGHTER. If (G, r, k) is a yes-instance then there is a strategy for firefighter on (G, r) that saves at least k vertices. We know from Observation 3 that this same strategy will also save at least k vertices in TEMPORAL FIREFIGHTER on $((G, \lambda'), r)$ where $\lambda'(e) = \{1, 2, ..., n-2\}$, and then from Lemma 5 we have that it is possible to save at least k vertices on $((G', \lambda), r)$, and thus $((G', \lambda), r', k')$ is a yes-instance, as $r' = r$ and $k' = k$.

To show the converse, we recall from Observation 4 that TEMPORAL FIREFIGHTER is always over by timestep $n - 1$ where $n = |V(G)|$, and thus the fire will only ever spread along edges that are active before this time. If $((G', \lambda), r', k')$ is a yes-instance then there is a strategy for TEMPORAL FIREFIGHTER on $((G', \lambda), r')$ that saves at least k' vertices. If the same strategy is used in firefighter on (G, r') then exactly the same vertices will burn on each timestep, as every edge present in G is active at all times until the fire stops spreading in (G', λ). Furthermore, as previously stated the fire will never spread along the edges present only in (G', λ), as these are only active at time $n - 1$. Thus (G, r, k) is also a yes-instance as $r' = r$ and $k' = k$. □

As a result we can deduce that TEMPORAL FIREFIGHTER is **NP**-Complete when the underlying graph belongs to several classes containing cliques for which FIREFIGHTER is in **P**. For the same reason, we can determine that TEMPORAL FIREFIGHTER is **NP**-Complete when the underlying graph belongs to the AT-free graphs, a class for which the complexity of FIREFIGHTER is still an open problem.

Corollary 2. *TEMPORAL FIREFIGHTER is **NP**-Complete when the underlying graph belongs to any of the following static classes: split graphs, unit interval graphs, cographs, P_k-free graphs for $k > 2$, and AT-free graphs.*

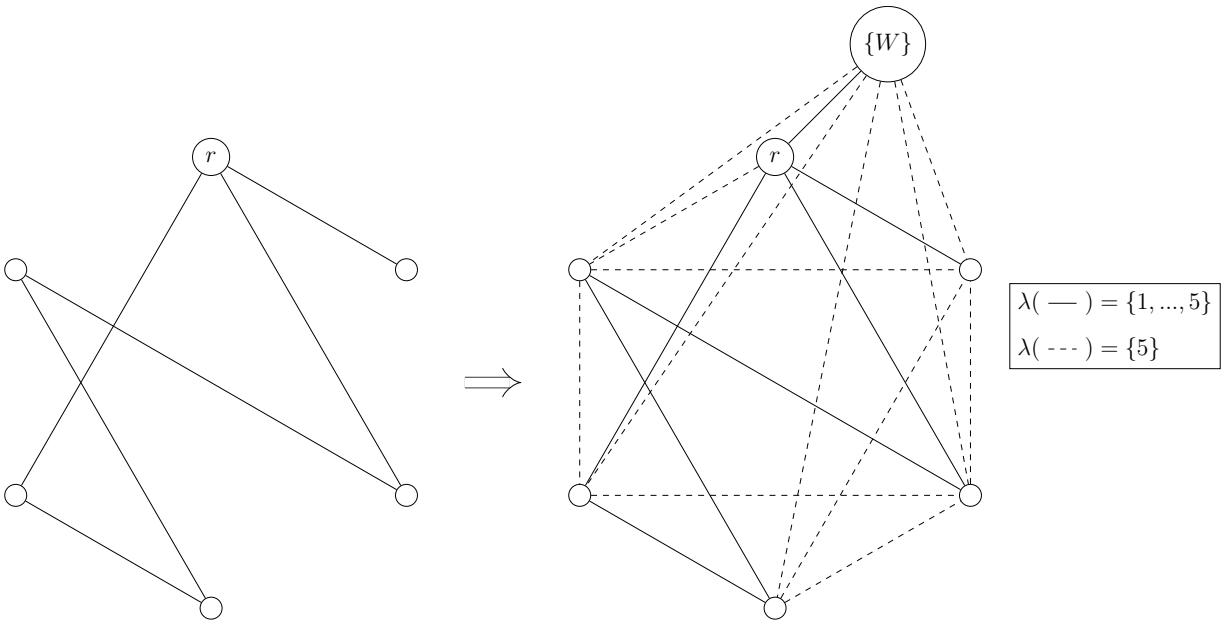

Figure 2.2: An example of the reduction for Temporal Firefighter on cliques with bounded lifetime. The vertex marked $\{W\}$ represents the set W containing $|V(G)|^c - |V(G)|$ vertices.

Given this reduction operates by adding edges active only at times greater than the final timestep on which the fire can spread, it is reasonable to consider if we might obtain polynomial-time algorithms for cliques where the lifetime is bounded by some function of the number of the vertices in the graph. We find that this is not the case, and now prove the stronger result that Temporal Firefighter is hard on cliques of n vertices with lifetime of less than $n^{\frac{1}{c}}$ for any positive integer constant c. This reduction operates by adding $n^c - n$ vertices to a static graph, and assigning times in such a way that they will all burn immediately, without affecting the spread of the fire over the existing graph. All defences then take place on a clique constructed in the same manner as that described in the proof of Theorem 6.

Theorem 7. *For any constant $c \in \mathbb{N}$, Temporal Firefighter is **NP**-complete when restricted to temporal graphs whose underlying graph is a clique and whose lifetime is at most $n^{\frac{1}{c}}$ where n is the number of vertices in the graph.*

Proof. We give a reduction from Firefighter; given an instance $((G, r), k)$ of Firefighter and a constant c, where G is a non-trivial connected graph, we construct an instance $(((G', \lambda), r), k)$ of Temporal Firefighter which is a yes-instance if and only if $((G, r), k)$ is a yes-instance of Firefighter.

Letting $\ell = |V(G)|$, we now construct an instance $(((G', \lambda), r), k)$ of Temporal Firefighter with ℓ^c vertices as follows. Let W be a set of $\ell^c - \ell$ vertices not in G. Then let G' be the graph (V', E'), with $V' = V(G) \cup W$, and E' containing edges connecting every

pair of distinct vertices in V', making the graph a clique, as shown in Fig. 2.2. We then define λ as follows:

$$\lambda(e) = \begin{cases} \{1, 2, ..., \ell - 2\} & \text{if } e \in E(G) \\ & \text{or } e = \{r, v\} \text{ and } v \in W \\ \{\ell - 1\} & \text{otherwise.} \end{cases}$$

Note that the lifetime of this instance is $\ell - 1$, which is less than $|V(G')|^{\frac{1}{c}} = \ell$, as required.

We now show that if $((G, r), k)$ is a yes-instance of Firefighter then $(((G', \lambda), r), k)$ is a yes-instance of Temporal Firefighter. If $((G, r), k)$ is a yes-instance then there is a strategy S for Firefighter on (G, r) that saves at least k vertices. We claim that we can save at least k vertices in Temporal Firefighter on $((G', \lambda), r)$ by first playing the defences from S, and then defending in arbitrary order the remaining unburnt vertices in $V(G') \setminus W$.

If we play the defences from S, then every vertex in W burns on the first timestep, and we are left with only the vertices in $V(G)$ to defend. Any path from a vertex in W to a vertex in $V(G)$ other than those that go via the root has an arrival time of at least $\ell - 1$, and we have at most $\ell - 2$ vertices left to defend after W burns, so the process must have ended before the fire spreads from W into $V(G)$. We can then restrict our attention to the spread of the fire over the subgraph of $((G', \lambda), r)$ induced by $V(G)$, and this is the temporal graph $((V(G), E(G) \cup F), \lambda')$ where F contains edges connecting every pair of vertices in $V(G)$ not connected by edges in $E(G)$, and λ' is defined as follows.

$$\lambda'(e) = \begin{cases} \{1, 2, ..., \ell - 2\} & \text{if } e \in E(G) \\ \{\ell - 1\} & \text{otherwise} \end{cases}$$

We know from Observation 3 that the strategy S will save at least k vertices in any temporal graph with G as the underlying graph, and then from Lemma 5 we know that it is possible to save at least k vertices on $((V(G), E(G) \cup F), \lambda')$, and thus $(((G', \lambda), r), k)$ is a yes-instance.

To show the converse, we first argue that if it is possible to save k vertices in Temporal Firefighter on $((G', \lambda), r)$ then in particular it is possible to do this without defending any vertices in W. It is only possible to defend a vertex in W on the first timestep, as every undefended vertex in W will burn on timestep 1. As G is non-trivial and connected, there must be at least one vertex v in $V(G)$ on $((G', \lambda), r)$ that is connected to r by an edge active at times $\{1, 2, ..., \ell - 2\}$, and so v will burn on the first timestep if a vertex in W is defended. Thus, defending v instead of a vertex in W on the first timestep saves at least as many vertices.

Next we observe that in any strategy that does not defend a vertex in W, the fire stops spreading by timestep $\ell - 1$, as every vertex in W burns instantly on timestep 1, and by timestep $\ell - 1$ it must be the case that every vertex in $V(G)$ is burnt or defended.

It follows that if $(((G', \lambda), r), k)$ is a yes-instance then there is a strategy for Temporal Firefighter on $((G', \lambda), r)$ that saves at least k vertices, and does not defend any vertices in W.

We now see that this same strategy is valid for Firefighter on (G, r). Firstly, it does not defend any vertices in W. Secondly, if we consider any defence v_i in the strategy, then we can see that all paths from the root r to v_i in (G, r) are also present in $((G', \lambda), r)$ and have arrival times equal to their length. If v_i is unburnt on timestep i when S is played in Temporal Firefighter on $((G', \lambda), r)$ then every such path is either defended, or has length greater than i. Thus, if we inductively assume the first $i - 1$ defences from S are valid on (G, r), we can see that v_i is unburnt on timestep i when these defences are played in Firefighter on (G, r), as every undefended path from r to v_i must have length greater than i.

Furthermore if a vertex v does not burn in Temporal Firefighter on $((G', \lambda), r)$ then it must not burn in Firefighter on (G, r), as every temporal path between r and v in (G', λ) with an arrival time of less than $\ell - 1$ has a corresponding path between r and v in G that traverses the same vertices. If v does not burn in Temporal Firefighter then each of the temporal paths from r to v either traverse a defended vertex, or have an arrival time of $\ell - 1$, and therefore have no corresponding path present in G. Therefore it is possible to save at least k vertices on (G, r), and $((G, r), k)$ is also a yes-instance. □

We have seen that Temporal Firefighter is **NP**-Complete on several graph classes for which Firefighter is in **P**. We now show that there exists a non-trivial class for which both Firefighter and Temporal Firefighter are in **P**, that being the class of graphs of maximum degree three, with a root of degree at most two.

A proof that Firefighter is polynomial-time solvable on this class is given by Finbow et al. [34]. This proof works due to the fact that it is always optimal to restrict the fire to spreading down only one path on such a graph. An algorithm need only find the shortest path at which the fire can be contained at the end, and then defend accordingly. Exactly the same can be done for Temporal Firefighter, the only difference being in calculating where the fire can be contained – sometimes the active times of the edges allow the fire to be contained at a vertex in Temporal Firefighter where it could not be contained in Firefighter played on the underlying graph.

We present a strategy S for Temporal Firefighter on a temporal graph $((G, \lambda), r)$ where G has maximum degree three and r has degree two in G, and show that this strategy is optimal and computable in polynomial time. This strategy and proof only requires slight modifications from that given by Finbow et al. for Firefighter [34].

Throughout, for any two vertices v and u in a temporal graph (G, λ) let $\text{dist}(v, u)$ be the number of edges on the shortest path between v and u in the underlying graph G.

After defining the strategy S we show that no strategy that does not always defend next to the fire can outperform S, and thus that there always exists an optimal strategy which only defends next to the fire. Such a strategy, due to the degree restriction, limits the fire to spreading along a single path. An optimal strategy then finds the shortest of such paths to a vertex at which the spread of the fire can be stopped. We observe that this can be done at any vertex u where there are one or fewer incident edges not on the path and active on timestep $\text{dist}(r, u) + 1$. Stated otherwise, as soon as the fire reaches a vertex at which the temporal nature of the graph delays its spread, it is possible to contain it, and in an optimal strategy the fire will spread along the path to such a vertex at a rate of one vertex per timestep, just as in Firefighter. We then show that strategy S is exactly this strategy, and then that the number of vertices saved by such an optimal strategy can be computed in polynomial time, as required.

We begin by defining three sets that will be used in the strategy: V_0, V_1, and V_c.

- Let V_0 contain any vertex u that is not adjacent at time $\text{dist}(r, u) + 1$ to any vertex that is not on any shortest underlying path between r and u.
- Let V_1 contain any vertex u that is adjacent at time $\text{dist}(r, u) + 1$ to exactly one vertex not on any shortest underlying path between r and u.
- Finally, let V_c be the set of all vertices that lie on a cycle and are not in V_0 or V_1.

Additionally, for any vertex u that lies on a cycle, let $C(u)$ denotes the length of the shortest cycle containing u.

Strategy S operates by first finding a vertex $u \in V_0 \cup V_1 \cup V_c$ that minimises the function $f(u)$, defined below.

$$f(u) = \begin{cases} \text{dist}(r, u) + 1 & \text{if } u \in V_0 \cup V_1 \\ \text{dist}(r, u) + C(u) - 1 & \text{if } u \in V_c \end{cases}$$

Now, let P be the shortest path from r to u on the underlying graph G. Furthermore, if $u \in V_c$, let C be the shortest cycle containing u.

If $u \in V_0 \cup V_1$, assume for a contradiction there is no temporal path on (G, λ) that traverses the same vertices as P and has arrival time $\text{dist}(r, u)$. If every vertex v on P is adjacent to the next vertex on P at timestep $\text{dist}(r, v) + 1$, then such a temporal path exists, so there must exist a vertex v on P and closer to r than u which is not adjacent to the next vertex on P at timestep $\text{dist}(r, v) + 1$. Thus, $v \in V_0 \cup V_1$, and $f(v) < f(u)$, contradicting the minimality of $f(u)$. Therefore there exists a temporal path on (G, λ) that traverses the same vertices as P and has arrival time $\text{dist}(r, u)$.

Otherwise, if $u \in V_c$, assume that P traverses a vertex v of C other than u. As v lies on the shortest path between r and u, it is necessarily closer to r than u. Additionally, as v lies on the same cycle as u, we would have that $C(v) \leq C(u)$, and thus $f(u) < f(u)$ which contradicts the minimality of u. Therefore P must not traverse any vertices of C other than u.

Now when $u \in V_0 \cup V_1$ on every timestep $1 \leq t < f(u)$, strategy S defends the vertex adjacent to the fire on timestep t and not on P. On timestep $f(u)$, S defends either a non-burning neighbour of u adjacent on timestep $f(u)$ if one exists, or otherwise any other non-burning neighbour of u. If there exists any further non-burning neighbour of u, S defends this on timestep $f(u)+1$. Once the fire stops spreading, the burnt vertices will be all those on P, meaning that in total $f(u)$ vertices will be burnt.

Otherwise, if $u \in V_c$ then on every timestep $1 \leq t < \text{dist}(r,u)+1$, S defends the vertex adjacent to the fire on timestep t and not on P. Then on timestep $\text{dist}(r,u)+1$, S defends one of the two non-burning vertices on C adjacent to the fire on timestep $\text{dist}(r,u)+1$. Then for every remaining timestep S defends the vertex adjacent to the fire on that timestep and not on C. Once the fire stops spreading, the burnt vertices will be all those on P and all those on C except one, meaning once again $f(u)$ vertices are burnt in total.

We now show that there is an optimal strategy that always defends next to the fire. The argument here is equivalent to that for Firefighter [34], but uses comparison to our newly defined strategy S to show optimality.

Lemma 8. *Given a rooted temporal graph $((G,\lambda),r)$ where G has maximum degree 3 and r has degree at most 2 in G, if k is the greatest number of vertices that can be saved in Temporal Firefighter on $((G,\lambda),r)$, then there exists a strategy for Temporal Firefighter that saves k vertices and always defends adjacent to the fire.*

Proof. Assume for a contradiction that there is a rooted temporal graph $((G,\lambda),r)$ with maximum degree 3 and root of degree 2 such that there exists a strategy S for Temporal Firefighter that saves k vertices, but no strategy S' that saves k vertices and always defends adjacent to the fire, that is a counterexample, and that this counterexample is minimal in the number of vertices in G. Let x_1 and x_2 be the two neighbours of r. If there is a strategy that saves k vertices in which the first vertex defended is a neighbour of r, say x_1 without loss of generality, then $((G-\{r,x_1\},\lambda),x_2)$ is a smaller counterexample – a contradiction.

Let T be some strategy for Temporal Firefighter on $((G,\lambda),r)$ that saves k vertices, and let u be the closest vertex to r defended in T. This cannot be a neighbour of r, and thus $\text{dist}(r,u) \geq 2$.

If no neighbours of u are burning once the fire stops spreading, then there are no temporally admissible paths between r and u. In this case T wastes a defence defending

u, a vertex that will never burn, and thus this defence can be replaced with a defence of a vertex that burns when T is played, producing a strategy that saves $k+1$ vertices, a contradiction.

If only one neighbour of u is burning once the fire stops spreading, then defending this neighbour instead of u saves one more vertex, and thus we again have a contradiction.

If once the fire stops spreading two or more neighbours of u are burning, then u lies on a cycle that is completely burnt except for u. In this case we argue that the strategy S must save at least as many vertices. Strategy S finds a vertex v that minimizes $f(v)$, and always causes $f(v)$ vertices to burn, saving the rest. If T performs better than S, then there is a vertex $w \in V_c$ lying on the same cycle as u where the entire path between r and w has burnt, as well as the entire cycle except for u, thus meaning that $f(w) < f(v)$ vertices burn. This is impossible – $f(v)$ is a minimum. As S always defends next to the fire, its optimality contradicts the assumption. □

We now show that strategy S is an optimal strategy.

Theorem 9. *Temporal Firefighter can be solved in polynomial time on a rooted temporal graph $((G,\lambda),r)$ of maximum degree 3 with a root of degree at most 2.*

Proof. First we note that if strategy S is played on the graph $((G,\lambda),r)$ then $\min\{f(u) \mid u \in V_0 \cup V_1 \cup V_c\}$ vertices will burn. We now show that strategy S is optimal.

By Lemma 8 there is an optimal strategy T in which each vertex defended is next to the fire, thus restricting the fire to spreading down a single path. Let w be the final vertex to burn, at the end of this path.

Due to the degree restriction there are at most two vertices adjacent to w that do not lie on the path from r down which the fire burnt to reach w. There are two ways in which the fire can stop spreading at w. In the first case both of these vertices are defended after the fire reaches w, and in the second at least one of these vertices has already been defended before the fire reaches w, and any undefended neighbours are defended afterwards.

In the first case, we must have that $w \in V_0 \cup V_1$, as there must have been time to make these defences after the fire reached w. Furthermore at least $\text{dist}(r,w)+1$ vertices must have burnt, and strategy S performs at least as well, and is therefore optimal.

Otherwise, in the second case, w must lie on a cycle that is fully burnt except for one vertex, as the already defended vertex must be adjacent to some burning vertex, and therefore adjacent to a vertex that lies on the burnt path from r to w, as these are the only vertices that burn. Let v be the first vertex on C to have burnt. A path from r to v must be fully burnt, as is all of C except for one vertex. Thus $f(v)$ vertices have burnt in total. As strategy S finds a vertex u that minimises $f(u)$, and allows $f(u)$ vertices to burn, it must be the case that $f(u) = f(v)$, and thus strategy S is optimal. □

We now give a polynomial algorithm that computes how many vertices S saves, showing that Temporal Firefighter is in **P** for temporal graphs of maximum degree 3 with roots of degree 2.

Algorithm 1 Temporal Firefighter on temporal graphs of maximum degree 3 with a root of maximum degree 2

Input: A rooted temporal graph $((G, \lambda), r)$ and integer k where G has maximum degree 3 and r has maximum degree 2.

Output: Returns true if at least k vertices can be saved in the temporal firefighter game on $((G, \lambda), r)$.

$V_0, V_1, V_c \leftarrow \emptyset$
for each vertex $v \in V(G)$ **do**
  $P \leftarrow$all vertices on the shortest underlying path between r and v
  if $N_{\text{dist}(r,u)+1}[v] \setminus P = \emptyset$ **then**
    $V_0 \leftarrow V_0 \cup \{v\}$
  else if $|N_{\text{dist}(r,u)+1}[v] \setminus P| = 1$ **then**
    $V_1 \leftarrow V_1 \cup \{v\}$
  else if v is on a cycle **then**
    $V_c \leftarrow V_c \cup \{v\}$
  end if
end for
return $k \geq |V(G)| - \min\{f(v) : v \in V_0 \cup V_1 \cup V_c\}$

Theorem 10. *Algorithm 1 will run in time $O(n^3)$, where n is the number of vertices in the graph.*

Proof. For each vertex v in the input graph (G, λ), Algorithm 1 finds the shortest underlying path between r and v, and thus also the underlying distance between r and v. This can be computed in time $O(n^2)$ for each vertex. The algorithm also finds the shortest cycle containing v if it exists, and thus the number of vertices on this cycle, which is again possible in time $O(n^2)$ for each vertex, giving an overall runtime of $O(n(n^2+n^2))$. Finally, the algorithm iterates over every vertex v, computing the value of the function $f(v)$ to find the minimum. As the distances between r and v, as well as the size of the shortest cycle containing v have already been computed, $f(v)$ can be can be computed in constant time. Thus this final step can be achieved in time $O(n)$, giving an overall runtime of $O(n(n^2 + n^2) + n) = O(n^3)$. □

Corollary 3. *Temporal Firefighter is in **P** for temporal graphs where the underlying graph has maximum degree 3 and the root has maximum degree 2.*

2.3 Conclusions

In this chapter, we defined the problems Temporal Firefighter and Temporal Graph Burning. These problems are natural temporal extensions of the Firefighter

and Graph Burning problems. We showed that in both problems, fires can never spread faster in a temporal graph than they can on the underlying static graph, and that both problems remain **NP**-Complete.

We then considered restricting the underlying graph class for Temporal Firefighter, and found this approach to be limited in its ability to produce tractable algorithms. For all but one class of static graphs where Firefighter is in **P**, Temporal Firefighter remains **NP**-Complete, as Temporal Firefighter is **NP**-Complete on cliques. We were only able to find a polynomial time algorithm for the very sparse class containing graphs of maximum degree 3 where the fire begins at a vertex of degree 2.

Evidently, in the case of Firefighter, restrictions on the underlying graph class are too weak to produce tractable algorithms. This is because the class of the underlying graph tells us very little about the actual behaviour of the temporal graph. For example, in our reduction in Theorem 6 we produce a temporal graph where the underlying graph is a clique. However, several of the edges that make this graph a clique are only active on one very late timestep, and as such do not actually affect the spread of the fire in Firefighter. Due to the lack of correlation between the underlying graph class and the behaviour of a temporal graph, it is also possible that restricting the underlying graph class is would be too strong a restriction in other settings, and would needlessly rule out temporal graphs where properties of the temporal structure actually allow for tractable algorithms despite the underlying graph class.

Motivated by the unsuitability of restricting the underlying graph class, the remaining chapters will use the tools of paramterised complexity to consider restrictions to the temporal structure of a graph. We begin in the next chapter by considering two parameters that are small when the activity of the edges in a temporal graph is limited.

Chapter 3

Parameterised Complexity of Temporal Firefighter on Sparse Temporal Graphs

In Chapter 2 we saw that Temporal Firefighter remained **NP**-Complete when the underlying graph belongs to many of the classes for which Firefighter is known to be in **P**, with one main exception when the underlying graph has maximum degree 3 and a root of degree 2. In this chapter we again consider the complexity of Temporal Firefighter, but now restrict the temporal structure of the input graph, rather than just the underlying static structure. We achieve this using the tools of parameterised complexity, in particular we determine the parameterised complexity of Temporal Firefighter for two existing parameters: vertex-interval-membership-width and edge-interval-membership-width. Intuitively, a temporal graph with small (vertex or edge)-interval-membership-width can be thought of as having low temporal activity. The parameters measure the maximum number of vertices or edges respectively that overlap in the time intervals in which they are active. Both of these parameters are small only when every snapshot graph of the temporal graph is sparse, but it is worth noting that this is not a sufficient condition for the parameters to be small. For example, these width measures can be unbounded even when the underlying graph is a path. In this way, both parameters can be said to capture a form of sparsity, but specifically sparsity of activity. We find that Temporal Firefighter is in **FPT** when parameterised by vertex-interval-membership-width, but remains **NP**-Complete when the edge-interval-membership-width is bounded.

We first define the parameters of vertex-interval-membership-width and edge-interval-membership-width, and give some useful lemmas about their properties and the complexity of computing them. We then present an algorithm for Temporal Firefighter when parameterised by vertex-interval-membership-width. Finally, we prove that Temporal Firefighter is **NP**-Complete even when the edge-interval-membership-width is

bounded.

3.1 Background

We begin by recalling the definition of edge-interval-membership-width, due to Bumpus and Meeks [20]. For each edge, we consider its *active interval* to be the sequence of timesteps between the first timestep on which it is active, and the final timestep on which it is active. Note that there may exist timesteps within an edge's active interval on which the edge is not active, however on such a timestep it is always true that the edge will again be active at some point in the future. The parameter then measures the maximum number of edges in their active interval on any given timestep.

Definition 18 (Edge-Interval-Membership-Width (Bumpus and Meeks [20]))**.** *The edge-interval-membership-sequence of a temporal graph (G, λ) with lifetime Λ is a sequence $(F_t)_{t\in[\Lambda]}$ of subsets of $E(G)$ where $F_t = \{e \in E(G) : \min(\lambda(e)) \leq t \leq \max(\lambda(e))\}$.*

The edge-interval-membership-width of a temporal graph is then the integer $\psi = \max_{t\in[\Lambda]} |F_t|$.

Note that if the edge-interval-membership-width is bounded, then so is the number of active edges on any given timestep. However the converse is not true: it is possible for a temporal graph to have a constant number of active edges on every timestep, but an arbitrarily large edge-interval-membership-width.

It is possible to compute the edge-interval-membership-width in time polynomial in the lifetime of the temporal graph and the edge-interval-membership-width itself.

Lemma 11 (Bumpus and Meeks [20])**.** *There is an algorithm that, given a temporal graph $\mathcal{G}$ with every edge active at least once and with lifetime Λ, computes the edge-interval-membership-sequence of $\mathcal{G}$ in time $O(\omega\Lambda)$ where ω is the edge-interval-membership-width of $\mathcal{G}$.*

The vertex-interval-membership-width, also defined by Bumpus and Meeks, is very similar, but considers the number of vertices, rather than edges, in their active interval. The active interval of a vertex v refers to the inclusive sequence of timesteps from the earliest timestep on which any edge incident at v is active to the latest timestep on which any edge incident at v is active. We denote these two timesteps with $\text{mintime}(v) = \min(\{\min(\lambda(e)) : e \text{ is incident at } v\})$, and $\text{maxtime}(v) = \max(\{\max(\lambda(e)) : e \text{ is incident at } v\})$.

Definition 19 (Vertex-Interval-Membership-Width (Bumpus and Meeks [20]))**.** *The vertex-interval-membership-sequence of a temporal graph (G, λ) with lifetime Λ is the sequence $(F_t)_{t\in[\Lambda]}$ of vertex-subsets of G where $F_t = \{v \in V(G) : \text{mintime}(v) \leq t \leq \text{maxtime}(v)\}$.*

The vertex-interval-membership-width of a temporal graph (G, λ) is then the integer $\omega = \max_{t\in[\Lambda]} |F_t|$.

The vertex-interval-membership-width and edge-interval-membership-width are incomparable. In some cases the edge-interval-membership-width maybe higher than the vertex-interval-membership-width: take for example a temporal graph where the underlying graph is a clique, and there is at least one timestep on which every edge is active. The edge-interval-membership-width of this graph is equal to the number of edges, which is $\frac{1}{2}n(n-1)$ where n is the number of vertices. The vertex-interval-membership-width is n. In other cases, the vertex-interval-membership-width is higher than the edge interval-membership-width, and in fact can be arbitrarily higher, as shown by Bumpus and Meeks [20].

Just as with the edge-interval-membership-width, we can compute the vertex-interval-membership-width in time polynomial in the lifetime of the temporal graph and the vertex-interval-membership-width.

Lemma 12 (Bumpus and Meeks [20])**.** *There is an algorithm that, given a temporal graph $\mathcal{G}$ with every edge active at-least once and with lifetime Λ, computes the vertex-interval-membership-sequence of $\mathcal{G}$ in time $O(\omega\Lambda)$ where ω is the vertex-interval-membership-width of $\mathcal{G}$.*

Finally, we see that any entries from the edge-interval-membership-sequence containing an edge e must always occur consecutively within the edge-interval-membership-sequence, as it is not possible for an edge to ever be active again after it has left its active interval.

Lemma 13. *Given a temporal graph $\mathcal{G}$ with edge-interval-membership-sequence $[F_t]_{t\leq\Lambda}$, for any edge e and times s, t, and u with $s \leq t \leq u$, if e is in F_s and F_u, it must be in F_t.*

Proof. If $e \in F_s$, then by definition $\min(\lambda(e)) \leq s$. Furthermore, if $e \in F_u$ then by definition $\max(\lambda(e)) \geq u$. Now $\min(\lambda(e)) \leq s \leq t \leq u \leq \max(\lambda(e))$, and therefore $e \in F_t$. □

The same is also true of the vertex-interval-membership-sequence. The entries that contain a vertex v must always occur consecutively within the sequence, as by definition it is not possible for a vertex to ever be active again after it has left its active interval.

Lemma 14. *Given a temporal graph $\mathcal{G}$ with vertex-interval-membership-sequence $[F_t]_{t\leq\Lambda}$, for any vertex v and times s, t, and u with $s \leq t \leq u$, if v is in F_s and F_u, it must be in F_t.*

Proof. If $v \in F_s$, then by definition $\text{mintime}(v) \leq s$. Furthermore, if $e \in F_u$ then by definition $\text{maxtime}(v) \geq u$. Now $\text{mintime}(v) \leq s \leq t \leq u \leq \text{maxtime}(v)$, and therefore $e \in F_t$. □

Having introduced the parameters of edge-interval-membership-width and vertex-interval-membership-width, we are now ready to show that Temporal Firefighter is in **FPT**

when parameterised by vertex-interval-membership-width, before going on to show that it remains **NP**-Complete even when the edge-interval-membership-width is bounded.

3.2 A Fixed Parameter Tractable Algorithm

We find that Temporal Firefighter is in **FPT** when parameterised by vertex-interval-membership-width. To simplify our analysis when showing this, we actually use the related problem Temporal Firefighter Reserve.

Temporal Firefighter Reserve is the temporal extension of the Firefighter Reserve problem described by Fomin et al. [38]. In Temporal Firefighter Reserve, it is not required to make a defensive move every timestep. Rather, each timestep a budget is incremented by 1, and it is then possible to defend any number of vertices less than or equal to the budget simultaneously, subtracting from the budget appropriately. Stated more formally, the problem ask whether it is possible to save k vertices from burning in the following process:

1. At time $t = 0$, the root r is labeled as burning, and the budget is 0.
2. At all times $t \geq 1$, the budget is incremented by 1, and then a chosen (possibly empty) set of vertices D are labeled as defended, provided $|D|$ is at most equal to the budget. The budget is then decremented by $|D|$, and the fire spreads to all undefended vertices adjacent to the fire on timestep t.
3. The process ends once the fire cannot spread on the current timestep or any future timesteps.

A strategy is then defined as a sequence of sets of valid defences, such that on each timestep there is sufficient budget to defend all the vertices in the corresponding set.

Definition 20. *Given a rooted temporal graph $(\mathcal{G}, r)$, a strategy $S = D_1, D_2, ..., D_\ell$ is a sequence of sets of vertices from $\mathcal{G}$, such that $\Sigma_{i=1}^{t}|D_i| \leq t$ for all timesteps t, and if the fire begins at r on timestep 0 and each set of vertices is defended in turn, all the vertices in each D_i are unburning and undefended on timestep i, and the fire stops spreading on or before timestep ℓ.*

The decision problem is then phrased identically to Temporal Firefighter.

Temporal Firefighter Reserve
Input: A rooted temporal graph $(\mathcal{G}, r)$ and an integer k.
Output: Does there exist a strategy that saves at least k vertices on $\mathcal{G}$ when the fire starts at vertex r?

Just as in the static Firefighter Reserve, allowing the defence to build up a reserve in this manner does not affect the number of vertices than can be saved. We prove this using the same argument as that for the static case as given by Fomin et al. [38].

Lemma 15. *An instance $((G, \lambda), r)$ along with an integer k of Temporal Firefighter Reserve is a yes-instance if and only if $((G, \lambda), r)$ along with k is a yes-instance of Temporal Firefighter.*

Proof. Given a temporal graph $((G, \lambda), r)$, assume there is a strategy for Temporal Firefighter Reserve that saves k vertices. Any strategy for Temporal Firefighter is a valid strategy for Temporal Firefighter Reserve, and thus it is also possible to save k vertices in Temporal Firefighter Reserve by playing the same strategy.

Now assume that there is a strategy that saves k vertices in Temporal Firefighter Reserve on $((G, \lambda), r)$. We can transform this strategy into a valid strategy for Temporal Firefighter that saves the same number of vertices as follows: if at any timestep t the strategy defends $d > 1$ vertices, the budget on timestep t must be at least d, and there therefore must have been at least $d - 1$ timesteps prior to t where no defences were made. By making exactly one of these d defences on the latest $d - 1$ each of these $d - 1$ prior timesteps, and on timestep t itself, we produce a valid strategy for Temporal Firefighter. Modifying the strategy in this manner creates a valid strategy for Temporal Firefighter, as, if defending vertex v is valid on timestep t, it must also be valid at any timestep less than t. Finally, as the exact same defences occur, only at an earlier time, the modified strategy must also save at least k vertices. □

Additionally, we note that in Temporal Firefighter Reserve there is always an optimal strategy that only defends vertices temporally adjacent to the fire, as any defence can be delayed until the turn upon which the defended vertex would burn. More generally, there is always an optimal strategy which defends only vertices at time t if they have an incident edge active at time t. From now on when we refer to strategies for Temporal Firefighter Reserve, we assume that they all have this property.

We now give an algorithm for Temporal Firefighter Reserve that iterates over the vertex-interval-membership sequence of the input graph, and show that it is fixed parameter tractable with respect to vertex-interval-membership-width.

The algorithm takes as input a rooted temporal graph $((G, \lambda), r)$ with lifetime Λ and vertex-interval-membership-sequence $[F_i]_{i \in [\Lambda]}$, and an integer k, and determines if it is possible to save k vertices in temporal firefighter reserve played on the graph. For any edge set $E \subseteq E(G)$, let $V(E)$ be the set of vertices with an incident edge in E. Also, for any set X let $\mathcal{P}(X)$ denote the powerset of X: $\{X' : X' \subseteq X\}$. The algorithm then operates by recursively computing a sequence of sets $L_i \in \mathcal{P}(F_i) \times \mathcal{P}(F_i) \times \{1, 2, ..., \Lambda\} \times \{1, 2, ..., n\}$ for each F_i in the vertex-interval-membership-sequence.

An element of L_i is a 4-tuple (D, B, g, c) where D is a set of defended vertices in F_i, B is a set of burnt vertices in F_i, g is the budget that will be available on timestep $i+1$, and c is the total count of vertices that have burnt at time i.

To determine the spread of the fire it is only necessary to keep track of the root, along with the vertices that have burned or been defended in F_i, as if a vertex is not in F_i all its incident edges must either only be active before time i, or after time i. If the former is the case, and the vertex is not the root, then the fire cannot spread from or to it after time i, meaning that whether it is burning or defended does not affect the spread of the fire after this point. If the latter is the case, and the vertex is not the root, then the vertex cannot be burning, as the fire cannot have reached it yet, and as we only defend vertices with incident edges active at time i, it cannot be defended either.

Additionally, it is possible to compute these defended and burning sets recursively from only a previous entry in the sequence, as we know from Lemma 14 that a vertex v can only be in a sequence of consecutive F_is.

Finally, we check if there is any entry $(D, B, g, c) \in L_\Lambda$ where Λ is the lifetime of the graph, such that $|V(G)| - c \geq k$, returning true if so.

We recursively compute the sequence L_i, beginning by initialising $L_0 = (\emptyset, \{r\} \cap F_0, 1, 1)$. Let E_i be the set of edges active at time i, and $N_i(S)$ the set of all vertices adjacent at time i to the vertices in S, for any set $S \subseteq V(G)$. Note that $V(E_i) \subseteq F_i$, as the timestep i is within a vertex's active interval if the vertex has an incident edge active on i.

We then require that, for any set $A \subseteq V(E_i) \setminus (D \cup B)$ containing vertices to be defended on timestep i, $(D', B', g - |A| + 1, c') \in \mathcal{P}(F_i) \times \mathcal{P}(F_i) \times \{1, 2, ..., \Lambda\} \times \{1, 2, ..., n\}$ is in L_i if and only if there is a tuple (D, B, g, c) in L_{i-1}, such that:

(1) $D' = (D \cap F_i) \cup A$

(2) $B' = (B \cap F_i) \cup N_i(B \cup \{r\}) \setminus D'$

(3) $g - |A| + 1 > 0$

(4) $c' = c + |N_i(B \cup \{r\}) \setminus (D' \cup B \cup \{r\})|$

That is, given sets of burning and defended vertices in F_{i-1}, we create a set of burning and defended vertices in F_i appropriately for each possible set of defences on vertices with incident edges active at time i.

Condition **1** ensures that the defended set contains only previously defended vertices in F_i, along with the set of new defences A.

Condition **2** specifies that the burning set contains only previously burnt vertices in F_i, allong with all non-defended vertices temporally adjacent to the fire.

Condition **3** ensures that the budget is correct. The budget available at the end of timestep i will be $g - d + 1$ if at the end of timestep i the budget is g, and d vertices are to be defended, as the budget decreases by the number of defences made, but increases by 1 per timestep.

Condition **4** counts the number of newly burnt vertices.

We show that computing these sets correctly answers the Temporal Firefighter problem. That is, that entries exist in the sequence if and only if there is a corresponding strategy, and thus it is possible to check if k vertices can be saved in temporal firefighter on the rooted temporal graph $((G, \lambda), r)$.

Theorem 16. *Given a temporal graph $((G, \lambda), r)$ there is an entry $(D, B, g, c) \in L_i$ if and only if there exists a strategy for Temporal Firefighter Reserve on $((G, \lambda), r)$ such that D and B are the vertices in F_i that are defended and burnt respectively by timestep i, g is the budget that will be available on timestep $i+1$, and c is the total number of vertices that burn by timestep i.*

Proof. We proceed by induction on i. After timestep 0, only one vertex (the root) has burnt, and $L_0 = \{(\emptyset, \{r\}, 1, 1)\}$. We suppose that for every entry $(D, B, g, c) \in L_{i-1}$, there is a corresponding strategy S_{i-1}, and show that if $(D', B', g - d + 1, c') \in L_i$ then there is a corresponding strategy S_i.

Assume that there is an entry $(D', B', g - d + 1, c') \in L_i$, we will now show that there exists a corresponding strategy. If $(D', B', g - d + 1, c') \in L_i$ there must be an entry $(D, B, g, c) \in L_{i-1}$ and set of vertices $A \subseteq V(E_i) \setminus (B \cup D)$ with $d = |A|$ such that Conditions **1** through **4** hold. By our induction hypothesis there is a corresponding strategy S_{i-1} for this entry such that D and B are the vertices in F_{i-1} that are defended and burnt respectively by timestep $i - 1$, g is the budget available at the end of timestep $i - 1$, and c is the total number of vertices burnt by timestep $i - 1$. If we take S_{i-1} and extend it by defending the set of vertices A on timestep i, then we obtain a strategy S_i that we claim corresponds to $(D', B', g - d + 1, c')$.

First see that by the definition of A, all the defences it contains are valid, as A contains only vertices in $V(E_i) \subseteq F_i$ that have not either already burnt or been defended, and $g - |A| + 1 > 0$, and g is the budget available at the end of timestep $i - 1$.

The vertices that are newly defended on timestep i when S_i is played are only those in A. Thus the vertices that are defended in F_i on timestep i when S_i is played are those that were already defended and are also in F_i, that is $D \cap F_i$, and those that are newly defended. Thus by condition **1** the vertices in F_i that are defended by timestep i when S_i is played are those in D'.

The vertices that then burn on timestep i when S_i is played are all those adjacent to the fire on timestep i and not defended by S_i. Additionally, any vertex from which the fire spreads on timestep i must be in F_i, as it must have an incident edge active at time i.

For the same reason, any defended vertex that the fire would otherwise burn on timestep i must also be in F_i. Thus, $N_i(B) \setminus D'$ is the set of vertices that newly burn on timestep i. Therefore the vertices that have burnt in F_i on timestep i when S_i is played are those that had already burnt and are also in F_i, that is $B \cap F_i$, and those that newly burn: $N_i(B) \setminus D'$. Thus by condition **2** the vertices in F_i that have burnt by timestep i when S_i is played are those in B'.

The budget available on timestep $i+1$ when S_i is played is the budget available on timestep i incremented by 1, with $|A|$, the number of defences made on timestep i, subtracted. Thus by condition **3** the budget available at timestep $i+1$ when S_i is played is $g - d + 1$.

The set of vertices that newly burn after timestep i when S_i is played is all those adjacent to the fire on timestep i and not defended or already burning, so the number of such vertices is $|N_i(B) \setminus (B \cup D')|$. The total number of vertices to have burnt after timestep i when S_i is played is then $c + |N_i(B) \setminus (B \cup D')|$, thus by condition **4** the total number of vertices to have burnt is c'.

We now show the converse: that if there is a strategy S_i such that after time i the sets of vertices that have been defended and burnt are D_S and B_S respectively, g' is the available budget, and c' is the total number of vertices to have burnt, then there is a corresponding entry $(D', B', g', c') \in L_i$, such that $D' = D_S \cap F_i$ and $B' = B_S \cap F_i$.

Consider the state at timestep $i-1$ if strategy S_i is played. By our induction hypothesis there is a corresponding entry $(D, B, g, c) \in L_{i-1}$ where D is the set of vertices in F_{i-1} that are defended at time $i-1$, B is the set of vertices in F_{i-1} that are burnt at time $i-1$, g is the budget that will be available at time i, and c is the total number of vertices to have burnt at time $i-1$.

Let A be the set of vertices defended at time i when strategy S_i is played. As we consider only strategies that only defend vertices at time i with incident edges at time i, and A is a valid defence, we have that $A \subseteq V(E_i) \setminus (B \cup D)$.

By our induction hypothesis D is the set of vertices in F_{i-1} that are defended by the end of timestep $i-1$, so the set of vertices in F_i defended by time i is $D_S \cap F_i = (D \cap F_i) \cup A$, satisfying condition **1**.

Again by the induction hypothesis B is the set of vertices in F_{i-1} that are burnt by the end of timestep $i-1$. The only vertices from which the fire can spread on timestep i are those that have an incident edge active at time i, and thus are in F_i. For the same reason the only defended vertices that would otherwise burn on timestep i are in F_i. Therefore the vertices in F_i burnt by time i are $B_S \cap F_i = (B \cap F_i) \cup N_i(B) \setminus D_S = (B \cap F_i) \cup N_i(B) \setminus (D \cup A)$, satisfying condition **2**.

Finally, the budget available at time i is g, and so the budget at time $i+1$ is $g' = g - |A| + 1$, satisfying condition **3**, and the number of vertices burnt after time $i-1$

is c, so the number of vertices burnt after time i is $c' = c + |N_i(B) \setminus (B \cup D_S)| = c + |N_i(B) \setminus (B \cup (D \cap F_i) \cup A)|$, satisfying condition **4**.

Thus we see that, given (D, B, g, c) is an entry in L_{i-1}, we have that $(D_S \cap F_i, B_S \cap F_i, g', c')$ satisfies conditions **1**-**4**, and thus is an entry in L_i. □

We now determine the runtime of computing all sets L_i, thus showing that Temporal Firefighter is FPT when parameterised by vertex-interval-membership-width.

Theorem 17. *It is possible to solve Temporal Firefighter in time $O(8^\omega \omega \Lambda^3)$ for a rooted temporal graph $((G, \lambda), r)$ where Λ is the lifetime of the graph, and ω is the vertex-interval-membership-width.*

Proof. Temporal Firefighter Reserve, and therefore Temporal Firefighter, can be answered by computing all sets L_i, and then checking if there exists an entry $(D, B, g, c) \in L_\Lambda$ with $|V(G)| - c \geq k$. Thus, it suffices to show that each of these sets can be computed in the required time.

First observe that the total number of burnt vertices on any given timestep i is at most $\Sigma_{j=1}^{i} |V(E_j)| = O(\omega\Lambda)$, as on each timestep j only vertices in $V(E_j)$ can burn, and $|V(E_j)| \leq \omega$, and for any timestep i we have that $i \leq \Lambda$.

Now see that for any i, we have that $|L_i| = O(4^\omega \omega \Lambda^2)$ as $L_i \subseteq \mathcal{P}(F_i) \times \mathcal{P}(F_i) \times \{1, ..., \Lambda\} \times \{1, ..., \omega\Lambda\}$, and $|\mathcal{P}(F_i)| \times |\mathcal{P}(F_i)| = 2^{2\omega} = 4^\omega$.

Furthermore, on each timestep i we only consider defending vertices in $V(E_i)$, and $|V(E_i)| \leq \omega$. Thus for each timestep there are at most 2^ω defences to consider.

As described, for each timestep i in the lifetime Λ of the graph, it is necessary to compute every possible set of defences for every entry in L_i, and then to check if these defences are valid and compute the resulting state after these defences are made. Checking if a set of defences is valid can be achieved in $O(\omega)$ time by checking if every vertex to be defended is unburnt. Computing the resulting state can then be achieved in $O(\omega^2)$ time by checking if each vertex in F_i is adjacent to a burning vertex. The overall complexity is therefore $O(4^\omega \omega \Lambda^2 \times 2^\omega \times \omega \times \omega^2 \times \Lambda) = O(8^\omega \omega^4 \Lambda^3)$ as required. □

3.3 Hardness for Edge-Interval-Membership-Width

Motivated by the existence of a fixed-parameter-tractable algorithm for Temporal Firefighter when parameterised by vertex-interval-membership-width, we now answer whether a fixed-parameter-tractable algorithm for Temporal Firefighter exists when parameterised by edge-interval-membership-width. When the vertex-interval-membership-width is small, the number of possible useful defences per timestep is also small, which gives us a fixed-parameter-tractable algorithm.

Despite the similarity between the two parameters, the same is not true of edge-interval-membership-width. In fact, we show that Temporal Firefighter is **NP**-Complete even when the edge-interval-membership-width is at most 2. We show this by reduction from the Max-2-SAT variant of the classic SAT problem.

Satisfiability problems ask whether there is a truth assignment to the variables of a boolean formula that satisfy a certain requirement. Formulas are usually given in conjunctive normal form (abbreviated to CNF), where the formula is a conjunction of clauses: disjunctions of literals.

Max-2-Sat asks us to determine if a given number of clauses can be satisfied in a CNF formula, in which each clause contains two literals. This problem was shown to be **NP**-Complete by Garey et al. [41].

Max-2-SAT
Input: An integer k, and a pair (B, C) where B is a set of boolean variables, and C is a set of clauses over B in CNF, each containing 2 variables.
Output: Is there a truth assignment to the variables such that at least k clauses in C are satisfied?

Theorem 18 (Garey et al. [41])**.** *Max-2-SAT is **NP**-Complete.*

We are now ready to give our reduction. Given a CNF formula in which each clause has 2 literals, we produce a temporal graph in which the underlying graph is a tree of depth 2, and each edge is active exactly once, and at most two edges are active on every timestep. The fire begins at a root vertex r, and every vertex adjacent to the root corresponds to a literal from the formula. We attach leaves to these literal vertices and assign times to their incident edges such that the firefighters are forced to defend exactly one of each pair of literal vertices corresponding to a variable. Such a defence then corresponds to a truth assignment for the variables in the formula. We construct our instance such that a defence saves the desired number of vertices in Temporal Firefighter if and only if the corresponding truth assignment satisfies the desired number of clauses.

Theorem 19. *Temporal Firefighter is NP-Complete even when restricted to the class of temporal trees with each edge active exactly once, and at most two edges active per timestep.*

Proof. We reduce from Max-2-SAT. Given an instance $((B, C), k)$ of Max-2-SAT we construct an instance $(((G, \lambda), r), k')$ of Temporal Firefighter where G is a tree, each edge is active exactly once, and there are at most two edges active per timestep, such that $(((G, \lambda), r), k')$ is a yes-instance if and only if $((B, C), k)$ is also a yes-instance.

Let $v = |B|$, the number of variables, and $w = |C|$, the number of clauses. Our vertex $V(G)$ set consists of $1 + 2v + 2wv + 4w$ vertices:

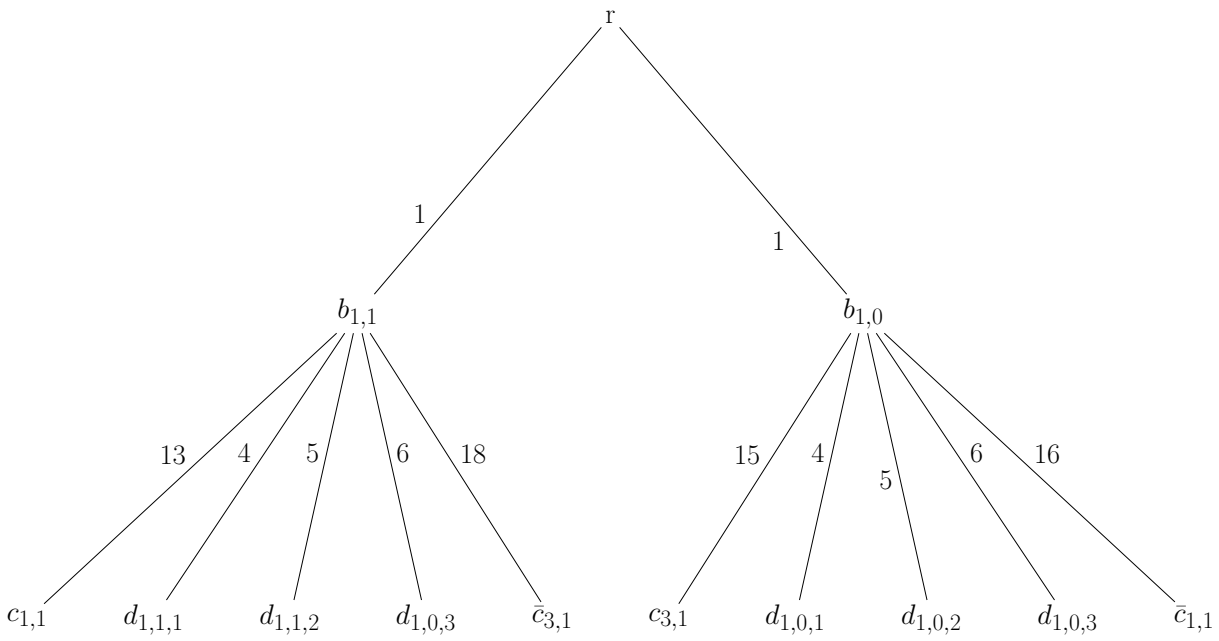

Figure 3.1: The section of the tree corresponding to the appearances of variable b_1 in the MAX-2-SAT instance $(b_1 \vee b_2) \wedge (\neg b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_3)$

- one root vertex r,
- $2v$ variable vertices $\{b_{i,x} : i \in [v], x \in \{1, 0\}\}$,
- $2wv$ forcing leaf vertices $\{d_{i,x,j} : i \in [v], x \in \{1, 0\}, j \in [w]\}$,
- $4w$ clause leaves, two for each appearance of a literal in a clause, $\{c_{j,i}, \bar{c}_{j,i} : i \in [v], j \in [w], b_i$ appears in clause $c_j\}$.

Our set of time edges then connects every variable vertex to the root, and every forcing and clause leaf to a variable vertex:

$$\begin{aligned}\{(e,t) : e \in E(\mathcal{G}), t \in \lambda(e)\} = &\{(\{b_{i,x}, r\}, i) : i \in [v], x \in \{1,0\}\} \\ &\cup \{(\{d_{i,x,j}, b_{i,x}\}, v + (i-1)w + j) : i \in [v], x \in \{1,0\}, j \in [w]\} \\ &\cup \{(\{c_{j,i}, b_{i,1}\}, v + wv + j), (\{\bar{c}_{j,i}, b_{i,0}\}, v + wv + w + j) \\ &: i \in [v], j \in [w], b_i \text{ occurs positively in } c_j\} \\ &\cup \{(\{c_{j,i}, b_{i,0}\}, v + wv + j), (\{\bar{c}_{j,i}, b_{i,1}\}, v + wv + w + j) \\ &: i \in [v], j \in [w], b_i \text{ occurs negatively in } c_j\}\end{aligned}$$

Now, set $k' = (1 + 2v + 2wv + 4w) - (1 + v + (w - k)) = v + 2wv + 3w + k$, and this along with the above temporal graph is our instance $(((G, \lambda), r), k')$. As required, G is a tree, each edge is active on exactly one timestep, and there are two edges active on every timestep. For any timestep between 1 and v inclusive, both of these edges are between

the root and a variable vertex. For any timestep between $v+1$ and $v+vw$ inclusive, these edges are between a variable vertex and a forcing leaf. For any timestep between $v+vw+1$ and $v+vw+w$ inclusive these edges are between a variable vertex and a positive clause vertex. Finally, for any timestep between $v+wv+w+1$ and $v+wv+2w$ inclusive these edges are between a variable vertex and a negative clause vertex. An example construction of such a graph can be seen in Fig. 3.1.

Now assume that $((B,C),k)$ is a yes-instance, that is that there is a truth assignment $\phi : B \to \{T,F\}$ to the variables in B such that at least k of the clauses in C are satisfied. Given this truth assignment we then define a strategy σ, and show it to save $k' = v(2w+1)+3w+k$ vertices on $((G,\lambda),r)$, thus demonstrating that $(((G,\lambda),r),k')$ is also a yes-instance. This strategy defends as follows:

Definition 21 (Strategy σ)**.**

- *For each timestep $t \in [v]$, σ, if $\phi(b_t) = T$ then σ defends $b_{t,1}$, and if $\phi(b_t) = F$ then σ defends $b_{t,0}$,*
- *for each timestep $t \in [v+1, v+vw]$, σ defends $d_{\lceil \frac{t-v}{w} \rceil,0,((t-v-1) \mod w)+1}$ if $\phi(b_{\lceil \frac{t-v}{w} \rceil}) = T$, and $d_{\lceil \frac{t-v}{w} \rceil,1,((t-v-1) \mod w)+1}$ if $\phi(b_{\lceil \frac{t-v}{w} \rceil}) = F$,*
- *for each timestep $t \in [v+wv+1, v+wv+w]$, σ defends any clause leaf in $\{c_{t-(v+wv),i} : b_i \text{ occurs in } c_{t-(v+wv)}\}$ that has an undefended parent. If neither of these two clause leaves have an undefended parent, then σ defends a clause leaf in $\{\bar{c}_{t-(v+wv),i} : b_i \text{ occurs in } c_{t-(v+wv)}\}$,*
- *finally, for each timestep $t \in [v+wv+w+1, v+wv+2w]$, σ defends any clause leaf in $\{\bar{c}_{t-(v+wv+w),i} : b_i \text{ occurs in } c_{t-(v+wv+w)}\}$ that has an undefended parent. If neither of these two leaves have an undefended parent, then σ defends one of them arbitrarily.*

Now consider the number of vertices that burn under σ. To begin with the root and half of the variable vertices burn, before all of the forcing leaves are saved. Now consider some clause $c_j \in C$ containing variables b_x and b_y. If c_j is satisfied, then neither clause leaf $c_{j,x}$ or $c_{j,y}$ burns, as at least one of these leaves will have a defended parent, and if either leaf does not have a defended parent, the leaf will be defended on timestep $v+wv+j$. If c_j is not satisfied, then neither of $c_{j,x}$ and $c_{j,y}$ will have a defended parent, and one of them will burn, and the other will be defended on timestep $v+wv+j$.

Finally consider a pair of negative clause leaves $\bar{c}_{j,x}$ and $\bar{c}_{j,y}$. If the parents of both of these leaves burn, the neither of the parents of the corresponding leaves $c_{j,x}$ and $c_{j,y}$ burn, and one of $\bar{c}_{j,x}$ and $\bar{c}_{j,y}$ will be defended on timestep $v+wv+j$, and the other on timestep $v+wv+w+j$. If one or fewer of the parents burn, then either of the leaves with a burning parent will be defended on timestep $v+wv+w+j$. Therefore no negative clause leaf

$\bar{c}_{j,i}$ will burn. Thus in total the root, half of the variable vertices, and one clause leaf per unsatisfied clause burn, that being at most $1 + v + (w - k)$ vertices. This means that at least $(1 + 2v + 2wv + 4w) - (1 + v + (w - k)) = v + 2wv + 3w + k$ vertices are saved as required.

We now show that if $(((G, \lambda), r), k')$ is a yes-instance, that is that there is some strategy σ that saves $v + 2wv + 3w + k$ vertices then $((B, C), k)$ is also a yes-instance. We begin by showing that if there exists a strategy that saves k' vertices, then there exists a strategy that on every timestep $t \in [v]$ defends one of the vertices $b_{t,0}$ and $b_{t,1}$, and also saves k' vertices. Given a strategy σ with this property, we then define a truth assignment ϕ, such that $\phi(b_t) = T$ if σ defends $b_{t,1}$ on timestep t, and $\phi(b_t) = F$ if σ defends $b_{t,0}$ on timestep t.

First assume that there exists a strategy that saves k' vertices, but no strategy that does so by defending only variable vertices on every timestep $t \in [v]$. Now let σ be a strategy that saves k' vertices and is maximal in the number of timesteps $t \in [v]$ on which a variable vertex is defended. Let t be the earliest timestep on which σ defends a leaf vertex l, and let $\{b_{i,0}, b_{i,1}\}$ be a pair of variable vertices undefended by σ, noting that such a pair must exist - if σ defends at least one vertex from every pair of variable vertices, then it must do so by timestep v at the latest, by which time every variable vertex burns. Furthermore, if it defends at least one vertex from every pair of variable vertices, then this requires at least v defences, and so σ would only defend variable vertices on every timestep $t \in [v]$, contradicting its definition. Consider now the strategy σ' that defends $b_{i,0}$ on timestep t, and makes the same defences as σ otherwise. See that σ' saves at least all the vertices saved by σ, with the possible exception of l, and also saves $b_{i,0}$, which was not saved by σ. Therefore σ' also saves at least k' vertices and contradicts the maximality of σ, and so there always exists a strategy that only defends variable vertices for every timestep $t \in [v]$.

We now show by induction on the variable index i that any strategy σ that saves k' vertices and defends only variable vertices on the first v timesteps must defend exactly one of every pair of vertices $b_{i,0}$ and $b_{i,1}$ on timestep i. When $i = 1$ if neither of $b_{i,0}$ and $b_{i,1}$ are defended both of these vertices will burn. There are then $2w$ forcing leaf vertices adjacent to these variable vertices, and all of these forcing leaf vertices will burn by timestep $v + w$. Our strategy only defends variable vertices for the first v timesteps, so can then only defend at most w of the forcing leaf vertices by timestep $v + w$, meaning at least w forcing vertices burn. Even assuming the remaining vertices in the graph are saved, the root, v of the variable vertices, and w forcing vertices burn, meaning that the number of saved vertices is $(1 + 2v + 2wv + 4w) - (1 + v + w) = v + 2wv + 3w < v + 2wv + 3w + k$, contradicting the assumption that σ saves k' vertices. Therefore σ must defend exactly one of $b_{1,0}$ and $b_{1,1}$ on timestep 1. Otherwise, if $i > 1$ then by the inductive assumption

σ defends one of each pair of vertices $b_{t,0}$ and $b_{t,1}$ on every timestep $t < i$. Assume that σ does not defend either of $b_{i,0}$ or $b_{i,1}$ on timestep i, so both of these vertices will burn. There are then $iw+w$ forcing leaf vertices adjacent to the burning variable vertices $b_{t,0}, b_{t,1}$ with $t \leq i$, and all of these forcing leaf vertices will burn by timestep $v+iw$. Our strategy only defends variable vertices for the first v timesteps, so can then only defend at most iw of the forcing leaf vertices by timestep $v + iw$, meaning that at least w forcing leaves burn. Even assuming the remaining vertices in the graph are saved, the root v of the variable vertices, and w forcing vertices burn, meaning that the number of saved vertices is $(1 + 2v + 2wv + 4w) - (1 + v + w) = v + 2wv + 3w < v + 2wv + 3w + k$, contradicting the assumption that σ saves k' vertices. Therefore σ must defend one of $b_{i,0}$ and $b_{i,1}$ on timestep i.

Therefore, if there exists a strategy that saves k' vertices, there must exist a strategy that only defends variable vertices during the first v timesteps, and this strategy must defend exactly one of each pair of variable vertices $b_{i,0}$ and $b_{i,1}$. Thus given such a strategy σ we then define a truth assignment ϕ, such that $\phi(b_i) = T$ if σ defends $b_{i,1}$ on timestep i, and $\phi(b_i) = F$ if σ defends $b_{i,0}$ on timestep i.

Now assume that there exists a strategy that defends one of the vertices $b_{i,0}$ or $b_{i,1}$ on each timestep $i \leq v$, and saves at least k' vertices, but no strategy that saves at least k' vertices, defends either $b_{i,0}$ or $b_{i,1}$ on each timestep $i \leq v$, and saves every forcing leaf. Let σ be a strategy that saves at least k' vertices, defends one of the vertices $b_{i,0}$ or $b_{i,1}$ on each timestep $i \leq v$, and is maximal in the integer ℓ such that every defence made by σ on a timestep $v < t \leq v + \ell$ is made at a forcing leaf with a burning parent and an active incident edge active on timestep t. Consider the strategy σ' which defends as σ but on timestep $v + \ell + 1$ defends a forcing leaf with an incident edge active on timestep $v + \ell$. Note that such a leaf must exist, as for every timestep $v < t \leq v + vw$, there are two forcing leaves with incident edges active on t, and one of these leaves is the child of a variable vertex $b_{i,0}$, and the other the child of $b_{i,1}$, only one of which will be defended by σ. Also see that such a leaf cannot have burnt at the start of timestep $v + \ell$, as its only incident edge is only active on this timestep. See then that any leaf that does not burn and is not defended when σ is played also does not burn when σ' is played, as σ' defends the same non-leaf vertices as σ. Furthermore, the number of leaves that are defended is the same when σ is played is the same as the number of leaves that are defended when σ' is played, and therefore σ' saves the same number of vertices as σ. This contradicts the maximality of σ, and therefore if there exists a strategy that saves k' vertices, there exists a strategy that saves k' vertices, defends one of the vertices $b_{i,0}$ or $b_{i,1}$ on each timestep $i \leq v$, and saves every forcing leaf.

When such a strategy σ is played, vw forcing leaves will have burning parents, and each of these leaves will burn by timestep $v + vw$ if undefended, meaning that on every

timestep $v < t \leq v + vw$, σ will defend a forcing leaf. Furthermore, σ saves v variable vertices, $2vw$ forcing leaves, and must therefore save at least $3w + k$ clause leaves, as σ saves $k' = v + 2wv + 3w + k$ vertices. There are $2w$ negative clause leaves, and $2w$ positive clause leaves, meaning that at least $w + k$ positive clause leaves must be saved by σ. Every undefended positive clause leaf with a burning parent will burn by timestep $v + vw + w$, and σ can only defend clause leaves from timestep $v + vw + 1$ onwards. Therefore at most w positive clause leaves can be saved by being defended. The remaining required k positive clause leaves must therefore have parents defended by σ. If the parent $b_{i,x}$ of any positive clause leaf $c_{j,i}$ is defended, then clause c_j must be satisfied by our truth assignment. Therefore at least k clauses are satisfied by the truth assignment corresponding to σ as required. □

3.4 Conclusions

In this chapter, we considered the tractability of Temporal Firefighter when parameterising by vertex-interval-membership-width and edge-interval-membership-width. Bounding either of these parameters bounds the activity of the graph, the former bounds the number of vertices that can be in their active interval on any given timestep, and the latter the number of edges.

For vertex-interval-membership-width we provide a dynamic programming algorithm for Temporal Firefighter Reserve that is fixed-parameter-tractable. Temporal Firefighter Reserve is a variant of Temporal Firefighter that allows a budget of defences to be built up so that multiple defences can be made in a single turn. Using this problem allows us to consider only strategies that always defend adjacent to the fire, and it is equivalent to Firefighter in terms of the number of vertices that can be saved.

We then showed that Temporal Firefighter is **NP**-Complete even on a tree of depth 2, with every vertex active on exactly 1 timestep, and at most 2 vertices active per timestep. Such a tree has edge-interval-membership-width 2, so Temporal Firefighter cannot be in **FPT** when parameterised by edge-interval-membership-width. However, the tree does have unbounded degree, and as future work it may be worth to consider parameterising by both the edge-interval-membership-width and the maximum degree.

Both of the parameters we have considered so far are small only when a temporal graph is in some way sparse, or limited in activity. In Chapter 4 we consider parameters that may be small even on dense temporal graphs. Finally, in Chapter 5 we return our attention to vertex-interval-membership-width and edge-interval-membership-width, and define classes of problems for which these parameters produce tractable algorithms.

Chapter 4

Parameterised Complexity on Dense Temporal Graphs

In Chapter 2 we presented an efficient algorithm Temporal Firefighter when the underlying graph belongs to a particular sparse graph class. Then in Chapter 3 we found that Temporal Firefighter is in **FPT** with respect to the parameter of vertex-interval-membership-width, a parameter that is small when the temporal graph is temporally sparse, with few vertices relevant on any given timestep. In this chapter we determine the complexity of Temporal Firefighter and Temporal Graph Burning when parameterising by temporal neighbourhood diversity, and also determine the complexity of Temporal Graph Burning when parameterising by the related parameter of temporal modular width. Both of these parameters can be small on dense temporal graphs, and are temporal analogues of the known static parameters of neighbourhood diversity and modular width. The definitions of these temporal parameters were first introduced in joint work with Enright, Larios-Jones, and Meeks [29].

We begin by recalling the definitions for neighbourhood diversity and modular width, before stating the definitions for temporal neighbourhood diversity and temporal modular width, and seeing that the temporal neighbourhood diversity of a temporal graph is always at least the temporal modular width, that is a graph with a small temporal neighbourhood diversity will have a small temporal modular width, but not vice-versa. We then give algorithms for both Temporal Firefighter and Temporal Graph Burning when parameterised by the temporal neighbourhood diversity. Finally, we show that Temporal Graph Burning remains **NP**-Complete when the temporal modular width is bounded.

4.1 Introducing the Parameters

The neighbourhood diversity of a static graph measures the number of classes of vertices where any pair of vertices in the same class are indistinguishable - they are both adjacent

to exactly the same vertices excluding each other. We say that such a pair of vertices has the same type. Lampis originally introduced this parameter specifically for coloured graphs [59], and it was then generalised by Ganian to uncoloured graphs [40]. We now recall the formal definitions of type due to Ganian, and neighbourhood diversity due to Lampis.

Definition 22 (Type (Ganian [40]))**.** *Two vertices v and v' have the same type if and only if $N(v) \setminus \{v'\} = N(v') \setminus \{v\}$.*

Definition 23 (Neighourhood Diversity (Lampis [59]))**.** *A graph G has neighbourhood diversity at most k if and only if there exists a partition of $V(G)$ into at most k sets where all vertices in each set have the same type.*

The definition of type neither requires or disallows two adjacent vertices to be of the same type. It is worth noting that in any set where all vertices have the same type, either all vertices in the set are adjacent to each other (the set is a clique), or no vertices in the set are adjacent to each other (the set is an independent set).

Modular width, like neighbourhood diversity, measures the number of sets produced when we group the vertices of a graph according to their neighbourhoods. Unlike neighbourhood diversity, this grouping is not a partition, and is in fact a recursive decomposition of the graph. In particular the well known notion of modular decomposition is used, which recursively decomposes a graph into sets of vertices known as modules. A module of a graph is a set of vertices X such that any two vertices within X have the same set of neighbours outside X. Every modular decomposition has an associated decomposition tree, where every vertex is a module, and the children of a vertex its sub-modules.

The modular width measures the maximum degree of this modular decomposition tree, and was first briefly considered as a parameter by Courcelle et al. [25], and then explored in further detail by Gajarskỳ et al. [39], from whom we obtain our definition given below.

Definition 24 (Modular Width (Gajarskỳ et al. [39]))**.** *Suppose the graph G can be constructed by an algebraic expression A that uses only the following operations.*

1. $\odot$ *- Creates an isolated vertex.*
2. $\oplus$ *- Takes the disjoint union of two graphs. $G_1 \oplus G_2$ is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_2) \cup E(G_2)$.*
3. $\otimes$ *- Takes the complete join of two graphs. $G_1 \otimes G_2$ is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2) \cup \{\{v, w\} : v \in V(G_1)$ and $w \in V(G_2)\}$.*
4. *$G(G_1, ..., G_n)$ - Substitutes the vertices $v_1, ...v_n$ of G with the graphs $G_1, ...G_n$. $G(G_1, ..., G_n)$ is the graph with vertex set $\bigcup_{1 \leq i \leq n} V(G_i)$ and edge set $\bigcup_{1 \leq i \leq n} E(G_i) \cup \{\{u, w\} : \{v_i, v_j\} \in E(G)$ and $u \in V(G_i)$ and $w \in V(G_j)\}$.*

The width of A is the maximum number of operands in an occurrence of operator $\downarrow$ in A. The modular width of G is then the minimum width of any expression A that constructs G.

Temporal neighbourhood diversity and temporal modular width are both defined analogously to their static counterparts, except we consider the temporal neighbourhood of the vertices, rather than the static neighbourhood. Two vertices share the same temporal neighbourhood if they are adjacent to the same vertices on every timestep.

Definition 25 (Temporal Neighourhood)**.** *The temporal neighbourhood of a vertex v in a temporal graph $\mathcal{G} = (G, \lambda)$ is the set $TN(v)$ of vertex time pairs (w, t) where $(w, t) \in TN(v)$ if and only if $\{v, w\} \in E(G)$, and $t \in \lambda(\{v, w\})$.*

We then define the temporal type and temporal neighbourhood diversity according to this definition of neighbourhood. The following definitions first appeared in joint work with Enright, Larios-Jones, and Meeks [29].

Definition 26 (Temporal Type (Enright et al. [29]))**.** *Two vertices u, v have the same temporal type if and only if $\{(w, t) \in TN(v) : w \neq u\} = \{(w, t) \in TN(u) : w \neq v\}$.*

Definition 27 (Temporal Neighourhood Diversity (Enright et al. [29]))**.** *A temporal graph $\mathcal{G}$ has temporal neighbourhood diversity at most k if and only if there exists a partition $\mathbf{X}$ of $V(\mathcal{G})$ into at most k sets where every vertex in each set has the same temporal type.*

We refer to the partition $\mathbf{X}$ as the temporal neighbourhood partition, and will use $[v]_{\mathbf{X}}$ to refer to the class in $\mathbf{X}$ to which v belongs.

Similar definitions are given by Bui-Xuan et al. [19] who refer to two vertices of the same type as "eternal twins", and a class in the temporal neighbourhood partition as an eternal module. They also consider a variant of these concepts referred to as a Δ-module where vertices are only required to share a neighbourhood for a certain time window of length Δ.

We now see that as every pair of vertices in a class must have identical temporal neighbourhoods excluding each other, every class must either form an independent set or a clique on each timestep. If two vertices u and v in the same class are adjacent on a timestep t, then all other vertices in the class must both be adjacent to u and v on timestep t. If this were not the case, there would exist vertices in the class with different types to u and v.

Lemma 20 (Enright et al. [29])**.** *At any snapshot $\mathcal{G}_t$ of $\mathcal{G}$, the subgraph induced by the vertices in a class $\mathbf{X}$ of a temporal neighbourhood partition of $\mathcal{G}$ either forms an independent set or a clique.*

We see that we can compute the temporal neighbourhood partition, and therefore the temporal neighbourhood diversity, in polynomial time by checking every pair of vertices to see if they have the same temporal type.

Lemma 21 (Enright et al. [29])**.** *The temporal neighbourhood diversity of a temporal graph $\mathcal{G}$ can be computed in $O(\Lambda n^3)$ time, where n is the number of vertices in the graph, and Λ the lifetime.*

We then define the temporal modular width using an analogous substitution operator to that in Definition 24. Using this operator to substitute graphs $\mathcal{G}_1, ..., \mathcal{G}_n$ into $\mathcal{G}$ joins each pair of graphs $\mathcal{G}_i, \mathcal{G}_j$ with edges active at all the times on the edge between v_i and v_j in $\mathcal{G}$. Furthermore, we note that the complete join and disjoint union operators from Definition 24 can be expressed using substitutions into a clique and independent set respectively, and as such we omit these operators.

Definition 28 (Temporal Modular Width (Enright et al. [29]))**.** *Suppose a temporal graph $\mathcal{G}$ can be constructed by an algebraic expression A using only the following operations.*

- $\odot$ *- Creates an isolated vertex.*
- $\mathcal{G}(\mathcal{G}_1, ..., \mathcal{G}_n)$ *- Substitutes the vertices $v_1, ...v_n$ of $\mathcal{G}$ with the temporal graphs $\mathcal{G}_1, ...\mathcal{G}_n$. $\mathcal{G}(\mathcal{G}_1, ..., \mathcal{G}_n)$ is the graph with vertex set $\bigcup_{1\leq i\leq n} V(\mathcal{G}_i)$ and time-edge set $\bigcup_{1\leq i\leq n} \varepsilon(\mathcal{G}_i) \cup \{(\{u, w\}, t) : (\{v_i, v_j\}, t) \in \varepsilon(\mathcal{G})$ and $u \in V(\mathcal{G}_i)$ and $w \in V(\mathcal{G}_j)\}$.*

The width of A is the maximum number of operands in an occurrence of operator 2 in A. The temporal modular width of $\mathcal{G}$ is then the minimum width of any expression A that constructs $\mathcal{G}$.

We now show that the unique decomposition of the graph into maximally sized temporal modules, and therefore the width, can be computed in polynomial time. Habib and Paul [42] describe a simple algorithm for finding the modular decomposition of a static graph. This operates by finding and repeatedly adding *splitters* to a candidate module. We use a similar method to find the temporal modular decomposition. We begin by recalling the definition of splitters, and use this to give a definition for temporal modules.

Definition 29 (Splitter (Habib and Paul [42]))**.** *Given a set of vertices S, a vertex x is said to be a* splitter *for S if there exist vertices $u, v \in S$ such that x is adjacent to exactly one of u and v.*

Definition 30 (Static Module (Habib and Paul [42]))**.** *A set of vertices M is a* module *of the static graph G, if and only if for every vertex $x \notin M$, x is not a splitter for M.*

Definition 31 (Temporal Module)**.** *A set of vertices M is a* temporal module *of (G, λ) if and only if for every timestep $t \in [\Lambda]$ and vertex $x \notin M$, x is not a splitter for M in the snapshot graph $(V(G), \{e \in E(G) : t \in \lambda(e)\})$.*

We now show that every temporal module of $\mathcal{G}$ is a module of every static snapshot graph of $\mathcal{G}$, and conversely, that any module of a snapshot graph is a temporal module of $\mathcal{G}$.

Lemma 22. *M is a temporal module of (G, λ) if and only if for every timestep $t \in [\Lambda]$, M is a module of the snapshot graph $(V(G), \{e \in E(G) : t \in \lambda(e)\})$.*

Proof. If M is a temporal module of (G, λ), then for any timestep t there are no splitters of M in $(V(G), \{e \in E(G) : t \in \lambda(e)\})$, and thus M is a module of $(V(G), \{e \in E(G) : t \in \lambda(e)\})$.

Conversely, if for every timestep t, M is a module of $(V(G), \{e \in E(G) : t \in \lambda(e)\})$, it must have no splitters in $(V(G), \{e \in E(G) : t \in \lambda(e)\})$, and thus is a temporal module of (G, λ). □

We further demonstrate that, as in the static case, the set of maximal modules for a temporal graph is unique.

Lemma 23. *The set of maximal temporal modules $\mathcal{M}$ for the temporal graph (G, λ) is unique and partitions $V(G)$.*

Proof. Assume that such a set does not partition $V(G)$, that is $\bigcup \mathcal{M} = V(G)$ but there exists $M_i, M_j \in \mathcal{M}$ such that $M_i \cap M_j \neq \emptyset$. By the previous lemma, we have that M_i and M_j are modules in every snapshot of (G, λ), and hence $M_i \cup M_j$ is also a module in every snapshot of (G, λ) and is therefore a temporal module, but $M_i \cup M_j \supset M_i$, contradicting the maximality.

Now assume that such a set is not unique, that is there exists a set of maximal temporal modules $\mathcal{P}$ with $\mathcal{P} \neq \mathcal{M}$. Now let $P_i \in \mathcal{P}$ be a module such that $P_i \notin \mathcal{M}$, and consider a vertex $v \in P_i$. As $\mathcal{M}$ partitions $V(G)$, there exists some module $M_j \in \mathcal{M}$ such that $v \in M_j$. Then $P_i \cup M_j$ is a module in every snapshot, and therefore a temporal module, but this contradicts the maximality of $\mathcal{M}$ and $\mathcal{P}$. □

A simple way to compute the unique maximal modular decomposition is then provided by the following observation.

Observation 7. *Let S be a subset of the vertices of a temporal graph (G, λ). If S has a splitter x in any snapshot of (G, λ), then any module of (G, λ) containing S also contains x.*

We now recursively compute the modular decomposition as follows. Given a set $\mathcal{M}$ of modules, repeatedly test if a non-trivial module containing a pair $M_1, M_2 \in \mathcal{P}$ exists, by considering $M_1 \cup M_2$ as a candidate module, and repeatedly adding any remaining modules in $\mathcal{M}$ containing a vertex that splits $M_1 \cup M_2$. If a trivial module is obtained the pair is discounted, and otherwise the set of modules is updated with the newly found non-trivial module. We find the maximal modular decomposition by initialising the set of modules to all the singleton trivial modules, that is $\{\{v\} : v \in V(G)\}$.

Theorem 24. *We can find the maximal temporal modular decomposition in time $O(n^4\Lambda)$, where n is the number of vertices in the temporal graph.*

Proof. The above algorithm begins with a set of n modules, and repeatedly considers pairs from this set. After each iteration the size of the list will either be reduced by one, or a pair will be discounted, thus the process must terminate after $O(n^2)$ iterations. On each iteration each remaining module is checked to see if it splits the current candidate module on any timestep, which is possible in $O((n+m)\Lambda) = O(n^2\Lambda)$, giving an overall runtime of $O(n^4\Lambda)$. □

Finally, we observe that the temporal neighbourhood diversity bounds the temporal modular width. This result is again due to joint work with Enright et al., and follows from Lemma 20. Each class in the temporal neighbourhood decomposition is also a module, and each of these classes can be created by repeatedly substituting into a two vertex temporal graph, where the underlying graph is either a clique or independent set.

Lemma 25 (Enright et al. [29])**.** *The modular width of a temporal graph $\mathcal{G}$ is at most the temporal neighbourhood diversity of $\mathcal{G}$.*

In the remainder of the chapter we explore the tractability of both Temporal Firefighter and Temporal Graph Burning with respect to the two parameters of temporal neighbourhood diversity and temporal modular width. We begin by showing that Temporal Firefighter is in **FPT** with respect to temporal neighbourhood diversity.

4.2 Temporal Firefighter Parameterised by Temporal Neighbourhood Diversity

We now present a fixed parameter tractable algorithm for Temporal Firefighter when parameterised by the temporal neighbourhood diversity. This algorithm operates by determining a sequence of classes from the temporal neighbourhood partition in which to defend every vertex. From such a sequence it is possible to determine in which classes every undefended vertex will burn, as all vertices of a class from the temporal neighbourhood partition share the same temporal adjacencies, and we can therefore in some respects

view each class as a single vertex. In particular, we observe that in order for any undefended vertex that is temporally reachable from the root to be saved, it is necessary to defend every vertex in at least one class from the temporal neighbourhood partition. Our algorithm then determines if a strategy exists that defends according to the sequence of classes, and if so how many vertices such a strategy can save.

Given a rooted temporal graph $(\mathcal{G}, r)$ along with a sequence of defences S for Temporal Firefighter, we throughout denote the set of classes from the temporal neighbourhood partition $\mathbf{X}$ of $\mathcal{G}$ in which every vertex is defended by S with $\mathbf{C}_S = \{X_i \in \mathbf{X} \mid \forall v \in X_i \cdot v \in S\}$, and refer to these classes as *totally defended*. Furthermore, we denote the set of any other classes in which at least one vertex is defended by S with $\mathbf{I}_S = \{X_i \in \mathbf{X} : \exists v \in X_i \cdot v \in S \text{ and } X_i \notin \mathbf{C}_S\}$, we refer to the defences made by S in $\mathbf{I}_S$ as *individual* defences. Note that both of these definitions refer to a sequence of defences and not necessarily an entire strategy. Furthermore, throughout we consider a sequence of defences S to be a bijection between the first $|S|$ natural numbers and the vertices in the sequence, and use $S^{-1}(A)$ to denote the set of timesteps on which S defends the vertices in a set A.

We begin by formally proving the observation stated above, showing that it is necessary to totally defend some classes in order to save any undefended vertices temporally reachable from the root.

Lemma 26. *Let $(\mathcal{G}, r)$ be a rooted temporal graph with temporal neighbourhood decomposition $\mathbf{X}$, and let v be a vertex both undefended and unburnt on timestep $t > |S|$ after a sequence of defences S is played.*

Consider any temporal path P from r to v with earliest arrival time $t' \leq t$. There exists a vertex x on P such that $[x]_{\mathbf{X}} \in \mathbf{C}_S$.

Proof. Let $r, p_1, ..., p_i, ..., p_n, v$ be the vertices traversed by P, and assume there is no such vertex x.

Then for every p_i there exists a $p'_i \in [p_i]_{\mathbf{X}}$ undefended by S. Each of these vertices shares the same temporal adjacencies as the corresponding vertex from P. Therefore, there is a temporal path with the same earliest arrival time as P that traverses the undefended vertices $r, p'_1, ..., p'_i, ..., p'_n, v$. So, v would burn on timestep $t' \leq t$, and we have contradiction. □

We now further show that given a strategy S it is never harmful to add classes to $\mathbf{C}_S$. Doing so will always save at least all the undefended vertices previously saved by S.

Lemma 27. *Let $(\mathcal{G}, r)$ be a rooted temporal graph, and let S and S' both be valid sequences of defences with $\mathbf{C}_S \subseteq \mathbf{C}_{S'}$, and v a vertex not defended by S.*

If v is not burning immediately after S is played, then it is also not burning immediately after S' is played.

Proof. Assume that there is a counterexample, that is a rooted temporal graph $(\mathcal{G}, r)$, with sequences of defences S and S' such that and $\mathbf{C}_S \subseteq \mathbf{C}_{S'}$ such that a vertex v is not burning or defended immediately after S is played, but is burning immediately after S' is played.

As v burns when S' is played, there must exist a temporal path P from r to v with arrival time $t' < |S'|$ on which every vertex burns, and thus that contains no vertices defended by S'. By Lemma 26 when S is played P contains a vertex x with $[x]_{\mathbf{X}} \in \mathbf{C}_S$. However, $\mathbf{C}_S \subseteq \mathbf{C}_{S'}$ but $[x]_{\mathbf{X}} \notin \mathbf{C}_{S'}$ as x is not defended by S', and we have a contradiction. □

Corollary 4. *Let $(\mathcal{G}, r)$ be a rooted temporal graph, and let S and S' both be valid sequences of defences with $\mathbf{C}_S \subseteq \mathbf{C}_{S'}$, and v a vertex not defended by S.*

If v never burns, and is thus unreachable by the fire after S is played, it never burns and is thus unreachable by the fire after S' is played.

We now show that given a strategy S we can remove classes from $\mathbf{C}_S$, and any undefended vertex saved by S will not burn until the removed classes burn.

For any class $X \in \mathbf{X}$ let $b_S(X)$ be the earliest timestep t such that the vertices of X are adjacent to the fire at the start of timestep t when S is played. That is if $X \notin \mathbf{C}_S$, $b_S(X)$ is the timestep on which all undefended vertices of X burn.

Note that as a consequence of Lemma 27 the timestep on which any given undefended vertex burns is determined only by the classes totally defended by the strategy being played, and thus for any class X_i and pair of strategies S and S' with $\mathbf{C}_S = \mathbf{C}_{S'}$ we have that $b_S(X_i) = b_{S'}(X_i)$.

Lemma 28. *Let $(\mathcal{G}, r)$ be a rooted temporal graph, and let S and S' both be valid sequences of defences with v a vertex not defended by S. Let t be a timestep such that $t \leq b_{S'}(X)$ for every class $X \in \mathbf{C}_S \setminus \mathbf{C}_{S'}$.*

If v is not burning on timestep t when S is played, then it is also not burning on timestep t when S' is played.

Proof. Assume that there is a counterexample, that is a rooted temporal graph $(\mathcal{G}, r)$, and let S and S' be sequences of defences with $\mathbf{C}_{S'} \subseteq \mathbf{C}_S$ such that a vertex v is not burning or defended on timestep t after S is played, but is burning on timestep t after S' is played, where $t \leq b_{S'}(X)$ for every class $X \in \mathbf{C}_S \setminus \mathbf{C}_{S'}$.

As v burns when S' is played, there exists a temporal path P from r to v with arrival time $t' < t$ on which every vertex burns, and thus that contains no vertices defended by S'. By Lemma 26 when S is played P contains a vertex x with $[x]_{\mathbf{X}} \in \mathbf{C}_S$. However, $t < b_{S'}(X)$ for every class $X \in \mathbf{C}_S \setminus \mathbf{C}_{S'}$. If $[x]_{\mathbf{X}} \in \mathbf{C}_{S'}$ then we have contradiction, as x would be defended and thus not burn when S' is played. Otherwise, $[x]_{\mathbf{X}} \in \mathbf{C}_S \setminus \mathbf{C}_{S'}$, and thus $t \leq b_{S'}([x]_{\mathbf{X}})$ and again x cannot be burning on timestep t and we have contradiciton. □

We now show that for any strategy S, there exists a strategy S' that totally defends the same classes, saves at least the same number of vertices, and makes all defences within a totally defended class on a contiguous set of timesteps. If a strategy S has this property, that is for every class $X_i \in \mathbf{C}_{S'}$ we have that $\max(S^{-1}(X_i)) - \min(S^{-1}(X_i)) = |X_i|$, we refer to it as *well formed*. Furthermore, we say that the sequence of every class in $X \in \mathbf{C}_S$ ordered by $\min(S^{-1}(X))$ is the *total defence sequence* of S.

Lemma 29. *Let $((G, \lambda), r)$ be a rooted temporal graph, and let S be a strategy for Temporal Firefighter on this graph that saves at least k vertices.*

There exists a strategy S' that saves at least k vertices with $\mathbf{C}_{S'} = \mathbf{C}_S$, such that for every class $X_i \in \mathbf{C}_{S'}$ we have $\max(S^{-1}(X_i)) - \min(S^{-1}(X_i)) = |X_i|$.

Proof. Assume that there is a counterexample: a temporal graph $((G, \lambda), r)$ and a strategy S that saves k vertices, such that there is no strategy S' that saves at least k vertices with $\mathbf{C}_{S'} = \mathbf{C}_S$, and for every class $X_i \in \mathbf{C}_{S'}$ we have $\max(S^{-1}(X_i)) - \min(S^{-1}(X_i)) = |X_i|$. Furthermore, assume that S is maximal in the number of classes X_i for which $\max(S^{-1}(X_i)) - \min(S^{-1}(X_i)) = |X_i|$.

Let $X_i \in \mathbf{C}_S$ be a class such that $\max(S^{-1}(X_i)) - \min(S^{-1}(X_i)) > |X_i|$, and let $t_1 = \min(S^{-1}(X_i))$ and $t_2 = \max(S^{-1}(X_i))$.

Now let R be the subsequence of vertices defended by S between times t_1 and t_2 that are not in X_i.

Consider now the sequence S' that contains the same first $t_1 - 1$ defences as S, but then from timestep t_1 defends the sequence R, before defending all the vertices of X_i, and then defends as S again from timestep t_2 onwards.

We now show that S' is a valid strategy. As S is a valid strategy the initial $t_1 - 1$ defences of S' are valid.

Now consider a vertex v in R. Let t be the time at which S defends this vertex, and t' the time at which S' defends this vertex. See that $t' < t$ as S' defends the vertices of R in the same order as S, but begins defending them on timestep t_1, on which S defends a vertex of X_i. Let $S_{t'-1}$ and $S'_{t'-1}$ denote the sequences containing the first $t' - 1$ moves of S and S' respectively. See that $\mathbf{C}_{S_{t'}} \subseteq \mathbf{C}_{S'_{t'-1}}$ and that v is unburnt and undefended on timestep t' when S is played and is therefore by Lemma 27 also unburnt and undefended on timestep t' when the first $t' - 1$ moves of S' are played. Therefore the defence of v by S' on timestep t' is valid.

We now show that the defences S' makes in X_i are valid. As every undefended vertex of X_i will always burn on the same timestep, it suffices to show that the final defence S' makes in X_i is valid. This defence is made on timestep t_2, and we have that $\mathbf{C}_S^{t_2-1} = \mathbf{C}_{S'}^{t_2-1}$, and thus by Lemma 27 this defence is valid.

Finally consider a vertex v defended by S' on a timestep $t' > t_2$. S also defends v on this timestep and thus v is not burning on t' when S is played. Furthermore we have that

$\mathbf{C}^{t'}_{S'} = \mathbf{C}^{t'}_{S}$, and hence by Lemma 27 v is not burning on t' after the first $t'-1$ moves of S' are played, and thus its defence by S' is valid.

Then, as S and S' defend the exact same vertices, any vertex unreachable by the fire after S is played is also unreachable by the fire after S' is played. Furthermore, S' defends exactly the same vertices as $|S|$. Therefore, S' also saves at least k vertices, and we have contradiction. □

We now further show that each full defence of a class can be finished "as late as possible", meaning the defence of the class is completed on the last possible timestep before either the class burns, or there is not enough time to defend the remaining classes. We refer to such a strategy as *strongly well formed*.

Lemma 30. *Let (G, λ) be a temporal graph with root r and temporal neighbourhood decomposition $(X_i)_{i\in I}$, and let S be a well formed strategy that saves k vertices on this graph, with total defence sequence $Z = Z_1, ..., Z_\ell$.*

There then exists a well formed strategy S' also with total defence sequence Z that saves at least k vertices, such that if we define the function ld as follows:

$$ld(Z_i) = \begin{cases} b_{S'}(Z_i) & i = \ell \\ \min(b_{S'}(Z_i), ld(Z_{i+1}) - |Z_{i+1}|) & i \neq \ell \end{cases}$$

for all classes $Z_i \in Z$ we have that $\max(S'^{-1}(Z_i)) = ld(Z_i)$.

Proof. Assume that the temporal graph (G, λ) with root r along with a well formed strategy S that saves k vertices is a counterexample. Let $Z = Z_1, ..., Z_\ell$ be the total defence sequence of S, and assume that S is maximal in length of the postfix subsequence $Z_i, ..., Z_\ell$ of classes for which $\max(S'^{-1}(Z_i)) = \mathrm{ld}(Z_i)$.

Consider the class Z_{i-1}. It is the case that:

$$\max(S'^{-1}(Z_{i-1})) \leq \mathrm{ld}(Z_{i-1}).$$

This is as S would not be valid if $\max(S'^{-1}(Z_{i-1})) > b_S(Z_{i-1})$, and S would not be well formed with total defence sequence Z if $\max(S'^{-1}(Z_{i-1})) > \min(S'^{-1}(Z_i)) - 1$. Then $\min(S'^{-1}(Z_i)) - 1 = \max(S'^{-1}(Z_i)) - |Z_i| = \mathrm{ld}(Z_i) - |Z_i|$, by the assumption and the fact that S is well formed. Furthermore, as Z_{i-1} does not satisfy the expression in the lemma $\max(S'^{-1}(Z_{i-1})) \neq \mathrm{ld}(Z_{i-1})$ and therefore $\max(S'^{-1}(Z_{i-1})) < \mathrm{ld}(Z_{i-1})$.

Let R be the sequence of defences made by S between times $\max(S'^{-1}(Z_{i-1}))$ and $\mathrm{ld}(Z_{i-1})$.

Consider the strategy S' which defends as S, but then from timestep $\min(S^{-1}(Z_{i-1}))$ defends the entire sequence R, before defending all the vertices of Z_{i-1}, and then from timestep $\mathrm{ld}(Z_{i-1})$ onwards defending as S does again.

In S the defences of the vertices of R all occur strictly later than they do in S', and the set of totally defended classes is the exactly the classes in $Z_1, ..., Z_{i-1}$ when the defences are made in S, and exactly the classes $Z_1, ..., Z_{i-2}$ when the defences are made in S'. Then, as S' defends the vertices of R prior to $b_{S'}(Z_{i-1})$, by Lemma 28 these defences are valid.

Furthermore S' defends all the vertices of Z_{i-1} prior to $b_{S'}(Z_{i-1})$, and thus these defences are valid.

Then S' is a valid sequence of defences that defends the same vertices as S, and therefore also saves k vertices.

However in S' the sequence of classes $Z_{i-1}, ..., Z_n$ satisfies the lemma, contradicting the maximality of S, and thus the existence of a counter-example. □

Finally, we turn our attention to the defences made in non-totally defended classes. We show that if it is possible to save k vertices, then there always exists a strategy that does so and only non-totally defends one class, leaving the vertices in all other classes undefended.

Lemma 31. *Let $((G, \lambda), r)$ be a rooted temporal graph, and let S be a strategy for Temporal Firefighter on this graph that saves at least k vertices.*

There exists a strategy S' that saves at least k vertices with $\mathbf{C}_S \subseteq \mathbf{C}_{S'}$, such that $\mathbf{I}_{S'}$ contains only a single class.

Proof. Assume there is a counterexample, that is a rooted temporal graph $((G, \lambda), r)$ on which k vertices can be saved, but no strategy S for saving k vertices has $|\mathbf{I}_S| = 1$. Let S be a strategy such that $|\mathbf{I}_S|$ is minimised, noting that $|\mathbf{I}_S| \geq 2$ by the assumption that $((G, \lambda), r)$ is a counterexample.

We first show that the undefended vertices of every class in $\mathbf{I}_S$ must burn when S is played. If there existed a class $I \in \mathbf{I}_S$ for which the undefended vertices did not burn, then a shorter strategy that omits the defences in I, but ortherwise defends the same vertices as S in the same order would also save at least k vertices, contradicting the minimality of $|\mathbf{I}_S|$.

Let X_i be a class in $\mathbf{I}_S$ such that $b_S(X_i)$ is maximised, and d_i be the number of vertices of X_i that S defends. Then let T be the latest set of $\min(|\bigcup \mathbf{I}_S \setminus X_i|, |X_i| - d_i)$ timesteps on which S defends vertices in $\bigcup \mathbf{I}_S \setminus X_i$.

Now let S' be a strategy that defends exactly as S, except that for every timestep in T, S' defends a vertex in X_i that is not defended by S.

We now show that every defence S' makes of a vertex in X_i is valid. Let t be the final timestep on which S defends a vertex not in $\bigcup \mathbf{C}_S$. As $b_S(X_i)$ was a maximum, we have that no vertex of X_i is burning on timestep t when S is played. Then, for every timestep $t' \leq t$ on which S' defends a vertex of X_i we have that $\mathbf{C}_S^{t'} \subseteq \mathbf{C}_{S'}^{t'}$, and then by Lemma 27 the defence made by S' on timesetep t' is valid.

See that $\mathbf{C}_S \subseteq \mathbf{C}_{S'}$, and thus by Corollary 4 any vertices that never burn and are not defended when S is played, also never burn after S' is played, and are unreachable by the fire. Furthermore, see that S' defends the exact same number of vertices as S. Finally, no vertex defended by S' is unburnt and undefended when S is played, as the only vertices that S' defends that S does not are vertices of X_i, and the undefended vertices of this class burn when S is played. Therefore S' also saves at least k vertices.

See now that if $|\bigcup \mathbf{I}_S \setminus X_i| < |X_i| - d_i$ then $\mathbf{I}_{S'}$ contains only X_i, and we have contradiction.

Otherwise, if $|X_i| - d_i \leq |\bigcup \mathbf{I}_S \setminus X_i|$ then S' defends all vertices of X_i, and thus $\mathbf{C}_S \subset \mathbf{C}_{S'}$, and we again have a contradiction. □

We now present an algorithm for Temporal Firefighter, and show that this algorithm is in **FPT** when parameterising by temporal neighbourhood diversity.

Algorithm 2 Temporal Firefighter Parameterised By Temporal Neighbourhood Diversity

Input: A rooted temporal graph $((G, \lambda), r)$ and an integer k.
Output: Whether there exists a strategy that saves k vertices in Temporal Firefighter on $((G, \lambda), r)$.

```
Θ ←the temporal neighbourhood partition of (G, λ)
for all Sequences C of subsets of Θ do
    for all Classes I ∈ Θ and not in the sequence C do
        if The unique strongly well formed sequence S with C_S = C and I_S = {I} is a strategy that saves at least k vertices then
            return True
        end if
    end for
end for
return False
```

The correctness of Algorithm 2 follows from Lemma 29, Lemma 30, and Lemma 31. If there exists a strategy S that saves at least k vertices then we know from Lemma 31 that there exists a strategy S' that saves at least k vertices such that $\mathbf{I}_{S'}$ contains only a single class. Then by Lemma 29 and Lemma 30 there exists a strongly well formed strategy S'' that saves at least k vertices, maintaining the condition that $\mathbf{I}_{S''}$ contains only a single class.

Theorem 32. *We can solve Temporal Firefighter on a temporal graph (G, λ) in time $O(\Lambda n^3 \psi \psi!)$ where ψ is the temporal neighbourhood diversity, and n is the number of vertices in the graph.*

Proof. The algorithm described above solves Temporal Firefighter. It begins by computing the temporal neighbourhood decomposition, which we know from Lemma 21

that we can do in time $O(\Lambda n^3)$. Furthermore there are at most $\Sigma_{i=1}^{\psi}({}^{\psi}P_i) = O(\psi\psi!)$ possible sequences $\mathbf{C}$, and at most ψ sets I. We can then simulate temporal firefighter on a graph with any sequence of defences in time $O(n^2\Lambda)$ by checking every pair of vertices on each timestep, giving us an overall running time of $O(\Lambda n^3 + \psi\psi! n^2\Lambda) = O(\Lambda n^3 \psi\psi!)$. □

4.3 Temporal Graph Burning Parameterised by Temporal Neighbourhood Diversity

We now show that Temporal Graph Burning is solvable efficiently when the temporal neighbourhood diversity of the input graph is bounded. Throughout we assume that the lifetime Λ of the input temporal graph is at most the number of vertices n, as it is possible to burn any temporal graph in n timesteps by placing a fire at every vertex in turn.

We begin by defining notation for the burning set of vertices on a given timestep when a strategy is played.

Definition 32 (Burning Set)**.** *Given a strategy S the* burning set $B_t(S)$ *at timestep t is the set of vertices immediately after a fire is placed on timestep t when S is played.*

We now prove a lemma that shows that if for two strategies S and R and some timestep t_1 we have $B_{t_1}(S) \subseteq B_{t_1}(R)$, then we are able to use an initial segment from strategy R and find remaining moves from times $t_1 + 1$ onwards to obtain a strategy that will burn the graph at least as fast as S.

Thus throughout, in order to show that the existence of a strategy that burns the graph at least as fast as another, we need argue only about the existence of such a timestep t and the first t moves made by the strategies.

Lemma 33. *Let S be a successful strategy for (G, λ). Suppose that there is some timestep $t_1 < |S|$ and strategy R with $|R| = |S|$ such that $B_{t_1}(S) \subseteq B_{t_1}(R)$ and on every timestep after t_1, R places a fire at the same vertex as S. Then R is also successful.*

Proof. We prove by induction on $t_2 - t_1$ that for any timestep $t_2 \geq t_1$ we have that $B_{t_2}(S) \subseteq B_{t_2}(R)$.

Our base case is given, as R is defined such that $B_{t_1}(S) \subseteq B_{t_1}(R)$. We now assume that $B_t(S) \subseteq B_t(R)$, for some $t_1 \leq t < |S|$, and show that $B_{t+1}(S) \subseteq B_{t+1}(R)$.

Now let $v \in B_{t+1}(S)$. If $v \in B_t(S)$ then $v \in B_t(R)$ and therefore $v \in B_{t+1}(R)$. Otherwise if $v \notin B_t(S)$ then either the fire spreads to v on timestep $t+1$ when S is played, or S places a fire at v on timestep $t + 1$. In the former case v is adjacent to a vertex in u in $B_t(S)$ on timestep $t + 1$. It must be the case that u is also in $B_t(R)$, and therefore the fire will spread to v on $t + 1$ when R is played if v has not already burnt. In the latter

case either v is burning before R places a fire a on timestep $t+1$ when R is played, or $r_{t+1} = s_{h+1} = v$.

Therefore $B_{t+1}(S) \subseteq B_{t+1}(R)$. We then have that $V(G) = B_{|S|}(S) \subseteq B_{|S|}(R)$, and therefore R is successful. □

We now show that we can delay placing a fire at a vertex belonging to a class in which a fire is already burning, without causing a large effect on the set of vertices that will burn at each timestep.

Lemma 34. *Let (G, λ) be a temporal graph with temporal neighbourhood partition $(X_i)_{i \in I}$. Now let S be a strategy that burns this graph, and let u be a vertex at which S places a fire on a timestep t_1, and X_i be the class to which this vertex belongs. Fix a timestep $t_2 > t_1$, and let S' be a strategy which plays as follows until timestep t_2:*

$$s'_t = \begin{cases} s_t & \text{if } t < t_1 \\ s_{t+1} & \text{if } t_1 \leq t < t_2 \\ u & \text{if } t = t_2 \end{cases}$$

If there exists a vertex $w \in X_i$ which is burning before the end of timestep t_1 when S' is played, we have that $B_{t_2}(S) \subseteq B_{t_2}(S')$.

Proof. We will show that for any vertex $x \in B_{t_2}(S)$ we have that $x \in B_{t_2}(S')$.

Consider the case where x is a vertex at which S places a fire. If $x \neq u$ then S' will place a fire at x on the same timestep or earlier than S, and thus $x \in B_{t_2}(S)$. Otherwise, if $x = u$ then S' places a fire at it on timestep t_2, and again $x \in B_{t_2}(S)$.

Now, consider the case where S does not place a fire at x before timestep t_2. It must then burn because the fire spreads to it. Note, as $x \in B_{t_2}(S)$, there exists timesteps t_α and t_β with $t_\alpha < t_\beta$ such that there exists a temporal path P with arrival time t_β that traverses the vertices $y_1, ..., y_h$, $y_h = x$, and y_1 is a vertex at which S places a fire prior to t_α, and $t_\beta \leq t_2$.

If $y_1 \neq u$ then $x = y_h$ will still burn before the end of timestep $t_\beta \leq t_2$ when S' is played, as S' will also place a fire at y_1 prior to timestep t_α. Thus $x \in B_{t_2}(S')$.

Otherwise if $y_1 = u$ then u is adjacent to y_2 on timestep $t_\alpha > t_1$. As $w, u \in X_i$, w is also adjacent to y_2 on this timestep, and there is also a temporal path $P' = w, y_2, ..., y_h$ starting at time $t_\alpha > t_1$, and identical to P in all but the first vertex. Therefore in this case when S' is played $x = y_h$ will burn before the end of timestep t, as w burns by the end of timestep $t_1 < t_\alpha$ when S' is played. Thus, again, $x \in B_{t_2}(S')$. □

We go on to show that any time we place a fire at a vertex, we may instead place a fire at another unburnt vertex in the same class, if such a vertex exists, and obtain a strategy that burns the graph in the same time.

Lemma 35. *Let (G, λ) be a temporal graph with temporal neighbourhood partition $(X_i)_{i \in I}$.*

Let S and S' both be strategies with $|S'| = |S|$, such that on every timestep, S and S' both place fires in the same class, that is, for any $i \leq \min(|S|, |S'|)$ we have that there exists a class X_j where $\{s_i, s'_i\} \subseteq X_j$. Furthermore, suppose that S' places a fire at an already burning vertex on a timestep i if and only if S also places a fire at an already burning vertex on timestep i.

Then S is successful if and only if S' is.

Proof. We show that on each timestep t, the number of burning vertices in each class from the temporal neighbourhood partition is the same when S and S' are played. We proceed by induction on the timestep.

On the first timestep only s_1 is burning when S is played, and only s'_1 is burning when S' is played. By assumption these two vertices are both in the same class in the temporal neighbourhood partition.

Then, assume that on some timestep t the number of burning vertices in each class from the temporal neighbourhood partition is the same when S and S' are played. Now given any class X_i from the temporal neighbourhood partition, let b_i be the number of vertices burning in X_i at the end of timestep $t + 1$ when S is played. The number of vertices burning in X_i at the end of timestep $t + 1$ when S' is played is then the number of vertices that were burning on timestep t, plus the number of vertices to which the fire spreads, plus one if a fire was placed in X_i by S' on timestep $t + 1$. All of the vertices in X_i will be burning by the end of timestep $t + 1$ if the fire spreads to any vertex $v \in X_i$, as any burning vertex u adjacent to v on $t + 1$ is also adjacent to all other vertices in X_i. Furthermore, such a vertex $u \in X_j$ exists if and only if there is a burning vertex in X_j at the end of timestep t when S is played, as the number of vertices burning in X_j at the end of timestep t is the same when both S and S' are played. Then, all of the vertices of X_i are burning on timestep $t + 1$ when S is played if and only if all of the vertices of X_i are burning on timestep $t + 1$ when S' is played. Furthermore, S' places a fire in X_i if and only if S does also. Therefore, on timestep $t + 1$ the number of vertices burning when S is played is the same as the number of vertices burning when S' is played.

Then, by induction, if there exists a timestep t on which all the vertices are burning when S is played, the same number of vertices are burning when S' is played, so S is successful if and only if S' is.

□

Definition 33 (Placement Classes)**.** *The placement classes for a strategy S denoted $C(S)$ is the set of classes from the temporal neighbourhood partition in which S places fires.*

We now show that we can reorder any successful burning strategy S, so that initially one fire is placed in every *placement class* for S. This reordering gives a new strategy which is still successful.

Lemma 36. *Given a temporal graph (G, λ), let S be any successful strategy. There is then a successful strategy S' with $|S'| = |S|$, and $C(S') = C(S)$, such that the first $|C(S)|$ burns are in distinct equivalence classes in the temporal neighbourhood partition.*

Proof. Assume that (G, λ) is a counterexample. Then let R be a successful strategy minimal in the timestep t_c, such that at the end of timestep t_c there is a fire placed in every class in $C(S)$. Now, let $u \in X_i$ be a vertex at which S places a fire on timestep $t_1 \leq |C(S)|$, such that a fire has already been placed at a vertex $w \in X_i$ prior to t_1.

Consider the strategy R' which makes moves as follows:

$$r'_t = \begin{cases} r_t & \text{if } t < t_1 \text{ or } t > t_c \\ r_{t+1} & \text{if } t_1 \leq t < t_c \\ u & \text{if } t = t_c. \end{cases}$$

By Lemma 34, we have that $B_{t_c}(R) \subseteq B_{t_c}(R')$. Then, by Lemma 33, R' burns (G, λ) in the same or less time as R. This contradicts the assumption that R was minimal in the number of moves played until a fire is placed in every class in $C(R)$, so no such counterexample (G, λ) can exist. □

Finally we show that, given a strategy S that places fires only in distinct classes for the first $|C(S)|$ moves, we can arbitrarily reorder all subsequent moves made after timestep $|C(S)|$.

Lemma 37. *Let (G, λ) be a temporal graph, and S a successful strategy such that the first $|C(S)|$ fires placed by S are placed in distinct classes from the temporal neighbourhood partition. Let $f : [|C(S)| + 1, |S|] \to [|C(S)| + 1, |S|]$ be any bijection.*

Then the strategy S' given by

$$s'_t = \begin{cases} s_t & \textit{if } t \leq |C(S)| \\ s_{f(t)} & \textit{otherwise} \end{cases}$$

is successful, and burns the graph in the same or less time as S.

Proof. Let (G, λ), along with a strategy S and bijection f be a counterexample.

Then let R be a successful strategy with $|R| \leq |S|$, $C(R) = C(S)$, and $r_t = s_t$ for all $t \leq |C(S)|$. Furthermore, assume that R is the strategy minimal in the timestep t_2 such that, for every timestep $t \geq t_2$, $r_t = s_{f(t)}$. (Note that it is possible that $t_2 = |R| + 1$, and there is no terminal sub-sequence on which R agrees with the permutation of S.) Let t_1 be the timestep on which R places a fire at $s_{f(t_2-1)} = s'_{t_2-1}$.

Now let R' be the strategy that makes moves as follows:

$$r'_t = \begin{cases} r_t & \text{if } t < t_1 \text{ or } t \geq t_2 \\ r_{t+1} & \text{if } t_1 \leq t < t_2 - 1 \\ s_{f(t_2-1)} & \text{if } t = t_2 - 1. \end{cases}$$

Then, as R' places a fire at a vertex in every class in $C(S)$ prior to timestep t_1, by Lemma 34 we have that $B_{t_2-1}(R') \subseteq B_{t_2-1}(R)$. Then, by Lemma 33, R' burns (G, λ) in the same or less time as R. This contradicts the assumption that R was minimal in the timestep t_2, so no such counterexample (G, λ), strategy S, and bijection f can exist. □

We now present an algorithm for Temporal Graph Burning, and show that this algorithm is an fpt-algorithm with respect to temporal neighbourhood diversity.

Algorithm 3 TND Graph Burning Algorithm

Input: A temporal graph $\mathcal{G}$, and an integer h.
Output: True if there exists a successful burning strategy of length at most h, and false otherwise.

```
Compute the temporal neighbourhood partition Θ of (G, λ).
for all possible subsets A ⊆ Θ do
    for all possible orderings of A do
        for all possible subsets B ⊆ A do
            Let S be the strategy that first places a fire in order in every class from A, and then places fires at every unburnt vertex in B.
            if S is successful and consists of h or fewer moves then
                return true.
            end if
        end for
    end for
end for
return false.
```

We now prove correctness of this algorithm, using the following lemmas.

Lemma 38. *Algorithm 3 returns true for a temporal graph (G, λ) and integer h if and only if there exists a strategy S that burns the graph in h or fewer timesteps.*

Proof. Suppose there exists a strategy S that burns (G, λ) in h or fewer timesteps. By Lemma 36 and Lemma 35 we may assume without loss of generality that S first places a fire at an arbitrary vertex from every class in $C(S)$. The remaining moves must then place fires at every other vertex of any class to which the fire will not spread before the graph is burnt. These classes must be some subset of the classes from $C(S)$, and from Lemma 37 we know that these fires can be placed in any order.

The algorithm exhaustively checks every such strategy, and thus will return true if any strategy exists that burns (G, λ) in h or fewer moves, and false otherwise. □

This allows us to obtain fixed parameter tractability, as bounding the temporal neighbourhood diversity bounds the number of such strategies that we have to check.

Theorem 39. *Temporal Graph Burning is solvable in time $O(\Lambda n^3 \phi\phi! 2^\phi)$, where n is the number of vertices in $\mathcal{G}$, Λ the lifetime, and ϕ the temporal neighbourhood diversity. If the temporal neighbourhood partition is given, we obtain a runtime of $O(\Lambda n^2 \phi\phi! 2^\phi)$.*

Proof. The TND Graph Burning Algorithm solves Temporal Graph Burning. It begins by computing the temporal neighbourhood partition, which we know from Lemma 21 that we can do in time $O(\Lambda n^3)$. Furthermore, there are at most $\phi\phi!$ possible orderings A of subsets of the temporal neighbourhood partition Θ, and at most 2^ϕ sets B. We then simulate temporal graph burning on the graph, which is possible in $O(n^2\Lambda)$ time, where n is the number of vertices in the input graph. This gives us an overall running time of $O(\Lambda n^3 + \phi\phi! 2^\phi n^2 \Lambda) = O(\Lambda n^3 \phi\phi! 2^\phi)$, and Λ is the lifetime of the input graph. If the decomposition is given, we instead obtain a runtime of $O(\Lambda n^2 \phi\phi! 2^\phi)$, as we may drop the extra factor of n needed to compute it. □

4.4 Hardness for Temporal Modular Width

In this section we show that Temporal Graph Burning is NP-hard even on graphs of bounded temporal modular-width. This is achieved by reducing from (3, 2B)-SAT, an NP-hard variant of the Boolean satisfiability problem in which each variable appears exactly twice both positively and negatively [8], defined formally as follows.

(3, 2B)-SAT

Input: A pair (B, C) where B is a set of Boolean variables, and $C = C_1 \wedge ... \wedge C_m$ is a set of clauses over B in CNF, each containing 3 literals $C_j^1 \vee C_j^2 \vee C_j^3$, such that each variable appears exactly twice negatively and exactly twice positively.

Output: Is there a truth assignment to the variables such all of the clauses in C are satisfied?

Our reduction produces a graph where every connected component has bounded temporal neighbourhood diversity, and hence the graph overall has bounded temporal modular-width. Each connected component can be constructed with a use of the substitution operation, where the number of operands equal to the temporal neighbourhood diversity of the connected component, as each class in a temporal neighbourhood partition is either a clique or an independent set. We note that this hardness result contrasts with the recent proof that the static version of the problem is in **FPT** parameterised by modular-width

[56]: this difference arises from the fact that, in the static setting, the length of a longest induced path in the graph (which is upper bounded by the modular-width) gives an upper bound on the time taken to burn the graph, whereas in the temporal setting the times assigned to edges mean that even temporal graphs with an underlying graph of small diameter may take many steps to burn.

Theorem 40. *Temporal Graph Burning is **NP**-complete even when restricted to graphs with constant temporal modular-width.*

Proof. Temporal Graph Burning is in **NP**, as stated in Theorem 4. We now show that it is **NP**-hard by reduction from (3, 2B)-SAT. As such we begin by describing how to construct an instance $((G, \lambda), h)$ of Temporal Graph Burning, given an instance (B, C) of (3, 2B)-SAT with $C = C_1 \wedge \cdots \wedge C_m$, and $|B| = n$. We then go on to show that (B, C) is a yes-instance of (3, 2B)-SAT if and only if $((G, \lambda), h)$ is a yes-instance of Temporal Graph Burning.

To construct $((G, \lambda), h)$, begin by setting $h = 2n + 3m + 1$. Now let the vertex set of G be given by the union of the following sets:

- the set of *literal vertices*: $\{x_i, \neg x_i : i \in [n]\}$,
- the set of *clause vertices*: $\{c_j^1, c_j^2, c_j^3 : j \in [m]\}$,
- the set of *appearance vertices*: $\{u_{i,j}, w_{i,j} : x_i \text{ or } \neg x_i \text{ appears in } C_j\}$,
- the set of *leaf vertices*: $\{y_{i,d}, \neg y_{i,d} : i \in [n], d \in [h+1]\} \cup \{z_{j,d}^1, z_{j,d}^2, z_{j,d}^3 : j \in [m], d \in [h+1]\}$.

We represent the set of time-edges as a set of pairs of edges and a single timestep, such that if $(\{u, v\}, t)$ is a time-edge then $\{u, v\} \in E(G)$ and $\lambda(\{u, v\}) = \{t\}$. This set is then given by the union of the following sets:

- $\{(\{x_i, y_{i,d}\}, 2i + 1), (\{\neg x_i, \neg y_{i,d}\}, 2i + 1) : i \in [n], d \in [h+1]\}$,
- $\{(\{c_j^1, z_{j,d}^1\}, 2n + 3j + 1), (\{c_j^2, z_{j,d}^2\}, 2n + 3j + 1), (\{c_j^3, z_{j,d}^3\}, 2n + 3j + 1) : j \in [m], d \in [h+1]\}$,
- $\{(\{x_i, u_{i,j}\}, 2i), (\{u_{i,j}, w_{i,j}\}, h), (\{w_{i,j}, c_j^\ell\}, 2n + 3j) : x_i \text{ is the } \ell^{\text{th}} \text{ literal in } C_j\}$,
- $\{(\{\neg x_i, u_{i,j}\}, 2i), (\{u_{i,j}, w_{i,j}\}, h), (\{w_{i,j}, c_j^\ell\}, 2n + 3j) : \neg x_i \text{ is the } \ell^{\text{th}} \text{ literal in } C_j\}$.

Note that this graph consists of $2n$ connected components, each corresponding to a literal x_i or $\neg x_i$. We denote by H_i, and $\neg H_i$ the connected components containing x and $\neg x_i$ respectively. One of these connected components can be seen in Fig. 4.1.

In order to show that it is possible to burn this graph in h timesteps if and only if there is a satisfying truth assignment for (B, C), we first prove the following claims.

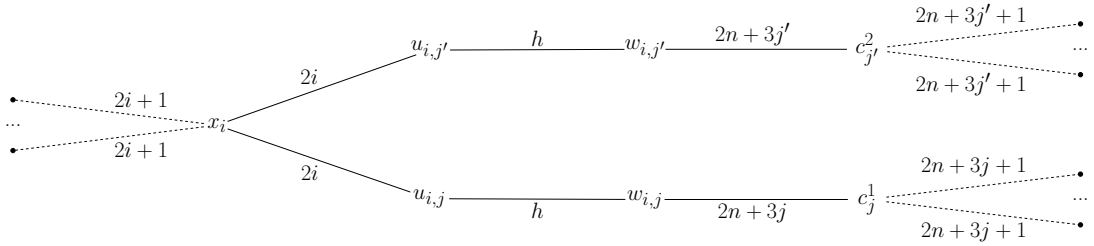

Figure 4.1: The connected component corresponding to the literal x_i, appearing in clauses C_j and $C_{j'}$.

Claim 1. *In order to burn (G, λ) in h or fewer timesteps, for each pair of literal vertices x_i and $\neg x_i$, a fire must be placed at or adjacent to one on timestep $2i-1$, and at the other on timestep $2i$.*

Proof. Begin by observing that in order for (G, λ) to burn in h or fewer timesteps, every literal vertex x_i or $\neg x_i$ must be burning by the end of timestep $2i$, as every literal vertex has $h+1$ adjacent leaves with edges active at timestep $2i+1$. Observe that the fire can only spread to a literal vertex x_i or $\neg x_i$ by the end of timestep $2i$ if it originates at one of the two adjacent non-leaf vertices $u_{i,j}$ and $u_{i,j'}$. As a result, for each literal vertex x_i or $\neg x_i$ a fire must either be placed at the literal vertex by the end of timestep $2i$, or at one of the vertices $u_{i,j}$ and $u_{i,j'}$ by the end of timestep $2i-1$. We now continue by induction on the variable index i. For the base case it follows immediately that a fire must be placed at x_1 on timestep 1 or 2, or the adjacent vertex $u_{1,j}$ on timestep 1. The same is true of the vertex $\neg x_1$. Therefore on timestep 1 a fire must be placed at or adjacent to one of these vertices, and on timestep 2 a fire must be placed at the other vertex. For the inductive step assume that the claim is true for all literal vertices x_a and $\neg x_a$ with $a < i$. Then, on timestep $2i-1$, there must be no fires already placed either at or adjacent to x_i or $\neg x_i$. Then, a fire must be placed at x_i on timestep $2i-1$ or $2i$, or at the adjacent vertex $u_{i,j}$ on timestep $2i-1$. Again, the same is true of vertex $\neg x_i$, and therefore a fire must be placed at one of these vertices on timestep $2i$, and at or adjacent to the other on timestep $2i-1$. □

Claim 2. *In order to burn (G, λ) in h or fewer timesteps, for each triple of clause vertices c^1_j, c^2_j, c^3_j there must exist a permutation π of $\{1,2,3\}$, such that a fire is placed at or adjacent to $c^{\pi(1)}_j$ on timestep $2n+3j-2$, at or adjacent to $c^{\pi(2)}_1$ on timestep $2n+3j-1$, and at $c^{\pi(3)}_1$ on timestep $2n+3j$.*

Proof. Begin by observing that in order for (G, λ) to burn in h or fewer timesteps, every clause vertex c^ℓ_j must be burning by the end of timestep $2n+3j$, as every clause vertex has $h+1$ adjacent leaves with edges active at timestep $2n+3j+1$. Observe that the fire can only spread to a clause vertex c^ℓ_j by the end of timestep $2n+3j$ if it originates at the

adjacent non-leaf vertex $w_{i,j}$. As a result, for every clause vertex c_j^ℓ a fire must either be placed at the clause vertex by the end of timestep $2n + 3j$, or at $w_{i,j}$ before the end of timestep $2n + 3j - 1$. We now continue by induction on the clause index j. For the base case we see from Claim 1 that no fire can be placed at either c_1^1 or $w_{i,1}$ prior to timestep $2n + 1$, and it therefore follows that a fire must be placed at c_j^1 on one of the timesteps $2n+1, 2n+2$, or $2n+3$, or the adjacent vertex $w_{i,1}$ on timesteps $2n+1$ or $2n+2$. The same is true of the two other clause vertices c_1^2 and c_1^3. Therefore there must exist a permutation π of $\{1, 2, 3\}$, such that a fire is placed at or adjacent to $c_1^{\pi(1)}$ on timestep $2n + 1$, at or adjacent to $c_1^{\pi(2)}$ on timestep $2n + 2$, and at $c_1^{\pi(3)}$ on timestep $2n + 3$. For the inductive step assume that the claim is true for all clause vertices c_a^1, c_a^2, c_a^3 with $a < j$. Then on timestep $2n + 3j - 2$, there must be no fires already placed either at or adjacent to c_j^1. Then, a fire must be placed at c_j^1 on one of the timesteps $2n+3j, 2n+3j-1, 2n+3j-2$, or at the adjacent vertex $w_{i,j}$ on one of the timesteps $2n+3j-1, 2n+3j-2$. The same is true of the vertices c_j^2 and c_j^3, and therefore there must exist a permutation π of $\{1, 2, 3\}$, such that a fire is placed at or adjacent to $c_j^{\pi(1)}$ on timestep $2n + 3j - 2$, at or adjacent to $c_1^{\pi(2)}$ on timestep $2n + 3j - 1$, and at $c_1^{\pi(3)}$ on timestep $2n + 3j$. □

Claim 3. *Suppose that there is a successful strategy for (G, λ) of length at most h, and consider the connected component H_i or $\neg H_i$ in which a fire is placed on timestep $2i$. Let c_j^ℓ be a clause vertex in this connected compoment. Then a fire must be placed at or adjacent to c_j^ℓ prior to timestep $2n + 3j$.*

Proof. If a fire is placed in H_i or $\neg H_i$ on timestep $2i$, then it must be placed at the corresponding literal vertex x_i or $\neg x_i$ by Claim 1. The fire cannot spread to the adjacent non-leaf vertex $u_{i,j}$ from the literal vertex, as the edge between $u_{i,j}$ and the literal vertex is only active at time $2i$. Thus, the fire must spread to $u_{i,j}$ from $w_{i,j}$. Furthermore, by Claim 2, a fire is either placed at c_j^ℓ one of the timesteps $2n + 3j - 2, 2n + 3j - 1, 2n + 3j$ or at $w_{i,j}$ on one of the timesteps $2n + 3j - 2, 2n + 3j - 1$. This fire cannot be placed at c_j^ℓ on timestep $2n + 3j$, as otherwise it would not reach $w_{i,j}$. In every other case, the fire is placed prior to timestep $2n + 3j$ as required. □

We are now ready to prove the correctness of our reduction, and begin by showing that if $((G, \lambda), h)$ is a yes-instance, then so is (B, C).

Assume that $((G, \lambda), h)$ is a yes-instance, and consider a strategy that burns the graph in h or fewer timesteps. We then assign a value to each variable x_i according to when the strategy places fires in the connected components H_i or $\neg H_i$. If on timestep $2i - 1$ a fire is placed in H_i, then x_i is assigned true, otherwise, if a fire is placed in $\neg H_i$ then x_i is assigned false. Note that by Claim 1 this truth assignment is well defined. Now consider any clause $C_j \in C$, and see that by Claim 2 one of the corresponding vertices c_j^1, c_j^2, or c_j^3 must have a fire placed at it on timestep $2n + 3j$. Then, by Claim 3, this vertex must be

in the connected component in which a fire was placed on timestep $2i-1$. This literal is therefore assigned true, and thus the clause is satisfied, as required.

Now assume that (B, C) is a yes-instance, and consider a satisfying assignment of the variables in B. We now describe a burning strategy for (G, λ), and show that it burns the graph in h or fewer timesteps. We place fires such a fire is placed at the vertex corresponding to the truthful literal in $\{x_i, \neg x_i\}$ on timestep $2i-1$, and a fire is placed at the vertex corresponding to the literal assigned false in $\{x_i, \neg x_i\}$ on timestep $2i$. As we have a satisfying assignment, each clause $C_j = C_j^1 \vee C_j^2 \vee C_j^3$ must contain a literal C_j^ℓ that evaluates to true, and we place a fire at the corresponding clause vertex c_j^ℓ on timestep $2n+3j$. Fires are placed at the other two clause vertices $\{c_j^1, c_j^2, c_j^3\} \setminus \{c_j^\ell\}$ on timesteps $2n+3j-2$ and $2n+3j-1$ in either order. In this strategy, every literal vertex x_i or $\neg x_i$ is burnt by the end of timestep $2i$, and therefore all literal leaves $y_{i,d}$ burn by the end of timestep h. Every clause vertex c_j^ℓ is burnt by the end of timestep $2n+3j$, and therefore all clause leaves $z_{j,d}^\ell$ burn by the end of timestep h. Finally, if a vertex pair $u_{i,j}$ and $w_{i,j}$ belongs to a connected component H_i or $\neg H_i$ in which a fire is placed on timestep $2i-1$, the fire spreads from the literal vertex to $u_{i,j}$ on timestep $2i$, and then from $u_{i,j}$ on timestep h. Otherwise, the vertex pair must be on a path with a clause vertex at which a fire is placed by timestep $2n+3j-1$, then the fire spreads from the clause vertex to $w_{i,j}$ on timestep $2n+3j$, and from $w_{i,j}$ to $u_{i,j}$ on timestep h. Thus in either case both of these vertices will burn by the end of timestep h as required. □

4.5 Conclusions

In this chapter we considered the parameters of temporal neighbourhood diversity and temporal modular width, analogues of the static parameters of neighbourhood diversity and modular width. Both of these parameters can be small even on dense temporal graphs, and the temporal modular width is a generalisation of the temporal neighbourhood diversity, such that the temporal neighbourhood diversity always bounds the modular width.

We find that both Temporal Firefighter and Temporal Graph Burning admit fixed parameter tractable algorithms when parameterised by the temporal neighbourhood diversity, and also that Temporal Graph Burning remains **NP**-Complete when parameterised by temporal modular width.

In the following chapter we turn our attention again to the parameters of edge-interval-membership-width and vertex-interval-membership-width, and explore more problems to which these parameters may be applied.

Chapter 5

A Meta-Algorithm For Vertex Interval Membership Width

In Chapter 3 we used the parameter of vertex-interval-membership-width to obtain a fixed parameter tractable algorithm for Temporal Firefighter. We believe this parameter is applicable to a wide variety of similar problems concerning spreading, and more generally to any problem in which vertices can be affected only by their temporal neighbours. In this chapter we provide a framework for defining problems that obey this constraint, and prove a meta-theorem that shows that any problem definable in this framework admits a fixed parameter tractable algorithm when parameterised by the vertex-interval-membership-width. In defining this framework, we provide tools to easily show that a problem is in **FPT** when parameterised by vertex-interval-membership-width, and more generally explore the properties of problems that allow for tractability using this parameter.

This framework is based on labellings of the vertices of a temporal graph that change over discrete time. We define a set of conditions constraining the label on a vertex to change only when at least one incident edge is active, and refer to any problem that obeys these conditions as locally temporally uniform. We provide an algorithm capable of solving locally temporally uniform problems and give bounds on the runtime of this algorithm in terms of the vertex-interval-membership-width, showing that the algorithm is efficient when this parameter is small. Then, in order to obtain a fixed parameter tractable algorithm using vertex-interval-membership-width for a given problem, we need only show that it is locally temporally uniform. We provide examples of using this method to obtain algorithms for both Temporal Firefighter and Temporal Graph Burning, and also for Temporal Hamiltonian Path and Temporal Dominating Set, both temporal extensions of classical graph theory problems that do not involve spreading, demonstrating the versatility of this method.

5.1 Locally Temporally Uniform Problems

We consider temporal graph problems that can be expressed in terms of labellings on the vertices of the input graph that change over discrete time. We refer to the labelling of the vertices of a graph at a particular timestep as a state for the graph. Also included in a state is an additional k-length vector of integer counters.

Definition 34 ((X,k)-State)**.** *A (X,k)-state on a vertex set V is a pair (l,c), where $l : V \to X$ is a labelling of the vertices in V using the labels from set X, and c is a vector of k integers, containing integers of size at most a polynomial of the size of the graph.*

In order to obtain tractability with respect to the vertex-interval-membership-width, we specifically consider problems that can be expressed in terms of sequences of states on the vertices in the vertex-interval-membership-sequence, the definition of which we recall from Chapter 3.

Definition 35 (Vertex-Interval-Membership-Width (Bumpus and Meeks [20]))**.** *The vertex-interval-membership-sequence of a temporal graph (G,λ) is the sequence $(F_t)_{t\in[\Lambda]}$ of vertex-subsets of G where $F_t = \{v \in V(G) : \text{mintime}(v) \leq t \leq \text{maxtime}(v)\}$ and Λ is the lifetime of (G,λ).*

The vertex-interval-membership-width of a temporal graph (G,λ) is then the integer $\omega = \max_{t\in[\Lambda]} |F_t|$.

For convenience, we let $F_0 = F_1$, and furthermore, we let A_t denote the set of vertices with incident edges active on timestep t, and note that $A_t \subseteq F_t$. We first define temporally uniform problems, for which the sequences of states can be uniformly produced by a transition routine which when given two states returns true if the second state can follow the first. We will then provide a definition for *locally* temporally uniform problems that further restricts such problems, such that one state can only follow from another if only the labels on the vertices in their active interval change, and the transition routine ignores any label not on a vertex in its active interval. We go on to show that all locally temporally uniform problems have a tractable algorithm with respect to the vertex-interval-membership-width.

Definition 36 ((X,k)-Temporally Uniform Problem)**.** *We say that a problem P is temporally uniform if and only if there exists:*

1. *a polynomial time transition algorithm **Tr** that takes a static graph, and two (X,k)-States for the vertices of this graph and returns **true** or **false**,*
2. *a polynomial time accepting algorithm **Ac** that takes a (X,k)-State and the problem instance, and returns **true** or **false**, and*

3. *a set of (X,k)-States $S_{0,x}$ for every instance x,*

such that an input $x = (\mathcal{G}, ...)$ is a yes instance of P if and only if there exists a sequence $s_0, ..., s_\Lambda$ of (X,k)-States with $s_0 \in S_{0,x}$, $\boldsymbol{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \boldsymbol{true}$ for all timesteps $1 \leq t \leq \Lambda$, and $\boldsymbol{Ac}(s_\Lambda, x) = \boldsymbol{true}$.

We say that two states $s = (l, c)$ and $s' = (l', c')$ for vertex set V agree on a vertex set U if and only if $U \subseteq V$, l gives the same label as l' to every vertex in U, and $c = c'$.

We are now ready to define locally temporally uniform problems. Locality refers to the fact that we restrict temporally uniform problems such that the transition routine may only consider vertices in their active interval in the vertex-interval-membership-sequence.

Definition 37 ((X,k)-Locally Temporally Uniform Problem)**.** *We say that a (X,k)-temporally uniform problem P is (X,k)-locally temporally uniform if and only if for any temporal graph $\mathcal{G}$, and instance of the problem x:*

1. *There exists a label U such that every initial state $s_0 \in S_0$ gives label U to all vertices not in F_0.*

2. *For any pair of states s and s', if $\boldsymbol{Tr}(s, s', G) = true$ then s and s' give the same label to every isolated vertex in G.*

3. *For every quadruple of states r, r', s, and s', if r and s agree on the non-isolated vertices of G, each pair of states r, r' and s, s' give the same label to every isolated vertex of G, and r' and s' agree on the non-isolated vertices of G, then $\boldsymbol{Tr}(r, r', G) = \boldsymbol{Tr}(s, s', G)$.*

4. *For every pair of states s_Λ and s'_Λ that agree on the vertices in A_Λ, $\boldsymbol{Ac}(s_\Lambda, x) = \boldsymbol{true}$ if and only if $\boldsymbol{Ac}(s'_\Lambda, x) = \boldsymbol{true}$.*

We now give a meta-algorithm that solves any locally temporally uniform problem using the transition routine and the accepting routine, when given the input $x = (\mathcal{G}, ...)$ and the associated set of initial states S_0. This algorithm uses locality to avoid having to consider every possible state on each timestep. Instead, on each timestep t the algorithm considers only one state for each possible state for F_t, the number of which is bounded by a function of the vertex-interval-membership-width. These states for F_t are extended to states for the vertex set of the input graph by giving every other vertex label U. We first prove a lemma that shows that states extended in this manner will agree with any state in a sequence following from an initial state, on all vertices not yet in their active interval.

Lemma 41. *Consider any instance $(\mathcal{G}, ...)$ of a locally temporally uniform problem, such that S_0 is the set of initial states, $\boldsymbol{Tr}$ is the transition algorithm and $[F_t]_{t \leq \Lambda}$ is the vertex-interval-membership-sequence of $\mathcal{G}$.*

If $s_0, ..., s_t$ is a sequence of states such that $s_0 \in S_0$, and $\boldsymbol{Tr}(s_{i-1}, s_i, \mathcal{G}_i) = true$ for $1 \leq i \leq t$, then s_t gives label U to every vertex $v \in \bigcup_{t'>t} F_{t'} \setminus F_t$.

Proof. We continue by induction on the timestep t. If $t = 0$, then as the problem is locally temporally uniform, s_0 gives label U to every vertex not in F_0.

Then, assume by induction that s_{t-1} gives label U to every vertex $v \in \bigcup_{t'>t-1} F_{t'} \setminus F_{t-1}$. If $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \text{true}$, then s_{t-1} and s_t give the same label to any isolated vertex in $\mathcal{G}_t$, and therefore give label U to every vertex $v \in \bigcup_{t'>t} F_{t'} \setminus F_t \subseteq \bigcup_{t'>t-1} F_{t'} \setminus F_{t-1}$ as any vertex $v \in \bigcup_{t'>t} F_{t'} \setminus F_t$ is not in F_t, and therefore cannot have any active incident edges active on timestep t, so is isolated in $\mathcal{G}_t$. □

Algorithm 4 LOCALLY TEMPORALLY UNIFORM ALGORITHM

Input: A problem input $x = (\mathcal{G}, ...)$ with Λ the lifetime of $\mathcal{G}$, and an associated set S_0 of initial (X, k)-States.
Output: Whether x is a yes-instance.

```
1: Fix a label U ∈ X
2: for t = 1, ..., Λ do
3:     S_t ←{}
4:     for all Possible states s_{F_t} for F_t do
5:         s_t ←the state agreeing with s_{F_t} such that all vertices not in F_t are given label U
6:         for all s_{t-1} ∈ S_{t-1} do
7:             r_{t-1} ←the state agreeing with s_{t-1} on F_t such that all other vertices are given label U
8:             if Tr(r_{t-1}, s_t, G_t) then
9:                 S_t ←S_t ∪ {s_t}
10:            end if
11:        end for
12:    end for
13: end for
14: for all s_Λ ∈ S_Λ do
15:    if Ac(s_Λ, x) then
16:        return True
17:    end if
18: end for
19: return False
```

On a timestep t, Algorithm 4 considers every possible state for F_t, and then extends these states to the entire graph by giving every other vertex some fixed label. We first show that in doing so, the algorithm does not omit any required states, and for any state produced by repeated applications of the transition routine to an initial state, the algorithm will produce an agreeing state.

Lemma 42. *Let x be an instance of a (X, k)-locally temporally uniform problem P with transition routine $\boldsymbol{Tr}$ and acceptance routine $\boldsymbol{Ac}$, along with an associated set S_0 of initial*

*(X,k)-States. There exists a sequence of states $s_0,...s_t$ with $s_0 \in S_0$ and $\boldsymbol{Tr}(s_{i-1},s_i,\mathcal{G}_i) =$ **true** for all timesteps $1 \leq i \leq t$, if and only if there exists a state s'_t in the set S_t produced by Algorithm 4 that agrees with s_t on F_t.*

Proof. We in fact prove a stronger result, that not only does there exist a state s'_t in S_t that agrees with s_t on F_t, but that this state gives label U to every vertex not in F_t. We proceed by induction on the length of the sequence t. The base case when $t = 0$ is trivial, as S_0 is given as input to the algorithm, and by Definition 37 every state $s_0 \in S_0$ gives label U to every vertex not in F_0.

Assume by induction that there exists a sequence of states $s_0,...s_{t-1}$ with $s_0 \in S_0$ and $\mathbf{Tr}(s_{i-1},s_i,\mathcal{G}_i) = \mathbf{true}$ for all timesteps $1 \leq i \leq t-1$, if and only if there exists a state s'_{t-1} in the set S_{t-1} produced by Algorithm 4 that agrees with s_{t-1} on F_{t-1}, and gives label U to every vertex not in F_{t-1}.

Now, consider any state s_t such that there exists a sequence of states $s_0,...s_t$ with $s_0 \in S_0$ and $\mathbf{Tr}(s_{i-1},s_i,\mathcal{G}_i) = \mathbf{true}$ for all timesteps $1 \leq i \leq t$. By induction there exists a state $s'_{t-1} \in S_{t-1}$ that agrees with s_{t-1} on F_{t-1}, and gives all vertices not in F_{t-1} label U. This state will be used to produce a state r_{t-1} which agrees with s'_{t-1} on F_t. By Lemma 41, any vertices in $F_t \setminus F_{t-1}$ will be given label U by s_{t-1}, and r_{t-1} also gives these vertices label U, and therefore r_{t-1} agrees with s_{t-1} on F_t, and gives label U to every vertex not in F_t. Now, Algorithm 4 will consider a state s'_t that agrees with s_t on F_t, such that every vertex not in F_t is given label U, as it considers every possible state for F_t, extending these states by labelling the remaining vertices with U. Now as $\mathbf{Tr}(s_{t-1},s_t,\mathcal{G}_t) =$ true and A_t is exactly the set of non-isolated vertices of $\mathcal{G}_t$, s_{t-1} and s_t give the same label to every vertex not in A_t by Definition 37. Now, consider any vertex $v \notin A_t$. If $v \in F_t$ then s_t gives the same label to v as s'_t, as s'_t and s_t agree on F_t. Then s_{t-1} also gives the same label to v as $v \notin A_t$ and s_{t-1} and s_t give the same label to any vertex not in A_t. Finally, r_{t-1} gives the same vertex to v as it agrees with s_{t-1} on F_t. Otherwise, if $v \notin F_t$, both s'_{t-1} and r_{t-1} give label U to v. Therefore r_{t-1} and s'_{t-1} give the same label to every vertex not in A_t. Furthermore, s'_t and s_t agree on A_t, as $A_t \subseteq F_t$, as do r_{t-1} and s_{t-1}. Then by Definition 37, $\mathbf{Tr}(r_{t-1},s'_t,\mathcal{G}_t) =$ true, and line 9 of Algorithm 4 will place the state s'_t in S_t.

Finally, consider any state $s'_t \in S_t$, and see that there must exist some state $s'_{t-1} \in S_{t-1}$ such that if r_{t-1} is the state that agrees with s'_{t-1} on F_t and gives label U to all vertices not in F_t, then $\mathbf{Tr}(r_{t-1},s'_t,\mathcal{G}_t) =$ true. By induction s'_{t-1} agrees on F_{t-1} with some state s_{t-1} where there exists a sequence of states $s_0,...s_{t-1}$ with $s_0 \in S_0$ and $\mathbf{Tr}(s_{i-1},s_i,\mathcal{G}_i) = \mathbf{true}$ for all timesteps $1 \leq i \leq t-1$. Furthermore, by definition, s'_{t-1} gives label U to every vertex not in F_{t-1}, and so r_{t-1} gives label U to every vertex in $F_t \setminus F_{t-1}$. By Lemma 41 s_{t-1} also gives label U to every vertex in $F_t \setminus F_{t-1}$, and r_{t-1} agrees with s_{t-1} on $F_t \cap F_{t-1}$, so therefore r_{t-1} agrees with s_{t-1} on F_t. As $\mathbf{Tr}(r_{t-1},s'_t,\mathcal{G}_t) =$ true we have that r_{t-1} and

s'_t give the same label to every isolated vertex in $\mathcal{G}_t$. Furthermore r_{t-1} and s_{t-1} agree on the non-isolated vertices of $\mathcal{G}_t$, as these are $A_t \subseteq F_t$. Consider now the state s_t that agrees with s'_t on the non-isolated vertices of $\mathcal{G}_t$, and gives the same label as s_{t-1} to every isolated vertex in $\mathcal{G}_t$. By Definition 37 we have that $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \text{true}$ as $\mathbf{Tr}(r_{t-1}, s'_t, \mathcal{G}_t) = \text{true}$. Finally see that for any vertex $v \in F_t \setminus A_t$, that is for any isolated vertex of $\mathcal{G}_t$ in F_t, s_t and s_{t-1} give the same label to v. Now, as r_{t-1} and s_{t-1} agree on F_t, r_{t-1} also gives the same label to vertex v. Finally s'_t gives the same label to vertex v, as it gives the same label as r_{t-1} to every isolated vertex of $\mathcal{G}$. Therefore s'_t and s_t agree on F_t as required. □

We are now ready to prove our overall meta-theorem, showing that our algorithm solves any locally temporally uniform process, and does so in a time that is exponential only in the vertex-interval-membership-width, and otherwise polynomial in the size of the input.

Theorem 43. *Let x be an instance of a (X, k)-locally temporally uniform problem P with transition routine **Tr** and acceptance routine **Ac**, along with an associated set S_0 of initial (X, k)-States. We can determine if x is a yes-instance of P in time $O(\Lambda f(n, \Lambda) b^{2k} |X|^{2\omega})$, where ω is the vertex-interval-membership-width, b is the maximum size of any counter variable in a (X, k)-state, and f is a function such that **Tr** and **Ac** both run in $O(f(n, \Lambda))$ time.*

Proof. We show that Algorithm 4 returns true if and only if it is given a yes-instance as input, and furthermore that Algorithm 4 runs in the required time.

If Algorithm 4 returns true, then there exists a state $s_\Lambda \in S_\Lambda$ such that $\mathbf{Ac}(s_\Lambda, x) = \text{true}$. Furthermore, as Algorithm 4 only places a state s_t in S_t if there exists a state $s_{t-1} \in S_{t-1}$ with $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \mathbf{true}$, there exists a sequence $s_0, ..., s_\Lambda$ of states, such that $s_t \in S_t$ for every timestep t, and $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \mathbf{true}$ for all timesteps $t \geq 1$. Therefore, by Definition 37, x is a yes-instance.

If x is a yes-instance, then there exists a sequence $s_0, ..., s_\Lambda$ of (X, k)-States with $s_0 \in S_0$, and $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \mathbf{true}$ for all timesteps $t \geq 1$, and $\mathbf{Ac}(s_\Lambda, x) = \mathbf{true}$. Then, by Lemma 42 there exists a state $s'_\Lambda \in S_\Lambda$ that agrees with s_Λ on A_Λ. Then, by Definition 37 $\mathbf{Ac}(s'_\Lambda, x) = \mathbf{true}$, as $\mathbf{Ac}(s_\Lambda, x) = \mathbf{true}$.

For every timestep t, there is at most one entry in S_t for every possible state of F_t, of which there at most $b^k |X|^\omega$. Now for each timestep t Algorithm 4 runs the transition routine for every pair of a possible state in S_t, and a state in S_{t-1}. This can be achieved in time $O(f(n, \Lambda) b^{2k} |X|^{2\omega})$. Finally, Algorithm 4 runs the acceptance routine for every state in S_Λ, which can be achived in time $O(f(n, \Lambda) b^k |X|^\omega)$, giving an overall runtime of $O(\Lambda f(n, \Lambda) b^{2k} |X|^{2\omega} + f(n, \Lambda) b^k |X|^\omega) = O(\Lambda f(n, \Lambda) b^{2k} |X|^{2\omega})$. □

5.2 Examples of Temporally Local Processes

Using the machinery we have just defined, it is possible to find tractable algorithms for a range of problems. In order to show that a given problem admits a tractable algorithm when parameterised by the vertex-interval-membership-width, we are required to show that it is locally temporally uniform by providing the transition and accepting routines. It is also necessary to provide bounds on the runtime of these routines. We may then use Theorem 43 to obtain a tractable algorithm for our problem, thus negating the need to construct a dynamic programming algorithm from scratch. Whilst many problems are locally temporally uniform, it is worth noting that this is not always the case. Any problem where the state of a vertex at a given time depends not just on the other relevant vertices at that time is not locally temporally uniform. For example, consider a problem where we are given a temporal graph $\mathcal{G}$, a set of vertices V, and an integer k, and wish to determine if there exists a set of vertices V' of size at least k such that each vertex in V can be reached by every vertex in V'. This problem is not locally temporally uniform, since we need to know if the set of vertices that reach a vertex $v \in V$ contains all of the vertices in a candidate V'. We continue by giving some examples of problems for which we can find tractable algorithms using the approach outlined above.

5.2.1 Temporal Hamiltonian Path

We begin by considering a temporal extension of the Hamiltonian Path problem, which we refer to as Temporal Hamiltonian Path, and define below.

Temporal Hamiltonian Path
Input: A temporal graph $\mathcal{G}$.
Output: Does there exist a temporal path on $\mathcal{G}$ containing every vertex in the graph?

We produce an algorithm that will find any such path beginning on a vertex in F_1, the first entry in the vertex-interval-membership sequence. If no such path exists, then we can re-run our algorithm on $\mathcal{G}$, except with all temporal edges active on timestep 1 removed, and all other temporal edges active one timestep earlier. If any temporal hamiltonian path exists, we would be able to find it in this manner using at most $O(\Lambda)$ executions of the algorithm.

We use $(X, 1)$-states, where the label set $X = \{V, U, C\}$ contains a label for visited, unknown, and current vertices respectively, and the counter vector contains a single integer h, which counts the total number of visited vertices. We will define our transition routine and initial states such that each state produced by repeated applications of the transition routine corresponds to the existence of a temporal path that traverses h vertices, such that if there is a vertex given label C, the path is currently at that vertex, and any

vertices labelled V are traversed by the path. The accepting routine then returns true if $h = |V(\mathcal{G})|$. We now show that Temporal Hamiltonian Path is locally temporally uniform, by giving the transition and acceptance routines, and the set of initial states.

Algorithm 5 Temporal Hamiltonian Path Transition

Input: A static graph G and states $(l_1, (h_1))$ and $(l_2, (h_2))$ for $V(\mathcal{G})$.

Output: Returns true when $(l_2, (h_2))$ corresponds to a path that traverses zero or one further vertices than the path corresponding to $(l_1, (h_1))$ and false otherwise.

1: Let U_1 and U_2 be the set of vertices labelled U by l_1 and l_2 respectively, and equivalently for V_1, V_2, and C_1, C_2.
2: **if** $C_1 \setminus C_2$ contains a single vertex c_1, and $C_2 \setminus C_1$ contains a single vertex c_2 **then**
3: **if** $\{c_1, c_2\} \in E(G)$, $c_2 \in U_1$, $h_2 = h_1 + 1$, and $V_1 \cup \{c_1\} = V_2$ **then**
4: **return** True
5: **end if**
6: **end if**
7: **if** $(l_1, (h_1)) = (l_2, (h_2))$ **then**
8: **return** True
9: **end if**
10: **return** False

Given a state $(l, (h))$ and an instance of Temporal Hamiltonian Path, that is a temporal graph $\mathcal{G}$, the acceptance routine returns true if and only if $h = |V(\mathcal{G})|$. We use $|F_0|$ initial states. For each vertex $v \in F_0$, construct a state for F_0 where v is labelled C, all other vertices in F_0 are labelled U, and $h = 1$.

We now show that there exists a correspondence between the temporal paths on $\mathcal{G}$ and sequences of states beginning with initial states and related by the transition routine. We say that a sequence $s_0, ..., s_t$ of states *corresponds* to a temporal path beginning on a vertex in F_0 if and only if:

1. s_0 is an initial state,
2. $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_i) =$ true for every $1 \leq i \leq t$,
3. if s_t gives a single vertex label C, and this is the final vertex on the path, and the path has arrival time $t' \leq t$, and
4. the vertices traversed by the path are exactly the vertices given label V by s_t,
5. the value of h given by s_t is equal to the number of vertices traversed by the path.

Lemma 44. *For any timestep t, there exists a temporal path beginning on a vertex in F_0, and arriving on a timestep $t' \leq t$ if and only if there exists a corresponding sequence of states $s_0, ...s_t$.*

Proof. We proceed by induction on the timestep t. Any temporal path that begins on a vertex in F_0 and has arrival time 0 can only contain a single vertex. Our initial states contain a state for every vertex in F_0 that gives label C to this vertex, and sets h to 1, giving the remaining vertices label U, and thus the base case holds.

Otherwise, first assume that there is a temporal path P beginning on a vertex in F_0 and arriving on a timestep $t' \leq t$ at a vertex v, and traversing ℓ vertices. If $t' < t$, then by induction there exists a sequence of states $s_0, ...s_{t-1}$ corresponding to P. Now take $s_t = s_{t-1}$, and see that $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t[F_t]) =$ true as line 8 of Algorithm 5 will return true. If $t' = t$, there must exist some temporal path beginning on a vertex in F_0 and arriving on a timestep $t' \leq t-1$ at a vertex u adjacent to v on timestep t, and hence by induction there exists a sequence of states $s_0, ...s_{t-1}$ corresponding to this path. Let s_t be the state that has a value of h one greater than s_{t-1}, gives label C to v, label V to u, and labels all other vertices as they are by s_{t-1}. See that $\{u, v\} \in E(\mathcal{G}_t[F_t])$, as P traverses $\{u, v\}$ on timestep t, and therefore $t \in \lambda(\{u, v\})$. Therefore $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t[F_t]) =$ true, as line 7 of Algorithm 5 will return true.

Now assume there is a sequence of states $s_0, ...s_t$ such that s_0 is an initial state, and $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_t[F_t]) =$ true for every $1 \leq i \leq t$, and if such a vertex exists let v be the vertex given label C by s_t. Now consider the sequence $s_0, ..., s_{t-1}$. By induction, there must exist a temporal path P corresponding to this sequence, and let u be the final vertex on this path. See that $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t[F_t]) =$ true. If this is the case because line 4 of Algorithm 5 returns true then $\{u, v\} \in \mathcal{G}_t$, and therefore $t \in \lambda(\{u, v\})$. There is then a temporal path that traverses the same edges as P, before traversing $\{u, v\}$ on timestep t. This path will traverse all the vertices traversed by P, along with the additional vertex u, and therefore corresponds to s_t. Otherwise, if line 11 of Algorithm 5 returns true, then $s_t = s_{t-1}$, and therefore P corresponds to $s_0, ..., s_t$. □

Theorem 45. *Temporal Hamiltonian Path is locally temporally uniform.*

Proof. Each initial state gives label C to one vertex in F_0, and label U to all other vertices, thus all initial states give label U to any vertex not in F_0, as required. Line 4 of Algorithm 5 is the only line of the algorithm that returns true if the two input states are labeled differently. This line returns true if only the two vertices c_1 and c_2 have different labels, with all other vertices labeled identically. Furthermore if line 4 returns true then $\{c_1, c_2\} \in E(G)$, and neither c_1 and c_2 are isolated in G, and the algorithm returns true only if all isolated vertices in G are given the same label as required.

Consider any graph G and pair of states s and s' such that $\mathbf{Tr}(s, s', G) =$ true. Let C_s, U_s be the sets of vertices labeled C and U by s respectively, and equivalently for $C_{s'}, U_{s'}$ and s'. Now consider any vertex v isolated in G. If $v \in C_s$, then $v \in C_{s'}$, as $C_s \setminus C_{s'}$ only contains one vertex, and this vertex is not isolated in G. If $v \in C_{s'}$, then similarly $v \in C_s$ as $C_{s'} \setminus C_s$ only contains one vertex, and this vertex is not isolated in G. If $v \in V_s$, then

$v \in V_{s'}$ as $V_s \subseteq V_{s'}$. If $v \in V_{s'}$, then $v \in V_s$, as $V_{s'} \setminus V_s$ only contains one vertex, and this vertex is not isolated in G. Finally, as C_s, V_s, U_s partitions the vertices of G, and so does $C_{s'}, V_{s'}, U_{s'}$, we must have that if $v \in U_s$ if and only if $v \in U_{s'}$. Therefore s' gives the same label as s to every non-isolated vertex of G.

Consider any graph G and a quadruple of states r, r', s, and s' such that r and s agree on the non-isolated vertices in G, r' and s' agree on the non-isolated vertices in G, and the pairs s, s' and r, r' both give the same label to every isolated vertex not in G. Assume without loss of generality that $\mathbf{Tr}(r, r', G) = \text{true}$.

If this is because line 8 of Algorithm 5 returns true, see that $r = r'$. Then as s agrees with r on the non-isolated vertices of G, s also agrees with r' on the non-isolated vertices of G, and r' agrees with s' on the non-isolated vertices of G. Therefore s agrees with s' on the non-isolated vertices of G, and by definition s and s' give the same label to every isolated vertex of G, and so $s = s'$ and line 8 of Algorithm 5 will return true when given s, s' and G as input.

Otherwise, if $\mathbf{Tr}(r, r', G) = \text{true}$ because line 4 of Algorithm 5 returns true, let I be the non-isolated vertices in G, and C_s, $C_{s'}$, C_r, and $C_{r'}$ be the vertices labeled C by s, s', r, and r' respectively. See that $C_s \setminus I = C_{s'} \setminus I$, and therefore $C_s \setminus C_{s'} = (C_s \cap I) \setminus (C_{s'} \cap I)$, and $C_{s'} \setminus C_s = (C_s \cap I) \setminus (C_{s'} \cap I)$. Furthermore, $C_s \cap I = C_r \cap I$ and $C_{s'} \cap I = C_{r'} \cap I$, and therefore $C_s \setminus C_{s'} = (C_r \cap I) \setminus (C_{r'} \cap I)$, and $C_{s'} \setminus C_s = (C_{r'} \cap I) \setminus (C_r \cap I)$. Then as r and r' give the same label to every vertex not in I, $C_s \setminus C_{s'} = C_r \setminus C_{r'}$, and $C_{s'} \setminus C_s = C_{r'} \setminus C_r$.

Thus, $C_s \setminus C_{s'}$ and $C_{s'} \setminus C_s$ both contain the same single vertices c_1 and c_2 as $C_r \setminus C_{r'}$ and $C_{r'} \setminus C_r$ respectively. We have that $\{c_1, c_2\} \in E(G)$ as $\mathbf{Tr}(r, r', G) = \text{true}$, and c_2 is given label U by s, as it is given label U by r, $c_2 \in I$, and s and r agree on I. Also, s' has the same value of h as r', and s has the same value of h as r, so $h_{s'} = h_s + 1$. Finally, c_1 is given label V by s', as it is given label V by r', and s' and r' agree on I. Any vertex $v \in I$ and not equal to c_1 or c_2 is given the same label by s and s', as r and r' give the same label to v, and s agrees with r on I, and s' agrees with r' on I. Any vertex $v \notin I$ is given the same label by s and s' by definition, and hence $V_s \cup c_2 = V_{s'}$, $\mathbf{Tr}(s, s', G) = \text{true}$ because line 4 of Algorithm 5 returns true when given s, s' and G as input.

Finally, see that the acceptance routine checks only the value of a counter variable, and therefore if it returns true when given a state s, then it will return true when given any state agreeing with s on any vertex set. □

Then, as Algorithm 5 runs in time $O(n)$ by checking the label on each vertex in turn, and we use 1 counter variable of size at most n and 3 labels, we finally obtain the following corollary from Theorem 43.

Corollary 5. *Temporal Hamiltonian Path can be solved in time* $O(\Lambda n n^2 3^{2\omega}) = O(\Lambda^2 n^3 3^{2\omega})$*, where* Λ *is the lifetime of the input temporal graph,* n *the number of vertices, and* ω *the vertex-interval-membership-width.*

5.2.2 Temporal Firefighter

In Chapter 3 we showed that Temporal Firefighter is in FPT with respect to the parameter of vertex-interval-membership-width, and gave a dynamic programming algorithm to solve it using this parameter. We now show how we can use our framework of temporally local processes to obtain the same result, and show the existence of an algorithm without the necessity to construct a dynamic program. As in Chapter 3 we use the equivalent problem of Temporal Firefighter Reserve, which allows us to only consider strategies that defend temporally adjacent to the fire. Thereby ensuring that every defence made on a timestep t is made at a vertex in A_t. Furthemore we assume that we are given an instance $(\mathcal{G}, r)$ of Temporal Firefighter Reserve such that r has an incident edge active on timestep 1. Note that if we are given an instance where this is not the case, we could take the earliest timestep at which r has an active incident edge to be timestep 1, and increase the starting budget according to the number of ommited timesteps, as the fire cannot leave r before its first incident edge is active.

We use $(X, 2)$-states, where the label set $X = \{B, D, U\}$ contains a label for burning, defended, and unburning vertices respectively, and the counter vector contains an integer h which counts the total number of burnt vertices, and an integer b which counts the current available budget. We will define our transition routine and initial states such that each state produced by repeated applications of the transition routine corresponds to the existence of a stratagy for Temporal Firefighter Reserve where there are b burning vertices and these are all given label B, the strategy has defended every vertex labeled D, and all unburning and undefended vertices are given label U. Note that both b and h can be at most the number of vertices in the input graph.

Given an instance $((\mathcal{G}, r), k)$, we use one initial state, r is labelled B, all other vertices are labelled U, and b and h are both equal to one. We now give the transition algorithm for this process, and thus prove it is temporally local.

Algorithm 6 Temporal Firefighter Reserve Transition

Input: A static graph G and states $(l_1, (b_1, h_1))$ and $(l_2, (b_2, h_2))$ for $V(\mathcal{G})$.

Output: Returns true when $(l_2, (b_2, h_2))$ corresponds to state of the graph when the fire spreads after an in budget defence is made following the state corresponding to $(l_1, (b_1, h_1))$ and false otherwise.

1: Let U_1 and U_2 be the set of vertices labelled U by the l_1 and l_2 rspectively, and equivalently for D_1, D_2, and B_1, B_2.
2: Let I be the non-isolated vertices of G
3: **if** $D_2 \setminus D_1 \subseteq I \cap U_1$ and $b_2 = b_1 - |D_2 \setminus D_1| + 1$ and $b_2 \geq 1$ and $B_2 = N_G[B_1] \setminus D_2$ and $h_2 = h_1 + |B_2 \setminus B_1|$ and $U_2 = U_1 \setminus (B_2 \cup D_2)$ **then**
4: **return** True
5: **else**
6: **return** False
7: **end if**

Given a state $(l, (b, h))$ and an instance of Temporal Firefighter Reserve, that is a rooted temporal graph $(\mathcal{G}, r)$ along with an integer k, the acceptance routine returns true if and only if $|V(\mathcal{G})| - h \geq k$.

We now show that there exists a correspondence between sequences of defences for Temporal Firefighter Reserve that only defend active vertices on $(\mathcal{G}, r)$ and sequences of states beginning with initial states and related by the transition routine. We say that a sequence $s_0, ..., s_t$ of states *corresponds* to a t-length sequence S of defences for Temporal Firefighter Reserve that only defends active vertices on $(\mathcal{G}, r)$ if and only if:

1. s_0 is an initial state,
2. $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_i) =$ true for every $1 \leq i \leq t$,
3. the vertices given label B by s_t are exactly the burning vertices after the defences from S are played,
4. the vertices given label D by s_t are exactly the vertices defended by S,
5. the vertices given label U by s_t are exactly the vertices that are unburning and undefended after the defences from S are played,
6. the value of h given by s_t is equal to the number of burning vertices after the defences from S are played,
7. the value of b given by s_t is equal to the budget available at the start of timestep $t + 1$ after the defences from S are played.

Lemma 46. *For any timestep t, there exists a t-length sequence of defences for Temporal Firefighter Reserve that only defends active vertices on $(\mathcal{G}, r)$, if and only if there exists a corresponding sequence of states $s_0, ...s_t$.*

Proof. We proceed by induction on the timestep t. On timestep 0, only the root is burning and no defences can have been made, and the budget will be 1. Our initial states contain only one state which gives label B to r, label U to every other vertex, and sets h and b to 1, and thus the base case holds.

Otherwise first let S be a t-length sequence of defences that only defends active vertices for Temporal Firefighter Reserve, letting S_{t-1} be the first $t - 1$ defences from S, and A the set of vertices defended by S on timestep t. See that by induction there must be some sequence of states $s_0, ..., s_{t-1}$ corresponding to S_{t-1}. Now, let s_t be the state such that all vertices burning after the defences from S are played are given label B, all vertices defended by S are given label D, and all remaining vertices are given label U. Furthermore, let h be the number of burning vertices after the defences from S are played, and b the

budget available at the start of timestep $t+1$ after the defences from S are played. See that $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) =$ true, as by the defintion of Temporal Firefighter Reserve, and our assumption that all defences are made at active vertices, all vertices in A must be unburning and undefended at the end of timestep $t-1$, and have active incident edges on timestep t and are therefore not isolated in $\mathcal{G}_t$, the budget available after the vertices in A are defended is equal to $b_{t-1} - |A| + 1$ where b_{t-1} is the previous budget, the budget available after the vertices in A are defended must be at least 1, and the burnt vertices after the fire spreads are the undefended vertices in the temporal neighbourhood of the vertices burning before the vertices in A are defended. Therefore $s_0, ..., s_t$ corresponds to S.

Now let $s_0, ..., s_t$ be a sequence of states such that s_0 is an initial state and $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_i) =$ true for every $1 \leq i \leq t$. By induction there exists a sequence of defences S_{t-1} corresponding to $s_0, ..., s_{t-1}$. Consider the set of vertices $D_2 \setminus D_1$. As $\mathbf{Tr}(s_t, s_t, \mathcal{G}_t) =$ true every vertex in this set is undefended and unburning after the fire spreads on timestep $t-1$ and has an active incident edge on timestep t. Furthermore, the budget after defending these vertices would be $b_{t-1} - |A| + 1$ where b_{t-1} is the value of b given by s_{t-1}, and this value is greater than or equal to 1. Thus, the vertices in $D_2 \setminus D_1$ can be defended on timestep t after the defences from S_{t-1} are made. Furthermore the vertices burning after the fire spreads after these defences are made are all the undefended vertices in the closed temporal neighbourhood of the vertices burning the defences were made. Therefore if we let S be the sequence of defences such that the first $t-1$ defences are those in S_{t-1}, and the tth defence defends the vertices in $D_2 \setminus D_1$, this corresponds to the sequence $s_0, ...s_t$. □

Theorem 47. *Temporal Firefighter Reserve is locally temporally uniform.*

Proof. The initial state gives label B to r, which we assume is in F_0, and label U to all other vertices. Thus all initial states give label U to any vertex not in F_0, as required.

Consider any pair of states s and s' and graph G such that $\mathbf{Tr}(s, s', G) =$ true. Let D_s, B_s, U_s be the sets of vertices labeled D, B, and U by s respectively, and equivalently for $D_{s'}, B_{s'}, U_{s'}$ and s'. Let v be any vertex isolated in G. If $v \in B_s$ then $v \notin D_{s'}$, as $D_{s'} \setminus D_s \subseteq I \cap U_s \subseteq I$, $v \notin D_s$, and v is isolated in G. Therefore $v \in N_G[B_s] \setminus D_{s'} = B_{s'}$. If $v \in B_{s'}$, then $v \in B_s$, as $v \in N_G[B_s]$, and v is isolated in G. If $v \in U_s$, then $v \notin D_{s'}$, as $D_{s'} \setminus D_s \subseteq I \cap U_s \subseteq I$, and $v \notin D_s$ and v is isolated in G. Furthermore, $v \notin B_{s'}$, as $v \notin B_s$, and we just showed that any isolated vertex in $B_{s'}$ is in B_s. Therefore $v \in U_{s'}$, as $U_{s'}, B_{s'}, D_{s'}$ partition the vertices of G. If $v \in U_{s'}$, then $v \in U_s \setminus (B_{s'} \cup D_{s'}) \subseteq U_s$. Finally, as D_s, B_s, U_s partitions the vertices of G, and so does $D_{s'}, B_{s'}, U_{s'}$, we must have that if $v \in D_s$ if and only if $v \in D_{s'}$. Therefore s' gives the same label as s to every non-isolated vertex of G.

Consider any graph G and a quadruple of states r, r', s, and s' for $V(G)$, such that r and s agree on the non-isolated vertices in G, r' and s' agree on the non-isolated vertices

in G, and the pairs s, s' and r, r' both give the same label to every isolated vertex not in G. Assume without loss of generality that $\mathbf{Tr}(r, r', G) = \text{true}$.

Let I be the non-isolated vertices in G, and D_s, $D_{s'}$, D_r, and $D_{r'}$ be the vertices labeled D by s, s', r, and r' respectively. See that $D_{s'} \setminus D_s = D_{r'} \setminus D_r$, as $D_{r'} \setminus D_r \subseteq I$, s and s' give the same label to every vertex not in I, and s and r agree on I, as do s' and r'. Furthermore, $U_s \cap I = U_r \cap I$ and $D_{r'} \setminus D_r \subseteq U_r \cap I$, so therefore $D_{s'} \setminus D_s \subseteq U_s \cap I$. Consider any vertex $v \in B_{s'}$, if $v \in I$, then $v \in B_{r'} = N_G[B_r] \setminus D_{r'}$, and as both r and s, and r' and s' agree on I, $v \in N_G[B_s] \setminus D_{s'}$. Otherwise if $v \notin I$, then $v \in B_s$ as s and s' give the same label to every vertex not in I. Furthermore $v \notin D_{s'}$ as $v \in B_{s'}$, therefore $v \in N_G[B_s] \setminus D_{s'}$ and so $B_{s'} \subseteq N_G[B_s] \setminus D_{s'}$. Conversely, consider any vertex $v \in N_G[B_s] \setminus D_{s'}$ if $v \in I$ then $v \in N_G[B_r] \setminus D_{r'} = B_{r'}$, and then $v \in B_{s'}$ as $B_{r'}$ and $B_{s'}$ agree on I. Otherwise, if $v \notin I$, then $v \in B_s \setminus D_{S'} \subseteq B_s$. Then $v \in B_{s'}$ as s and s' give the same label to every vertex not in I, so $B_{s'} = N_G[B_s] \setminus D_{s'}$.

Now consider any vertex $v \in U_{s'}$. If $v \in I$ then $v \in U_{r'} = U_r \setminus (B_{r'} \cup D_{r'})$, and $v \in U_s \setminus (B_{s'} \cup D_{s'})$. Otherwise if $v \notin I$ then $v \in U_s$. Therefore $U_{s'} \subseteq U_s \setminus (B_{s'} \cup D_{s'})$. Conversely consider any vertex $v \in U_s \setminus (B_{s'} \cup D_{s'})$. If $v \in I$ then $v \in U_r \setminus (B_{r'} \cup D_{r'}) = U_{r'}$, and then $v \in U_{s'}$. Otherwise if $v \notin I$ then $v \in U_{s'}$ as $v \in U_s$, and s and s' give the same label to every vertex not in I. Then $U_{s'} = U_s \setminus (B_{s'} \cup D_{s'})$.

Finally $D_{s'} \setminus D_s \subseteq I$, and $B_{s'} \setminus B_s \subseteq I$ as s and s' give the same label to every vertex not in I. Therefore $D_{s'} \setminus D_s = D_{r'} \setminus D_r$, and $B_{s'} \setminus B_s = B_{r'} \setminus B_r$, and so $h_{s'} = h_{r'} = h_r + |B_{r'} \setminus B_r| = h_s + |B_{s'} \setminus B_s|$, and $b_{s'} = b_{r'} = b_r - |B_{r'} \setminus B_r| + 1 = b_s - |B_{s'} \setminus B_s| + 1$. So $\mathbf{Tr}(s, s', G) = \text{true}$ as required.

Finally, see that the acceptance routine checks only the value of a counter variable, and therefore if it returns true when given a state s, then it will return true when given any state agreeing with s on any vertex set. □

Then, as Algorithm 6 runs in time $O(n)$ by checking the label on each vertex in turn, and we use 2 counter variable of size at most n and 3 labels, we finally obtain the following corollary from Theorem 43.

Corollary 6. *Temporal Firefighter Reserve can be solved in time $O(\Lambda n n^4 3^{2\omega}) = O(\Lambda n^5 3^{2\omega})$, where Λ is the lifetime of the input temporal graph, n the number of vertices, and ω the vertex-interval-membership-width.*

5.2.3 Temporal Graph Burning

We now show that Temporal Graph Burning is in FPT with respect to the parameter of vertex-interval-membership-width. As with Temporal Firefighter, we actually use the equivalent problem of Temporal Graph Burning Reserve, where it is not required to place exactly one fire at each timestep, and rather a budget increases by one

each timestep, and at any point the budget can be depleted by placing up to the number of fires given by its value. Similarly to Temporal Firefighter Reserve, this allows us to assume that any fire placed on timestep t is placed at a vertex in F_t.

Temporal Graph Burning Reserve
Input: A temporal graph $\mathcal{G}$, and integer k.
Output: Does there exist a strategy S, that is a k-length sequence of sets of vertices such that if fires are placed at all the vertices in each set in turn, every vertex of the graph is burning by the end of timestep k.

We use $(X, 3)$-states, where the label set $X = \{B, U\}$ contains a label for burning, and unburning vertices respectively, and the counter vector contains an integer h which counts the total number of burnt vertices, an integer b which counts the current available budget, and an integer i which counts the total number of timesteps until the graph is burnt. We will define our transition routine and initial states such that each state produced by repeated applications of the transition routine corresponds to the existence of a stratagy for Temporal Graph Burning Reserve where there are b burning vertices and these are all given label B, and all unburning vertices are given label U. Note that both b and h can be at most the number of vertices in the input graph.

Given an instance $(\mathcal{G}, k)$, we use one initial state, where all vertices are labelled U, and b is equal to zero, h is equal to one, and i is equal to 0. We now give the transition algorithm for this process, and thus prove it is temporally local.

Algorithm 7 Temporal Graph Burning Reserve Transition

Input: A static graph G and states $(l_1, (b_1, h_1, i_1))$ and $(l_2, (b_2, h_2, i_1))$ for $V(\mathcal{G})$.

Output: Returns true when $(l_2, (b_2, h_2, i_2))$ corresponds to the fire spreading after an in budget number of fires is placed following the state corresponding to $(l_1, (b_1, h_1, i_1))$ and false otherwise.

1: Let U_1 and U_2 be the set of vertices labelled U by the l_1 and l_2 rspectively, and equivalently for B_1 and B_2.
2: Let I be the non-isolated vertices of G
3: **if** $\exists A \subseteq I \cap (U_1 \setminus N_G[B_1])$ such that $B_2 = N_G[B_1] \cup A$ and $b_2 = b_1 - |A| + 1$ and $b_2 \geq 1$ and $h_2 = h_1 + |B_2 \setminus B_1|$ and $U_2 = U_1 \setminus B_2$ and i_2 is equal to i_1 if $h_2 = |V(G)|$ and equal to $i_1 + 1$ otherwise **then**
4: **return** True
5: **else**
6: **return** False
7: **end if**

Given a state $(l, (b, h, i))$ and an instance of Temporal Graph Burning Reserve, that is a temporal graph $\mathcal{G}$ along with an integer k, the acceptance routine returns true if and only if $i \leq k$.

We now show that there exists a correspondence between sequences of moves for Tem-

poral Graph Burning Reserve that only place fires at active vertices on $\mathcal{G}$ and sequences of states beginning with initial states and related by the transition routine. We say that a sequence $s_0, ..., s_t$ of states *corresponds* to a t-length sequence S of moves for Temporal Graph Burning Reserve that only places fires at active vertices on $\mathcal{G}$ if and only if:

1. s_0 is an initial state,
2. $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_i) = \text{true}$ for every $1 \leq i \leq t$,
3. the vertices given label B by s_t are exactly the burning vertices after the moves from S are played,
4. the vertices given label U by s_t are exactly the vertices that are unburning after the moves from S are played,
5. the value of h given by s_t is equal to the number of burning vertices after the moves from S are played,
6. the value of b given by s_t is equal to the budget available at the end of timestep t after the moves from S are played,
7. the value of i given by s_t is equal to the number of moves in S after which the graph is fully burnt, and equal to t if the moves from S do not burn the graph.

Lemma 48. *For any timestep t, there exists a t-length sequence of moves for Temporal Graph Burning Reserve that only places fires at active vertices on $\mathcal{G}$, if and only if there exists a corresponding sequence of states $s_0, ...s_t$.*

Proof. We proceed by induction on the timestep t. When $t = 0$, any sequence of moves will not have placed any fires, so every vertex is unburnt, the budget available on the next timestep is 1, and the graph is not burnt. The only initial state labels all vertices with U, sets $h = 0$, $b = 1$, and $i = 0$, as required.

Now assume that there exists a sequence $s_0, ..., s_t$ of states such that s_0 is the initial state, and $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_i) = \text{true}$ for every $1 \leq i \leq t$. By induction there exists a sequence of moves S_{t-1} corresponding to the sequence $s_0, ..., s_{t-1}$. Now as $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \text{true}$, there exists a set A of vertices with active incident edges on timestep t such that all of the vertices in A are unburnt after the moves in S_{t-1} are played, none of the vertices in A are adjacent on timestep t to a vertex burning after the moves in S_{t-1} are played, and $|A|$ does not exceed the budget available at the end of timestep $t-1$. Thus, placing a fire at every vertex in A is a valid move on timestep t after the moves in S_{t-1} are played. Let S be the sequence of moves that plays the moves in S_{t-1} for the first $t-1$ timesteps, and then places fires at the vertices of A on timestep t. The burnt vertices after the moves in

S are played are then the vertices in the closed temporal neighbourhood of the vertices that are burning after the moves in S_{t-1} are played, along with the vertices in A. The budget available at the end of timestep t is equal to $b_{t-1} - |A| + 1$ where b_{t-1} is the budget available at the end of timestep $t+1$, and the graph was burnt in t timesteps if not all of the vertices of the graph were burning after the moves in S_{t-1} were played, but all the vertices of the graph are burning after the moves in S are played.

Otherwise assume there exists a t-length sequence S of moves for Temporal Graph Burning and see that by induction there exists a sequence of states $s_0, ..., s_{t-1}$ corresponding to the first $t-1$ moves of this sequence. Let A be the set of vertices at which fires are placed by S on its tth move, and consider the state s_t that gives label B to all vertices labeled B by s_{t-1}, all vertices in A, and all vertices adjacent on timestep t to a vertex labeled B by s_{t-1}, and label U to all other vertices. Furthermore let the value of h given by s_t be equal to the number of burning vertices after the moves from S are played, the value of b given by s_t be the budget at the end of timestep t, and the value of i be the number of moves of S taken to fully burn the graph. Then as A is a set of vertices unburning after the fire spreads on timestep t with a cardinality that does not exceed the budget available and the end of timestep $t-1$, we have that $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \text{true}$, as required. □

Theorem 49. *Temporal Graph Burning Reserve is locally temporally uniform.*

Proof. The initial state gives label U to all vertices as required.

Consider any graph G and pair of states s and s' such that $\mathbf{Tr}(s, s', G) = \text{true}$. Let U_s and B_s be the sets of vertices labeled U and B by s respectively, and equivalently for $U_{s'}$ and $B_{s'}$. Then consider any vertex v isolated in G. If $v \in B_s$, then $v \in N_G[B_s] \cup A$ for any set A, and therefore $v \in B_{s'}$. If $v \in B_{s'}$, then $v \in B_s$ as v is isolated in G. Now as U_s and B_s partition the vertices of G, as do $U_{s'}$ and $B_{s'}$, we have that $v \in U_{s'}$ if and only if $v \in U_s$. Therefore s and s' give the same label to any isolated vertex in G.

Consider any graph G and a quadruple of states r, r', s, and s' for $V(G)$, such that r and s agree on the non-isolated vertices in G, r' and s' agree on the non-isolated vertices in G, and the pairs s, s' and r, r' both give the same label to every isolated vertex not in G. Assume without loss of generality that $\mathbf{Tr}(r, r', G) = \text{true}$.

Let B_r, $B_{r'}$, B_s, and $B_{s'}$ be the sets of vertices labeled B by r, r', s and s' respectively, and equivalently for U_r, $U_{r'}$, U_s, and $U_{s'}$. Also, let I be the non-isolated vertices of G, and A the set of vertices that must exist obeying the conditions on line 3 of Algorithm 7, as $\mathbf{Tr}(r, r', G) = \text{true}$.

Consider any vertex $v \in B_{s'}$. If $v \in I$ then $v \in B_{r'} = N_G[B_r] \cup A$ as s' and r' agree on I. Then $v \in N_G[B_s] \cup A$, as $I \subseteq I$, and s and r also agree on I. Otherwise if $v \notin I$ then $v \in B_s \subseteq N_G[B_s] \cup A$, as s and s' give the same label to every vertex not in I. Therefore $B_{s'} \subseteq N_G[B_S] \cup A$. Now consider any vertex $v \in N_G[B_s] \cup A$. If $v \in I$ then $v \in N_G[B_r] \cup A$,

as s and r agree on I, and $A \subseteq I$. Then $N_G[B_r] \cup A = B_{r'}$, and $v \in B_{s'}$ as s' and r' agree on I. Otherwise if $v \notin I$, then $v \in B_s$, as $N_G[B_s] \setminus B_s \subseteq I$ and $A \subseteq I$. Then $v \in B_{s'}$ as s and s' give the same label to any vertex not in I. Therefore $N_G[B_s] \cup A \subseteq B_{s'}$, and $B_{s'} = N_G[B_s] \cup A$.

Furthermore, consider any vertex $v \in U_{s'}$. If $v \in I$ then $v \in U_{r'} = U_r \setminus B_{r'}$ as r' and s' agree on I. Then $v \in U_s \setminus B_{s'}$, as s and r also agree on I. Otherwise if $v \notin I$ then $v \in U_s$ as s and s' give the same label to every vertex not in I. Furthermore $v \notin B_{s'}$, and so $v \in U_s \setminus B_{s'}$. Therefore $U_{s'} \subseteq U_s \setminus B_{s'}$. Now consider any vertex $v \in U_s \setminus B_{s'}$, if $v \in I$ then $v \in U_r \setminus B_{r'} = U_{r'} = U_{s'}$, otherwise if $v \notin I$ then $v \in U_{s'}$. Therefore $U_s \setminus B_{s'} \subseteq U_{s'}$ and $U_{s'} = U_s \setminus B_{s'}$.

Finally, see that $B_{s'} \setminus B_s = (B_{s'} \cap I) \setminus (B_s \cap I)$, as s' and s give the same label to every vertex not in I. Equivalently see that $B_{r'} \setminus B_r = (B_{r'} \cap I) \setminus (B_r \cap I)$, and therefore $B_{s'} \setminus B_s = B_{r'} \setminus B_r$ as s' and r' agree on I, as do s and r.

Now $h_{s'} = h_{r'} = h_r + |B_{r'} \setminus B_r| = h_s + |B_{s'} \setminus B_s|$, $b_{s'} = b_{r'} = b_r - |A| + 1 = b_s - |A| + 1$, and $i_{s'} = i_{r'}$, and $i_{r'} = i_r$ if $h_r = |V(G)|$ and $i_r + 1$ otherwise, and if $h_r = |V(G)|$ then $h_s = |V(G)|$, and $i_r = i_s$, so $i_{s'} = i_s$ if $h_s = |V(G)|$ and $i_s + 1$ otherwise.

Finally, see that the acceptance routine checks only the value of a counter variable, and therefore if it returns true when given a state s, then it will return true when given any state agreeing with s on any vertex set. □

Then, as Algorithm 7 runs in time $O(n)$ by checking the label on each vertex in turn, and we use 3 counter variable of size at most n and 2 labels, we finally obtain the following corollary from Theorem 43.

Corollary 7. *Temporal Graph Burning Reserve can be solved in time* $O(\Lambda n n^6 2^{2\omega}) = O(\Lambda n^7 2^{2\omega})$*, where* Λ *is the lifetime of the input temporal graph,* n *the number of vertices, and* ω *the vertex-interval-membership-width.*

5.2.4 Temporal Dominating Set

We now consider the problem of Temporal Dominating Set. This problem asks if it is possible to find a set D of size k or less consisting of vertex-time pairs (vertex appearances) such that every vertex v is *covered*, that is it either appears in a pair in D, or there exists a $(u, t) \in D$ such that v is adjacent to u on timestep t. We assume that the underlying graph $\mathcal{G}\downarrow$ contains no isolated vertices, and that for any vertex appearance $(u, t) \in D$, u has an incident edge active on timestep t.

Temporal Dominating Set
Input: A temporal graph $\mathcal{G}$ and an integer k.
Output: Does there exist a set D of vertex-appearances such that $\|D\| \leq k$ and the appearances in D cover every vertex in $\mathcal{G}$?

We use $(X, 2)$-states, where the label set $X = \{U, C\}$ contains a label for uncovered and covered vertices respectively, and the counter vector contains an integer c which counts the number of covered vertices, and an integer d which counts the size of the dominating set D. We will define our transition routine and initial states such that each state produced by repeated applications of the transition routine corresponds to the existence of a set D of size d of vertex appearances which covers c vertices and these are all given label C, and all uncovered vertices are given label U. Note that both c and d can be at most the number of vertices in the input graph.

We use one initial state, in which every vertex is labelled U, and c and d are both equal to 0. We now give the transition algorithm for this process, and thus prove it is temporally local.

Algorithm 8 Temporal Dominating Set Transition

Input: A static graph G and states $(l_1, (c_1, d_1))$ and $(l_2, (c_2, d_2))$ for $V(\mathcal{G})$.
Output: Returns true when $(l_2, (c_2, d_2))$ corresponds to adding $d_2 - d_1$ vertex appearances to the vertex appearances corresponding to $(l_1, (c_1, d_1))$ and false otherwise.

1: Let U_1 and U_2 be the set of vertices labelled U by l_1 and l_2 respectively, and equivalently for C_1 and C_2.
2: Let I be the non-isolated vertices of G
3: **if** $\exists D \subseteq I$ such that $d_2 = d_1 + |D|$ and $C_2 = C_1 \cup N_G[D]$ and $c_2 = c_1 + |C_2 \setminus C_1|$ **then**
4: **return** True
5: **else**
6: **return** False
7: **end if**

Given a state $(l, (c, d))$ and an instance of Temporal Dominating Set, that is a temporal graph $\mathcal{G}$ along with an integer k, the acceptance routine returns true if and only if $d \leq k$ and $c = |V(\mathcal{G})|$.

We now show that there exists a correspondence between sets of vertex appearances and sequences of states beginning with initial states and related by the transition routine. We say that a sequence $s_0, ..., s_t$ of states *corresponds* to a set D of vertex appearances up to timestep t from $\mathcal{G}$ if and only if:

1. s_0 is an initial state,
2. $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_i) =$ true for every $1 \leq i \leq t$,
3. the vertices given label C by s_t are exactly those covered by D,
4. D contains d vertices,
5. D covers c vertices.

Lemma 50. *For any timestep t, there exists a set of vertex appearances up to timestep t from $\mathcal{G}$, if and only if there exists a corresponding sequence of states $s_0, ...s_t$.*

Proof. We proceed by induction on the timestep t. When $t = 0$ there are 0 vertices that appear on or before timestep 0, and so 0 vertices are covered. Our initial state gives label U to every vertex in the graph, and sets d and c to 0, as required.

Now assume that there exists some set D consisting of vertex appearances on or before timestep t. By induction there exists a sequence of states $s_0, ..., s_{t-1}$ corresponding to $D_{t-1} = \{(v, i) \in D : i \leq t-1\}$. Now let s_t be the state giving all vertices covered by D label C, all other vertices label U, and with $d = |D|$, and c the number of vertices covered by D. Let $D_t = \{v : (v, t) \in D\}$ be the set of vertices appearing on timestep t in D, and see that D_t is a subset of the non-isolated vertices of $\mathcal{G}_t$. Then $d_2 = |D| = |D_{t-1}| + |D_t| = d_1 + |D_t|$. The vertices labeled C by s_t are those covered by D, so those covered by D_{t-1}, which are the vertices labeled C by s_{t-1}, and any vertex in the closed temporal neighourhood of D_t on timestep t. Finally, c_2 is the number of vertices covered by D, that is the number of vertices covered by D up to timestep $t-1$, so c_1, plus the number of vertices covered by D on timestep t and not before, so $|C_2 \setminus C_1|$. Therefore $\mathbf{Tr}(s_{t-1}, s_t, \mathcal{G}_t) = \text{true}$, and $s_0, ..., s_t$ corresponds to D as required.

Otherwise assume that there exists some sequence of states $s_0, ..., s_t$ such that $\mathbf{Tr}(s_{i-1}, s_i, \mathcal{G}_i) = \text{true}$ for every $1 \leq i \leq t$, and s_0 is the initial state. By induction there exists a set D_{t-1} of vertex appearances corresponding to the sequence $s_0, ..., s_{t-1}$. Let A be a set of non-isolated vertices in $\mathcal{G}_t$ such that $d_2 = d_1 + |A|$, $C_2 = C_1 \cup N_{\mathcal{G}_t}[A]$, and $c_2 = c_1 + |C_2 \setminus C_1|$, seeing that such a set exists as Algorithm 8 returns true when given s_{t-1}, s_t, and $\mathcal{G}_t$ as input. Now let D_t be the set of vertex appearances $\{(v, t) : v \in A\}$.

See that $|D_{t-1} \cup D_t| = d_1 + |D_t| = d_2$. Also, the vertices covered by $D_{t-1} \cup D_t$ are those covered by D_{t-1} along with those in the closed temporal neighourhood of A on timestep t, so $C_1 \cup N_{\mathcal{G}_t}[A] = C_2$. The number of vertices covered by $D_{t-1} \cup D_t$ is then the number of vertices covered by D_{t-1}, so c_1, plus the number of vertices covered by D_t but not D_{t-1}, so $|C_2 \setminus C_1|$. We have that $c_1 + |C_2 \setminus C_1|$, and thus $D_{t-1} \cup D_t$ corresponds to $s_0, ..., s_t$. □

Theorem 51. *Temporal Dominating Set is locally temporally uniform.*

Proof. The initial state gives label U to all vertices as required.

Consider any graph G and pair of states s and s' such that $\mathbf{Tr}(s, s', G) = \text{true}$. Let U_s and C_s be the vertices labeled U and C by s, and equivalently for $U_{s'}$ and $C_{s'}$ and s'. Consider any vertex v isolated in G. If $v \in C_s \subseteq C_s \cup N_G[D]$ for any set D then $v \in C_{s'}$. If $v \in C_{s'}$ then $v \in C_s$, as $C_{s'} = C_s \cup N_G[D]$, for some set D of non-isolated vertices of G. Now as U_s and C_s partition the vertices of G, as do $U_{s'}$ and $C_{s'}$, we have that $v \in U_{s'}$ if and only if $v \in U_s$. Therefore s and s' give the same label to any isolated vertex in G.

Consider any graph G and a quadruple of states r, r', s, and s' for $V(G)$, such that r and s agree on the non-isolated vertices in G, r' and s' agree on the non-isolated vertices in G, and the pairs s, s' and r, r' both give the same label to every isolated vertex not in G. Assume without loss of generality that $\mathbf{Tr}(r, r', G) = \text{true}$.

Let $C_{s'}$, C_s, $C_{r'}$, and C_r be the vertices given label C by s', s, r', and r respectively. Also let I be the non-isolated vertices of G, and A a set of vertices such that $d_{r'} = d_r + |A|$, $C_{r'} = C_r \cup N_G[A]$, and $c_{r'} = c_r + |C_{r'} \setminus C_r$, noting that such a set must exist as $\mathbf{Tr}(r, r', G) =$ true.

Consider any vertex $v \in C_{s'}$. If $v \in I$ then $v \in C_{r'} = C_r \cup N_G[A]$ as s' and r' agree on I. Then $v \in C_s \cup N_G[A]$ as s and r agree on I. Otherwise if $v \notin I$ then $v \in C_s$ as s and s' give the same label to every vertex not in I. Therefore $C_{s'} \subseteq C_s \cup N_G[A]$. Consider now any vertex $v \in C_s \cup N_G[A]$, if $v \in I$ then $v \in C_r \cup N_G[A] = C_{r'}$ and then $v \in C_{s'}$. Otherwise if $v \notin I$ then $v \in C_s$, as $N_G[A] \subseteq I$. Then $v \in C_{s'}$ as s and s' give the same label to any vertex not in I. Therefore $C_s \cup N_G[A] \subseteq C_{s'}$ and $C_{s'} = C_s \cup N_G[A]$. Also, $d_{s'} = d_{r'} = d_r + |A| = d_s + |A|$.

Finally, see that $C_{s'} \setminus C_s = (C_{s'} \cap I) \setminus (C_s \cap I)$, as s' and s give the same label to every vertex not in I. Equivalently see that $C_{r'} \setminus C_r = (C_{r'} \cap I) \setminus (C_r \cap I)$, and therefore $C_{s'} \setminus C_s = C_{r'} \setminus C_r$ as s' and r' agree on I, as do s and r. Therefore $c_{s'} = c_{r'} = c_r + |C_{r'} \setminus C_r| = c_s + |C_{s'} \setminus C_s|$, and we have that $\mathbf{Tr}(s, s', G) =$ true as required.

Finally, see that the acceptance routine checks only the value of a counter variable, and therefore if it returns true when given a state s, then it will return true when given any state agreeing with s on any vertex set. □

Then, as Algorithm 8 runs in time $O(n)$ by checking the label on each vertex in turn, and we use 2 counter variable of size at most $n\Lambda$ and 2 labels, we finally obtain the following corollary from Theorem 43.

Corollary 8. *Temporal Firefighter Reserve can be solved in time* $O(\Lambda n(n\Lambda)^4 2^{2\omega}) = O(\Lambda^5 n^5 2^{2\omega})$*, where* Λ *is the lifetime of the input temporal graph,* n *the number of vertices, and* ω *the vertex-interval-membership-width.*

5.3 Conclusions

In this chapter we showed that Temporal Firefighter, Temporal Graph Burning, Temporal Hamiltonian Path, and Temporal Dominating Set are in **FPT** with respect to the parameter of vertex-interval-membership-width. These results were obtained through the use of a meta-theorem, in which we presented a fixed parameter tractable algorithm for solving any problem expressible in terms of a *locally temporally uniform* pair of transition and acceptance routines. By expressing problems in this way we may avoid the need to construct an individual algorithm for a problem that we wish to show is in **FPT** with respect to vertex-interval-membership-width. We believe that many more temporal graph problems are expressible in this manner, and therefore that this will prove a useful toolkit for finding temporal problems in **FPT**.

Furthermore, we believe that this toolkit will carry over to the parameter of edge-interval-membership-width. If we label the edges, rather than the vertices, of a temporal graph, and allow the label on an edge to change only according to the labels on edges temporally incident to the same vertices, then we will then find analogous algorithmic results to those for the locally temporally uniform problems, except that the runtime bounds will be given in terms of the edge-interval-membership-width. We suggest as future work to define edge-locally temporally uniform problems, and prove an equivalent theorem to Theorem 43. We believe that Temporal Eulerian Walk and Temporal Vertex Cover would be examples of such problems, as these are analogous to Temporal Hamiltonian Path and Temporal Dominating Set respectively, except that a solution consists of a set of edges, rather than vertices.

Chapter 6

Conclusions

The developing field of temporal graphs studies graphs that have been augmented with temporal information, modeling the *when* rather than just the *where* of a connection. This thesis explored two spreading problems on temporal graphs: Temporal Firefighter and Temporal Graph Burning. These problems are natural extensions of Firefighter and Graph Burning respectively to temporal graphs, and usefully model spreading on real world networks, where the topology is often dynamic. Furthermore, both of these problems are **NP**-complete, and we explored various ways to tame this complexity, including using the tools provided by parameterised complexity to obtain fixed-parameter-tractable algorithms. We explored several parameters specific to temporal graphs, and found making use of the temporal structure of the graph in this way fruitful for solving our problems efficiently. We believe the temporal graph parameters we considered to have wide applicability to problems on temporal graphs, beyond those considered in this thesis.

6.1 Contributions and Future Work

In Chapter 2 we defined Temporal Firefighter and Temporal Graph Burning, and proved that they are both **NP**-complete. We also considered the complexity of Temporal Firefighter when the underlying graph is a clique, and found it to remain **NP**-complete in this case. In doing so we showed that Temporal Firefighter is **NP**-complete for almost every class of underlying graph where Firefighter is known to be in **P**. This demonstrates an informative principle: the underlying structure of a temporal graph determines very little about its behaviour. We did not consider restricting the underlying graph class for Graph Burning, and suggest this for future work. We believe that a similar proof technique to that used for the hardness of Temporal Firefighter on cliques could be used to show that Temporal Graph Burning is **NP**-complete whenever the underlying graph class contains long paths, as Graph Burning is **NP**-complete

on path forests [10].

In Chapter 3 we considered the complexity of Temporal Firefighter when parameterised by vertex-interval-membership-width and edge-interval-membership-width: parameters which respectively measure the number of *relevant* vertices and edges on each timestep [20]. We found that Temporal Firefighter is in **FPT** when parametersed by the vertex-interval-membership-width but remains **NP**-complete when even when the edge-interval-membership-width is at most 2. The reduction used for the hardness result produces an instance of Temporal Firefighter with a root of large degree, and we suggest investigating the complexity of Temporal Firefighter when parameterised by the edge-interval-membership-width and the maximum degree of the underlying graph as future work. Furthermore, although we resolve the complexity of Temporal Graph Burning when parameterised by the vertex-interval-membership-width in Chapter 5, we leave its complexity when parameterised by the edge-interval-membership-width unresolved and suggest this as future work. We expect that like Temporal Firefighter, Temporal Graph Burning will remain **NP**-complete even when the edge-interval-membership-width is small.

In Chapter 4 we gave fixed-parameter-tractable algorithms for both Temporal Firefighter and Temporal Graph Burning when parameterised by temporal neighbourhood diversity. We also considered the parameter of temporal modular width, and found that Temporal Graph Burning is **NP**-complete when the temporal modular width is small. This differs from the static case; Graph Burning is in **FPT** when parameterised by modular width [56]. We did not determine the complexity of Temporal Firefighter when parameterised by temporal modular width, and suggest this as further work. Both temporal neighbourhood diversity and temporal modular width are temporal analogues of static graph parameters, and as is the case with the static parameters, they can be small even when the temporal graph is dense. This contrasts to many of the other temporal graph parameters considered thus far in the literature, which are only small when the temporal graph is sparse.

Finally, in Chapter 5 we considered how the parameter of vertex-interval-membership-width might be more generally applied to other problems. We defined a class of *locally temporally uniform* problems, and proved a meta-theorem showing that all locally temporally uniform problems are in **FPT** when parameterised by the vertex-interval-membership-width. Our definition of locally temporally uniform problems should capture many natural spreading problems, and we show that Temporal Graph Burning is locally temporally uniform, thus obtaining that it is in **FPT** when parameterised by vertex-interval-membership-width. We also show that Temporal Firefighter is locally temporally uniform, recreating the result from Chapter 3. Furthermore, we show that Temporal Hamiltonian Path and Temporal Dominating Set, both prob-

lems unrelated to spreading, are locally temporally uniform, and therefore in **FPT** when parameterised by vertex-interval-membership-width. This demonstrates the wide applicability of our meta-theorem, and we suggest applying the theorem to other temporal graph problems as future work. Furthermore, we believe that an equivalent result can be shown for edge-interval-membership-width, and also leave this for future work.

This thesis analysed the temporal graph problems of Temporal Firefighter and Graph Burning, with a focus on complexity theory. We proved hardness results, showing that our problems are **NP**-complete, and therefore hard to solve in the worst case. To achieve tractability, we developed fixed-parameter-tractable algorithms, meaning that feasible runtimes can be achieved if a parameter other than the input size is small. As future work it is worth implementing these algorithms, and executing them on real datasets to see how well they perform in practice, and how useful these parameters are in a non-theoretical setting. It is also worth noting that although we showed that our problems are hard to solve in the worst case, solvers exist for **NP**-complete problems, such as integer programming and boolean satisfiability, that can quickly solve a wide range of instances. We also suggest as future work to formulate our problems so that they can be input into one of these solvers, and see what problem inputs are solvable in practice.

Overall, we find that temporal graph parameters are a useful tool for building efficient algorithms for temporal graph problems, and show that several problems admit fixed-parameter-tractable algorithms when parameterised by the temporal graph parameters considered in this thesis. These parameters should be applicable to problems beyond those we explore, and determining the complexity of other problems when parameterised by the temporal parameters discussed in this thesis is an avenue for future research. To aid this, we proved a meta-theorem that can be applied generally to a wide range of temporal graph problems. As future work, parameterised complexity results for temporal graphs could be further generalised, by proving a result of the kind provided by Courcelle, where any problem expressible in a certain logic is shown to yield fixed-parameter-tractable algorithms for a given parameter [24].

Bibliography

[1] David Adjiashvili, Andrea Baggio, and Rico Zenklusen. "Firefighting on trees beyond integrality gaps". In: *ACM Transactions on Algorithms (TALG)* 15.2 (2018), pp. 1–33.

[2] Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Viktor Zamaraev. "Temporal vertex cover with a sliding time window". In: *J. Comput. Syst. Sci.* 107 (2020), pp. 108–123. DOI: 10.1016/j.jcss.2019.08.002. URL: https://doi.org/10.1016/j.jcss.2019.08.002.

[3] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN: 978-0-521-42426-4.

[4] David Arthur, Raphaël Clifford, Markus Jalsenius, Ashley Montanaro, and Benjamin Sach. "The Complexity of Flood Filling Games". In: *Fun with Algorithms, 5th International Conference, FUN 2010, Ischia, Italy, June 2-4, 2010. Proceedings*. Ed. by Paolo Boldi and Luisa Gargano. Vol. 6099. Lecture Notes in Computer Science. Springer, 2010, pp. 307–318. DOI: 10.1007/978-3-642-13122-6_30. URL: https://doi.org/10.1007/978-3-642-13122-6_30.

[5] Giorgio Ausiello, Pierluigi Crescenzi, and Marco Protasi. "Approximate solution of NP optimization problems". In: *Theoretical Computer Science* 150.1 (1995), pp. 1–55.

[6] Robert Axelrod. "The dissemination of culture: A model with local convergence and global polarization". In: *Journal of conflict resolution* 41.2 (1997), pp. 203–226.

[7] Cristina Bazgan, Morgan Chopin, Marek Cygan, Michael R. Fellows, Fedor V. Fomin, and Erik Jan van Leeuwen. "Parameterized complexity of firefighting". In: *J. Comput. Syst. Sci.* 80.7 (2014), pp. 1285–1297. DOI: 10.1016/J.JCSS.2014.03.001. URL: https://doi.org/10.1016/j.jcss.2014.03.001.

[8] Piotr Berman, Marek Karpinski, and Alex D. Scott. "Approximation Hardness of Short Symmetric Instances of MAX-3SAT". In: *Electron. Colloquium Comput. Complex.* TR03-049 (2003). ECCC: TR03-049. URL: https://eccc.weizmann.ac.il/eccc-reports/2003/TR03-049/index.html.

[9] Stéphane Bessy, Anthony Bonato, Jeannette C. M. Janssen, Dieter Rautenbach, and Elham Roshanbin. "Bounds on the burning number". In: *Discret. Appl. Math.* 235 (2018), pp. 16–22. DOI: 10.1016/J.DAM.2017.09.012. URL: https://doi.org/10.1016/j.dam.2017.09.012.

[10] Stéphane Bessy, Anthony Bonato, Jeannette C. M. Janssen, Dieter Rautenbach, and Elham Roshanbin. "Burning a graph is hard". In: *Discret. Appl. Math.* 232 (2017), pp. 73–87. DOI: 10.1016/j.dam.2017.07.016. URL: https://doi.org/10.1016/j.dam.2017.07.016.

[11] Cristiano Bocci, Chiara Capresi, Kitty Meeks, and John Sylvester. "A New Temporal Interpretation of Cluster Editing". In: *Combinatorial Algorithms - 33rd International Workshop, IWOCA 2022, Trier, Germany, June 7-9, 2022, Proceedings*. Ed. by Cristina Bazgan and Henning Fernau. Vol. 13270. Lecture Notes in Computer Science. Springer, 2022, pp. 214–227. DOI: 10.1007/978-3-031-06678-8_16. URL: https://doi.org/10.1007/978-3-031-06678-8_16.

[12] Hans L. Bodlaender. "Treewidth: Characterizations, Applications, and Computations". In: *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*. Ed. by Fedor V. Fomin. Vol. 4271. Lecture Notes in Computer Science. Springer, 2006, pp. 1–14. DOI: 10.1007/11917496_1. URL: https://doi.org/10.1007/11917496_1.

[13] Anthony Bonato, Jeannette C. M. Janssen, and Elham Roshanbin. "Burning a Graph as a Model of Social Contagion". In: *Algorithms and Models for the Web Graph - 11th International Workshop, WAW 2014, Beijing, China, December 17-18, 2014, Proceedings*. Ed. by Anthony Bonato, Fan Chung Graham, and Pawel Pralat. Vol. 8882. Lecture Notes in Computer Science. Springer, 2014, pp. 13–22. DOI: 10.1007/978-3-319-13123-8_2. URL: https://doi.org/10.1007/978-3-319-13123-8_2.

[14] Anthony Bonato and Shahin Kamali. "An improved bound on the burning number of graphs". In: *arXiv preprint arXiv:2110.01087* (2021).

[15] Anthony Bonato and Shahin Kamali. "Approximation Algorithms for Graph Burning". In: *Theory and Applications of Models of Computation*. Ed. by T.V. Gopal and Junzo Watada. Cham: Springer International Publishing, 2019, pp. 74–92. ISBN: 978-3-030-14812-6.

[16] Anthony Bonato and Thomas Lidbetter. "Bounds on the burning numbers of spiders and path-forests". In: *Theor. Comput. Sci.* 794 (2019), pp. 12–19. DOI: 10.1016/J.TCS.2018.05.035. URL: https://doi.org/10.1016/j.tcs.2018.05.035.

[17] J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory with Applications*. Macmillan Education UK, 1976. ISBN: 978-1-349-03521-2. DOI: 10.1007/978-1-349-03521-2. URL: https://doi.org/10.1007/978-1-349-03521-2.

[18] Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. "Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks". In: *Int. J. Found. Comput. Sci.* 14.2 (2003), pp. 267–285. DOI: 10.1142/S0129054103001728. URL: https://doi.org/10.1142/S0129054103001728.

[19] Binh-Minh Bui-Xuan, Hugo Hourcade, and Cédric Miachon. "Computing small temporal modules in time logarithmic in history length". In: *Social Network Analysis and Mining* 12.1 (2022), p. 19.

[20] Benjamin Merlin Bumpus and Kitty Meeks. "Edge Exploration of Temporal Graphs". In: *Algorithmica* 85.3 (2023), pp. 688–716. DOI: 10.1007/s00453-022-01018-7. URL: https://doi.org/10.1007/s00453-022-01018-7.

[21] Leizhen Cai, Elad Verbin, and Lin Yang. "Firefighting on Trees: (1-1/e)-Approximation, Fixed Parameter Tractability and a Subexponential Algorithm". In: *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*. Ed. by Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga. Vol. 5369. Lecture Notes in Computer Science. Springer, 2008, pp. 258–269. DOI: 10.1007/978-3-540-92182-0_25. URL: https://doi.org/10.1007/978-3-540-92182-0_25.

[22] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. "Time-varying graphs and dynamic networks". In: *Int. J. Parallel Emergent Distributed Syst.* 27.5 (2012), pp. 387–408. DOI: 10.1080/17445760.2012.668546. URL: https://doi.org/10.1080/17445760.2012.668546.

[23] Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. "Finding Temporal Paths Under Waiting Time Constraints". In: *Algorithmica* 83.9 (2021), pp. 2754–2802. DOI: 10.1007/s00453-021-00831-w. URL: https://doi.org/10.1007/s00453-021-00831-w.

[24] Bruno Courcelle. "The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs". In: *Inf. Comput.* 85.1 (1990), pp. 12–75. DOI: 10.1016/0890-5401(90)90043-H. URL: https://doi.org/10.1016/0890-5401(90)90043-H.

[25] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. "Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width". In: *Theory Comput. Syst.* 33.2 (2000), pp. 125–150. DOI: 10.1007/s002249910009. URL: https://doi.org/10.1007/s002249910009.

[26] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN: 978-3-319-21274-6. DOI: 10.1007/978-3-319-21275-3. URL: https://doi.org/10.1007/978-3-319-21275-3.

[27] Marek Cygan, Fedor V. Fomin, and Erik Jan van Leeuwen. "Parameterized Complexity of Firefighting Revisited". In: *Parameterized and Exact Computation - 6th International Symposium, IPEC 2011, Saarbrücken, Germany, September 6-8, 2011. Revised Selected Papers*. Ed. by Dániel Marx and Peter Rossmanith. Vol. 7112. Lecture Notes in Computer Science. Springer, 2011, pp. 13–26. DOI: 10.1007/978-3-642-28050-4_2. URL: https://doi.org/10.1007/978-3-642-28050-4_2.

[28] Mike Develin and Stephen G Hartke. "Fire containment in grids of dimension three and higher". In: *Discrete Applied Mathematics* 155.17 (2007), pp. 2257–2268.

[29] Jessica Enright, Samuel D. Hand, Laura Larios-Jones, and Kitty Meeks. "Structural Parameters for Dense Temporal Graphs". In: *arXiv e-prints*, arXiv:2404.19453 (Apr. 2024), arXiv:2404.19453. arXiv: 2404.19453 [cs.DM].

[30] Jessica Enright, Kitty Meeks, and Hendrik Molter. "Counting temporal paths". In: *arXiv preprint arXiv:2202.12055* (2022).

[31] Jessica A. Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. "Deleting edges to restrict the size of an epidemic in temporal networks". In: *J. Comput. Syst. Sci.* 119 (2021), pp. 60–77. DOI: 10.1016/j.jcss.2021.01.007. URL: https://doi.org/10.1016/j.jcss.2021.01.007.

[32] Jessica A. Enright, Kitty Meeks, and Fiona Skerman. "Assigning times to minimise reachability in temporal graphs". In: *J. Comput. Syst. Sci.* 115 (2021), pp. 169–186. DOI: 10.1016/j.jcss.2020.08.001. URL: https://doi.org/10.1016/j.jcss.2020.08.001.

[33] Thomas Erlebach and Jakob T. Spooner. "Parameterised temporal exploration problems". In: *J. Comput. Syst. Sci.* 135 (2023), pp. 73–88. DOI: 10.1016/J.JCSS.2023.01.003. URL: https://doi.org/10.1016/j.jcss.2023.01.003.

[34] Stephen Finbow, Andrew D. King, Gary MacGillivray, and Romeo Rizzi. "The firefighter problem for graphs of maximum degree three". In: *Discret. Math.* 307.16 (2007), pp. 2094–2105. DOI: 10.1016/j.disc.2005.12.053. URL: https://doi.org/10.1016/j.disc.2005.12.053.

[35] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. ISBN: 978-3-540-29952-3. DOI: 10.1007/3-540-29953-X. URL: https://doi.org/10.1007/3-540-29953-X.

[36] Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. "Temporal graph classes: A view through temporal separators". In: *Theor. Comput. Sci.* 806 (2020), pp. 197–218. DOI: 10.1016/j.tcs.2019.03.031. URL: https://doi.org/10.1016/j.tcs.2019.03.031.

[37] Fedor V. Fomin, Petr A. Golovach, and Jan Kratochvíl. "On tractability of Cops and Robbers game". In: *Fifth IFIP International Conference On Theoretical Computer Science - TCS 2008, IFIP 20th World Computer Congress, TC 1, Foundations of Computer Science, September 7-10, 2008, Milano, Italy*. Ed. by Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong. Vol. 273. IFIP. Springer, 2008, pp. 171–185. DOI: 10.1007/978-0-387-09680-3_12. URL: https://doi.org/10.1007/978-0-387-09680-3_12.

[38] Fedor V. Fomin, Pinar Heggernes, and Erik Jan van Leeuwen. "The Firefighter problem on graph classes". In: *Theor. Comput. Sci.* 613 (2016), pp. 38–50. DOI: 10.1016/j.tcs.2015.11.024. URL: https://doi.org/10.1016/j.tcs.2015.11.024.

[39] Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. "Parameterized Algorithms for Modular-Width". In: *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*. Ed. by Gregory Z. Gutin and Stefan Szeider. Vol. 8246. Lecture Notes in Computer Science. Springer, 2013, pp. 163–176. DOI: 10.1007/978-3-319-03898-8_15. URL: https://doi.org/10.1007/978-3-319-03898-8_15.

[40] Robert Ganian. "Using Neighborhood Diversity to Solve Hard Problems". In: *CoRR* abs/1201.3091 (2012). arXiv: 1201.3091. URL: http://arxiv.org/abs/1201.3091.

[41] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. "Some Simplified NP-Complete Graph Problems". In: *Theor. Comput. Sci.* 1.3 (1976), pp. 237–267. DOI: 10.1016/0304-3975(76)90059-1. URL: https://doi.org/10.1016/0304-3975(76)90059-1.

[42] Michel Habib and Christophe Paul. "A survey of the algorithmic aspects of modular decomposition". In: *Comput. Sci. Rev.* 4.1 (2010), pp. 41–59. DOI: 10.1016/J.COSREV.2010.01.001. URL: https://doi.org/10.1016/j.cosrev.2010.01.001.

[43] Samuel D. Hand, Jessica A. Enright, and Kitty Meeks. "Brief Announcement: The Temporal Firefighter Problem". In: *1st Symposium on Algorithmic Foundations of Dynamic Networks, SAND 2022, March 28-30, 2022, Virtual Conference*. Ed. by James Aspnes and Othon Michail. Vol. 221. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 22:1–22:3. DOI: 10.4230/LIPICS.SAND.2022.22. URL: https://doi.org/10.4230/LIPIcs.SAND.2022.22.

[44] Samuel D. Hand, Jessica A. Enright, and Kitty Meeks. "Making Life More Confusing for Firefighters". In: *11th International Conference on Fun with Algorithms, FUN 2022, May 30 to June 3, 2022, Island of Favignana, Sicily, Italy*. Ed. by Pierre Fraigniaud and Yushi Uno. Vol. 226. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 15:1–15:15. DOI: 10.4230/LIPICS.FUN.2022.15. URL: https://doi.org/10.4230/LIPIcs.FUN.2022.15.

[45] Stephen G Hartke. "Attempting to narrow the integrality gap for the firefighter problem on trees". In: *DIMACS series in discrete mathematics and theoretical computer science* 70 (2006), p. 225.

[46] Bert Hartnell. "Firefighter! an application of domination". In: *the 24th Manitoba Conference on Combinatorial Mathematics and Computing, University of Minitoba, Winnipeg, Cadada, 1995*. 1995.

[47] Bert Hartnell. "Firefighting on trees: how bad is the greedy algorithm?" In: *Congressus Numerantium* 145 (2000), pp. 187–192.

[48] Michaela Hiller, Eberhard Triesch, and Arie M. C. A. Koster. "On the Burning Number of p-Caterpillars". In: *CoRR* abs/1912.10897 (2019). arXiv: 1912.10897. URL: http://arxiv.org/abs/1912.10897.

[49] Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. "Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs". In: *Soc. Netw. Anal. Min.* 7.1 (2017), 35:1–35:16. DOI: 10.1007/s13278-017-0455-0. URL: https://doi.org/10.1007/s13278-017-0455-0.

[50] Petter Holme and Jari Saramäki. "Temporal Networks". In: *Physics reports* 519.3 (2012), pp. 97–125.

[51] Anjeneya Swami Kare and I. Vinod Reddy. "Parameterized Algorithms for Graph Burning Problem". In: *Combinatorial Algorithms - 30th International Workshop, IWOCA 2019, Pisa, Italy, July 23-25, 2019, Proceedings*. Ed. by Charles J. Colbourn, Roberto Grossi, and Nadia Pisanti. Vol. 11638. Lecture Notes in Computer Science. Springer, 2019, pp. 304–314. DOI: 10.1007/978-3-030-25005-8_25. URL: https://doi.org/10.1007/978-3-030-25005-8_25.

[52] Richard 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*. Ed. by Raymond E. Miller and James W. Thatcher. The IBM Research Symposia Series. Plenum Press, New York, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007/978-1-4684-2001-2_9.

[53] David Kempe, Jon M. Kleinberg, and Amit Kumar. "Connectivity and Inference Problems for Temporal Networks". In: *J. Comput. Syst. Sci.* 64.4 (2002), pp. 820–842. DOI: 10.1006/jcss.2002.1829. URL: https://doi.org/10.1006/jcss.2002.1829.

[54] Andrew D. King and Gary MacGillivray. "The firefighter problem for cubic graphs". In: *Discret. Math.* 310.3 (2010), pp. 614–621. DOI: 10.1016/J.DISC.2009.05.007. URL: https://doi.org/10.1016/j.disc.2009.05.007.

[55] István Z Kiss, Joel C Miller, Péter L Simon, et al. "Mathematics of epidemics on networks". In: *Cham: Springer* 598.2017 (2017), p. 31.

[56] Yasuaki Kobayashi and Yota Otachi. "Parameterized Complexity of Graph Burning". In: *Algorithmica* 84.8 (2022), pp. 2379–2393. DOI: 10.1007/S00453-022-00962-8. URL: https://doi.org/10.1007/s00453-022-00962-8.

[57] David C. Kutner and Laura Larios-Jones. "Temporal Reachability Dominating Sets: Contagion in Temporal Graphs". In: *Algorithmics of Wireless Networks - 19th International Symposium, ALGOWIN 2023, Amsterdam, The Netherlands, September 7-8, 2023, Revised Selected Papers*. Ed. by Konstantinos Georgiou and Evangelos Kranakis. Vol. 14061. Lecture Notes in Computer Science. Springer, 2023, pp. 101–116. DOI: 10.1007/978-3-031-48882-5_8. URL: https://doi.org/10.1007/978-3-031-48882-5_8.

[58] Michael Lampis. "Algorithmic Meta-Theorems for Graphs of Bounded Vertex Cover". In: *CoRR* abs/0910.0582 (2009). arXiv: 0910.0582. URL: http://arxiv.org/abs/0910.0582.

[59] Michael Lampis. "Algorithmic meta-theorems for restrictions of treewidth". In: *Algorithmica* 64 (2012), pp. 19–37.

[60] Max R. Land and Linyuan Lu. "An Upper Bound on the Burning Number of Graphs". In: *Algorithms and Models for the Web Graph - 13th International Workshop, WAW 2016, Montreal, QC, Canada, December 14-15, 2016, Proceedings*. Ed. by Anthony Bonato, Fan Chung Graham, and Pawel Pralat. Vol. 10088. Lecture Notes in Computer Science. 2016, pp. 1–8. DOI: 10.1007/978-3-319-49787-7_1. URL: https://doi.org/10.1007/978-3-319-49787-7_1.

[61] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*. Ed. by Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett. ACM, 2005, pp. 177–187. DOI: 10.1145/1081870.1081893. URL: https://doi.org/10.1145/1081870.1081893.

[62] Huiqing Liu, Xuejiao Hu, and Xiaolan Hu. "Burning number of caterpillars". In: *Discret. Appl. Math.* 284 (2020), pp. 332–340. DOI: 10.1016/J.DAM.2020.03.062. URL: https://doi.org/10.1016/j.dam.2020.03.062.

[63] Gary MacGillivray and Ping Wang. "On the firefighter problem". In: *Journal of Combinatorial Mathematics and Combinatorial Computing* 47 (2003), pp. 83–96.

[64] George B. Mertzios, Othon Michail, and Paul G. Spirakis. "Temporal Network Optimization Subject to Connectivity Constraints". In: *Algorithmica* 81.4 (2019), pp. 1416–1449. DOI: 10.1007/s00453-018-0478-6. URL: https://doi.org/10.1007/s00453-018-0478-6.

[65] George B. Mertzios, Hendrik Molter, Rolf Niedermeier, Viktor Zamaraev, and Philipp Zschoche. "Computing maximum matchings in temporal graphs". In: *J. Comput. Syst. Sci.* 137 (2023), pp. 1–19. DOI: 10.1016/j.jcss.2023.04.005. URL: https://doi.org/10.1016/j.jcss.2023.04.005.

[66] George B. Mertzios, Hendrik Molter, and Viktor Zamaraev. "Sliding window temporal graph coloring". In: *J. Comput. Syst. Sci.* 120 (2021), pp. 97–115. DOI: 10.1016/j.jcss.2021.03.005. URL: https://doi.org/10.1016/j.jcss.2021.03.005.

[67] SA Moeller and P Wang. "Fire control on graphs". In: *Journal of Combinatorial Mathematics and Combinatorial Computing* 41 (2002), pp. 19–34.

[68] Maziar Nekovee, Yamir Moreno, Ginestra Bianconi, and Matteo Marsili. "Theory of rumour spreading in complex social networks". In: *Physica A: Statistical Mechanics and its Applications* 374.1 (2007), pp. 457–470.

[69] Sergey Norin and Jérémie Turcotte. "The burning number conjecture holds asymptotically". In: *Journal of Combinatorial Theory, Series B* 168 (2024), pp. 208–235. ISSN: 0095-8956. DOI: https://doi.org/10.1016/j.jctb.2024.05.003. URL: https://www.sciencedirect.com/science/article/pii/S009589562400042X.

[70] Richard J. Nowakowski and Peter Winkler. "Vertex-to-vertex pursuit in a graph". In: *Discret. Math.* 43.2-3 (1983), pp. 235–239. DOI: 10.1016/0012-365X(83)90160-7. URL: https://doi.org/10.1016/0012-365X(83)90160-7.

[71] Steven Skiena. *The Algorithm Design Manual, Third Edition*. Texts in Computer Science. Springer, 2020. ISBN: 978-3-030-54255-9. DOI: 10.1007/978-3-030-54256-6. URL: https://doi.org/10.1007/978-3-030-54256-6.

[72] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. ISBN: 978-0-521-19527-0.

[73] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[74] Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. "The complexity of finding small separators in temporal graphs". In: *J. Comput. Syst. Sci.* 107 (2020), pp. 72–92. DOI: 10.1016/j.jcss.2019.07.006. URL: https://doi.org/10.1016/j.jcss.2019.07.006.