THE UNIVERSITY *of* EDINBURGH

Firefighting multiple disease outbreaks under different defence policies

Catriona Wedderburn

Doctor of Philosophy
University of Edinburgh
30^{th} November 2023

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Catriona Wedderburn*)

To my parents.

Abstract

Variations of the Firefighter problem offer deterministic, discrete time models for disease control. Classic Firefighter (CF) offers a model for disease control using vaccination. In the first section of the thesis, we review the literature, compare CF to a related graph problem and present some results on extremal graphs. We then introduce a polynomial time algorithm for optimal solutions of certain instances of CF.

In the next section, we introduce the Edge-Defence Firefighter Problem (EDF), which models disease control via contact reduction. We present some results on its properties and on how certain results on CF can be translated into EDF results. We present polynomial heuristic algorithms for EDF defence strategies on both finite and infinite square grids. We derive an Integer Program (IP) for EDF, along with results on valid inequalities that can be used to speed up the calculation time. Finally in this section, we present computational results on several heuristics for EDF defence strategies, derived from our CF algorithm.

In the final chapter, we introduce the Cost-Value Firefighter Problem (CVF) and the Distance-Limited Firefighter Problem (DLF). Both variations are based on CF. In CVF, we assume that vertices can have different costs and values to vaccinate, modelling the logistics of vaccinating interconnected population centres of varying size. In DLF, we relax the assumption in CF that defenders can travel an arbitrary distance between vertices they defend, instead fixing a maximum distance defenders can travel between time steps. We present various IP formulations for both, along with valid inequalities and computational results. Finally, we present polynomial time algorithms for DLF on paths.

Lay Summary

During a disease outbreak, it is important to know where best to deploy different control measures - especially if resources are limited. Network Epidemiology uses the mathematics of networks to model how an infection spreads between individuals or population centres by focusing on the connections between each individual or each population centre. Individuals are linked if the infection could spread between them - e.g. for airborne transmission, this could be if they regularly come within a certain physical distance. Population centres could be linked if they regularly have individuals travelling between them, potentially spreading the disease. The Firefighter Problem offers a simple model for something spreading through a network, tempered by a control technique. While originally conceived as a way to model fire spread controlled by firefighters, there has been increasing interest in using it to model infection control. This is the focus of this thesis. We investigate four variants of the Firefighter Problem, each modelling different disease control methods with different restrictions:

- Classic Firefighter Problem - the original variant, most studied in the literature, models disease spread, limited by a restricted budget of vaccines.
- Edge-Defence Firefighter Problem - this models disease spread where the only possible intervention is via contact reduction, or testing on transport routes between population centres to prevent transmission between them. This is especially useful if a vaccine is not available.
- Cost-Value Firefighter Problem - this models disease spread between population centres of varying sizes, such as cities or villages, where the entire population would be vaccinated. This models weighs the extra costs of vaccinating a larger population against the extra benefits of protecting more individuals.
- Distance-Limited Firefighter Problem - this models disease spread, limited by a vaccine where the vaccinators can only travel a certain distance in a given amount of time. This could be due to rough terrain or the need to maintain a cold chain for temperature sensitive vaccines.

As the Classic Firefighter Problem is the variant that has been most widely studied, we discuss which results in the literature on that variant apply to the other three, possibly with modifications. For the Classic, Edge-Defence and Distance-Limited Firefighter Problems, we present fast algorithms to find either the best or very good disease control plans under a range of circumstances. These circumstances are related to the properties of the network and sometimes also to

where exactly the disease breaks out. Much of the literature has focused on a single disease outbreak. It is not always possible to start control interventions when there is only one infected patient and the disease has not already spread. We focus heavily on multiple outbreaks of the same disease to model this. For Edge-Defence we present some heuristic algorithms to find disease control plans. Heuristic algorithms trade off making plans which are not necessarily optimal with much faster computation time. One of the biggest issues in using the variants of the Firefighter Problem for real world decision making is its computational complexity. If there are too many individuals or population centres being modelled, the calculation would take an unpractical amount of time to run on linear optimisation software. For the Edge-Defence, Cost-Value and Distance-Limited Firefighter Problems, we present ways to significantly speed up the calculations to find the optimal disease control methods. We do this through a combination of reformulating the problem and new theoretical results about what properties an optimal solution would have, so the software can find it more quickly. This means that calculations can run in a practical amount of time even if they consider more individuals or population centres.

Acknowledgement

My biggest thanks go to my main supervisor in the later stages of this project, Jörg Kalcsics, for stepping in and supporting me and without whom I would not have reached this stage. He made useful contributions in Chapters 3 for the section on square grids and throughout Chapters 4 and 5. Jessica Enright was the supervisor with whom I had the most contact while writing Chapter 2; Chapter 3 up to the square-grid material; and the early stages of Chapter 4 - which was finished with Jörg Kalcsics. In writing Chapter 5, I had the pleasure of working together with Marta Baldomero Naranjo, Jörg Kalcsics and Antonio Manuel Rodríguez Chía.

Many people contributed to helping me learn to code. I would especially like to thank Andy Law, Rudi Horn and Frank Emrich. I also want to give a big thanks to Jo Stevens for supporting me through the more difficult stages of this project. Rowland Kao and David Gally provided help and support on my thesis committee; and further proof reading was provided by William Pettersson, Jesse Sigal and Mary Cryan. I would also like to thank the College of Medicine and Veterinary Medicine and the Global Academy of Agriculture and Food Systems for funding me and the School of Informatics for hosting me in their building.

My examiners, Stephen Finbow and Sergio García Quiles, provided much useful feedback, which improved the final version of this thesis. Stephen Finbow made especially useful contributions in Chapter 2.

Nuiok Dicaire, Lauren Watson, Rose Blake and Vera have been excellent friends and I was lucky to have their support. My biggest supporters were my parents and my husband. I owe my love of mathematics to Mr. Harris. Finally, I would like to thank Janet Macska, Jason Kishka, Simona Kato, Josephine Miao, Augustus Catfiadhaich and Chidi Ichbinnekatze.

Contents

Chapter 1

Introduction

1.1 The Firefighter Problem and Network Epidemiology

Traditionally, epidemiological models have assumed that populations are well or completely mixed. More recently, explicit spatial or contact data has been incorporated into models. Network Epidemiology uses appropriate spatial and contact data to build *contact networks* to model disease spread - these are graphs where individuals or groups of individuals are represented by vertices and an edge connects two vertices if there is a possibility that disease could spread directly between them. A particularly comprehensive data set exists for cattle transport networks within the EU due to EU legislation, see Kao et al. [33]; the vertices could also model entire population centres, with edges modelling transport links between them.

The Firefighter problem and its variants offer a model which assigns all individuals to one of two states: a potential host is *susceptible* if they are not currently infected but could potentially become infected; a host is *infectious* if they are infected and are capable of passing on the disease to susceptible individuals. Such models are called Susceptible-Infected or SI models. In the Firefighter problem, a fire or virus breaks out on a set of vertices at time zero; then defenders can defend a fixed number of vertices; then the fire spreads deterministically to all undefended neighbours of burning vertices. Once vertices burn or are defended, they remain so for the rest of the game. The game continues with defenders defending vertices and the fire spreading to undefended neighbours until the fire can spread no further.

Defence in the Firefighter problem can be used to model the effects of vaccination. Although ring-vaccination (vaccinating those most at risk from infection and their contacts up to a certain number of degrees of separation) is widely used in real world epidemics, there are circumstances under which all individuals in a population centre are vaccinated, in which case the vertices in Firefighter can model these population centres. Firefighter can also model the effect of removing all individuals from an area under threat from disease spread. This could be especially useful for protecting endangered species from outbreaks like the current pandemics of Chytridiomycosis affecting amphibians and of White-nose

syndrome, which affects bats. It can also be used for applications other than disease control, such as modelling the spread and control of invasive species.

In this thesis, we study graphs that can provide a baseline for comparison with other networks by leading to the least burning if the fires start on the worst place and if the defence is the best possible. We then move on to study three new variants of the Firefighter problem, each of which are designed to address and model problems encountered with disease control in the real world.

1.2 *mmd* Graphs and Fire Path Equivalent Games

The overarching theme of the first part of this thesis is making the best of a bad situation. This is appropriate since we set out to answer some open questions, fail, and then develop useful theory for studying other problems. We begin by studying *mmd* or *minimal maximal damage* graphs, first studied by Finbow et al. in [19]. These are the graphs that have the least burning if the fires start in the worst places and the defenders do the best job possible. Having the fires start in the worst places could be of interest to policy makers who want to understand how an epidemic would develop in the worst case scenario. Understanding *mmd* graphs could provide a baseline for comparison with other networks, since they perform best under the assumption that an optimal defence can be found and implemented.

We shall look at the worst places for the fires to start - and note how different *mmd* graphs are from graphs which are optimal if the fires start uniformly at random. We also devote some attention to studying the links between *mmd* graphs and the Resistance Network Problem, first proposed in 1978 by Gunther and Hartnell in [28], around 17 years before the Firefighter Problem was proposed by Hartnell in [30]. The Resistance Network Problem considers how to organise an underground network to minimise the number of contacts that individuals could betray, if the worst possible set of conspirators betray their comrades. Finding optimal graphs for the Resistance Network Problem is relatively simple, if a little computationally expensive. We provide an algorithm for this and discover new *mmd* graphs by proving that under certain conditions, the optimal graphs we found for the Resistance Network Problem are also *mmd*.

Finding a theorem that describes the structure of *mmd* graphs and finding all the *mmd* graphs on up to 10 vertices for two fires and two defenders, were both listed as open problems in Finbow and MacGillivray [21]. While we find many new *mmd* graphs, and make some progress towards understanding their structure; identifying the worst place for the fire to start; and what optimal defence in such cases might look like, the questions of a full categorisation of their structure or how to find further examples sadly remain open. In future, the question of finding *mmd* graphs may be amenable to bilevel optimisation and we provide some new conjectures to work towards a theory of their structure.

In trying to find *mmd* graphs, we come across a certain scenario - instances of Firefighter where every vertex of degree greater than two is initially burning. This effectively renders the game equivalent to defending on a set of paths and we give a polynomial algorithm for finding optimal defence given any number of

starting fires (as long as they include all high degree vertices) and any number of defenders. The basic idea is to prioritise defending large path-like sections, leaving the smaller ones to burn. This algorithm proves to be very useful in a few situations, and will be modified and revisited in Chapters 4 and 5.

1.3 Edge-Defence Firefighter

The original version of the Firefighter Problem focuses on defending vertices, which can be used to model vaccination against an infectious disease. We introduce a different version of the Firefighter Problem, which instead focuses on defending edges.

Defending edges can have many different interpretations in terms of disease control. In cases where a vaccine has not been developed or is not yet readily available, restricting the interactions between individuals may be the only form of disease control available. Such social distancing was a common feature of many government responses in the 1918-1920 flu pandemic and in the first years of the COVID-19 pandemic. More generally, if the vertices model population centres rather than individuals, restricting travel between population centres (especially from places where an infectious disease has been detected) is a centuries-old response to pandemics and was widely used during the Black Death of 1346 to 1353.

In addition to cutting off some form of interaction entirely, edge defence can also be used to model many detection and intervention processes which generally allow the interaction but stop an infection from spreading. This could include the testing and quarantine procedures common for the movement of livestock and domestic animals across borders for various diseases including rabies; screening donated blood to avoid spreading hepatitis or HIV to the recipients; or the use of disinfectant mats to prevent people spreading foot and mouth disease or sudden oak death on their shoes.

Beyond disease control, edge defence can be used to model various measures for preventing the spread of invasive species between different areas. Examples include spraying airplanes with pesticide to reduce the risk of invasive insect species hitching a lift.

To the best of our knowledge, there is no published work dedicated to the Edge-Defence Firefighter Problem, as we have defined it. There are however, some results on Edge-Defence Firefighter in Comellas, Mitjana and Peters [9]. Their paper was focused on a model where a message was broadcast (or a virus spread) through a graph starting at time 0. Each infected vertex would stay *active* for A time steps and at each subsequent time step, for as long as a vertex was active, the infection would spread along k edges of the infected vertex. At the end of the paper, they introduced the idea of defending edges so the infection cannot spread along them and presented a result on circulant graphs allowing k to equal $\Delta(G)$, the maximum degree of the graph and thus making their broadcast model equivalent to the Edge-Defence Problem we propose here. They found that for $2 \leq k \leq \Delta(G)$, all vertices would be infected if the defence budget was zero (Theorem 5 of [9]). This similarity with the Firefighter problem was not noted

in the original paper, but was noted by Michalak and Knowles in [43].

Michalak also provides the other closest comparison to our problem definition in [42], which studies a version of the multiobjective Firefighter Problem with both vertex and edge defence. The paper focuses on metaheuristics and will be more fully discussed in Chapter 4.

The final published work on a similar question is found in Marzouk [38]. This looked at a variant of the Firefighter game where all edges are initially susceptible, and then are progressively randomly assigned to be either burning or defended, with the fire spreading instantly along susceptible edges until it reaches a defended edge.

1.4 Edge-Defence Computational Experiments

In this chapter, we focus on various heuristics for Edge-Defence Firefighter, adapted from the optimal vertex defence strategy for fire path equivalent games given in Theorem 16 of Section 2.6. To have optimal solutions with which to compare the heuristic solutions, we start by developing an integer program for Edge-Defence Firefighter and using various valid inequalities to improve its running time.

Our main motivation for investigating different variations of the Firefighter problem is to better model real world constraints. Optimising the run time for an integer program to find exact solutions and developing heuristic algorithms for instances that would take too long to solve exactly are important steps towards creating tools that could be used for policy decisions in real world epidemics.

An integer program for Classic Firefighter was first given in Develin and Hartke [15]. Subsequent papers have worked on modifying and improving the running times for this program, notably Finbow and MacGillivray [21], which modified it; García-Martínez et al. [24], which noted the importance of an upper limit for the number of time steps a game takes to complete for the running time and used an iterative approach to finding it; and Ramos et al. [47], which found improved upper bounds for this and added some valid inequalities, which we shall build upon.

An early result by Hartnell and Li [29] showed that for Classic Firefighter on trees with a single initially burning vertex and a single defender, defending the threshold vertex with the greatest number of descendants at each defender turn saves at least half the number of vertices saved by the optimal solution. There are further approximation algorithms for Classic Firefighter on trees, including in Cai, Verbin and Yang [6]; and Iwaikawa, Kamiyama and Matsui [32]. Since any vertex defence on a tree has an equivalent edge defence strategy which leads to the same amount of burning these results instantly carry over to Edge-Defence Firefighter.

García-Martínez et al. [24] summarised various heuristics for Classic Firefighter on random graphs, based on vertex degree and the number of descendants of a vertex.

There has been considerable work on *metaheuristics* for Classic Firefighter. A *metaheuristic* is a procedure which produces and applies heuristics in an attempt

to find a near optimal solution. Blum et al. [3] used a metaheuristic purely based on ant colony optimisation, and another which started with ant colony optimisation, then used solution polishing. Hu, Windbichler and Raidl [31] used a variable neighbourhood search.

The paper presenting heuristic algorithms for a problem which arguably comes closest to Edge-Defence Firefighter is Michalak [42]. This presented an evolutionary algorithm for a problem based on Classic Firefighter, but with a few key differences. Firstly, edges could also be defended. This edge defence is different from the one we model, in that our defenders choose exactly which edges to defend, whereas in Michalak [42] a decision is made to defend edges of a given vertex, and which of these edges are defended is probabilistic. Furthermore, vertex defence was also allowed as part of the same strategies as this edge defence, and the spread of the fire was probabilistic, whereas we model it as deterministic. Michalak has written several other papers using evolutionary algorithms to find solutions for the multiobjective Firefighter and its variants, including in [40] and [41] as sole author and in [43] with Knowles.

For a more in-depth review, see Wagner [48]. As far as we know, there are no published papers giving heuristics for the Edge-Defence Firefighter Problem as we have defined it.

1.5 Limited Defence Firefighter

We address two variants of the Firefighter problem: Cost-Value Firefighter (CVF) and Distance-Limited Defence Firefighter (DLF).

- In the Cost-Value Firefighter, every vertex has both a cost - a minimum amount of defence budget that must be spent to protect it; and a value - the utility gained by protecting that vertex. The objective is then to maximise the total value of the vertices saved. Moreover, instead of a fixed number of defenders, we are given a defence budget that we can spend each time step for defending vertices. Differing from the classic model, in this variant, the defence of a vertex can be built up gradually, i.e., over multiple time steps. There has been some work assigning a value to saved vertices, see Duffy and MacGillivray [16], but we believe this is the first work to combine that with a cost of saving each vertex.

- We also deal with the Distance-Limited Firefighter, which is based on CF but adds restrictions to how far defenders can travel between time steps. In Chen et al. [8], a version with the distance limit set to one was termed *continuous firefighting*, and they studied how many defenders are needed to contain fires on an infinite square grid. DLF has attracted some attention recently, see Days-Merrill [13], Burgess et al. [4] and [5]. The later introduced a different IP from the one we propose.

The defence strategy in the Classic Firefighter Problem - first introduced by Hartnell in 1995 in [30] - can model various interventions including a vaccine that prevents the vertex from being infected or from spreading the disease to other

vertices; or an intervention that simply removes individuals from the area and thus from the path of the disease. Such interventions have been used to save endangered species from the Chytridiomycosis pandemic [26].

The vertices themselves could model individuals, with edges connecting individuals between whom disease could spread (for instance if they regularly come within a certain physical distance for airborne transmission). Alternatively, the vertices could be modelling population centres, with edges joining centres that in real life are linked by transportation and regularly have individuals travelling between them, potentially spreading the disease.

Although the contact tracing and ring vaccination (vaccinating everyone who came into contact with an infected individual and everyone who came into contact with them etc.) within population centres are frequently used, sometimes it is preferable to vaccinate entire populations. This could be due to lack of information about contacts; a policy choice based on the cost of the vaccines and the risk to the public; or the logistical difficulty of having to visit the same population centres multiple times to administer further vaccines.

The variant CVF proposed in this chapter can better represent some real-word applications by setting:

- the cost proportional to the population size that must be vaccinated, and any extra costs related to that location; and
- the value proportional to the population size who would be protected.

This can model vaccinating population centres of varying sizes. There has been some interest in modelling the cost of vaccinating different vertices, notably in Michalak [42] - which was based on CF but uses probabilistic disease spread and control interventions beyond vaccination.

Of course, varying population sizes are not the only logistical constraint policy makers need to consider. In particular, in the second variant that we address (DLF), we focus on relaxing the assumption in CF that vaccination teams can travel arbitrarily large distances. The terrain covered by vaccination teams may be tough or simply very large; there may be extra constraints imposed by the need to maintain a cold chain for temperature sensitive vaccines; or when the health care workers require a security detail. The latter was notably the case in the $2018-2020$ Ebola outbreak in Kivu in the Democratic Republic of Congo [46]; and in polio vaccination campaigns in areas where the terrorist group Boko Haram is present [45].

1.6 Organisation of the Thesis

In Chapter 2, we initially focus on minimal maximal damage or *mmd* graphs - the graphs which have the least burning if the fire starts in the worst place and we use the best possible defence strategy. We discuss results from the related Resistance Network Problem (RNP) and how they relate to finding *mmd* graphs. We find some new optimal graphs for the RNP; then present some results on how these are related to *mmd* graphs, showing that some of our new RNP-optimal graphs are *mmd*.

We then focus on particular instances of the Firefighter Game, where the fire initially breaks out on all vertices of the graph with degree at least three. We call these *fire path equivalent* and give a polynomial time algorithm for optimal defence, which we will adapt and reuse in Chapters 4 and 5. We finish the chapter by relating this to the search for mew *mmd* graphs and suggesting how the search may be continued.

In Chapter 3, we introduce Edge-Defence Firefighter. We start by defining the problem and discussing the most similar work we could find in the literature. We then move on to discuss how Edge-Defence Firefighter generally compares with Classic Firefighter, in terms of the amount of burning with optimal defence and the same number of firefighters; and of computational complexity. We show that Edge-Firefighter and Classic Firefighter are equivalent for games on trees with a single fire. We then move on to focus on adapting various results from the literature on Classic Firefighter to Edge-Defence Firefighter. The remainder of the chapter is dedicated to Edge-Defence Firefighter on square grids with a single starting fire and a single defender.

In Chapter 4 we focus on computational experiments on Edge-Defence Firefighter. We first give an integer program, then optimise the run times for it. We do this using valid inequalities, with a particular focus on upper limits for how many time steps it takes for the game to finish using optimal defence. We then move on to several heuristic algorithms for Edge-Defence Firefighter, all based on the fire path equivalent Algorithm from Chapter 2. We discuss which centrality measures to use for these heuristics; which heuristics perform best; and how they compare with the optimal solutions.

In Chapter 5 we introduce two variants of the Firefighter Problem: Cost-Value Firefighter and Distance-Limited Firefighter. Both models are intended to address limitations in applying the Classic Firefighter Problem to real life conditions in an epidemic.

Cost-Value Firefighter (CVF) allows for vertices to have different costs to defend them and different values of saving them; modelling the different quantities of resources that need to be assigned to vaccinate variably sized population centres. We define the problem, introduce a mixed integer program (MIP) and then run computational experiments to optimise the runs times for the MIP. We do this by developing several upper bounds for the number of time steps it takes for the game to finish with optimal defence; by imposing or relaxing binary variables; and with other valid inequalities.

Distance-Limited Firefighter (DLF) restricts the distance that defenders can travel per time step. We define the problem, discuss relevant literature, briefly discuss computational complexity and then give an MIP to find exact solutions. We then present computational experiments, optimising the run times for the MIP by adding valid inequalities and relaxing the binary constraints on certain variables. We discuss upper bounds for the number of time steps an optimal solution on trees would take. We finish the main body of the thesis with a discussion of DLF on paths, with multiple initially burning vertices.

We round off the thesis with conclusions and a discussion of future research directions.

Chapter 2

mmd Graphs and Fire Path Equivalent Games

The overarching theme of this chapter is making the best of a bad situation. This is appropriate since we set out to answer some open questions, fail, and then develop useful theory for studying other problems. We begin by studying *mmd* or *minimal maximal damage* graphs, first studied by Finbow et al. in [19]. These are the graphs that have the least burning if the fires start in the worst places and the defenders do the best job possible. Having the fires start in the worst places could be of interest to policy makers who want to understand how an epidemic would develop in the worst case scenario. Understanding *mmd* graphs could provide a baseline for comparison with other networks, since they perform best under the assumption that an optimal defence can be found and implemented.

We shall look at where the worst places for the fires to start is - and note how different *mmd* graphs are from graphs which are optimal if the fires start uniformly at random. We also devote some attention to studying the links between *mmd* graphs and the Resistance Network Problem, first proposed in 1978 by Gunther and Hartnell in [28], around 17 years before the Firefighter Problem was proposed by Hartnell in [30]. The Resistance Network Problem considers how to organise an underground network to minimise the number of contacts that individuals could betray, if the worst possible set of conspirators betray their comrades. Finding optimal graphs for the Resistance Network Problem is relatively simple, if a little computationally expensive. We provide an algorithm for this and discover new *mmd* graphs by proving that under certain conditions, the optimal graphs we found for the Resistance Network Problem are also *mmd*.

Finding a theorem that describes the structure of *mmd* graphs; and finding all the *mmd* graphs on up to 10 vertices for two fires and two defenders were both listed as open problems in Finbow and MacGillivray [21]. While we find many new *mmd* graphs, and make some progress towards understanding their structure; where the worst place for fire to start is; and what optimal defence in such cases might look like, the questions of a full categorisation of their structure or how to find further examples sadly remain open. In future, the question of finding *mmd* graphs may be amenable to bilevel optimisation and we provide some new conjectures to work towards a theory of their structure.

In trying to find *mmd* graphs, we come across a certain scenario - instances of the Firefighter Game where every vertex of degree greater than two is initially burning. This effectively renders the game equivalent to defending on a set of paths and we give a polynomial algorithm for finding optimal defence given any number of starting fires (as long as they include all high degree vertices) and any number of defenders. The basic idea is to prioritise defending large path-like sections, leaving the smaller ones to burn. This algorithm proves to be very useful in a few situations, and will be modified and revisited in Chapters 4 and 5.

2.1 Problem Definition and Notation

We start with some basic definitions:

Definition. A *graph* G or $G = (V, E)$ consists of a set of vertices V and a set of edges E, each of which join two vertices.

The above terminology is standard for mathematics, in network epidemiology graphs, vertices and edges are often called *networks*, *nodes* and *links* respectively. Throughout this thesis we will be using the notation $n := |V|$, the number of vertices in a graph; and $m := |E|$, the number of edges in a graph.

Definition. A *walk* in a graph $G = (V, E)$ is an ordered list of vertices $v_1, v_2, v_3 \dots v_k \in V$ such that for all $1 \leq i \leq k-1$, $(v_i, v_{i+1}) \in E$; together with these edges. A *path* is a walk where all vertices are distinct.

Definition. A graph $G = (V, E)$ is *connected* if there exists a walk between every pair of vertices $(u, v) \in V$.

Definition. A graph $G = (V, E)$ is *simple* if there is at most one edge between each pair or vertices and no edges connect a vertex to itself.

Henceforth all graphs in this thesis are assumed to be connected and simple unless explicitly stated otherwise. We frequently examine *trees*:

Definition. A *cycle* is a non-empty walk where only the first and last vertices are equal. A *tree* is a connected graph that does not contain any cycles.

We often need to consider the *neighbours* of a vertex.

Definition. For two vertices $u, v \in V$, we say that u is a *neighbour* of v if $(u, v) \in E$, and the *neighbourhood* $N(v)$ of v is the set of all its neighbours. We call u, w *distinct neighbours* of v if they are neighbours of v and u does not equal w. For a subset $S \subset V$, we define $N(S)$ to be the set of all neighbours of all elements of S. The *degree* of a vertex is the number of its neighbours.

In the original version of the Firefighter Game as introduced in Hartnell [30] - which we shall term *Classic Firefighter* throughout this thesis and is the sole focus of this chapter - a fire breaks out on a single vertex of a simple, connected graph $G = (V, E)$. We shall generally allow it to break out on the f vertices in the set F, where $f \geq 1$. After this, up to d vertices can be defended at each defence

turn before the fire spreads to all undefended neighbours of burning vertices in the next time step. The game alternates between the fire spreading and up to d vertices being defended until the fire can spread no further. A vertex, once burning or defended, will remain in that state for the rest of the game, which ends when the fire can spread no further.

We call a vertex *susceptible* at a given time step if it is neither burning nor defended. A vertex is called *saved* if, at the end of the game, it is not burning.

Intuitively, we interpret a defence as a mapping of times to the vertices that the firefighters defend (if they are available for defending) at that time. Given a graph $G = (V, E)$ and a defence budget d, a *defence strategy* is a function $DV : \mathbb{N} \to DV(t) \subset V$ where $|DV(t)| \leq d$ for all t. In later chapters, we shall refer to such strategies as *vertex defence strategies* to differentiate them from other defence methods. Let $S = (G, F, d)$ be an instance of Classic Firefighter on graph $G = (V, E)$ with the set of f initially burning vertices $F \subseteq V$ and d defenders. We call a defence strategy for S *optimal* if it leads to the fewest vertices burning at the end of the game out of all possible defence strategies on S.

For a given instance S and defence strategy DV, let $BV(S, DV, t)$ be the set of vertices burning on instance S using defence strategy DV at time step t. Let T denote the final time step of the game. Then $|BV(S, opt, T)|$ denotes the total burning at the end of an instance of the fire firefighter game, using optimal defence.

2.2 Resistance Network Problem

A related problem which offers key insights into the Firefighter Problem is the Resistance Network Problem. The problem imagines a network of agents or cells of agents of a resistance organisation as follows: firstly all agents should be able to communicate with each other, so the graph must be connected; secondly, the resistance organisation expects that a certain number of its agents will be captured and betray all of their neighbours in the contact network. The Resistance Network Problem aims to find a way of arranging the graph so that, given that the set of f agents with the most distinct contacts will be captured, the fewest number of their colleagues will be betrayed. Stating this formally, the agents we expect to be captured constitute a B–set.

Definition. For some fixed $f \in \mathbb{N}$, a *B–set* of a graph is the set of size f which has the most distinct neighbours.

A given graph can have multiple distinct B-sets. Phrased this way, the Resistance Network Problem is to minimise the number of distinct neighbours of the B-sets.

Definition. For a graph G and some $f \in \mathbb{N}$, $K(G, f)$ is defined as the number of distinct neighbours of a B-set of size f of G plus f. For a fixed $n \in \mathbb{N}$, $K(n, f)$ is defined as the smallest value of $K(G, f)$ over all connected graphs G of size n.

The Resistance Network Problem aims to find the graphs on n vertices with the lowest possible values of $K(G, f)$ for a given value of f.

Definition. A $K(n, f)$*–optimal graph* is a graph in which all B-sets of size f have exactly $K(n, f) - |B|$ distinct neighbours.

The values of $K(n, f)$ for all $n, f \in \mathbb{N}$ are known and at least one $K(n, f)$–optimal graph has been identified for each combination of n and f. The values of $K(n, f)$ were calculated by Gunther and Hartnell in [28]:

Theorem 1 (Quoted from [19], originally in [28])**.**

1. *If* $n \leq 2f + 1$, $K(n, f) = n$
2. *If* $f \leq 3$ *and* $n \geq 4f$, $K(n, f) = 3f$
3. *If* $f \geq 4$ *and* $n \geq 5f - 4$, $K(n, f) = 3f$
4. *Case 1:* $f = 3t + 1$ *for some* $t \in \mathbb{Z}$
   (a) *If* $f = (6t + 3) + (2a - 1)$ *or* $f = (6t + 3) + 2a$ *where* $1 \leq a \leq 2t + 1$, $K(n, f) = (6t + 3) + (a - 1)$
   (b) *If* $f = 10t + 6 + x$ *where* $0 \leq x < 5t - 5$, $K(n, f) = (8t + 4) + \lceil \frac{x}{5} \rceil$
5. *Case 2:* $f = 3t + 2$ *for some* $t \in \mathbb{Z}$
   (a) *If* $f = (6t + 5) + (2a - 1)$ *or* $f = (6t + 5) + 2a$ *where* $1 \leq a \leq 2t + 2$, $K(n, f) = (6t + 5) + (a - 1)$
   (b) *If* $n = 10t + 10$, $K(n, f) = 8t + 6$
   (c) *If* $f = 10t + 11 + x$ *where* $0 \leq x < 5t - 5$, $K(n, f) = (8t + 7) + \lceil \frac{x}{5} \rceil$
6. *Case 3:* $f = 3t + 3$ *for some* $t \in \mathbb{Z}$
   (a) *If* $f = (6t + 7) + (2a - 1)$ *or* $f = (6t + 7) + 2a$ *where* $1 \leq a \leq 2t + 3$, $K(n, f) = (6t + 7) + (a - 1)$
   (b) *If* $n = 10t + 14$ *or* $n = 10t + 15$, $K(n, f) = 8t + 9$
   (c) *If* $f = 10t + 16 + x$ *where* $0 \leq x < 5t - 5$, $K(n, f) = (8t + 10) + \lceil \frac{x}{5} \rceil$

In the next section and Section 2.4, we will look at the relationship between $K(n, f)$-optimal graphs and a class of graphs that are analogous in the Firefighter Game.

2.3 mmd Graphs

The main focus of this section is *mmd* graphs. An *mmd* or *minimal maximal damage* graph is a graph of size n which has the least total burning if the fire starts in the worst place and the defenders defend optimally. Intuitively, it can be viewed as a way of optimising a contact network so that even under the worst case scenario for an initial outbreak, there will be relatively little spread as long

as the defenders use the best strategy, thus providing a good comparison or goal for other ways of arranging networks to limit potential disease spread.

Similarly to the Resistance Network Problem, in *mmd* graphs, we assume that the fire starts in the worst place:

Definition. For some fixed $f \in \mathbb{N}$, an *M–set* of a graph is a set of size f where if the fire is placed there it will cause the most possible damage to the graph over the course of the game if the firefighters defend optimally.

Like B-sets for the Resistance Network Problem, a given graph can have multiple distinct M-sets. We can now formally define an *mmd* graph:

Definition. An *$mmd(n, f, d)$–graph* is a graph on n vertices in which the fewest possible vertices burn (over all connected graphs on n vertices) if the fire starts in an M-set size of f and d defenders defend optimally.

Henceforth, if n, f and d are clear from context then we may write *mmd* rather than $mmd(n, d, f)$.

Optimal Defence

One of the challenges of studying *mmd* graphs lies in determining what the optimal defence strategy is. The computational complexity of this has been widely studied. The decision question is formally stated as follows:

FIREFIGHTER
Instance: A connected graph $G = (V, E)$, a set $F \subset V$ and two natural numbers $d, k \in \mathbb{N}$
Question: If the fire starts at F in G, is there a strategy for d defenders to save at least k vertices?

This is NP-complete if there is a single initial fire and single defender (as shown in Finbow et al. [20]) and for games with multiple defenders (as shown in Bazgan et al. [2]). We will cover both of these results in slightly more detail in Chapter 3.

Optimal defence strategies are known for very limited graph classes. The key concepts in these results are *distance* and *levels*:

Definition. The *length* of a path is the number of edges it contains. The *distance* between two vertices in a graph is the length of a shortest path between them.

Definition. Given a graph G and a set of initially burning vertices F, a vertex v is defined as being at level i if i is the smallest distance between v and any vertex in F.

The notion of levels is most meaningful when there is only one fire and it can only reach each given vertex along one path, i.e. if the graph is a tree.

Theorem 2 (Observation 4.1 of MacGillivray and Wang [37])**.** *Let G be a tree and let $F = \{v\}$ be a single fire at time 0. Then in optimal defence vertices at level i are defended at time i.*

This does not hold for games with more than one initial fire.

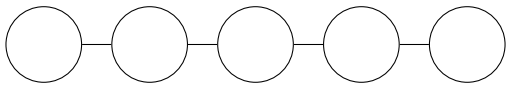


Figure 2.1: The path graph on five vertices.

Links to the Resistance Network Problem

In the Firefighter Problem, the fire spreads to all neighbours of vertices in F after one time frame, so the lower bound of burning at the end of the game for a given set of initial fires F and d defenders, assuming that $|N(F)| \geq d$, is given by $|N(F)| + |F| - d$. The highest value this could take would be in the case where F has the most distinct neighbours, i.e. if F is the B-set from the resistance fighter problem. If this is the most burning starting from any set of size $|F|$ against optimal defence then the B-sets of the graph are also the M-sets. This means the theoretical minimum for the total burning at the end of the game in the Firefighter Problem on a graph of size n with f fires starting at an M-set is given by is $K(n, f) - d$. Furthermore, this means that any graph shown to have at most $K(n, f) - d$ burning if the fire starts at any set size f and the defenders defend optimally must be $mmd(n, f, d)$. This proof technique was used extensively in Finbow et al. [19].

Another related question is that of finding optimal graphs if the fire starts uniformly at random and defence is optimal. For instance, if we consider *star graphs*:

Definition. A *star graph* on n vertices is a connected tree in which $n-1$ vertices are connected to one central vertex.

Theorem 3 (Theorem 2.2 of Finbow et al. [19])**.** *If $d = 1$ and $f \geq d$ with F chosen uniformly at random, then a star graph is one of the graphs on n vertices that lead to the least possible burning, given optimal defence.*

There was also work by Crosby et al. [12], which characterised the optimal graphs if the fire starts uniformly at random and defence is optimal for $f = 2 = d$.

It is worth noting the difference between optimal graphs when the fire starts at an M–set and optimal graphs where the fire starts uniformly at random. The M–set of a star graph trivially contains the central vertex, so for $n > f + d$, the theoretical maximum of $n - d$ vertices would burn regardless of defence strategy, making star graphs one of the worst graph classes in such conditions.

Known mmd Graphs

Several *mmd* graphs have been identified. The first type is a *path graph*:

Definition. A *path graph* on n vertices is a tree in which $n - 2$ vertices have degree 2 and two vertices have degree 1.

The path graph on five vertices is shown in Figure 2.1. Further *mmd* graphs have been identified which are related to star graphs. The star graph is often modified using subdivision:

Definition. A graph $G = (V, E)$ is *subdivided* if one or more edges $(u, v) \in E$ is replaced with a new vertex w and two new edges (u, w) and (w, v)

The paths radiating from the central vertex of a subdivided star are called *arms*. We can now define the first two classes to which the known *mmd*- graphs belong:

Definition. The *superstar graph on n vertices* is the subdivided star with all arms length 2 if n is odd or all arms length 2 and one arm length 1 is n is even.

An example of a superstar graph is given in Figure 2.2.

Definition. For $n \in \mathbb{N}$ the *fifth column on n vertices* is the graph formed by first taking the path length $\lceil \frac{n}{5} \rceil$ and adding two branches of length 2 to all but the last vertex in the path; the remaining vertices are added to the final vertex in the path in the order: first one leaf, then another, then a child of a leaf and finally a child of the other leaf until the new graph has n vertices.

An example of a fifth column graph is given in Figure 2.3.

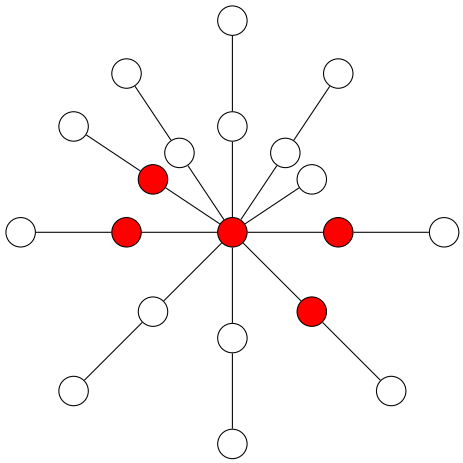


Figure 2.2: A superstar on 20 vertices. An M-set for $f = 5$ is indicated in red.

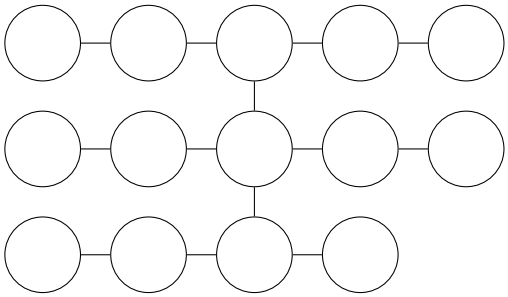


Figure 2.3: The fifth column graph on 14 vertices.

The final class of known *mmd* graphs are paths. All the currently identified *mmd* graphs are summarised in the following theorem:

Theorem 4 (Theorem 5.6 of Finbow et al. [19])**.** *For n vertices, f fires and d defenders, the following graphs are $mmd(n, f, d)$:*

1. *If $n \leq 2f + 1$, then all trees on n vertices are $mmd(n, f, d)$.*
2. *If $n \geq 4f$, $f \leq 3$ and $d \geq f$ then the path on n vertices is $mmd(n, f, d)$.*
3. *If $2f+1 < n \leq 4f-1$, $f \leq 3$ and $n = 2m+1$ or $n = 2m+2$ for some $m \in \mathbb{N}$ and $d \geq \lceil \frac{m-f+1}{2} \rceil$, then the superstar graph on n vertices is $mmd(n, f, d)$.*
4. *If $2f + 1 < n \leq 5f - 1$ and $f \geq 4$:*
   (a) *Case 1: $f = 3t + 1$ for some $t \in \mathbb{N}$*

i. *If* $6t+3 < n \leq 10t+5$ *and* $d \geq \lceil \frac{m-f+1}{2} \rceil$, *then the superstar graph on* n *vertices is* $mmd(n,f,d)$.

ii. *If* $10t+6 \leq n \leq 15t$ *and* $d \geq \lceil \frac{n}{5} \rceil$, *then the fifth column on* n *vertices is* $mmd(n,f,d)$.

(b) Case 2: $f = 3t+2$ *for some* $t \in \mathbb{N}$

i. *If* $6t+5 < n \leq 10t+9$ *and* $d \geq \lceil \frac{m-f+1}{2} \rceil$, *then the superstar graph on* n *vertices is* $mmd(n,f,d)$.

ii. *If* $10t+10 \leq n \leq 15+5t$ *and* $d \geq \lceil \frac{n}{5} \rceil$, *then the fifth column on* n *vertices is* $mmd(n,f,d)$.

(c) Case 3: $f = 3t+3$ *for some* $t \in \mathbb{N}$

i. *If* $6t+7 < n \leq 10t+13$ *and* $d \geq \lceil \frac{m-f+1}{2} \rceil$, *then the superstar graph on* n *vertices is* $mmd(n,f,d)$.

ii. *If* $10t+14 \leq n \leq 15t+10$ *and* $d \geq \lceil \frac{n}{5} \rceil$, *then the fifth column on* n *vertices is* $mmd(n,f,d)$.

5. *If* $n \geq 5f-4$, $f \geq 4$ *and* $d \geq f$ *then the path on* n *vertices is* $mmd(n,f,d)$.

This covers all combinations of n and f for the values of d given. For each of the $mmd(n,d,f)$ graphs in Theorem 4, the theoretical minimum of $K(n,f)-d$ burning is achieved if F is an M-set and the defenders defend optimally, as shown in Finbow et al. [19]. While Theorem 4 gives an $mmd(n,f,d)$ graph for each case where $d \geq f$, it is not generally understood if there are other classes of $mmd(n,f,d)$ graphs for each value of n, f and d.

In general the cases where $d < f$ are poorly understood and it is possible that no graph obtains $K(n,f)-d$ burning using optimal defence against a fire starting on a M-set, so a new way of proving a graph is $mmd(n,f,d)$ would have to be found. This will be the focus this chapter.

We begin by trying to get a deeper understanding on $K(n,f)$-optimal graphs. To do this we computationally generated all $K(8,2), K(9,2), K(10,2)$ and $K(10,3)$-optimal graphs. Algorithm 1 on page 27 shows how this was achieved for $f=2$, a very similar algorithm was used for $f=3$. After noting that many of these graphs were $mmd(n,f,d)$ optimal for certain values of d, we investigated the general relationship between $K(n,f)$-optimal and mmd graphs, with our main result given in Lemma 5. The generated graphs also informed our study of the structure of $K(n,f)$-optimal and mmd graphs, with results on their relation to their spanning trees given below. The main theoretical framework we have developed is to classify and study the simplest instances of the Firefighter Problem - which we call *fire path equivalent* instances - and to find optimal defence for them.

2.4 The Relationship Between mmd and $K(n,f)$-optimal Graphs

Our first result deepens the connections between $K(n,f)$-optimal and mmd graphs as outlined in Section 2.3 and capitalises on the fact that all $K(n,f)$ values are known [see Theorem 1].

Lemma 5. *For $K(n,f) \leq 2f+d$ all $K(n,f)$ – optimal graphs are $mmd(n)$–graphs.*

Proof. Let $n, f, d \in \mathbb{N}$ s.t. $K(n,f) \leq 2f+d$. Let G be a $K(n,f)$–optimal graph. Let the fire start at a B–set for the graph of size f. By definition of $K(n,f)$ the B-set has $K(n,f)-f$ distinct neighbours. Defend d of these. In the next move all the other $K(n,f)-f-d \leq f$ neighbours of the B-set burn. These new fires can have at most $K(n,f)-f$ distinct neighbours by definition of $K(n,f)$, but f of these are the original B-set which is already burning, hence they have $K(n,f)-f-f \leq d$ distinct neighbours which are not already burning. Defending these ends the game. In total f fires burned in the first time interval and $K(n,f)-d-f$ burned in the second time interval, giving $K(n,f)-d$ burned vertices.

Furthermore, in this case the B-set must also be an M-set. To show this, assume that the B-set is not an M-set. Then there exists some M-set with fewer than $K(n,f)-f$ neighbours but which will burn more vertices if the fire is started there. Following exactly the same procedure as above of defending d arbitrary neighbours of burning vertices will end up burning fewer vertices than the B-set would, a contradiction.

This strategy is optimal as it leads to the burning of the theoretical minimum of $K(n,f)-d$ over the course of the game.

Hence G is a graph in which starting the fire at an M–set and defending optimally gives the smallest number of burned vertices over the course of the game for a graph of its size, so it is an mmd–graph.

□

In particular this means that for $f = 2 = d$ and $n \geq 6$ then, since for $n = 6, 7$ we have $K(n,f) = 5$ and for $n \geq 8$ we have $K(n,f) = 6$, the above condition is fulfilled and all $K(n,f)$–optimal graphs are mmd–graphs.

This bound is tight. For instance for $n = 8$, $f = 2$, $d = 1$ we have $K(8,2) = 6$ giving a theoretical minimum of 5 vertices burning in the game. This lower bound is achieved by most $K(8,2)$–optimal graphs including the example in Figure 2.4. However, the path and the cycle on eight vertices; as well as the graph shown in Figure 2.5 are $K(8,2)$–optimal graphs where 6 vertices burn if 2 fires are started in the appropriate places and 1 defender defends optimally and are therefore not $mmd(8,2,1)$.

It is also not generally true that $mmd(n,f,d)$ graphs will be $K(n,f)$–optimal.

The proof of Lemma 5 also contributes to the limited results on known optimal strategies outlined in Section 2.3.

Corollary 6. *An optimal strategy for a $K(n,f)$-optimal graph where $K(n,f) \leq 2f+d$ is to defend d arbitrary neighbours of fires in each move.*

Proof. See proof of Lemma 5. □

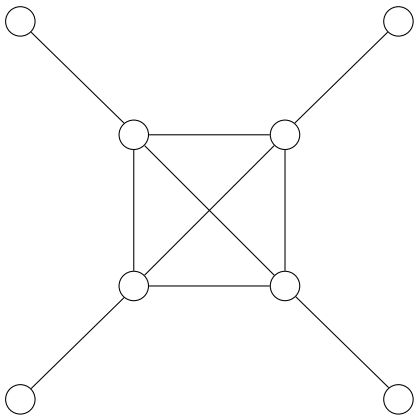


Figure 2.4: A $K(8,2)$–optimal graph which is also $mmd(8,2,1)$.

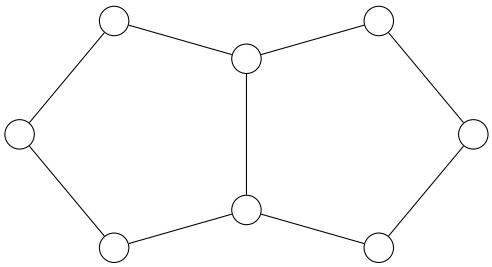


Figure 2.5: A $K(8,2)$–optimal graph which is not $mmd(8,2,1)$.

2.5 The Structure of mmd and $K(n,f)$-optimal Graphs

We shall now focus on the structure of $K(n,f)$-optimal and mmd graphs, in particular, their *spanning subgraphs*.

Definition. A *subgraph* of a graph $G=(V,E)$ is a graph $H=(V',E')$, where $V' \subset V$and $E' \subset E$. A subgraph $H(V',E')$ of $G=(V,E)$ is *spanning* if $V'=V$.

Lemma 7. *For any $n,f \in \mathbb{N}$ all spanning subgraphs of a $K(n,f)$–optimal graph are themselves $K(n,f)$–optimal.*

Proof. This is straightforward, the B-sets of G are also contained in any spanning subgraph and any other B-sets in the spanning subgraph are also contained in G. In both cases the set has at most as many neighbours in the spanning subgraph as it does in the original graph G. □

More interestingly, this relationship between a graph being $K(n,f)$-optimal and its spanning trees being $K(n,f)$-optimal goes the other way.

Lemma 8. *For any $n,f \in \mathbb{N}$, if all spanning subtrees of a graph G are $K(n,f)$–optimal then G is $K(n,f)$–optimal.*

Proof. We prove the contrapositive. Let $n,f \in \mathbb{N}$ and let G be a connected graph which is not $K(n,f)$-optimal. Let S be a set of size f of vertices which has the highest possible number of distinct neighbours, i.e. a B-set of G. Construct a spanning tree T of G by first assigning an arbitrary order to the vertices in S and then iterating through this list adding all edges adjacent to that vertex but not adjacent to a vertex which is already connected to the growing tree. Following the method of Kruskal's algorithm of then viewing all other vertices (i.e. the vertices which were not neighbours of S) as disconnected components and adding edges that connect two disconnected components of the graph finishes the spanning tree. By construction, the B-set of T has more than $K(n,f)-|S|$ distinct neighbours in T as they are 'inherited' from G, hence it is not $K(n,f)$-optimal. □

Lemmas 7 and 8 combined mean that a graph is $K(n, f)$-optimal if and only if all its spanning trees are optimal. This gives a way to classify all $K(n, f)$-optimal graphs according to their spanning trees. We then investigated whether similar results hold for *mmd* graphs.

Lemma 9. *For any $n, f, d \in \mathbb{N}$ all spanning subgraphs of an $mmd(n, f, d)$ graph G are themselves $mmd(n, f, d)$.*

Proof. Let G be an $mmd(n, f, d)$ graph for some given $n, f, d \in \mathbb{N}$. Let G' be an arbitrary spanning subgraph of G. Assume that G' is not $mmd(n, f, d)$. Clearly the M-sets in G' are also in G and any other M-sets in G are also in G'. The fire could progress at least as quickly on G as it could on G' given the same defense strategy, so the only way for fewer vertices to burn in G is for the 'extra' edges in G to allow a better defense strategy. At each stage of the game using this better strategy at least as many new vertices must burn on G as would burn in the same step of the analogous game on G' as each vertex has at least as many neighbours. Hence the only way G could allow a better strategy is if it is possible to end the game in fewer steps than on G'. Suppose we play the game using this better strategy on both G and G' with the fire starting on the same vertices. For the game to end we must reach a stage where all neighbours of burning vertices are either burning or defended, however at each stage of the game, the burning vertices in G have at least as many neighbours as the analogous burning vertices in G' hence the game cannot finish in strictly less time in G than it can in G'. This is a contradiction. Hence G' must be $mmd(n, f, d)$.

□

This gives us a useful corollary which can be used in further research.

Corollary 10. *Adding edges to a graph always results in at least as many vertices burning as in the original graph when starting the fire on the same set of vertices and defending with optimal strategy.*

Proof. See proof of Lemma 9. □

With a better understanding of the structure of *mmd* and $K(n, f)$-optimal graphs, we now move on to the key concept in our attempts to find new *mmd* graphs. We used Algorithm 1, together with the lists of all graphs in a certain classes in McKay [39] to find the new $K(n, f)$-optimal graph given in Appendix A.

2.6 Fire Path Equivalence

Intuitively, a stage of the Firefighter Game is *fire path equivalent* if it can be broken down into much simpler games, more precisely if removing all burning or defended vertices results in a graph consisting only of a disconnected collection of paths. This makes finding both M-sets and optimal defence strategies much easier.

Definition. A stage of the firefighter game $S = (G, F, D_{prev})$, where the graph is given by G the set of initially burning vertices is given by F and the set of previously defended vertices is given by D_{prev} is *fire path equivalent* if removing all burning or defended vertices leaves only path segments, such that only the end vertices in the path were originally connected to a burning or defended vertex.

We will often refer to a graph G or a game (G, F) being fire path equivalent or FPE when it is clear that what is meant is a stage of a game $S = (G, F, D_{prev})$. A game on a path or cycle is fire path equivalent at all stages of the game, independently of F or D_{prev}. A stage of a game on any graph is fire path equivalent if and only if all vertices of degree strictly greater than two are burning or defended.

Fire path equivalent games can be understood using a related game.

Definition. The *path representation* of a fire path equivalent stage of a game $S = (G, F, D_{prev})$ is an instance of the firefighter game in which the graph consists of a disconnected set of paths, as follows: First, delete all vertices burning or defended vertices, creating a collection of paths. Then add a burning vertex neighbouring any vertex that previously neighboured a burning vertex. This means each path will either have one or two burning ends.

Observe that defending the path representation is exactly equivalent to defending S and results in exactly the same amount of further burning. For an example of a fire path equivalent stage of the Firefighter Game and its path representation, see Figures 2.6 and 2.7 respectively.

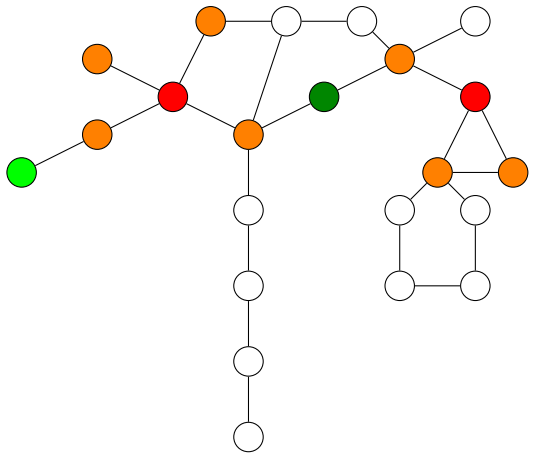


Figure 2.6: An example of a fire path equivalent stage. The initially burning vertices are indicated in red, the vertices that burn in time frame 1 are orange. The first vertex to be defended is dark green, the second is pale green.

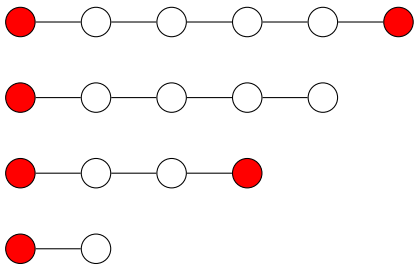


Figure 2.7: Its path representation. Burning vertices are indicated in red.

The relative simplicity of fire path equivalent games allows us to make conclusions about their optimal defence.

We call the susceptible neighbours of burning vertices *threshold vertices*. The set of susceptible neighbours of burning vertices is called the *threshold*.

Lemma 11. *Let $S = (G, F, D_{prev})$ be fire path equivalent. Then there exists an optimal defence which only defends threshold vertices.*

Proof. Let $S = (G, F, D_{prev})$ be a fire path equivalent stage of the Firefighter Game and consider a path P in its path representation. Let u, v be two not necessarily distinct burning vertices which are both on path P. Without loss of generality, suppose P contains no other burning vertices. Since S is fire path equivalent, all vertices on P excluding the endpoints have degree 2 and there is only one direction the fire can travel in from u and v respectively. Suppose that vertex w which lies on P but is not adjacent to either u or v is defended. Either another vertex on P will be defended in the course of the game or w will be the only defended vertex on P. If no other vertex on P is defended, then the placement of the single defender on P does not matter - the total burning on P will be equal to its length minus one and this will not affect the burning in any other part of the graph. Hence a vertex adjacent to u or v could be defended rather than defending w with no change to the total burning. However, if more than one vertex on P will eventually be defended then the most vertices are saved if the two defenders are as far apart from each other as possible - saving all the vertices sandwiched between them. This only occurs if both defenders were placed adjacent to the fires starting at u and v. □

By Lemma 11, we can focus solely on defence strategies that only defend threshold vertices when we look for optimal defence strategies against fire path equivalent stages.

Definition. A *ordering defence* on an FPE stage of the game S is defined first by arbitrarily assigning a label 'left' or 'right' to the burning vertices in each path with two burning vertices in the path representation and then permuting the single fire paths and the labelled vertices in paths with two fires.

This effectively assigns an order to the threshold vertices. As the threshold moves during the game, we then proceed to defend the next d threshold vertices in the list in each turn. If there are no remaining threshold vertices corresponding to an item in the ordering defence at the time step we want to defend them in, then we simply move to the next one in the list. Note that for fire path equivalent stages of the firefighter game, each ordering defence uniquely defines a defence strategy. For the rest of this section, we shall only consider such defences.

We say a vertex *ignites* at time step t if time step t is the earliest time step during which the vertex burns. The key concept in finding optimal defence for fire path equivalent instances is keeping track of how many vertices will ignite in each time step. Let S be a fire path equivalent instance of Firefighter. For a particular time step $t > 0$ and an ordering defence strategy D, we define $b(t, D)$ to be the number of vertices that ignite in time step.

To keep track of how many vertices will ignite in each time step, for $0 < t$ we subdivide the timestep t into t_1, when the defenders are placed and t_2, when the fire spreads to all susceptible neighbours of burning vertices. We can calculate $b(t, D)$ using three functions. We define $a_1(t, D)$ to be the number of paths in the path representation which have one burning leaf with no defender

on the neighbouring susceptible vertex at the end of time step t_1, using the ordering defence strategy D. For a path in the path representation to contribute to $a_1(t, D)$, either it had one burning end for the duration of time step $t-1$ and no defender was placed on its threshold vertex during time step t_1, or it had two burning leaves at the end of time step $t-1$, but then a defender was placed on exactly one of its threshold vertices in time step t_1. Similarly, we define $a_2(t, D)$ to be the number of paths in the path representation that have two burning leaves with no defender on the neighbouring susceptible vertex at the end of time t_1, using the ordering defence strategy D. Note that a path only contributes to $a_2(t, D)$ if it contains at least one susceptible vertex at the end of time t_1. Finally, we define $a'_2(t, D)$ to be the number of paths in the path representation with two burning leaves and no defenders has exactly one susceptible vertex at the end of time t_1, using the ordering defence strategy D. Note that $a'_2(t, D)$ counts a subset of the paths that $a_2(t, D)$ counts, so $a_2(t, D) \geq a'_2(t, D)$ for all $t > 0$ and any ordering defence strategy D.

Lemma 12. *For a fire path equivalent stage S, a time step $t > 0$ and an ordering defence strategy D, we have:*

$$b(t, D) = a_1(t, D) + 2 \cdot a_2(t, D) - a'_2(t, D). \tag{2.1}$$

Proof. Since we assume S is fire path equivalent, it has a path representation with the same amount of new burning after time step 0 as S and which consists only of disjoint paths with either one burning end or two burning ends and no other burning vertices. At a time step $t > 0$, a path P_1 in this path representation which has one burning end which is not defended in t_1 and at least one susceptible vertex at the start of t will contribute one to the sum of new vertices burned in t, one to $a_1(t, D)$ and zero to both $a_2(t, D)$ and $a'_2(t, D)$. A path P_2 with two burning ends will contribute two to the sum of new vertices burned in t if and only if was completely undefended in time step t_1 and if it contained at least two susceptible vertices at the start of t. In this case it contributes one to $a_2(t, D)$ and zero to both $a_1(t, D)$ and $a'_2(t, D)$, and thus two to the RHS of Equation (2.1). If P_2 was not defended in time step t_1 and contained exactly one susceptible vertex at the start of t then it will contribute one to the sum of newly burning vertices and it will contribute one each to $a_2(t, D)$ and $a'_2(t, D)$ and zero to $a_1(t, D)$, giving an overall contribution to the RHS of Equation (2.1) of one. If one threshold vertices in P_2 was defended at time t_1 and P_2 contained at least one susceptible vertex at the start of t then it contributes one to the sum of newly burning vertices and one to $a_1(t, D)$ and zero to both $a_2(t, D)$ and $a'_2(t, D)$. If all threshold vertices in a path are defended in time t_1, then the path will have no newly ignited vertices and will contribute zero to all of $a_1(t, D)$, $a_2(t, D)$ and $a'_2(t, D)$. Finally, if a path has no susceptible vertices at the start of t then it will contribute zero to the sum of vertices igniting in t and zero to all of $a_1(t, D)$, $a_2(t, D)$ and $a'_2(t, D)$. □

Lemma 13. *If $\forall t$, the value of $b(t, D)$ is minimised for a particular defence strategy D, then D is optimal.*

Proof. By definition, D is optimal if $\Sigma_t b(t, D)$ is minimised. This will occur if $b(t, D)$ is minimised for all $t > 0$. □

By Lemma 11, to find an optimal defence strategy, we only have to consider defence strategies that exclusively defend threshold vertices. In such strategies, each individual defender in our defence budget either completely stops the burning in a path which has one threshold vertex; or it can turn a path with two threshold vertices into a path with only one threshold vertex. In either case, the contribution of an individual defender reduces $b(t, D)$ by exactly one. The only other process that leads to a reduction in $b(t, D)$ is a path 'burns out' so that either a path has completely burned; or it goes from having two threshold vertices to only having one.

Therefore, a strategy that minimises $b(t, D)$ for all time steps - and so is optimal by Lemma 13, will do so precisely by maximising the number of paths that burn out. This can only be achieved by prioritising the defence of longer paths that would not burn out soon if left undefended, while leaving the shorter paths to burn out undefended. The exact relative lengths of paths we should prioritise defending are discussed in the following two lemmas:

Lemma 14. *If P_1 is a path with one burning leaf of length l_1 excluding the initially burning vertex and P_2 is a path with two burning leaves of length l_2 excluding the initially burning vertices and $l_2 \leq 2l_1$, then P_2 will completely burn or turn into a path with one susceptible vertex faster than P_1, if both are left undefended.*

Proof. P_1 will completely burn in time step l_1 if left undefended as the fire burns one of its vertices per time step. If l_2 is even, then P_2 will completely burn in time step $\frac{l_2}{2} \leq l_1$ if left undefended, as the fire burns two of its vertices per time step. If $l_2 > 1$ is odd, then P_2 will only have one susceptible vertex at the start of time step $\frac{l_2-1}{2} \leq l_1$ as two vertices burn per time step until then. Finally, if l_2 equals one, then it is already a path with only one susceptible vertex at the beginning of time step one, whereas P_1 must contain at least two susceptible vertices at the beginning of that time step. □

Lemma 15. *If P_1 is a path with one burning leaf of length l_1 excluding the initially burning vertex and P_2 is a path with two burning leaves of length l_2 excluding the initially burning vertices and $l_2 > 2l_1$, then P_1 will completely burn or turn into a path with one susceptible vertex faster than P_2, if both are left undefended.*

Proof. See proof of Lemma 14. □

The above discussion immediately suggests an algorithm that assigns paths a weight based on their length and number of initially burning ends, then defends threshold vertices according to that weight. It is worth noting that if a path has two initially burning vertices and still has two threshold vertices, then these should be defended immediately after each other. This follows from Lemmas 14 and 15. If one of the two threshold vertices of a path P with length (excluding initially burning vertices) l and x currently non-burning vertices is defended because it would only finish burning out in another $\lceil \frac{x}{2} \rceil$ time steps if left undefended, then P becomes a path with one remaining threshold vertex that will only burn out after another $x - 1$ time steps. This means that if the defenders had thus far been placed on the paths that were going to take the longest to burn out themselves, then P becomes the path with the highest priority as it will take

the longest out of all the remaining paths to burn out and should therefore be defended with the very next available defender.

Theorem 16. *Let $S = (G = (V, E), F, D_{prev})$ be fire path equivalent. Then the polynomial time Algorithm 3 (used together with Algorithm 2 - both on page 28) will give an optimal defence strategy.*

Proof. By Lemma 11 there is always an optimal strategy which only defends threshold vertices, so the choice of defence techniques becomes which order to defend them in. Lemma 12 gives a way of calculating how many vertices will ignite per time step and Lemma 13 shows that minimising this will give an optimal defence strategy. Finally, Lemmas 14, 15 and the discussion that follows then show which order threshold vertices should be defended in to minimise the number of igniting vertices per time step, and thus to find an optimal defence strategy. □

This algorithm is useful in a few cases, and will be modified and reused in Chapters 4 and 5.

Having an algorithm for optimal defence of fire path equivalent games makes it possible to make many conclusions about them, including identifying their M-sets.

Corollary 17. *If a connected graph G has fewer than f vertices of degree strictly greater than two and if an M-set of this graph contains all of the vertices of degree strictly greater than two then the remaining vertices in the M-set are precisely those which maximise the weights as given in Algorithm 3.*

Proof. Since all the high degree vertices are in the M-set, starting fires there will make the graph fire path equivalent. By the proof of Theorem 16 shorter paths in the path representation of a fire path equivalent game lead to less burning, so all weights should be maximal to ensure the most burning. □

Given an algorithm for optimal defence and some ideas about M-sets, it is now possible to calculate exactly how many vertices burn on certain types of graph.

2.7 Towards Finding New mmd Graphs

By considering sets of initially burning vertices on a graph that produce states that are fire path equivalent from time frame 0, i.e. those that include all high degree vertices, we can make conclusions about the graph.

Definition. A graph is an *optimal fire path equivalent graph* if it has the least burning over all combinations (G, F) that are fire path equivalent at time frame 0, given the fire starts at the set covering all high degree vertices that leads to the most damage and the defenders defend optimally.

That is, if it is the most "*mmd*-like" graph that can be fire path equivalent at the beginning of the game for a given value of f.

Lemma 18. *If any graph for which there exists an M-set which includes all high degree vertices is mmd then all mmd graphs with at most f high degree vertices are optimal fire path equivalent graphs.*

Proof. Let G be a graph with at most f vertices of degree at least three. Then there is a way to place the fires so that all high degree vertices are covered and the game is fire path equivalent and the remaining fires can be placed according to Corollary 17. This placement of the fires F may or may not be an M-set. If F is an M-set and G is an optimal fire path equivalent graph then it has the same amount of burning when the fire starts on an M-set and the defenders defend optimally as an *mmd* graph, and is therefore itself *mmd*. If F is not an M-set then starting the fire at an M-set instead and defending optimally would only lead to more burning by definition of an M-set. □

This leaves open the possibility of graphs with strictly more than f vertices of degree strictly greater than two also being *mmd*. We propose the following conjectures:

Conjecture 19. *For sufficiently large values of n, connected graphs with strictly more than f vertices of degree strictly greater than two are not $mmd(n, d, f)$ for any values of d.*

Conjecture 20. *For a connected graph with exactly f vertices of degree at least three, and a given value of d, an $M-$ set of size f consists of precisely these vertices.*

The next step in the research is to calculate the optimal fire path equivalent graphs, as candidate *mmd* graphs. The first two calculations completed for likely candidates are given below.

Lemma 21. *For paths and cycles on n vertices for some $n \in \mathbb{N}$, the number of vertices burned over the course of the fire, given optimal defence and the initial placement of the fires which maximises damage, no longer increases with increasing values of n for $n \geq 2f \cdot k + f$, where:*

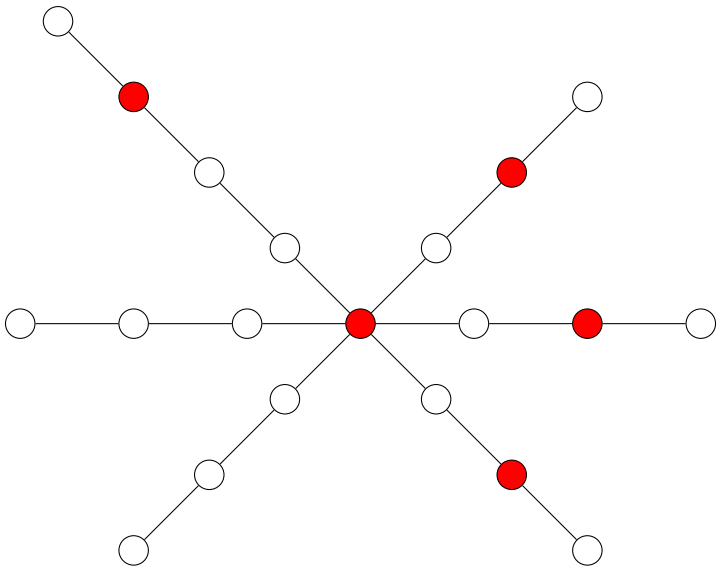
$$k = \begin{cases} \lfloor \frac{2f}{d} \rfloor & \text{if } \frac{2f}{d} \notin \mathbb{N} \\ \frac{2f}{d} - 1 & \text{if } \frac{2f}{d} \in \mathbb{N} \end{cases}$$

Proof. Since optimal defence always involves defending neighbours of the fire, the list of the distances a fire spreads from its initial position on one side spread of a fire beyond its initial position is given by:

$$\underbrace{\overbrace{0, \ldots 0}^{d \text{ times}}, 1, \ldots 1, \ldots, k}_{\text{length 2f}},$$

where the furthest spread of the fire is given by k. Multiplying this by $2f$ gives the distance the fires must be from each other for one fire never to reach another. □

Figure 2.8: The supernova graph on 20 vertices for 5 fires. An M-set is indicated in red.



This bound is in general tight, the only exception is if the fire reaches its maximum distance exactly once - in which case the fire is met by a fire that has travelled distance $k-1$ from its start vertex and the minimum size of graph above which the size of the fire no longer increases is given by $fk(k-1)+f$.

The second type of graph we are investigating are the *supernova* graphs.

Definition. Let $n, f \in \mathbb{N}$. The *supernova* graph for f fires on n vertices is formed by adding $f+1$ neighbours to a central vertex and then adding one new vertex to the end of each arm in turn until there are n vertices in the graph.

An example of a supernova graph is given in Figure 2.8.

Lemma 22. *For supernovae on n vertices for some $n \in \mathbb{N}$, the number of vertices burned over the course of the fire no longer increases with increasing values of n for $n \geq (f-1)(3k+1)+6k+1$ and optimal defence, where k is defined as in Lemma 21 and f fires start on an M-set.*

Proof. Since an M-set is given by placing one fire at the centre of the $f+1$ arms and all other fires on the arms, there will be $(f+1)+2\cdot(f-1)=3f-1$ directions for the fires to spread in. Since optimal defence always involves defending neighbours of the fire, the list of the distances a fire spreads from its initial position on one side spread of a fire beyond its initial position is given by:

$$\underbrace{\overbrace{0,\dots 0}^{d \text{ times}},1,\dots 1,\dots,k}_{\text{length 3f-1}},$$

where the furthest spread of the fire is given by k. Since an M-set is given by placing one fire on the centre and the other fires two thirds of the way along the arms, multiplying this by 3 and adding 1 gives the lengths of the $f-1$ arms with

fires for the fires to never reach another. The two arms that have no fires on can be one vertex shorter without affecting the path of the fire as they are defended first - giving a length of $3k$ each. Finally the vertex in the centre adds 1 to the total number of vertices. □

We conjecture that the supernova graph for f fires on n vertices is optimal fire path equivalent for many values of n, f and d.

2.8 Summary

We have enhanced our understanding of how $K(n, f)$-optimal and *mmd* graphs relate to each other and found some new *mmd* graphs along the way. We studied how both relate to their spanning subtrees and began a new framework for proving that a graph is *mmd*, even if there is more than $K(n, f) - d$ burning over the course if the game. Together with our new, polynomial time algorithm for certain instances of the Firefighter Game, this provides a basis from which to categorise all *mmd* graphs.

```
Input: n, f ∈ ℕ, K(n, f) (calculated elsewhere), a list G_all of all
       connected graphs on n vertices up to isomorphism
Output: A list G_{K(n,f)} of all the K(n, f) optimal graphs
for Graph G in G_all do
    if ∃ a vertex v in G with degree > K(n, f) then
        break;
    else
        initialise a Boolean variable b = True;
        initialise a list N that will contain the non–leaves;
        for vertex v in G do
            if degree the degree of v > 1 then
                append v to N
            end
        end
        for all distinct pairs (v, w) of vertices in N do
            initialise a list D that will contain all the distinct neighbours
             of (v, w);
            for vertex i in neighbours of v or w do
                if i not in D and i not equal to v or w then
                    append i to D
                end
                if |D| > K(n, f) then
                    b = False;
                    break;
                end
            end
        end
    end
    if b = True then
        append G to G_{K(n,f)}
    end
end
return G_{K(n,f)}
```

Algorithm 1: Finding the $K(n, f)$–Optimal Graphs

Input: A graph $G = (V, E)$; a subset of its vertices $F \subset V$
Output: The list of fire components of (G, F)
create an empty list *components*;
for *vertex* $v \in F \cup D$ **do**
  delete v;
end
call the remaining graph $G' = C_1 \cup C_2 \cup \ldots$, where each C_i is a connected component
for *connected component* $C_i \in G'$ **do**
  for $v \in F \cup D$ **do**
    for $w \in C_i$ *st.* $(v, w) \in E$ **do**
      add (v, w) to C_i
    end
  end
end
add C_i to *components*;
return *components*

Algorithm 2: Algorithm for finding the fire components.

Input: A list of fire components $C = C_1, C_2, \ldots$ for an instance of FIREFIGHTER on a graph G; the set of burning vertices F.
Output: A vertex defence strategy, DV
create an empty map *component weights*;
for *component* C_i *in* C **do**
  $weight = \frac{|C_i|}{|\{v \in C_i | v \in F\}|} n^2$;
  while *weight is in image of* component weights **do**
    $weight += 1$
  end
  component weights$(C_i) = weight$;
end
create a list *ordered components* of components ordered by their image in *component weights*;
start playing FIREFIGHTER;
for *defender move in FIREFIGHTER* **do**
  create an empty list *defence priorities*;
  for *component in* ordered components **do**
    create a list *thresh* of threshold vertices v;
    append *thresh* to *defence priorities*;
    defend the first d vertices in *defence priorities*;
  end
end

Algorithm 3: Polynomial algorithm that gives optimal defence for FPE games. It can also be used as a heuristic for other game instances, as will be further explored in Chapter 4.

Chapter 3

Edge-Defence Firefighter

The original version of the Firefighter Problem focuses on defending vertices, which can be used to model vaccination against an infectious disease. We introduce a different version of the Firefighter Problem, which instead focuses on defending edges.

Defending edges can have many different interpretations in terms of disease control. In cases where a vaccine has not been developed or is not yet readily available, restricting the interactions between individuals may be the only form of disease control available. Such social distancing was a common feature of many government responses in the 1918-1920 flu pandemic and in the first years of the COVID-19 pandemic. More generally, if the vertices model population centres rather than individuals, restricting travel between population centres (especially from places where an infectious disease has been detected) is a centuries-old response to pandemics and was widely used during the Black Death of 1346 to 1353.

In addition to cutting off some form of interaction entirely, edge defence can also be used to model many detection and intervention processes which generally allow the interaction but stop an infection from spreading. This could include the testing and quarantine procedures common for the movement of livestock and domestic animals across borders for various diseases including rabies; screening donated blood to avoid spreading hepatitis or HIV to the recipients; or the use of disinfectant mats to prevent people spreading foot and mouth disease or sudden oak death on their shoes.

Beyond disease control, edge defence can be used to model various measures for preventing the spread of invasive species between different areas. Examples include spraying airplanes with pesticide to reduce the risk of invasive insect species hitching a lift.

To the best of our knowledge, there is no published work dedicated to the Edge-Defence Firefighter Problem, as we have defined it. There are however, some results on Edge-Defence Firefighter in Comellas, Mitjana and Peters [9]. Their paper was focused on a model where a message was broadcast (or a virus spread) through a graph starting at time 0. Each infected vertex would stay *active* for A time steps, and at each subsequent time step, for as long as a vertex was active, the infection would spread along k edges of the infected vertex. At the end of the paper, they introduced the idea of defending edges so the infection

cannot spread along them and presented a result on circulant graphs allowing k to equal $\Delta(G)$, the maximum degree of the graph and thus making their broadcast model equivalent to the Edge-Defence Problem we propose here. They found that for $2 \leq k \leq \Delta(G)$, all vertices would be infected if the defence budget was zero (Theorem 5 of [9]). This similarity with the Firefighter problem was not noted in the original paper, but was noted by Michalak and Knowles in [43].

Michalak also provides the other closest comparison to our problem definition in [42], which studies a version of the multiobjective Firefighter Problem with both vertex and edge defence. The paper focuses on metaheuristics and will be more fully discussed in Chapter 4.

The final published work on a similar question is found in Marzouk [38]. This looked at a variant of the Firefighter game where all edges are initially susceptible, and then are progressively randomly assigned to be either burning or defended, with the fire spreading instantly along susceptible edges until it reaches a defended edge.

To introduce Edge-Defence Firefighter, in this chapter we shall focus on basic results comparing it to Classic Firefighter, before moving on to investigating how various published results on Classic Firefighter can be adapted to Edge-Defence Firefighter. We particularly focus on results on games on grids and on *survival rates*, which measure how much of a graph can be saved if the fires break out at random. This has received extensive attention in the literature, and we adapt only some of the many published results to Edge-Defence Firefighter. For more comprehensive overviews of grid and survival rates, see the literature reviews by Finbow and MacGillivray [21] and by Wagner [48]. We conclude the chapter with a new edge defence strategies for finite and infinite square grids, including a new take on survival rates.

3.1 Introductory Results

This chapter will focus on a modified version of the game traditionally called Firefighter, called *Edge-Defence Firefighter* (EDF). To avoid confusion, we shall henceforth refer to the traditional version of the Firefighter Game as *Classic Firefighter* (CF). In Edge-Defence Firefighter, the fire starts and spreads the same way as in Classic Firefighter, but the defenders defend edges rather than vertices. That is, a fire breaks out on a graph $G = (V, E)$ at time 0 on a set $F \subseteq V$ of f vertices. At most d edges are then defended at time 0 and remain defended for the rest of the game. Vertices, once burning, remain so for the rest of the game. At each subsequent time frame, the fire spreads deterministically along all undefended edges of burning vertices to the neighbouring vertices and then at most d more edges can be defended. The game ends when the fire can spread no further.

If an edge (u, v) is defended, where u is burning and v is susceptible, then the fire cannot spread from u to v though (u, v). However, v is not necessarily protected for the rest of the game as a fire may reach it through a different route. We call an edge *susceptible* if it is not defended. We will call an edge a *threshold edge* if it has one burning end and one susceptible end; and define the *threshold*

to be the set of all such edges at a given time step.

Given a graph $G = (V, E)$, and an edge defence budget d an *edge defence strategy* is a function $DE : \mathbb{N} \to DE(t) \subset E$ where $|DE(t)| \leq d$ for all t. Intuitively, we interpret an edge defence as a mapping of times to the edges that the firefighters defend (if they are available for defending) at that time and a vertex defence as a mapping of times to vertices that the firefighters defend. Let $S = (G, F, d)$ be an instance of Classic or Edge-Defence Firefighter on graph $G = (V, E)$ with the set of initially burning vertices $F \subseteq V$ and d defenders. We call a vertex or edge defence strategy for S *optimal* if it leads to the fewest vertices burning at the end of the game out of all possible such defence strategies on S.

For a given instance S and vertex defence strategy DV or an edge defence strategy DE, let $BV(S, DV, t)$ be the set of vertices burning on instance S using defence strategy DV at time step t and $BE(S, DE, t)$ be defined analogously for DE. Let T denote the final time step of the game and let $|BV(S, opt, T)|$ denote the total burning at the end of an instance of the Classic Firefighter, using optimal vertex defence, with $|BE(S, opt, T)|$ being defined analogously for the Edge-Defence Firefighter game.

In general, an optimal edge defence strategy with budget d for instance S will lead to more burning than a vertex defence strategy with the same budget. For instances with a single initially burning vertex v_f, we often use the *distance* of an edge or vertex to v_f. For an edge, we define this as the number of vertices in the shortest path between v_f and the edge's closest endpoint, including v_f itself. For a vertex v, we define it as the number of vertices in the shortest path between v_f and v - including both v_f and v.

Lemma 23. *For any graph $G = (V, E)$, vertex $v_f \in V$, positive integer d, for the instance of Classic Firefighter $S_1 = (G, \{v_f\}, d)$ and the instance of Edge-Defence Firefighter $S_2 = (G, \{v_f\}, d)$, we have $|BV(S_1, opt, T)| \leq |BE(S_1, opt, T)|$.*

Proof. Let DE be any defence strategy for the EDF instance $S_2 = (G, \{v_f\}, d)$. We construct a vertex defence strategy DV that for each edge e in $DE(t)$ defends the non-burning endpoint of e with the shortest distance to a burning vertex. If both ends of e are susceptible and equidistant from a burning vertex, we pick one of them arbitrarily.

We now argue inductively that the vertices burning after t time steps in the vertex defence instance are a subset of the burning vertices after t time steps in the edge defence instance. This is straightforwardly true at time 0, as both instances start with the same set of burning vertices. We now assume this is true at time t and show that this implies it is true at the subsequent time step $t + 1$.

By the inductive hypothesis, $BV(S_1, DV, t) \subset BE(S_2, DE, t)$. Now consider $v \in BV(S_2, DV, t + 1)$. We aim to show $v \in BE(S_2, DE, t + 1)$. If $v \in BE(S_2, DE, t)$ then this is trivial.

Assume $v \notin BE(S_2, DE, t)$. Then by the induction hypothesis, v is not in $BV(S_1, DV, t)$, so v must have at least one neighbour $w \in BV(S_1, DV, t) \subset BE(S_2, DE, t)$.

Suppose that $v \notin BE(S_2, DE, t + 1)$. Then DE must protect (v, w) in time t. However, by construction, this would mean that DV would protect v in time

$t+1$, which contradicts $v \in BV(S_1, DV, t+1)$.

Hence $v \in BV(S_1, DV, t+1) \implies v \in BE(S_2, DE, t+1)$. As this must also hold for an optimal edge defence, the claim is proven. □

The difference between $|BV(S, opt, T)|$ and $|BE(S, opt, T)|$ can be arbitrarily large. Consider a fire starting on two leaves on the star with n vertices and only one defender. Defending the single-non leaf vertex would limit $|BV(S, opt, T)|$ to two (see Figure 3.1), however in the edge defence version of the game, the fire always reaches the non-leaf vertex, spreading to at least $n-2$ vertices (see Figure 3.2). However, with a large enough difference in the defence budgets, an optimal edge defence strategy can perform at least as well as an optimal vertex defence strategy with a smaller defence budget.

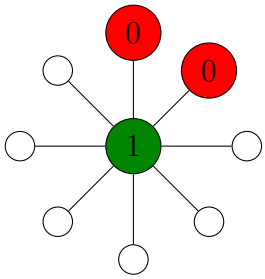


Figure 3.1: An example of vertex defence on a star. Red resp. green vertices burned resp. are defended in the time step indicated. Only two vertices burn in total, given one vertex defender.

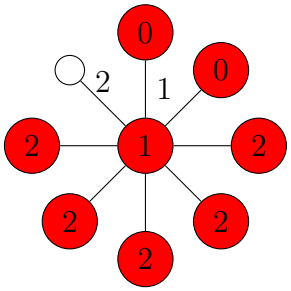


Figure 3.2: An example of edge defence on the same graph as Figure 3.1. Red vertices are burned in the time step indicated. Labelled edges are defended in the indicated time step. Only one vertex can be saved, given one edge defender.

Lemma 24. *Let $G = (V, E)$ be a graph with maximum degree Δ. Let $S_1 = (G, \{v_f\}, d)$ be an instance of Classic Firefighter with d vertex defenders where a single fire breaks out on a vertex $v_f \in V$ and let $S_2 = (G, \{v_f\}, \Delta \cdot d)$ be an instance of Edge-Defence Firefighter where $\Delta \cdot d$ edge defenders defend against a fire breaking out on the same vertex v_f as in S_1. Then $|BV(S_1, opt, T)| \geq |BE(S_2, opt, T)|$.*

Proof. Let DV be any defence strategy for the Classic Firefighter instance $S_1 = (G, \{v_f\}, d)$. We construct a defence DE for the Edge-Defence Firefighter instance $S_2 = (G, \{v_f\}, \Delta \cdot d)$ that defends every edge incident to v for each vertex v in $DV(t)$.

We argue inductively that the vertices burning after t time steps in the edge defence instance are a subset of the burning vertices after t time steps in the vertex defence instance. This is straightforwardly true at time 0, as both instances start with a single burning vertex. We now assume this is true at time t and show that this implies it is true at the subsequent time step $t+1$.

By the inductive hypothesis, $BE(S_2, DE, t) \subset BV(S_1, DV, t)$. Now consider $v \in BE(S_2, DE, t+1)$. We aim to show $v \in BV(S_1, DV, t+1)$. If $v \in BV(S_1, DV, t)$ then this is trivial.

Assume $v \notin BV(S_1, DV, t)$. Then by the induction hypothesis, v is not in $BE(S_2, DE, t)$, so v must have at least one neighbour $w \in BE(S_2, DE, t) \subset BV(S_1, DV, t)$.

Suppose that $v \notin BV(S_1, DV, t+1)$. Then v must have been protected in time t. However, by construction, this would mean that DE would protect every edge incident to v in time t, which contradicts $v \in BE(S_2, DE, t+1)$.

Hence $v \in BE(S_2, DE, t+1) \implies v \in BV(S_1, DV, t+1)$. As this must also hold for an optimal vertex defence, the claim is proven.

□

Intuitively, the reason optimal edge defence strategies do not perform as well as optimal vertex defence strategies with the same budget is that in edge defence, protecting a single edge does not ensure that the fire will not follow another path to its incident vertices. This is avoided in graphs without cycles as there is a direct 1-to-1 correspondence between defending a vertex v and defending the *pendant edge of* v, the edge incident to v and on the shortest path between the single initial fire v_f and v. For games with a single fire, defending any given vertex v on a tree T has the effect of ensuring that the fire never reaches v and any other vertices u if and only if v lies on the unique path between u and v_f, as there is no path of susceptible vertices the fire could reach u from. Defending the pendant edge of v will have the same effect as defending v itself. For a vertex defence strategy DV, we shall call this analogous strategy DE_v.

Observation 25. *For any tree $T = (V, E)$, vertex $v_f \in V$, positive integer d, for an instance of Classic Firefighter $S_1 = (T, \{v_f\}, d)$ and an instance of Edge-Defence Firefighter $S_2 = (T, \{v_f\}, d)$, we have $BV(S_1, DV, t) = BE(S_2, DE_v, t)$.*

Note that this observation does not hold if multiple vertices are burning in time step zero. Also, for instances where multiple vertices are burning in time step 0, DE_v may not be well defined for a vertex defence strategy DV. For instance, if DV defends a vertex v at time t which is equidistant along separate paths from two initially burning fires $f_1, f_2 \in F$, then it is not clear whether DE_v would defend the edge of v on the path to f_1 or the edge of v on the path to f_2 at time t.

Observation 26. *For any tree $T = (V, E)$, set of vertices $F \subset V$, with $|F| > 1$ and positive integer d, for an instance of Classic Firefighter $S_1 = (T, F, d)$ and an instance of Edge-Defence Firefighter $S_2 = (T, F, d)$, we have $|BV(S_1, opt, T)| \leq |BE(S_2, opt, T)|$.*

Now we have established some basic results on fires starting on a single given vertex v_f, we can investigate what we expect to happen if v_f is chosen uniformly at random. The idea of the *survival rate* of a graph was developed in Cai and Wang [7] for Classic Firefighter.

Definition. The *k-survival rate* of a graph is defined as: $\rho_k(G) := \frac{1}{n^2} \sum_{v_f \in V} n - |BV((G, \{v_f\}, k), opt, T)|$.

We modify this definition in the obvious way for Edge-Defence Firefighter:

Definition. The *k-survival rate (edge defence)* of a graph is defined as:

$$\rho_{e,k}(G) := \frac{1}{n^2} \sum_{v_f \in V} n - |BE((G, \{v_f\}, k), opt, T)|. \tag{3.1}$$

If $k = 1$, we may write ρ_e for $\rho_{e,1}$

In general, it has been shown that the survival rate of any given graph is likely to be very low. To formalise this, we say *almost all* graphs have property $\mathscr{P}$ if the proportion of all graphs on n vertices with property $\mathscr{P}$ tends to 1 as n tends to infinity.

Lemma 27 (Theorem 2 of Wang, Finbow and Wang [49])**.** *For $\varepsilon > 0$, almost all graphs have a survival rate less than ε with a single defender.*

As could be expected from the above lemmas, the edge survival rate is no higher.

Lemma 28. *For $\varepsilon > 0$, almost all graphs have an edge survival rate less than ε with a single defender.*

Proof. By Lemma 23, there will always be at least as much burning for any defence strategy on any instance of the edge defender game as there is on vertex defence. The result then follows immediately from Lemma 27. □

In addition to survival rates for almost all graphs being low, $|BV(S, opt, T)|$ is also very hard to compute. The decision version of the Classic firefighter problem is as follows:

CLASSIC FIREFIGHTER
Instance: A connected graph $G = (V, E)$, a set $F \subset V$ and two natural numbers $d, k \in \mathbb{N}$
Question: If each vertex $v_f \in F$ is burning at the initial step 0, is there a strategy for d defenders protecting vertices to save at least k vertices?

In general, this is computationally hard. There is a strong dividing line between graphs for which Classic Firefighter is computationally difficult and those for which it is not:

Theorem 29 (Theorems 2 and 4 of [20])**.** *CLASSIC FIREFIGHTER with a single defender is NP-Complete on trees of maximum degree three when a single fire starts at a vertex of degree three, but polynomial-time for graphs with maximum degree three if the fire breaks out at a vertex of degree two.*

This was generalised for games with multiple defenders in Bazgan, Chopin and Ries [2]:

Theorem 30 (Propositions 1 and 2 of [2])**.** *CLASSIC FIREFIGHTER with $d \geq 2$ defenders is NP-hard on trees of maximum degree $d + 3$ with a single fire, but polynomial-time on trees with maximum degree $d + 2$ when the fire breaks out at a vertex of degree $d + 1$.*

This transfers easily to Edge-Defence Firefighter. By Observation 25, it is clear that any instance of CLASSIC FIREFIGHTER on trees can be reduced to an instance on EDGE-DEFENCE FIREFIGHTER on trees. Hence, as the former problem is NP-hard on trees of maximum degree 3 rooted at a vertex of degree 3 by Theorem 2 [20], so is the later problem for the same class of graphs. Furthermore, as all instances of vertex-degree firefighter are in NP, it is NP-complete.

Theorem 4 of [20] gives an optimal vertex defence strategy for trees of maximal degree 3 rooted at a vertex of degree 2, calculated in polynomial time. Using the arguments in the discussion of Observation 25, this can be converted in polynomial time into an optimal edge defence strategy. The same logic applies to the results in Theorem 30, thus partially answering the decision question.

EDGE-DEFENCE FIREFIGHTER (EDF)
Instance: A connected graph $G = (V, E)$, a set $F \subset V$ and two natural numbers $d, k \in \mathbb{N}$
Question: If each vertex $v_f \in F$ is burning at the initial step 0, is there a strategy for d defenders protecting edges to save at least k vertices?

Observation 31. *EDF with a single defender and a single fire is NP-Complete on trees of maximum degree three when the fire starts at a vertex of degree three, but polynomial-time for graphs of maximum degree three if the fire breaks at a vertex of degree two. EDF with $d \geq 2$ defenders is NP-hard on trees of maximum degree $d+3$ with a single fire, but polynomial-time on trees with maximum degree $d+2$ when the fire breaks out at a vertex of degree $d+1$.*

3.2 The Hexagonal Grid

We shall now begin adapting some results form the literature on Classic Firefighter to Edge-Defence Firefighter. This section will focus on results of Dean et al. [14] on the infinite hexagonal grid. We define this by embedding the infinite hexagonal grid into the plane, with vertices (i, j) where $i, j \in \mathbb{Z}$ and $(i \bmod 2, j \bmod 6) \in \{(0,0), (0,2), (1,3), (1,5)\}$. Given vertices (i, j) and (k, l), an edge $e = ((i, j), (k, l))$ is then in the hexagonal grid if and only if exactly one of the following conditions is met:

1. $i = k$ and $|j - l| = 2$
2. $i - k = j - l = -1$
3. $i - k = -1$, $j - l = 1$

Intuitively, we tile the plane with a section of the hexagonal grid and the conditions ensure an edge consists of two unique vertices in the same tile. The words *left, right, higher, lower, above* etc. will be defined the usual way with respect to this coordinate system. In this section, we slightly adapt the vertex defence technique described in Dean et al. [14] to produce a strategy for edge defence Firefighter. This was used in their following result:

Theorem 32 (Theorem 1.2 of Dean et al. [14])**.** *Using one extra defender at time $2x'$, then one defender every turn is sufficient to contain the fire on the infinite hexagonal grid with a single initially burning vertex.*

To the best of our knowledge, this is the lowest known bound for the number of vertex defenders it takes to contain a fire on the infinite hexagonal grid. Both the strategy in Dean at al. [14] and the strategy given here are *online strategies* - we do not have to know in advance when the extra defender can be used. We adapt the strategy in Dean et al. [14] by defending the edge adjacent to the vertices they defend and on the unique shortest path between the vertex and the fire in each time step.

Note that to simplify the proof and the calculations of coordinates of burning vertices, in this section (only), we shall use a different definition of the time steps in the Firefighter game. The game will start with one vertex burning at time step 1. It will then continue with defenders defending on even time steps and the fire spreading on odd time steps until the end of the game.

We define x' so that the extra defender is received at time step $2x'$. The following proof will explicitly give a defence strategy that has one extra defender at time step $2x$, where x is the least odd integer such that $x \geq x'$. If $x > x'$, then a defence strategy with one extra defender at $2x'$ can easily be derived from the below by defending either of the two vertices or edges that are described as being defended at $2x$ at time $2x'$ instead.

Theorem 33. *Using one extra edge defender at time $2x'$, then one defender every turn is sufficient to contain the fire on the infinite hexagonal grid with a single initially burning vertex.*

The defence consists of several parts, the validity of which shall be proven separately. The outline of the defence is:

1. With one defender per even time step, form two walls of defenders, so that if this process were continued indefinitely $\frac{2}{3}$ of the grid would be protected, with the fire contained in a wedge shape with an angle of 120°.

2. At the time step with the extra defender, bend these walls so they become parallel to each other.

3. With one defender per even time step for the rest of the game, continue until this strip is sufficiently long.

4. Bend one wall of the strip four times into a spiral that meets back up with the previously defended vertices, containing the fire.

This results in the fire being contained within a snail shaped area.

The vertex version of the defence strategy as given in Dean et al. [14] is quoted below. It contains the fire by Theorem 1.1 of the same paper. The numbering corresponds to the outline above. We shall call this defence strategy DV_{hex}.

1. For $0 \leq k \leq \frac{x-3}{2}$, on turn $t = 4k+2$ protect vertex $(1-k, -1-3k)$, on turn $t = 4k + 4$ protect $(1 - k, 3 + 3k)$.

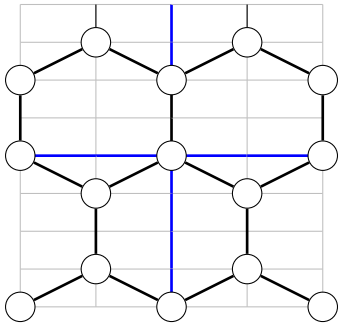


Figure 3.3: The coordinate system used in this section. Thick blue lines represent the axes.

2. At $t = 2x$ protect vertices $(1 - \frac{x-1}{2}, -1 - \frac{3(x-1)}{2})$ and $(1 - \frac{x-1}{2}, 3 + \frac{3(x-1)}{2})$.
3. For $0 \leq k \leq \frac{15x+11}{2}$ on turn $t = 2x + 4k + 2$ protect $(-\frac{x+1}{2} - 2k, -\frac{3(x+1)}{2})$, on turn $t = 2x + 4k + 4$ protect $(-\frac{x+1}{2} - 2k, 2 + \frac{3(x+1)}{2})$.
4. We then begin curving the fire back towards its starting point with a spiral of defenders, centred on $c = (-15x - 13, 0)$.
   (a) For $0 \leq k \leq \frac{x-1}{2}$, on turn $4k + 32x + 28$ protect $(\frac{-31x-27}{2} - 3k, \frac{-3x-3}{2} + 3k)$, on turn $4k + 32x + 30$ protect $(\frac{-31x-31}{2} - 3k, \frac{-3x+1}{2} + 3k)$. Finally, on turn $34x + 30$ protect $(-17x - 15, 0)$. This bends the bottom edge of the strip by 30°. We call vertices of this form for $k \geq 0$ *Line* 1.
   (b) For $0 \leq k \leq x$ on turn $4k+34x+32$ protect $(-17x-15, 6k+2)$, on turn $4k + 34x + 34$ protect $(-17x - 15, 6k + 6)$, this bends the outer line of defenders by 60°. We call vertices of this form for $k \geq 0$ *Line* 2.
   (c) For $0 \leq k \leq 2x + 1$ on turn $t = 4k + 38x + 36$ protect $(-17x - 13 + 3k, 6x + 8 + 3k)$, on turn $t = 4k + 38x + 38$ protect $(-17x - 12 + 3k, 6x + 9 + 3k)$, this bends the outer line of defenders by 60°. We call vertices of this form for $k \geq 0$ *Line* 3.
   (d) For $0 \leq k \leq \frac{7x+3}{2}$, on turn $t = 4k+46x+44$ protect $(-11x-8+3k, 12x+11-3k$, on turn $t = 4k + 46x + 46$ protect $(-11x - 6 + 3k, 12x + 9 - 3k)$, this bends the outer line of defenders by 60°. We call vertices of this form for $k \geq 0$ *Line* 4.

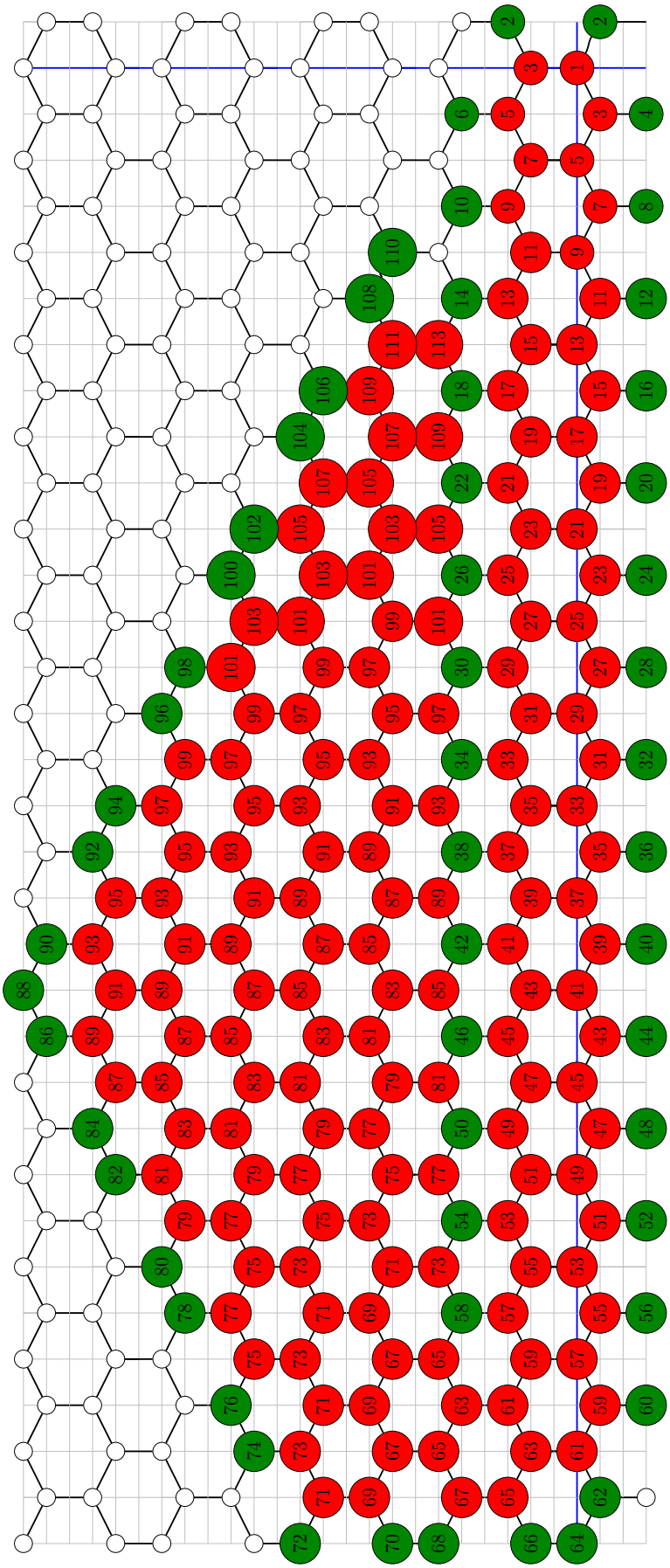


Figure 3.4: An example of DV_{hex} with $2x' = 2$. Axes are marked blue.

Relating Vertex Defence to Edge Defence

Let DV_a be a vertex defence strategy for an instance of Classic Firefighter with a single fire starting on $v_f \subset G$. If, for all vertices v defended by DV_a, there exists a unique edge incident to v which lies on the shortest path between v and v_f, we call this edge e_v. For such vertex defence strategies, we can define an edge defence strategy DE_a, which is derived from DV_a by defending $e_{v(i)}$ at time step i.

In this section, we seek to prove that for DV_{hex} the vertex defence strategy on the infinite hexagonal grid described above, DE_{hex} is valid and will contain the fire. We first define several areas of the graph. We define $V_d := \{v : v \in DV_{hex}(t), 0 \leq t \leq T\}$, the set of vertices which are defended by DV_{hex}. We are also interested in $B \subset V_d$, which we define as the vertices in V_d which at the end of a game, having used strategy DV_{hex}, have at least one burning neighbour and at least one susceptible neighbour. At the end of the game, the vertices in B will form a border between vertices that were burned or defended during the game and the rest of the grid. The end of the game is defined as the last time step during which an edge or vertex can be defended or a new vertex burns using a given defence strategy and denote it by T. We denote by I the set of vertices in the finite, connected component formed by removing B which contains all burning vertices at the end of the game if you use defence strategy DV_{hex}. And its subset $I_t \subset I$, the set of vertices burning at time t under the same conditions.

Observation 34. *Let DV be a vertex defence strategy that contains a fire starting at f on G. Let B be the set of vertices which, having used strategy DV, are defended and have at least one burning neighbour and at least one susceptible neighbour. Let I be the set of vertices in the finite, connected component formed by removing B. Then any edge defence strategy which ensures that no vertices outside I are burning at time t for all $1 \leq t \leq T$ will also contain the fire.*

Finding ways to differentiate between vertex defence strategies that can be easily adapted to edge defence strategies yielding the same result and those that cannot is one possible future research direction.

While this Observation will hold for all instances of the Firefighter Game, for the rest of this segment we shall only be considering an instance of Firefighter on the infinite hexagonal grid with the vertices and edges as defined above and a single initial fire at $(0, 0)$. We break the defence strategy down into different stages, starting with the stage from the initial defence turn until the second defender is available at time step $2x$. The following observation, taken from Dean et al. [14], will be useful:

Observation 35 (Observation 2.1 in [14])**.** *Let $v(1), v(2), \ldots, v(x-1)$ be the vertices protected in the manner described in part 1 of DV_{hex}. For all r with $1 \leq r \leq x-1$, we have $dist(f, v(r)) = r$.*

We can now prove that the first part of the edge defence strategy contains the fire within the same bounds as its vertex defence analog.

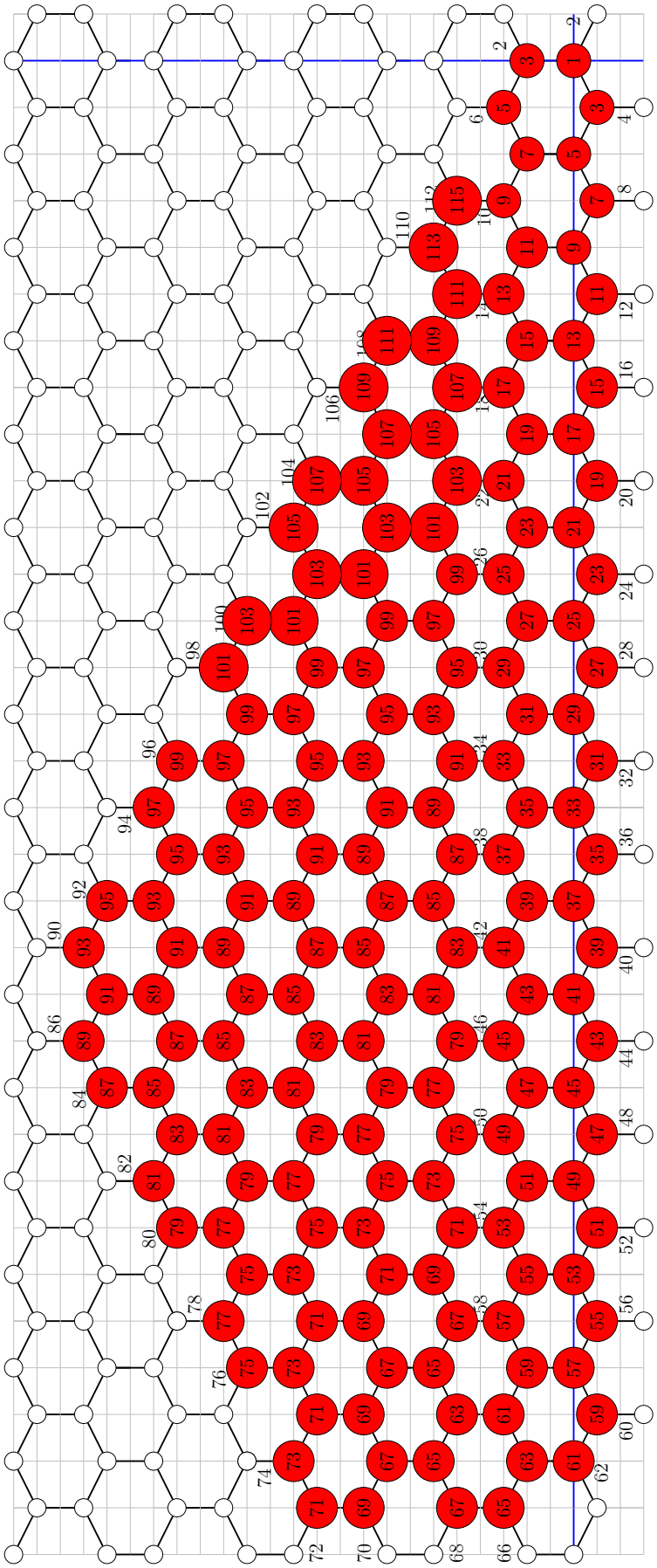


Figure 3.5: An example of DE_{hex} with $2x' = 2$. Axes are marked blue.

Using the above definition of DE_{hex}, for $0 \leq k \leq \frac{x-3}{2}$, it will protect edge $((-k,-3k),(1-k,-1-3k))$ on turn $t = 4k+2$ and protect edge $((-k,2+3k),(1-k,3+3k))$ on turn $t = 4k+4$. This culminates in protecting $((-\frac{x-1}{2},-1-\frac{3(x-1)}{2}),(1-\frac{x-1}{2},-1-\frac{3(x-1)}{2}))$ and $((-\frac{(x-1)}{2},2+\frac{3(x-1)}{2}),(1-\frac{x-1}{2},3+\frac{3(x-1)}{2}))$ at the time step with the extra defender.

Lemma 36. *The defence strategy DE_{hex} contains the fire within $I \setminus V_d$ for $2 \leq t \leq 2x$.*

Proof. The vertices of the form $(1-k,-1-3k)$ for $k \geq 0$ have a unique neighbour $(-k,-3k)$ on the shortest path between themselves and $(0,0)$ and similarly vertices $(1-k,3+3k)$ have a unique neighbour $(-k,2+3k)$ on the shortest path between themselves and the fire. Hence for each vertex v in V_d and each t in this stage of the game, v has a unique edge adjacent to I_t so the edge defence strategy DE_{hex} is well defined, using the definition at the start of Subsection 3.2.

At time 2 we defend the edge $((0,0),(1,-1))$, this prevents $(1,-1)$ from burning in time step 3, hence the claim holds as only $(0,0),(-1,0)$ and $(0,1)$ are burning.

Now assume that the claim holds for all time steps $2 \leq t < 2x$.

Case 1: $t = 4k+2$ for some $4 \leq k < 2x$. The next vertex to be defended by DV_{hex} is $v(d) = (1-k,3+3k)$. By Observation 35, it is not yet burning at this stage in the game using DE_{hex}. The neighbours of $v(d)$ are $(1-k,5+3k),(-k,2+2k)$ and $(2-k,2+2k)$, as can be seen by examining the conditions for the existence of a vertex or edge above. By the induction hypothesis, $(1-k,5+3k)$ and $(2-k,2+2k)$ are not burning but $(-k,2+2k)$ is, hence defending $e_{v(d)} = ((1-k,3+3k),(-k,2+2k))$ at $t+2$ will prevent $v(d)$ from burning in time step $t+3$. Furthermore, since no vertices outside V_d were burning at t by the induction hypothesis and the only vertex at distance $\frac{t}{2}$ from f protected by DV_{hex} was $v(d)$, only vertices in I_{t+3} can burn at step $t+3$. The proof for $t = 4k+4$ is similar, with the next vertex defended by DV_{hex} being $(1-k,-1-3k)$ with non-burning neighbours $(1-k,-3-3k)$ and $(2-k,-3k)$ and one burning neighbour at $(-k,-3k)$, so defending $e_{v(d)}$ contains the fire within the appropriate bounds. Hence overall, the claim holds by induction.

□

We next prove the fire is contained during the strip building phase of the defence strategy. So for $0 \leq k \leq \frac{15x+11}{2}$ we defend $((\frac{x-1}{2}-2k,1-\frac{3(x+1)}{2}),(-\frac{x-1}{2}-2k,-\frac{3(x+1)}{2}))$ on turn $t = 2x+4k+2$ and on turn $t = 2x+4k+4$ we defend $((-\frac{x-1}{2}-2k,1+\frac{3(x+1)}{2}),(-\frac{x-1}{2}-2k,2+\frac{3(x+1)}{2}))$. We shall use the following observation from Dean et al. [14]:

Observation 37 (Observation 2.2 in [14])**.** *Let $v(x+1),v(x+2),\ldots,v(16x+13)$ be the vertices protected in the manner described in part 2 of DV_{hex}. For all r with $x+1 \leq r \leq 16x+13$, we have $dist(f,v(r)) = r$.*

Lemma 38. *The defence strategy DE_{hex} contains the fire within $I \setminus V_d$ for $2x \leq t \leq 32x+26$.*

Proof. Intuitively, this proof follows the same technique as the proof of Lemma 36.

The vertices of the form $(-\frac{x-1}{2} - 2k, -\frac{3(x+1)}{2})$ for $k \geq 0$ have a unique neighbour $(-\frac{x-1}{2} - 2k, 2 - \frac{3(x+1)}{2})$ on the shortest path between themselves and $(0,0)$ and similarly vertices $(-\frac{x-1}{2} - 2k, 2 + \frac{3(x+1)}{2})$ have a unique neighbour $(-\frac{x-1}{2} - 2k, \frac{3(x+1)}{2})$ on the shortest path between themselves and the fire. Hence for each vertex v in V_d and each t in this stage of the game, v has a unique edge adjacent to I_t so BE is well defined.

By Lemma 36, at time $2x$ the fire is contained within $I \setminus V_d$. At time $2x$ we defend the edges $((-\frac{x-1}{2}, -1 - \frac{3(x-1)}{2}), (1 - \frac{x-1}{2}, -1 - \frac{3(x-1)}{2}))$ and $((-\frac{(x-1)}{2}, 2 + \frac{3(x-1)}{2}), (1 - \frac{x-1}{2}, 3 + \frac{3(x-1)}{2}))$. This prevents $(-\frac{x-1}{2}, -\frac{3(x+1)}{2})$ and $(-\frac{x-1}{2}, 2 + \frac{3(x+1)}{2})$ from burning in time step $2x+1$, as these were the only vertices at distance $4x$ defended by DV_{hex} the claim holds for $k = 0$. They were now previously burning by Observation 37.

Now assume that the claim holds for all time steps $2x \leq t < 32x + 26$. Case 1: $t = 2x + 4k + 2$ for some $2 \leq k < \frac{15x+11}{2}x$. The next vertex to be defended by DV_{hex} is $v(d) = (-\frac{x-1}{2} - 2k, 2 + \frac{3(x+1)}{2})$. By Observation 37, it is not yet burning at this stage in the game using DE_{hex}. The neighbours of $v(d)$ are $(-\frac{x-1}{2} - 2k, \frac{3(x+1)}{2}), (1 - \frac{x-1}{2} - 2k, 3 + \frac{3(x+1)}{2})$ and $(-1 - \frac{x-1}{2} - 2k, 3 + \frac{3(x+1)}{2})$. By the induction hypothesis, $(1 - \frac{x-1}{2} - 2k, 3 + \frac{3(x+1)}{2})$ and $(-1 - \frac{x-1}{2} - 2k, 3 + \frac{3(x+1)}{2})$ are not burning but $(-\frac{x-1}{2} - 2k, \frac{3(x+1)}{2})$ is, hence defending $e_{v(d)} = ((-\frac{x-1}{2} - 2k, 2 + \frac{3(x+1)}{2}), (-\frac{x-1}{2} - 2k, \frac{3(x+1)}{2}))$ at $t+2$ will prevent $v(d)$ from burning in time step $t+3$. Furthermore, since no vertices outside V_d were burning at t by the induction hypothesis and the only vertex at distance $\frac{t}{2}$ from f protected by DV_{hex} was $v(d)$, only vertices in I_{t+3} can burn at step $t+3$. The proof for $t = 2x+4k+4$ is similar, with the next vertex defended by DV_{hex} being $(-\frac{x-1}{2} - 2k, -\frac{3(x+1)}{2})$ with non-burning neighbours $(1 - \frac{x-1}{2} - 2k, -1 - \frac{3(x+1)}{2})$ and $(-1 - \frac{x-1}{2} - 2k, -1 - \frac{3(x+1)}{2})$ and one burning neighbour at $(-\frac{x-1}{2} - 2k, 2 - \frac{3(x+1)}{2})$, so defending $e_{v(d)}$ contains the fire within the appropriate bounds. Hence overall, the claim holds by induction. □

The final stage of the defence strategy is where we see the only significant deviation from the results of the vertex defence strategy. All vertices forming the upper edge of the parallel strip - ie $(-\frac{x-1}{2} - 2k, 2 + \frac{3(x+1)}{2})$ for $0 \leq k \leq \frac{15x+11}{2}$ - will burn using DE_{hex} as the fire can reach them from their two neighbours $(1 - \frac{x-1}{2} - 2k, 3 + \frac{3(x+1)}{2})$ and $(-1 - \frac{x-1}{2} - 2k, 3 + \frac{3(x+1)}{2})$, whereas they are protected by DV_{hex}. However, as these vertices are not in B, this does not affect the conditions of Lemma 34.

The defence strategy DE_{hex} for the rest of the game is as follows:

(a) For $0 \leq k \leq \frac{x-1}{2}$, on turn $4k + 32x + 28$ protect $((\frac{-31x-27}{2} - 3k, \frac{-3x-3}{2} + 3k + 3), (\frac{-31x-27}{2} - 3k, \frac{-3x-3}{2} + 3k))$, on turn $4k + 32x + 30$ protect $((\frac{-31x-31}{2} - 3k + 1, \frac{-3x+1}{2} + 3k + 1), (\frac{-31x-31}{2} - 3k, \frac{-3x+1}{2} + 3k))$.

(b) For $0 \leq k \leq x$ on turn $4k + 34x + 32$ protect $((-17x - 14, 6k + 3), (-17x - 15, 6k + 2))$, on turn $4k + 34x + 34$ protect $((-17x - 14, 6k + 5), (-17x -$

$15, 6k + 6)$).

(c) For $0 \leq k \leq 2x+1$ on turn $t = 4k+38x+36$ protect $((-17x-13+3k, 6x+5+3k), (-17x-13+3k, 6x+8+3k))$, on turn $t = 4k+38x+38$ protect $((-17x-11+3k, 6x+8+3k), (-17x-12+3k, 6x+9+3k))$.

(d) For $0 \leq k \leq \frac{7x+3}{2}$, on turn $t = 4k+46x+44$ protect $((-11x-8+3k, 12x+8-3k), (-11x-8+3k, 12x+11-3k))$, on turn $t = 4k+46x+46$ protect $((-11x-7+3k, 12x+8-3k), (-11x-6+3k, 12x+9-3k))$.

Observation 39 (Observation 2.4 and Lemma 2.5 in [14])**.** *Let $v(16x+14), v(16x+15), \ldots, v(30x+26)$ be the vertices protected in the manner described in part 4 DV_{hex}. For all r with $16x+14 \leq r \leq 30x+26$, we have that $v(r)$ is not burning at time $2r$ using defence strategy DV_{hex}.*

Lemma 40. *The defence strategy DE_{hex} contains the fire within I for $32x+26 < t \leq T$.*

Proof. This follows using the same technique as the proof of Lemma 38, so for brevity we shall omit some details, giving only neighbours of the vertices defended by DV_{hex} Let Lines 1 to 4 be as defined above.

The vertices in Line 1 of the form $v = (\frac{-31x-27}{2}-3k, \frac{-3x-3}{2}+3k)$, for some $k \geq 0$ have neighbours $(\frac{-31x-27}{2}-3k, \frac{-3x-3}{2}+3k+2), (\frac{-31x-27}{2}-3k+1, \frac{-3x-3}{2}+3k-1)$ and $(\frac{-31x-27}{2}-3k-1, \frac{-3x-3}{2}+3k-1)$. $(\frac{-31x-27}{2}-3k, \frac{-3x-3}{2}+3k+2)$ is the only vertex on the shortest path to $I(t)$ so protecting their edge will save v.

By choice of the coordinate system, the vertices in Line 1 have one unique neighbour above them which is not itself a member of Line 1; the vertices of Line 2 have one unique neighbour to their left; and the vertices of Line 3 and Line 4 have one unique vertex below them which is not itself a member of their own line. Note that as using V_d contains the fire with Lines 1 and 2 to the right of f; and Lines 3 and 4 above it, for each vertex v in V_d and each t in this stage of the game, v has a unique edge adjacent to I_t so DE_{hex} is well defined.

By Lemma 38, at time step $32x+26$ no vertex outside $I \setminus V_d$ is burning. The claim then follows from Lemma 2.5, which proves the validity of DV_{hex} for this stage of the game of [14] and the induction technique approach in Lemmas 36 and 38. □

This completes our proof of Theorem 33. The result is surprising as Edge-Defence Firefighter can take a much larger defence budget to contain a fire on other grids than is required for Classic Firefighter. We shall return to this when we look at square grids in Section 3.6.

3.3 Fractional Online Firefighting on Edges

In this section, we shall only be considering games with the fire breaking out on one initial vertex v_f. Moreover, we shall only be considering *online games* where the number of defenders available in a given time step is only known at the start of that time step. For this section and the rest of the thesis, we once again use the notation $n = |V|$.

Fractional online firefighting was originally introduced in Coupechoux et al. [11]. In their version of fractional firefighting, a fraction of a vertex can be defended and it is possible for only a fraction to be burning. For a time step $i \geq 0$ we denote the part of vertex v that is burning by $b_i(v)$ and the part that is defended by $d_i(v)$. It is possible for a vertex to be partially defended and partially burning simultaneously and it is possible for the fraction of a vertex that is defended or burning to increase further in later time steps. We define the *cumulative protection* of a vertex v at time i as the sum of the protection it has been given up to and including time i. Once a particular fraction of a vertex is defended or burning at least that fraction will remain so for the rest of the game. The game proceeds largely as in Classic Firefighter - a fire breaks out at the start of the game, then the game alternates between vertices being defended and the fire spreading to insufficiently defended neighbours of burning vertices and ends when the fire can spread no further. However, there are additional rules:

1. The sum of the burning and defence of a given vertex cannot exceed one.

2. At each time step, the new burning on vertex v is the maximum of its previous burning or the burning of any one of its neighbours, minus its protection.

Note that if the defender always defends whole vertices, then this has the same game tree and will have the same results as the Classic Firefighter game.

The original paper presented several results on this version of Firefighter on simple, connected graphs. We will show that these can be applied to an edge defence version of Fractional Firefighter. For an edge $(v, w) \in G$ let $d_i(v, w)$ be the part of (v, w) which is defended by the end of time step i and $b_i(v)$ denote the part of v which is burning by the end of time step i. We use the following adaptation of the rules in Coupechoux et al. [11], designed to make the game be analogous to the vertex defence version when the game is on a tree:

1. The protection on any given edge or the burning on any given vertex cannot exceed one. However, the protection on an edge places no limit on the burning on its incident vertices and vice versa.

2. At each time step, the new burning on vertex v is the maximum of its previous burning or the burning of any one of its neighbours, minus the cumulative protection on the edge joining the neighbour to v, ie $b_{i+1}(v) = b_i(w) - d_{i+1}(v, w) > b_i(v)$.

The first lemma we shall adapt from Coupechoux et al. [11] corresponds to their Proposition 1 and its proof. It relates to when the game terminates.

Let L_f be the length of a longest induced path on a finite graph G with one end at the unique initially burning vertex v_f. We shall adapt the proof technique used in Coupechoux et al. [11] for the following lemma:

Lemma 41 (Proposition 1 of Coupechoux et al. [11])**.** *Fractional firefighter on a finite graph will terminate in at most L_f time steps.*

The proof of Lemma 41 made use of the following claim:

Lemma 42 (From proof of Proposition 1 of Coupechoux et al. [11])**.** *Let G be a finite graph. For any time step $i > 0$ and for any vertex v with $b_i(v) > b_{i-1}(v)$, there is an induced path in G starting at v_f and ending at v such that the burning of each vertex along the path is non increasing from v_f to v.*

We shall show the following, narrower claim for Edge-Defence Firefighter:

Lemma 43. *Fractional edge defence firefighter on a finite tree will terminate in at most L_f time steps.*

Proof. We adapt the proof technique in Proposition 1 of Coupechoux et al. [11] and simplify for trees. Let T be a tree. We first show that for any time step $i > 0$ and for any vertex v with $b_i(v) > b_{i-1}(v)$, there is an induced path in T starting at v_f and ending at v such that the burning on each vertex is non increasing from v_f to v.

We induct on the time steps, i. For $i = 1$, if $b_1(v) > b_0(v)$ then v is a neighbour of v_f and $b_0(v_f) = 1 = b_1(v_f) \geq b_1(v)$.

We now suppose the property holds for time steps up to i and $b_{i+1}(v) > b_i(v)$ for some $v \in T$. Then v must have received some burning from a neighbour w, following the rule $b_{i+1}(v) = b_i(w) - d_{i+1}(v, w) > b_i(v)$. Any given vertex can only be burned by one neighbour, or else some subset of the union of the shortest paths linking any two of the multiple neighbours sending burning to v and v_f would form part of a cycle.

Furthermore, we must have $b_i(w) > b_{i-1}(w)$ or else we would have the following contradiction.

$$b_i(v) \geq b_{i-1}(w) - d_i(v, w) = b_i(w) - d_i(v, w) \geq b_i(w) - d_{i+1}(v, w) > b_i(v)$$

We now apply the induction hypothesis to w to find an induced path P from v_f to w with non-increasing burning. Finally, we can see that v shares no edges with u for $u \in P$ as T is a tree, so adding the edge (w, v) to P yields the claim.

Hence, if for any vertex v the burning increases at time step $i + 1$, we must have $i + 1 \leq L_f(G)$. So the fire can no longer spread at time step $L_f(T) + 1$. □

This is not true for graphs containing cycles.

Observation 44. *Fractional Edge-Defence Firefighter on a finite graph will not necessarily terminate in at most L_f time steps.*

Intuitively, this is because it is possible to have a small amount of burning reach a vertex v far from v_f at time $d(v, v_f)$, but for this burning to increase in later time steps as it follows a more circuitous route through the graph. An example of this is given in Figure 3.6.

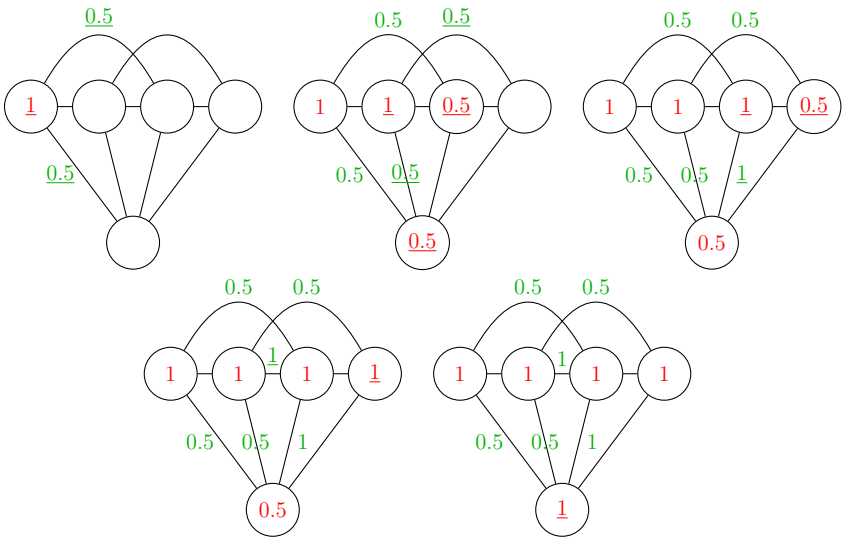


Figure 3.6: A game of fractional edge defence firefighter where the game lasts more than $L_f = 4$ time steps. There is one diagram per time step starting at $t = 0$ and changes to burning or defence are underlined.

A Defence Algorithm on Trees

We now focus on a defence algorithm for trees. Let T be a tree. We call the maximum length of a path from the initially burning vertex v_f to a leaf the *height* of T and denote it by $h(T)$. In this section, we show that the version of Edge-Defence Fractional Firefighter proposed above and the original version of Fractional Firefighter have many similarities on trees, and hence that the results from Coupechoux et al. [11], presenting a $\frac{1}{2}$-competitive defence algorithm for trees carry over to this version of the game.

From Lemma 43, we know that the game will end in at most $h(T)$ time steps. Furthermore, using the proof of 43, we know that burning is non-increasing along any path from the root of a tree v_f at any stage of the game and that for every vertex v in $Nbd_j(f)$ - the set of vertices distance j from v_f - the burning in v cannot increase in any time step $i > j$. Hence, no optimal defence strategy will protect the edge incident to v on the unique path connecting to to v_f after time j. As with the vertex defence version, allocating protection to an edge e will always result in less total burning at the end of the game than allocating the same amount of protection to an edge e' further from the fire such that e lies on the unique path between e' and v_f.

We call an edge e incident to vertex v *upstream to* v if it lies on the unique path between v and v_f. Each vertex v in a tree has at most one upstream edge, which we label $e(v)$. For a vertex defence strategy DV, let DE_v be the related edge defence strategy that defends the upstream edge of each vertex defended by DV at the same time step.

A parallel between the edge defence and vertex defence versions is that, as we exclude any defence strategy that defends an edge e_v if v already has a positive amount of burning, we have $d_i(e_v) + b(v) \leq 1$ for all time steps i - replicating the rule for vertex defence.

Lemma 45. *For Fractional Firefighter, for any tree $T = (V, E)$, vertex $v_f \in V$, positive integer d, $BV(S, DV, t, f, d) = BE(S, DE_v, t, f, d)$.*

Proof. Defending any given vertex v on T with protection p has the effect of ensuring that at most $1 - p$ burning will reach v; protecting its upstream edge will have the same result. Moreover, as we will not be defending edges upstream to vertices in $Nbd_j(v_f)$ (the vertices distance j from v_f) after time step j and the burning on these vertices will not increase after this time, the burning on each vertex at the end of game with defence strategy DE will be exactly the burning on its unique upstream neighbour, minus the protection on the edge joining them - the same as it would have been with DV.

It remains to show that that DE_v is optimal if and only if DV is. This follows from the discussion of Observation 25. □

To describe the defence strategy, we need to introduce some further notation. We write $v' \triangleleft v$ if v' lies on the unique path between v and v_f. Let $P_{iD}(v) := \sum_{v' \triangleleft v} d_T(v')$, where D is a defence strategy and T is the final time step of the game using this strategy. The defence algorithm in Coupechoux et al. [11] uses the vertex weights from Hartnell and Li [29], where the weight $w(v)$ of a vertex v is equal to the number of its descendants. Finally, [11] allows for the defence budget to vary through the game, so let d_i be the defence budget for time step i. The defence algorithm in [11] **GR**, is then the greedy algorithm which selects an optimal solution of the following linear program, P_i:

$$\begin{aligned} \text{maximize:} \quad & \sum_{v \in Nbd_i(v_f)} x(v)w(v) \\ \text{subject to:} \quad & \sum_{v \in Nbd_i(v_f)} x(v) \leq d_i \\ & x(v) + P_{iD}(v) \leq 1, \quad \forall v \in Nbd_i(v_f) \end{aligned}$$

We can now apply Lemma 45, to the following theorem:

Theorem 46 (Theorem 1 of Coupechoux et al. [11])**.** *The greedy algorithm* **GR** *is $\frac{1}{2}$ competitive for vertex defence fractional firefighter on finite trees.*

This gives us our final result of the section.

Corollary 47. *The algorithm $\mathbf{GR_e}$ given by defending e_v at time t for every vertex v defended by* **GR** *at time t is $\frac{1}{2}$ competitive for edge defence fractional firefighter on finite trees.*

Proof. This follows instantly from Theorem 46 and Lemma 45. □

3.4 P_k-free Graphs

In this section, we focus on P_k-free graphs. As in the rest of the thesis, P_k is the path on k vertices, and we call a graph *P_k-free* if it does not contain P_k as an *induced subgraph*, i.e.:

Definition. For a graph $G = (V, E)$, an *induced subgraph* is the subgraph $G' = (V_s, E_s)$ consisting of some subset $V_s \subseteq V$ of the vertices of G and including all edges $E_s : \{(u, v) \in E : u, v \in V_s\}$ - that is, all the edges that connected V_s in the original graph G.

The proof in Fomin, Heggernes and van Leeuwen [23] that firefighter on P_k-free graphs can be solved in polynomial time transfers instantly to Edge-Defence Firefighter. We shall use the following Lemma from that paper:

Lemma 48 (Lemma 4 of Fomin, Heggernes and van Leeuwen [23])**.** *Let $S = (G, f, d)$ be an instance of firefighter. Let l be the length of the longest induced path in G starting at f. Then no optimal defence strategy can save more than $l - 1$ vertices.*

We use this for the following theorem:

Theorem 49. *EDGE-DEFENCE FIREFIGHTER can be solved on P_k-free graphs with one edge defender in $O(n^k)$ time.*

Proof. The proof relies on the very small number of vertices that can be saved in a P_k-free graph - at most $k-2$ by Lemma 48. The algorithm in Theorem 5 of the same paper then enumerates all subsets of size at most $k-2$ in $O(n^{k-2})$- this will include all possible vertex defence strategies as defending a vertex saves it. As defending an edges always leads to at least as much burning as defending one of the vertices incident to it, the edge defence version will continue for at most $k-2$ time steps. Hence an enumeration of the subsets of edges of size at most $k-2$ will contain the edges to be defended in an optimal defence. Finally, checking if these are valid and how many vertices they will save takes $O(n + m)$-time by Lemma 1 of Fomin, Heggernes and van Leeuwen [23].

The proof of Lemma 1 in Fomin, Heggernes and van Leeuwen [23] is constructive. To see if a proposed defence strategy is valid, they perform a modified breadth first search starting from the fire while playing firefighter, to see if the vertex will or will not already be burning at the time step it is proposed it would be defended. Then they simply count the number of vertices that are saved at the end. This can clearly be transferred to Edge-Defence Firefighter by using the breadth first search to see if the vertices incident to the edge under consideration are already burning. □

3.5 Planar Graphs with Large Girth

In this subsection, we focus on *planar* graphs.

Definition. A *planar* graph is a graph that can be embedded in a plane so that none of its vertices or edges overlap.

Planar graphs are of special relevance to Network Epidemiology because maps of contiguous regions, such as farm boundaries, can always be drawn as planar. This means that a model which represents geographical areas as vertices and has edges between them if those regions are contiguous will be planar. Networks based on transport routes can often be approximated by planar networks - with the non-planarity of the graph coming from places like bridges and underpasses. Both of these applications were raised by Newman in [44].

Planar graphs have been studied extensively in both pure and applied mathematics and some of the most well–known results in graph theory pertain to them. Examples include Kuratowski's Theorem in [35]; the 4- Colour Theorem, proved by Appel and Haken in [1]; and the Planar Graph Separator Theorem of Lipton and Trajan [36]. This means there are many theoretical tools available for the further study of planar graphs.

There has been much work on planar graphs and the Firefighter Problem, specifically focusing on *k-goodness*, which relies on the definition of the *k-survival rate* of a graph, defined above as $\rho_k(G) := \frac{1}{n^2}\sum_{v_f \in V} n - BV((G, \{v_f\}, d), opt, T)$.

Definition. A graph class $\mathcal{G}$ is *k-good* if there exists some constant $c \in \mathbb{R}$ such that $\rho_k(G) \geq c$ for all $G \in \mathcal{G}$.

Planar graphs were proven to be 3-good in Kong, Zhang and Wang [34] using the Planar Graph Separator Theorem of Lipton and Tarjan [36]. Furthermore, Gordinowicz showed that more than $\frac{2}{11}$ of the vertices are expected to survive if there are three defenders in the first time frame and only two in consequent time frames in [27]. Planar graphs have been shown to not be 1-good in Wang et al. [51]. They were conjectured to be 2-good in Esperet et al. [17] – determining whether they are has been identified as one of the big open questions in the field, including by Finbow in [22]. Some results are known about the 2–goodness of particular sub–classes of planar graphs. One of the tools often used to simplify a graph problem is focusing on graphs with low diameter and it has been proven that planar graphs of diameter $2, 3$ or 4 are indeed 2–good [18]. Generalisations of planar graphs have also been studied and it has been shown that 1–planar graphs are 5–good [18]. Many other results rely on the idea of the *girth* of a graph:

Definition. The *girth* of a graph G is the length of its shortest cycle. We denote it by $g(G)$.

It has been shown that graphs of girth 7 are 1–good in Wang, Finbow and Wang [50] and conjectured that graphs of girth 8 may be 1–optimal by Finbow [22] and that planar graphs of girth 5 or 6 may be 1-good in Wang et al. [51].

Much work has been done on finding the survival rates of sparse graphs in the Classic Firefighter game, in particular, Wang, Finbow and Wang [49] found that for planar graphs with girth at least 9, $\rho(G) \geq \frac{2}{35}$ when vertices are defended. In this section of the thesis, we adapt their results to Edge-Defence Firefighter, reaching the same conclusion. Where their results are not obvious, a sketch of the proof will be provided, for more details, see Wang, Finbow and Wang [49].

Let $G = (V, E)$ be a graph with n vertices and m edges. Let $d(v)$ be the degree of a vertex, define a k-vertex to be a vertex of degree k and let V_k denote the set

of k-vertices and $n_k = |V_k|$. For a vertex v, let $n_2(v)$ be the number of 2-vertices which neighbour it. We call a 2-vertex v *good* if $n_2(v) \geq 1$ and *manageable* if $n_2(v) = 0$ and v is adjacent to a 3-vertex x with $n_2(x) \geq 2$. We denote the set of good and manageable 2-vertices by V_2' and V_2'' respectively, with $n_2' = |V_2'|$ and $n_2'' = |V_2''|$.

Good and manageable vertices are so called because if a fire breaks out on them, then there will be limited burning given optimal defence.

Lemma 50. *Let G be a simple, connected graph with $g(G) \geq 9$ and $m \leq (\frac{4}{3} - \epsilon)n$ for some $0 < \epsilon < \frac{5}{24}$. Then for $v_f \in V(G)$:*

$$|BE((G, v_f, d), opt, T)| \geq \begin{cases} n-1 & \text{if } v_f \in V_1, \\ n-2 & \text{if } v_f \in V_2', \\ n-3 & \text{if } v_f \in V_2'', \\ 1 & \text{else.} \end{cases}$$

Proof. Our proof follows the proof of Theorem 3 in Wang, Finbow and Wang [49], with minor adjustments. If the fire starts on a leaf $v_f \in V_1$, then defending the leaf's only edge will save all other $n-1$ vertices. If the fire starts on a good 2-vertex $v_f \in V_2'$, then first defend the edge leading to the neighbour with degree greater that 2 (if such a neighbour exists), then defend the edge connecting the now burning degree 2-vertex neighbour of v_f which does not join it to v_f. If the fire starts on a manageable 2-vertex $v_f \in V_2''$, then there always exists a path $\{v_f, x, u\}$ with $d(v_f) = d(u) = 2$ and $d(x) = 3$ by definition. Defend the edges of v_f, x and u respectively which are not on this path to get the result. Finally, suppose the fire starts at some $v_f \in V \setminus \{V_1, V_2', V_2''\}$. As $m \leq (\frac{4}{3} - \epsilon)n$, there exists at least two vertices u, w with $d(u) < 3$, $d(w) < 3$. Without loss of generality, suppose that u is not on fire at the start of the game. First, defend the edge of u which appears on the shortest path to the burning vertex v_f. As $g(G) \geq 9$, we have at least six more defence moves to save u before the fire can reach it. We defend the remaining edge of u in the next move, giving the result. □

We can also guarantee that if G has certain density constraints, then there will be a certain proportion of these special vertices. As the result is not obvious, a sketch of the proof from Wang, Finbow and Wang [49] is given.

Lemma 51 (Lemma 1 in Wang, Finbow and Wang [49])**.** *Let $0 < \epsilon < \frac{5}{24}$ be a real number. If G is a graph with n vertices and m edges such that $m \leq (\frac{4}{3} - \epsilon)n$, then*

$$n_1 + n_2' + n_2'' \geq \frac{6}{5}\epsilon n.$$

Proof. The following is a sketch of the proof given in Wang, Finbow and Wang [49]. We use a discharge method. By the sparseness conditions, we have:

$$\sum_{v \in V} d(v) = 2m \leq \left(\frac{8}{3} - 2\epsilon\right) n. \tag{3.2}$$

We also have:

$$\sum_{v\in V} d(v) = \sum_{v\in(V_1\cup V_2'\cup V_2'')} d(v) + \sum_{v\in V\setminus(V_1\cup V_2'\cup V_2'')} d(v) = n_1 + 2n_2' + 2n_2'' + \sigma,$$

where $\sigma = \sum_{v\in V\setminus(V_1\cup V_2'\cup V_2'')} d(v)$. We now claim the following:

$$\sigma := \sum_{v\in V\setminus(V_1\cup V_2'\cup V_2'')} d(v) \geq \frac{8}{3}(n - n_1 - n_2' - n_2'').$$

To prove the claim, we use the discharging method. We assign an initial weight $w(v)$ of $d(v)$ to each $v \in V \setminus (V_1 \cup V_2' \cup V_2'')$, and 0 to all other vertices and use the following discharge rule:

(R) Each vertex of degree at least 3 discharges $\frac{1}{3}$ to each 2-vertex in $V \setminus (V_1 \cup V_2' \cup V_2'')$

Let w' be the new weight function after discharge. Following the discharge, we have $w'(v) \geq \frac{8}{3}$ for every $v \in V \setminus (V_1 \cup V_2' \cup V_2'')$. This implies that

$$\sigma \geq \frac{8}{3}(|V| - |V_1| - |V_2'| - |V_2''|) = \frac{8}{3}(n - n_1 - n_2' - n_2''),$$

yielding the claim. By the claim and the inequality 3.2, we have:

$$\left(\frac{8}{3} - 2\epsilon\right) n \geq n_1 + 2n_2' + 2n_2'' + \sigma \geq \frac{8}{3} - \frac{5}{3}(n_1 + n_2' + n_2'').$$

This rearranges to yield the result. □

By combining Lemmas 50 and 51, we can reach conclusions about the survival rate of sparse graphs. The following Theorem was originally stated at Theorem 3 in Wang, Finbow and Wang [49] for Classic Firefighter. Since we are quoting it, we also state it here for Classic Firefighter. As the result requires some work, a sketch of the proof from Wang, Finbow and Wang [49] is given.

Theorem 52 (Theorem 3 in Wang, Finbow and Wang [49])**.** *Let $0 < \epsilon \leq \frac{5}{24}$ be a real number. If G is a graph with n (≥ 2) vertices and m edges such that $m \leq (\frac{4}{3} - \epsilon)n$, then*

$$\rho(G) \geq \frac{6}{5}\epsilon$$

Proof. The following is a sketch of the proof given in Wang, Finbow and Wang

[49]. By Lemmas 50 and 51 and the assumption $0 < \epsilon \leq \frac{5}{24}$, we have:

$$\begin{aligned}
\rho(G) \cdot n^2 &= \sum_{v \in V} |BE((G, v, d), opt, T)| \\
&= \sum_{v \in (V_1 \cup V_2' \cup V_2'')} |BE((G, v, d), opt, T)| \\
&+ \sum_{v \in V \setminus (V_1 \cup V_2' \cup V_2'')} |BE((G, v, d), opt, T)| \\
&\geq (n_1 + n_2' + n_2'')(n + 3) + (n - n_1 - n_2' - n_2'') \\
&\geq \frac{6}{5}\epsilon n(n-4) + n \\
&\geq \frac{6}{5}\epsilon n^2.
\end{aligned}$$

Finally, $\rho(G) \geq \frac{6}{5}\epsilon$. □

Stated for Edge-Defence Firefighter, Theorem 52 simply becomes:

Corollary 53 (Edge-Defence Firefighter Analogue of of Theorem 3 in Wang, Finbow and Wang [49])**.** *Let $0 < \epsilon \leq \frac{5}{24}$ be a real number. If G is a graph with n (≥ 2) vertices and m edges such that $m \leq (\frac{4}{3} - \epsilon)n$, then*

$$\rho_e(G) \geq \frac{6}{5}\epsilon$$

At no point does the proof of Theorem 52 given by Wang, Finbow and Wang in [49] for the Classic Firefighter version of the claim rely on anything that would be different for Edge-Defence Firefighter, other than that it refers to Lemma 50, which we adapt to Edge-Defence Firefighter above, yielding the same result.

Turning our thoughts now to planar graphs of a certain minimal girth, we find that they necessarily have density constraints we can use to make conclusions about their survival rates.

Lemma 54 (Lemma 2 in Wang, Finbow and Wang [49])**.** *If G is a connected planar graph with n vertices and m edges, then*

$$m \leq \frac{g(G)}{g(G) - 2}(n - 2).$$

Substituting in values for $g(G)$, we can easily reach results about the survival rates of planar graphs with a minimum girth size.

Corollary 55 (Corollary 5 in Wang, Finbow and Wang [49])**.** *Let G be a planar graph with at least two vertices.*

- *If $g(G) \geq 9$, then $\rho(G) \geq \frac{2}{35}$.*
- *If $g(G) \geq 11$, then $\rho(G) \geq \frac{2}{15}$.*
- *If $g(G) \geq 13$, then $\rho(G) \geq \frac{2}{11}$.*

In particular, this shows that planar graphs with girth at least 9 are 1-good for Edge-Defence Firefighter.

Corollary 56 (Edge-Defence Firefighter Analogue of Corollary 5 in Wang, Finbow and Wang [49])**.** *Let G be a planar graph with at least two vertices.*

- *If $g(G) \geq 9$, then $\rho_e(G) \geq \frac{2}{35}$.*
- *If $g(G) \geq 11$, then $\rho_e(G) \geq \frac{2}{15}$.*
- *If $g(G) \geq 13$, then $\rho_e(G) \geq \frac{2}{11}$.*

3.6 The Finite Square Grid

In this section, we focus on edge defence strategies on finite square grids. Grids in general have received a lot of attention in the literature for Classic Firefighter. For vertex defence strategies for both finite and infinite square grids, see Gavenčiak, Kratochvíl and Prałat [25].

Single Line Defence

To formally define the square grid, we first consider the *Cartesian graph product*

Definition. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs. The *Cartesian graph product*, denoted $G_1 \square G_2$, is the graph with vertices consisting of the Cartesian product of V_1 and V_2 where two vertices (u_1, v_1) and (u_2, v_2) are adjacent if and only if either $u_1 = u_2$ and $(v_1, v_2) \in E_2$; or if $v_1 = v_2$ and $(u_1, u_2) \in E_1$.

We consider the square grid $P_k \square P_k$ for $k \in \mathbb{N}$, with vertex coordinates as shown in Figure 3.7. As in earlier Chapters, let $S = (G, F, d)$ be an instance of the Edge-Defence Firefighter Game on a graph $G = (V, E)$, where a fire breaks out on the set $F \subset V$ of vertices and d edge defenders can defend per turn. In this section, we shall only consider cases with a single edge defender and a single fire, breaking out on $v_f = (a, b)$. For an instance S, let $BE(S, D, t)$ be the set of burning vertices at time t if a given defence strategy D for one edge defender is used. Let T be the final time step of the game, so that $|BE(S, D, T)|$ is the total amount of burning at the end of the game if edge defence strategy D is used on instance S. Assume (a, b) is in the lower left quadrant of the grid, i.e. $a, b \leq \frac{k-1}{2}$. In this section we shall only consider defence strategies which either only defend all the edges with the same x-coordinates, or all the edges with the same y-coordinates.

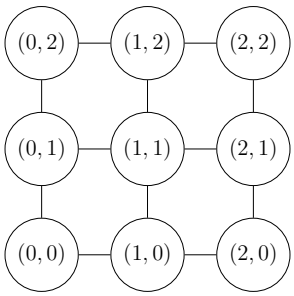


Figure 3.7: $P_3 \square P_3$ with the coordinates of each vertex labelled

Having restricted ourselves to a defence technique as described above, there are three choices to make:

- Whether to defend edges with the same x- or with the same y- coordinates - called defending *horizontally* resp. *vertically.*
- Which fixed x- or y- coordinates to chose.
- Which order the resulting set of edges should be defended in.

We propose that the best defence strategy of this form is given by Algorithm 4, which produces a strategy D_1 with $|BE(S, D_1, T)| = (a+b)k+k$ burning at the end of the game. If there is a greater distance between the starting fire and the right boundary than between the starting fire and the top boundary, it defends vertically; if there is more distance between the fire and the top boundary than between the fire and the right boundary, it defends horizontally. The distance away from the fire (i.e. which shared $x-$ resp. $y-$ coordinates the defended vertices have) is chosen so that the distance between v_f and s, the vertex closest to v_f which is adjacent to a defended edge, is equal to the distance between s and w the closest boundary vertex to s which has an edge defended by D_1, this means that the edges adjacent to the vertices on the path from s to w can be defended before the fire reaches s, and the defence turns can 'keep pace' with the fire, defending an edge just before it would have burned. An example of D_1 being used is shown in Figure 3.8.

After D is obtained from this algorithm, defend the edges it contains in order, one edge per time step. We shall call this defence technique D_1. Note that the number of vertices saved by this defence technique against a fire breaking out at (a, b) depends only of their coordinate sum $\sigma = a + b$. We can therefore write a function for how many vertices would survive using D_1 against a fire breaking out on a vertex with coordinate sum σ:

$$g(k, \sigma) = k^2 - k\sigma - k.$$

Note that for $\sigma = k - 1$ - which is only possible if k is odd - $g(k, \sigma) = 0$ and no vertices can be saved using D_1.

We are interested in how many vertices will be saved by this defence technique if the fire breaks out on one vertex chosen uniformly at random from $P_k \square P_k$.

To get the survival rate, we must find how many vertices in the first quadrant have a given coordinate sum. We define the set A_σ for $P_k \square P_k$ as the vertices

Input: $k, a, b \in \mathbb{N}$, with $a, b \leq \lfloor \frac{k-1}{2} \rfloor$.
Output: An ordered list D_1 of the edges to be defended for a single edge defender against a fire starting on (a, b) in the lower left quadrant of $P_k \square P_k$

initialise an empty list D_1;
initialise i = 0;
initialise j = 1;
if $a \leq b$ **then**
    `we defend vertically`
    while $i < k - b$ **do**
        append $((a + b, b + i), (a + b + 1, b + i))$ to D_1;
        $i+ = 1$;
        if $j \leq b$ **then**
            append $((a + b, b - j), (a + b + 1, b - j))$ to D_1;
            $j+ = 1$;
        end
    end
else
    `we defend horizontally`
    while $i < k - a$ **do**
        append $((a + i, a + b), (a + i, a + b + 1))$ to D_1;
        $i+ = 1$;
        if $j \leq a$ **then**
            append $((a - j, a + b), (a - j, a + b + 1)$ to D_1;
            $j+ = 1$;
        end
    end
end
return D_1

Algorithm 4: Algorithm for *single line* defence on $P_k \square P_k$.

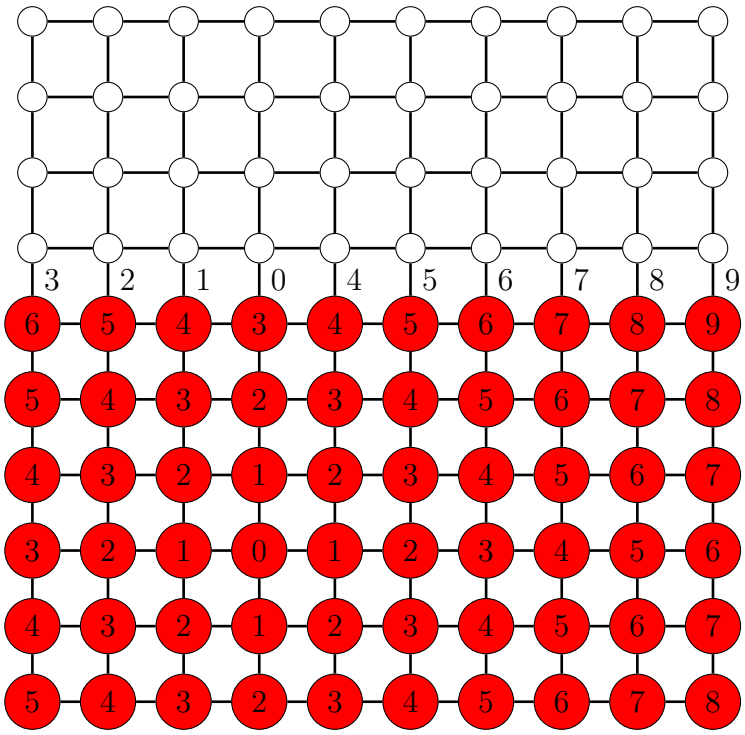


Figure 3.8: D_1 as described in Algorithm 4 on $P_{10}\square P_{10}$ against a fire starting at $v_f = (3, 2)$.

in the first quadrant with coordinate sum σ and their reflections across the lines $x = \frac{k-1}{2}$ and $y = \frac{k-1}{2}$, which will have the same number of saved vertices using a reflection of the given defence technique by symmetry of $P_k\square P_k$.

For even values of k, we have:

$$|A_\sigma| = \begin{cases} 4(\sigma+1) & \text{if } 0 \leq \sigma \leq \dfrac{k-2}{2} \\ 4(k-\sigma-1) & \text{if } \dfrac{k-2}{2} < \sigma \leq k-2 \end{cases} \tag{3.3}$$

For odd values of k, we have:

$$|A_\sigma| = \begin{cases} 4(\sigma+1) & \text{if } 0 \leq \sigma < \dfrac{k-1}{2} \\ 4(k-\sigma-1) & \text{if } \dfrac{k-1}{2} \leq \sigma < k-1 \\ 1 & \text{if } \sigma = k-1 \end{cases} \tag{3.4}$$

We now on the case where k is even. Since $n = k^2$, the edge survival rate as

defined in Equation 3.1 is given by:

$$\begin{aligned}
\rho_e(P_k \square P_k, D_1) &= \frac{1}{n^2}\sum_{\sigma=0}^{k-2}(|A_\sigma| \cdot g(k,\sigma)) \\
&= \frac{1}{k^4}\sum_{\sigma=0}^{\frac{k-2}{2}}((4\sigma+4)\cdot(k^2-k\sigma-k)) \\
&+ \frac{1}{k^4}\sum_{\sigma=\frac{k-2}{2}+1}^{k-2}((4k-4\sigma-4)\cdot(k^2-k\sigma-k)) \\
&= \frac{1}{k^4}\sum_{\sigma=0}^{\frac{k-2}{2}}(4k^2(\sigma+1)-4k(\sigma^2+2\sigma+1)) \\
&+ \frac{1}{k^4}\sum_{\sigma=\frac{k-2}{2}+1}^{k-2}(4k^3(1)-4k^2(2\sigma+2)+4k(\sigma^2+2\sigma+1)) \\
&= \frac{4}{k^2}\sum_{\sigma=0}^{\frac{k-2}{2}}(\sigma+1)-\frac{4}{k^3}\sum_{\sigma=0}^{\frac{k-2}{2}}(\sigma^2+2\sigma+1) \\
&+ \frac{4}{k}\sum_{\sigma=\frac{k-2}{2}+1}^{k-2}(1)-\frac{4}{k^2}\sum_{\sigma=\frac{k-2}{2}+1}^{k-2}(2\sigma+2)+\frac{4}{k^3}\sum_{\sigma=\frac{k-2}{2}+1}^{k-2}(\sigma^2+2\sigma+1) \\
&= \frac{k-2}{2k}+\frac{2}{k}-\frac{k^2+3k+3}{6k^2}-\frac{k-2}{k^2}-\frac{2}{k^2} \\
&+ \frac{2k-4}{k}-\frac{3k^2-10k+8}{k^2}-\frac{4k+8}{k^2}+\frac{7k^3-33k^2+50k-24}{6k^3} \\
&+ \frac{3k^2-10k+8}{k^3}+\frac{2k-4}{k^3} \\
&= -\frac{1}{3k^2}+\frac{1}{2k}+\frac{1}{3}+\frac{1}{3k^2}-\frac{1}{2k}+\frac{1}{6} \\
&= \frac{1}{2}.
\end{aligned}$$

Similarly, for the case where k is odd we get:

$$\begin{aligned}
\rho_e(P_k\Box P_k, D_1) &= \frac{1}{k^4}\sum_{\sigma=0}^{\frac{k-1}{2}-1}(4\sigma+4)\cdot(k^2-k\sigma-k)\\
&+\frac{1}{k^4}\sum_{\sigma=\frac{k-1}{2}}^{k-2}(4k-4\sigma-4)\cdot(k^2-k\sigma-k)\\
&+\frac{1}{k^4}(k^2-k(k-1)-k)\\
&=\frac{1}{k^4}\sum_{\sigma=0}^{\frac{k-1}{2}-1}(4k^2(\sigma+1)-4k(\sigma^2+2\sigma+1))\\
&+\frac{1}{k^4}\sum_{\sigma=\frac{k-1}{2}}^{k-2}(4k^3(1)-4k^2(2\sigma+2)+4k(\sigma^2+2\sigma+1))\\
&+0\\
&=\frac{4}{k^2}\sum_{\sigma=0}^{\frac{k-1}{2}-1}(\sigma+1)-\frac{4}{k^3}\sum_{\sigma=0}^{\frac{k-1}{2}-1}(\sigma^2+2\sigma+1)\\
&+\frac{4}{k}\sum_{\sigma=\frac{k-1}{2}}^{k-2}(1)-\frac{4}{k^2}\sum_{\sigma=\frac{k-1}{2}}^{k-2}(2\sigma+2)+\frac{4}{k^3}\sum_{\sigma=\frac{k-1}{2}}^{k-2}(\sigma^2+2\sigma+1)\\
&=\frac{k^2-1}{2k^2}-\frac{k^2-1}{6k^2}\\
&+\frac{2k-2}{k}-\frac{3k^2-4k+1}{k^2}+\frac{7k^2-12k+5}{6k^2}\\
&=\frac{k^2-1}{3k^2}+\frac{k^2-1}{6k^2}\\
&=\frac{1}{2}-\frac{1}{2k^2}.
\end{aligned}$$

So $\lim_{n\to\infty}\rho_e(P_k\Box P_k, D_1)=\frac{1}{2}$.

Two Line Defence

Again consider the square grid $P_k\Box P_k$ for $k\in\mathbb{N}$, with a fire breaking out at $v_f=(a,b)$ at time 0. Without loss of generality, assume $a\le b$ and $a,b\le\lfloor\frac{k-1}{2}\rfloor$ so the fire starts in the lower central left eighth of the grid, as shown in Figure 3.9. All results then apply to the rest of the grid by symmetry. Let $\sigma=a+b$.

We consider defence techniques that involve choosing two integers c and d; then defending c resp. d edges with the same y- resp. x-coordinates in the upper right corner of the grid. The first case where $k-c\le a$ and $k-d>b$ is shown in Algorithm 5 on page 71, with the remaining two cases - where $k-c>a$ and $k-d\le b$; and where $k-c>a$ and $k-d>b$ - in Appendix B. Note that we do

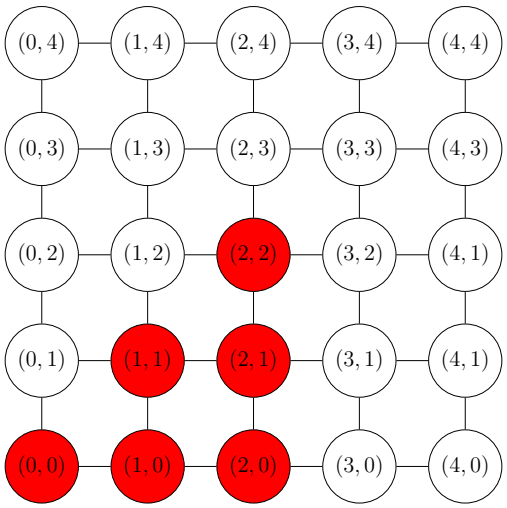


Figure 3.9: $P_5 \square P_5$ with the coordinates of each vertex labelled and vertices we will consider the fire starting from highlighted in red.

not cover the case $k - c \leq a$ and $k - d \leq b$ as it would not save any vertices, but this can be fixed by choosing appropriate values for c and d. A value of c will be called *valid* for a given value of d if at least one of the conditions $k - c \leq a$ and $k - d \leq b$ is met and it is possible to defend a rectangle containing $c \cdot d$ vertices.

Intuitively, we shall be defending in an L-shape in two stages, see Figure 3.10. We call the vertex adjacent to an edge to be defended and closest to the fire s and divide the edges to be defended into those whose vertices form a path between s and the top wall and those whose vertices form a path between s and the right wall. In the first stage of the defence, as the fire approaches the set of edges to be defended, we defend the smaller of these two groups of edges. After the fire has reached the edges to be defended, we defend the larger group, in order of their distance from the fire, thus keeping pace with it until we reach a boundary. Our goal is to find any cases where such a defence technique can save more vertices than the single line defence strategy from the previous section.

There are four options for which parts of the L-shape will be in the smaller group either: the part above s; the part to the right of s; the part that is to the left of and then above s; or the part that is below then right of s. Since we focus on considering initial fires in the octant $\{v_f = (a, b) \in P_k \square P_k : a, b \leq \lfloor \frac{k-1}{2} \rfloor, a \leq b\}$, there will always be more of $P_k \square P_k$ above v_f than to the right of it. We want to maximise the area saved while minimising its distance to the fire, so optimal values of c, d will always have $c \geq d$.

We shall call the vertex where the short part of the L-shape meets a boundary t, as seen in Figure 3.10. Since we defend the short part as the fire travels from f to s, for a value of c to be valid for a value of d we must have $dist(v_f, s) \geq dist(s, t)$. The maximum valid value of c will give $dist(v_f, s) = dist(s, t)$.

We shall derive a formula for the number of vertices saved by fixing d at a given integer; finding the maximum valid value of c; then seeing how the maximum value of c changes if we change d.

We begin by fixing $d = 1$, so the small section is either to the right of s or to the left of and above s.

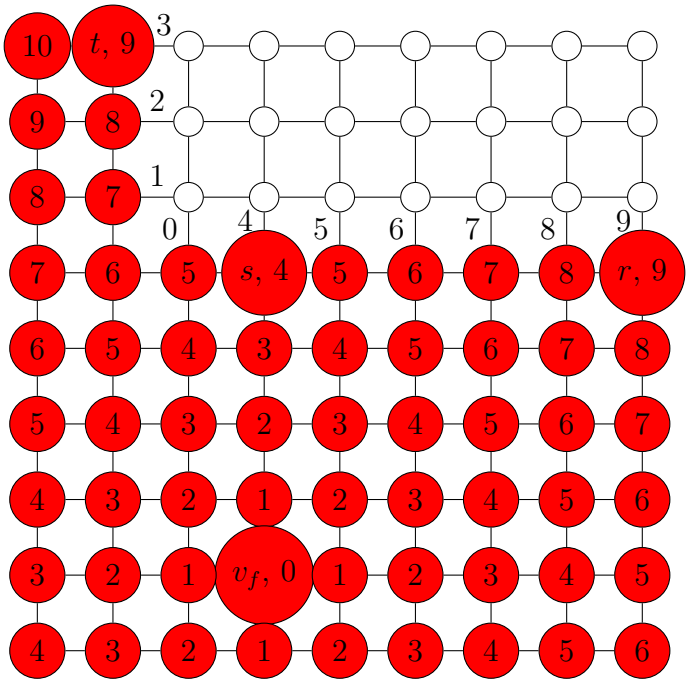


Figure 3.10: D_2 as described in Algorithm 5 (page 72) on $P_9 \square P_9$ against a fire breaking out at $(3, 1)$, with $c = 7$ and $d = 3$ - which is optimal. Red vertices burn and edges are defended during the time step in their label. The vertices v_f, s and t are also labelled. Since the number of edges defended left and above s is greater than the number of defended edges to the right of s, t is where the fire meets the upper boundary wall. We defend along the shorter section of the L-shape, left and above s until the fire reaches s in time step 4. We then defend the edges in right section of the L shape just before they were about to burn, keeping pace with the fire.

For the case where the section to the right of s is smaller and s has the same x-coordinate as v_f, then $dist(v_f, s) = dist(s, t)$ gives us $k-d-b-1 = k-a-1 \Leftrightarrow d = a - b$, with no restrictions on c beyond having to fit the graph, so we get $c = k-2$, the largest valid value. Since a defence technique is only valid if $d \geq 1$, we must also have $a > b$. Note that the resulting vertices saved $(a-b)(k-2)$ is always less than $g(k, \sigma) = k^2 - k\sigma - k$, so will never outperform one line defence. Since that is our ultimate goal, we will therefore not consider this case for the rest of the section.

We now focus on the remaining cases where the smallest part of the edges to be defended are to the left and above s.

Every time we increase the value of d by one, the section to the left and above s that we must defend while the fire travels from v_f to s gets bigger by one and the distance between v_f and s decreases by one. If the section to the left and above s was already as large as possible before increasing d, we must decrease the number of defenders to the left of s (and therefore c) by two.

Continuing this way will eventually give c, d values associated with a valid defence technique. We want to find a maximal (c, d) pair that maximises $c \cdot d$. If we fix $d = 1$, the maximal value of c must satisfy $dist(v_f, s) = dist(s, t)$. We have $dist(v_f, s) = k - b - d - 1 = k - b - 2$ and $dist(s, t) = c - (k - a - 1) - 1$, so $c = 2k - 3 - \sigma$. Let $a_0 = 2k - 3 - \sigma$. The product of the maximal pair after i iterations of increasing d by one decreasing c by two is then given by:

$$f(k, i, \sigma) = a_0 + ia_0 - 2i - 2i^2.$$

To maximise this, we want to find the largest value of i s.t. $f(i) \leq f(i+1)$. We have $f(k, i+1, \sigma) = f(k, i, \sigma) + a_0 - 1(1 + (2i+1))$. So $f(i+1) \geq f(i) \implies a_0 - 2 - 2i - 2 \leq 0 \implies \frac{a_0}{4} - 1 \geq i$, which is not necessarily an integer. The largest integer value of i satisfying this is then given by $i = \lfloor \frac{a_0}{4} \rfloor - 1$ and we want $i + 1 = \lfloor \frac{a_0}{4} \rfloor$. Hence the product will be maximal after $i_{max} = \lfloor \frac{a_0}{4} \rfloor$ iterations. Substituting this in gives:

$$\begin{aligned} f(k, i_{max}, \sigma) &= a_0 + a_0 \left\lfloor \frac{a_0}{4} \right\rfloor - 2 \left\lfloor \frac{a_0}{4} \right\rfloor + 2 \left\lfloor \frac{a_0}{4} \right\rfloor^2 \\ &= \left(2k - \sigma - 3 - 2 \left\lfloor \frac{2k - \sigma - 3}{4} \right\rfloor \right) \cdot \left(1 + \left\lfloor \frac{2k - \sigma - 3}{4} \right\rfloor \right), \end{aligned}$$

the product of the maximal (c, d) pair.

We shall now only consider the i_{max}-th iteration. Consider the case $a_0 = 2k - 3 - \sigma \equiv 0 \mod 4$. Then the number of vertices saved by the associated defence technique if it is valid will be given by:

$$\begin{aligned} f(k, \sigma) &= \left(2k - \sigma - 3 - 2\frac{2k - \sigma - 3}{4} \right) \cdot \left(1 + \frac{2k - \sigma - 3}{4} \right) \\ &= \frac{1}{8} (2k - \sigma - 3)(2k - \sigma + 1). \end{aligned}$$

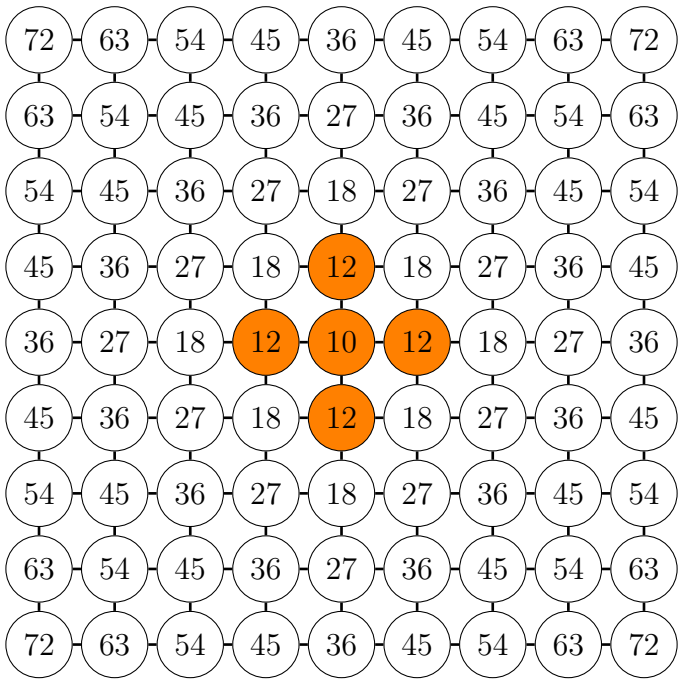


Figure 3.11: $P_9 \square P_9$, each vertex is marked with the amount of burning over the game is the fire starts there against D_3 or its reflection. The vertices are coloured white resp. orange if a D_1 resp. D_2 is better against a fire starting there.

Overall, we have:

$$f(k,\sigma) = \begin{cases} \frac{1}{8}(2k-\sigma-3)(2k-\sigma+1) & \text{if } a_0 = 2k-3-\sigma \equiv 0 \mod 4 \\ \frac{1}{8}(2k-\sigma-2)(2k-\sigma) & \text{if } a_0 \equiv 1 \mod 4 \\ \frac{1}{8}(2k-\sigma-1)^2 & \text{if } a_0 \equiv 2 \mod 4 \\ \frac{1}{8}(2k-\sigma)(2k-\sigma-2) & \text{if } a_0 \equiv 3 \mod 4 \end{cases}$$

Note that the cases $a_0 \equiv 1 \mod 4$ and $a_0 \equiv 3 \mod 4$ are equivalent.

Comparing Single Line and Two Line Defence

Performance for extreme values of σ

In general, D_1 will outperform D_2 for low values of σ and D_2 will outperform D_1 for high values of σ. For an example, see Figure 3.11. The difference between the performance of D_1 and D_2 will be largest at the two extreme values of $\sigma = 0$ and $\sigma = 2\lfloor\frac{k-1}{2}\rfloor$. For a vertex v with coordinate sum zero, we have $a_0 = 2k-\sigma-3 = 2k-3 \equiv 1 \mod 4$ or $a_0 \equiv 3 \mod 4$, so $f(k,0) = \frac{1}{8}2k(2k-2)$ and the difference in the number of vertices saved is given by $g(k,0) - f(k,0) = \frac{k^2}{2} - \frac{k}{2}$, which tends to infinity as k tends to infinity. For $\sigma = 2\lfloor\frac{k-1}{2}\rfloor$, we again get $a_0 = 2k-\sigma-3 \equiv 1 \mod 4$ or $a_0 \equiv 3 \mod 3$. This gives:

$$f\left(k, 2\left\lfloor\frac{k-1}{2}\right\rfloor\right) - g\left(k, 2\left\lfloor\frac{k-1}{2}\right\rfloor\right) = \begin{cases} \frac{1}{8}(k^2-1) & \text{if } k \text{ is odd,} \\ \frac{1}{8}(k^2-6k) & \text{if } k \text{ is even.} \end{cases}$$

both of these also tend to infinity as k tends to infinity. Hence overall, the difference in performance between D_1 and D_2 if the fire starts at a corner or in the middle of $P_k \square P_k$ can be arbitrarily large.

Average Performance

To find when two line defence is strictly better than single line defence, we need to solve $f(k,\sigma) > g(k,\sigma) = k^2 - \sigma k - k$, this gives:

$$\sigma > \begin{cases} 2\sqrt{2k^2+1} - 2k - 1 & \text{if } a_0 = 2k-3-\sigma \equiv 0 \mod 4 \\ \sqrt{8k^2+1} - 2k - 1 & \text{if } a_0 \equiv 1 \mod 4 \\ 2k(\sqrt{2}-1) - 1 & \text{if } a_0 \equiv 2 \mod 4 \\ \sqrt{8k^2+1} - 2k - 1 & \text{if } a_0 \equiv 3 \mod 4. \end{cases}$$

We denote by D_3 the defence strategy for $P_k \square P_k$ that consists of choosing whichever strategy of D_1 or D_2 for that particular fire starting position. We ran code that ran both calculated $g(k,\sigma)$ and $f(k,\sigma)$ - the number of vertices saved using D_1 and D_2 respectively - then used this to calculate the average burning on $P_k \square P_k$ for a fixed value of k, averaged over all possible starting points for the fire using D_1, D_2 or D_3. Averaging this over several values of k gives the results in Table 3.1.

Using D_3 rather than D_1 only lead to an extra 0.24% of all the vertices surviving in computational experiments; while the computational results for D_1 tallied with the lower bound for the edge survival rate calculated in Section 3.6 using D_1:

$$\lim_{n\to\infty} \rho_e(P_k \square P_k) \geq \frac{1}{2}.$$

	% Saved Using Different Defence Strategies			
k	D_1	D_2	D_3	% $v_f \in V$ s.t. $BE(D_1,T) < BE(D_2,T)$
$10-100$	49.98	28.62	50.22	5.89
$1000-1100$	50.00	28.65	50.24	5.89
$10000-10100$	50.00	28.65	50.24	5.89

Table 3.1: Results of computational experiments comparing different defence strategies on $P_n \square P_n$. The middle columns show the percentage of vertices saved using different defence strategies. The rightmost column shows the percentage of vertices v_f for which D_2 will save strictly more vertices that D_1 if the fire starts at v_f.

3.7 The Infinite Square Grid

We now consider the infinite square grid. Previous consideration of Firefighter on infinite graphs has focused on either containing the fire (e.g. Dean et al. [14]) or on a constant fraction of vertices distance t away from the fire that can be prevented from burning during time t. It does not seem possible to contain the fire, or to save a constant positive fraction of the grid using a single edge defender over infinite time steps. However, if a 'snapshot' of the game is taken at any given time point, it is possible to prepare to defend a constant positive fraction of the vertices under consideration. This will be the focus of this section.

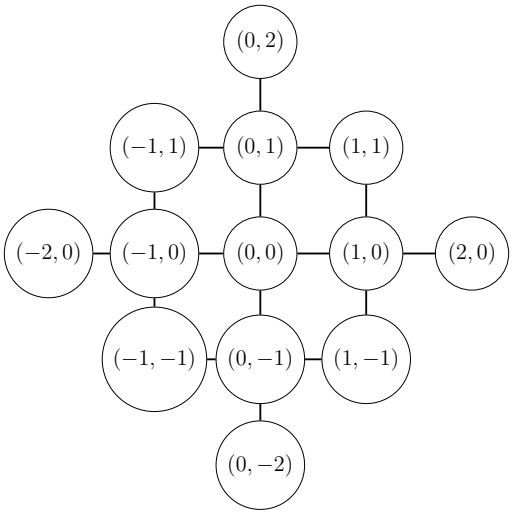


Figure 3.12: $P_{2,(0,0)} \square P_{2,(0,0)}$ with vertex coordinates labelled. In this section, the fire always breaks out at $v_f = (0, 0)$.

We define $P_{t,v_f} \square P_{t,v_f}$ as the section of the infinite square grid which is within distance t of the initially burning vertex v_f. Note that this is not equivalent to $P_k \square P_k$ for some integer k, but a diamond-shaped area, as shown in Figure 3.12, with the fire breaking out on vertex $v_f = (0, 0)$. We shall consider defence techniques on $P_{t,v_f} \square P_{t,v_f}$ which meet the following criteria:

- An edge can only be defended if both endpoints are within distance t of v_f.
- One edge may be defended per time step for any finite number T of time steps - possibly exceeding t.
- After T time steps, where T is finite, the defenders must have 'sealed off' any protected areas, so that even without any further defence, they would not burn if the game continued indefinitely.

An alternative way of viewing this is that we aim to find which fraction of the vertices within distance t of v_f on a square grid can be saved if the fire can spread without boundaries. We shall focus on defence techniques that involve defending a rectangle with width c and height d as seen in Figure 3.13. That is, defending $2c$ resp. $2d$ edges divided into two equal groups that share the same

y- resp. x-coordinates. Defending more than one rectangle would lead to the same number of defenders protecting fewer vertices, as each corner of a rectangle must have two protected edges. Note that it may take more that t time steps to complete the defence of the rectangle, as can be seen in Figure 3.13

As in Section 3.6, we shall start by finding the largest value of c for a fixed value of d that will yield a defence strategy satisfying our requirements.

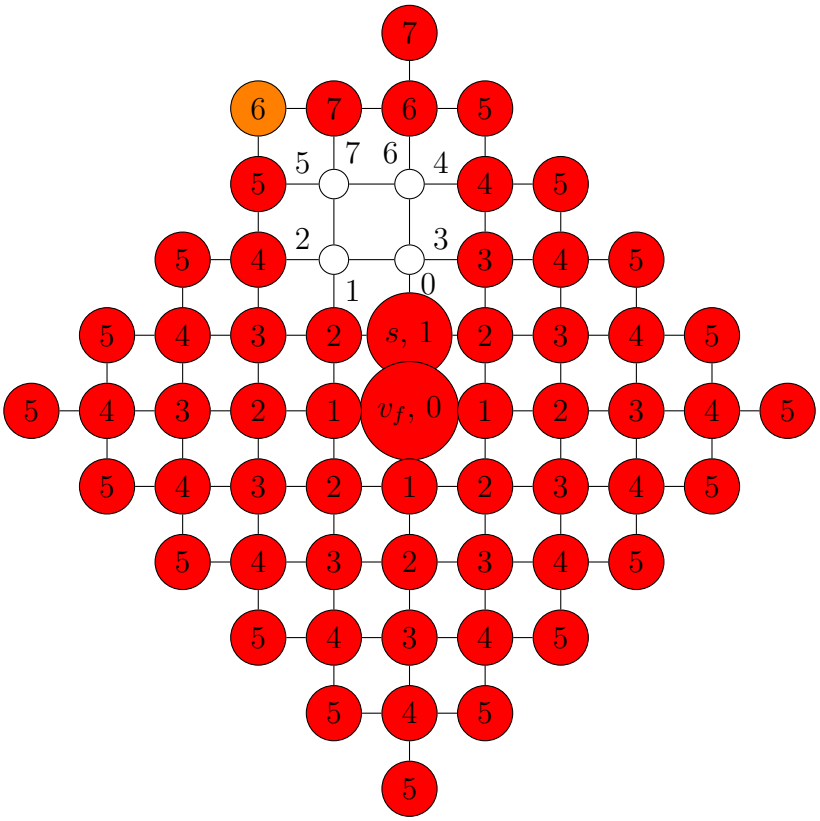


Figure 3.13: A game on $P_{5,(0,0)}\square P_{5,(0,0)}$ using defence technique D from Algorithm 6 with $c = 2 = d$, which is optimal. The vertices v_f and s are labelled. Red vertices burn and edges are defended during the labelled time steps. Note that for the fire to spread, it has to travel through $(-2, 4)$, which is outside $P_{5,(0,0)}\square P_{5,(0,0)}$ and indicated in orange. Also note that the rectangle is only sealed off after $T = 7$ time steps.

Consider a feasible defence technique, i.e. one that completely blocks off a rectangle before the fire can reach any of its interior vertices. As in subsection 3.6, we call the vertex with a defended edge which is closest to the fire s. An important consideration is the distance between v_f and s. We want to buy time to start our defence before the fire reaches the defended rectangle, so we place it as far away from $v_f = (0, 0)$ as possible, in a corner of $P_{t,v_f}\square P_{t,v_f}$. Without loss of generality, we put it in the top corner. This can be seen in Figure 3.13. The y-coordinate of the topmost vertex is $t - 1$. If $c = 1$, then the y-coordinate of s will be $t - d - 1$. If we fix d and increase c then the maximum possible y-coordinate of s will decrease by one as c increases by one from an odd number to an even number as the top part of the rectangle will no longer fit in the same

row and the rectangle has to start one row lower. It stays the same if c increases by one from an even number to an odd number as the top part of the rectangle can still fit in the same row. Continuing this way leads to the following expression for the distance between v_f and s:

$$Man(v_f, s) = \begin{cases} t - d - \left\lceil \frac{c}{2} \right\rceil & \text{if } c \text{ is odd} \\ t - d - \left\lceil \frac{c+1}{2} \right\rceil & \text{if } c \text{ is even,} \end{cases} \tag{3.5}$$

The game can be broken down into two stages. In Stage 1 of the game, while the fire approaches the area we want to defend, we can defend one edge per time step for $Man(v_f, s) + 1$ time steps. Say we defend the edges starting from the bottom of the rectangle, and then going left and then up. Once the fire reaches the defended rectangle in Stage 2, it then starts spreading left and right along its bottom, but the left direction has already been defended. We can then keep pace with the fire as it spreads to the right, defending one edge per time step. This would stop working after an additional $Man(v_f, s) + 1$ time steps as the fire on the left side would pass the edges defended in Stage 1 and start spreading along undefended edges. While the fire is spreading around the rectangle we intend to defend, at each corner the fire has to travel further around the boundary of the rectangle than the defenders do, giving us an extra four time steps to defend in. Hence for such a defence to be feasible, we must have:

$$2d + 2c \leq 2 \cdot (Man(v_f, s) + 1) + 4, \tag{3.6}$$

using the expression for $Man(v_f, s)$ from 3.5, this gives:

$$c \leq \begin{cases} \frac{2}{3}(t - 2d) + \frac{5}{3} & \text{if } c \text{ is odd} \\ \frac{2}{3}(t - 2d) + \frac{4}{3} & \text{if } c \text{ is even.} \end{cases} \tag{3.7}$$

Note that we can only defend in the four time steps we 'gained' on the fire as it went around a corner if they are used after the fire has travelled around all corners/if we have already defended three walls of the rectangle with total length $c + 2d$, and 'gained' two time steps on the fire as it travelled around the bottom two corners of the rectangle i.e. if:

$$2c + 2d - 2Man(v_f, s) - 2 > 3, \tag{3.8}$$

we need the extra condition:

$$c + 2d - 2Man(v_f, s) - 2 \leq 2. \tag{3.9}$$

We now consider the three cases for $t - 2d \mod 3 \in \{0, 1, 2\}$. For $t - 2d \equiv 0 \mod 3$, $\frac{2}{3}(t - 2d) + 1$ will be an odd integer, and in particular, the largest integer satisfying 3.7, so will be the optimal value for c. For $t - 2d \equiv 1 \mod 3$, we have $\frac{2}{3}(t-2d)+\frac{4}{3}$ is an even integer, so this satisfies inequality (3.7) and gives the value of c. Finally, for $t - 2d \equiv 2 \mod 3$, $\frac{2}{3}(t - 2d) + \frac{5}{3}$ is an odd integer, satisfying 3.7 and therefore the optimal value for c. Note that as t increases for a fixed value

of d then c will have the same odd value for consecutive values of t satisfying $t - 2d \mod 3 \in \{0, 2\}$; will then increase by one to an even value for $t - 2d \equiv 1 \mod 3$; and will then increase again by one for the next two consecutive values of t, continuing the cycle.

Multiplying this by d gives the following expression for the number of vertices saved for $t \in \mathbb{N}$:

$$h(t,d) = \begin{cases} \left(\frac{2}{3}(t-2d)+1\right)\cdot d & \text{if } t-2d \equiv 0 \mod 3 \\ \left(\frac{2}{3}(t-2d)+\frac{4}{3}\right)\cdot d & \text{if } t-2d \equiv 1 \mod 3 \\ \left(\frac{2}{3}(t-2d)+\frac{5}{3}\right)\cdot d & \text{if } t-2d \equiv 2 \mod 3. \end{cases} \tag{3.10}$$

For $t - 2d \equiv 0 \mod 3$, we have $h(t,d) = (\frac{2}{3}(t-2d)+1)\cdot d = \frac{3}{16}(\frac{2t}{3}+1)^2 - \frac{4}{3}(d-(\frac{t}{4}+\frac{3}{8}))^2$, which is maximised when $d = \frac{t}{4}+\frac{3}{8}$. Similarly, for $t-2d \equiv 1$ resp. $2 \mod 3$, $h(t,d)$ is maximised when d is equal to $\frac{t}{4}+\frac{1}{2}$ resp. $\frac{t}{4}+\frac{15}{24}$. The optimal valid integer value of d will not necessarily be either the floor or the ceiling of these expressions, as it still needs to fulfill the constraints on $t-2d \mod 3$. Since the constraints on d depend on the value of $t \mod 3$; and the closest integer to the respective turning points ($\frac{t}{4}+\frac{3}{8}$, $\frac{t}{4}+\frac{1}{2}$ and $\frac{t}{4}+\frac{15}{24}$ for $t-2d \equiv 0,1,2 \mod 3$ respectively) depends on the value of $t \mod 4$, we shall consider each value of $t \mod 12$ separately, thus covering all cases.

Let us consider the case $t \equiv 5 \mod 12$, or $t = 12l+5$ for some $l \in \mathbb{Z}$. Then $t \equiv 2 \mod 3$ and $t \equiv 1 \mod 4$. The first case of Equation (3.10) where $t-2d \equiv 0 \mod 3$ is maximised at the valid integer closest to $\frac{t}{4}+\frac{3}{8} = 3l+1+\frac{5}{8}$. Since $t \equiv 2 \mod 3$, to get $t-2d \equiv 0 \mod 3$, we must have $d \equiv 1 \mod 3$ so we want the closest integer to $3l+1+\frac{5}{8}$ which satisfies this. That gives $d = 3l+1 = \frac{t}{4}-\frac{1}{4}$. Similarly for the case $t-2d \equiv 1 \mod 3$, we want the integer closest to $\frac{t}{4}+\frac{1}{2} = 3l+1+\frac{1}{2}$ which satisfies $d \equiv 2 \mod 3$, giving $d = 3l+2 = \frac{t}{4}+\frac{3}{4}$. Finally, for $t-2d \equiv 2 \mod 3$ we want the integer closest to $\frac{t}{4}+\frac{15}{24} = 3l+1+\frac{21}{24}$ and congruent to $0 \mod 3$, which gives $d = 3l+3 = \frac{t}{4}+\frac{7}{4}$. Now that we have the optimal values of d for each value of $t-2d \mod 3$, we can now put them into the appropriate part of 3.10 to get an expression for the number of vertices saved just in terms of t. This gives $\frac{t^2}{12}+\frac{t}{4}-\frac{1}{3}$, $\frac{t^2}{12}+\frac{t}{3}+\frac{1}{4}$ and $\frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6}$ for $t-2d \equiv 0,1,2 \mod 3$ respectively. Finally, we can compare these expressions to find when $t-2d \equiv 2 \mod 3$ is the best option, ie when $\frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6} \geq \frac{t^2}{12}+\frac{t}{4}-\frac{1}{3}$ and when $\frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6} \geq \frac{t^2}{12}+\frac{t}{3}+\frac{1}{4}$. This gives us the conditions $t \geq 5$ and $t \geq 17$. So the optimal value for $h(t)$ the amount of vertices saved will be $\frac{t^2}{12}+\frac{t}{3}+\frac{1}{4}$ for $t = 5$ and $\frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6}$ for all other t s.t. $t \equiv 5 \mod 12$.

We can continue in a similar manner for all values of $t \mod 12$. The optimal valid integer values of d are summarised in Table 3.2.

Table 3.2: The valid integer values for d which lead to the greatest value of $h(t, d)$ as given in Equation (3.10).

	$t-2d \equiv 0 \mod 3$	$t-2d \equiv 1 \mod 3$	$t-2d \equiv 2 \mod 3$
$t \equiv 0 \mod 12$	$\frac{t}{4}$	$\frac{t}{4}+1$	$\frac{t}{4}+2$
$t \equiv 1 \mod 12$	$\frac{t}{4}+\frac{7}{4}$	$\frac{t}{4}-\frac{1}{4}$	$\frac{t}{4}+\frac{3}{4}$
$t \equiv 2 \mod 12$	$\frac{t}{4}+\frac{1}{2}$	$\frac{t}{4}+\frac{3}{2}$	$\frac{t}{4}-\frac{1}{2}$
$t \equiv 3 \mod 12$	$\frac{t}{4}-\frac{3}{4}$	$\frac{t}{4}+\frac{1}{4}$	$\frac{t}{4}+\frac{5}{4}$
$t \equiv 4 \mod 12$	$\frac{t}{4}+1$	$\frac{t}{4}-1$ or $\frac{t}{4}+2$	$\frac{t}{4}$
$t \equiv 5 \mod 12$	$\frac{t}{4}-\frac{1}{4}$	$\frac{t}{4}+\frac{3}{4}$	$\frac{t}{4}+\frac{7}{4}$
$t \equiv 6 \mod 12$	$\frac{t}{4}+\frac{3}{2}$	$\frac{t}{4}-\frac{1}{2}$	$\frac{t}{4}+\frac{1}{2}$
$t \equiv 7 \mod 12$	$\frac{t}{4}+\frac{1}{4}$	$\frac{t}{4}+\frac{5}{4}$	$\frac{t}{4}-\frac{3}{4}$
$t \equiv 8 \mod 12$	$\frac{t}{4}-1$	$\frac{t}{4}$	$\frac{t}{4}+1$
$t \equiv 9 \mod 12$	$\frac{t}{4}+\frac{3}{4}$	$\frac{t}{4}+\frac{7}{4}$	$\frac{t}{4}-\frac{1}{4}$
$t \equiv 10 \mod 12$	$\frac{t}{4}-\frac{1}{2}$	$\frac{t}{4}+\frac{1}{2}$	$\frac{t}{4}+\frac{3}{2}$
$t \equiv 11 \mod 12$	$\frac{t}{4}+\frac{5}{4}$	$\frac{t}{4}-\frac{3}{4}$	$\frac{t}{4}+\frac{1}{4}$

The values of $h(t)$ can then be found by substituting the values for d given in Table 3.2 into Equation (3.10), as shown in table 3.3.

Table 3.3: The value of $h(t)$, found by substituting the values in Table 3.2 into 3.10.

	$t-2d \equiv 0 \mod 3$	$t-2d \equiv 1 \mod 3$	$t-2d \equiv 2 \mod 3$
$t \equiv 0 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}$	$\frac{t^2}{12}+\frac{t}{3}$	$\frac{t^2}{12}+\frac{5t}{12}-2$
$t \equiv 1 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}-\frac{7}{3}$	$\frac{t^2}{12}+\frac{t}{3}-\frac{5}{12}$	$\frac{t^2}{12}+\frac{5t}{12}+\frac{1}{2}$
$t \equiv 2 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}+\frac{1}{6}$	$\frac{t^2}{12}+\frac{t}{3}-1$	$\frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6}$
$t \equiv 3 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}-\frac{3}{2}$	$\frac{t^2}{12}+\frac{t}{3}+\frac{1}{4}$	$\frac{t^2}{12}+\frac{5t}{12}$
$t \equiv 4 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}-\frac{1}{3}$	$\frac{t^2}{12}+\frac{t}{3}-\frac{8}{3}$	$\frac{t^2}{12}+\frac{5t}{12}$
$t \equiv 5 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}-\frac{1}{3}$	$\frac{t^2}{12}+\frac{t}{3}+\frac{1}{4}$	$\frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6}$
$t \equiv 6 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}-\frac{3}{2}$	$\frac{t^2}{12}+\frac{t}{3}-1$	$\frac{t^2}{12}+\frac{5t}{12}+\frac{1}{2}$
$t \equiv 7 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}+\frac{1}{6}$	$\frac{t^2}{12}+\frac{t}{3}-\frac{5}{12}$	$\frac{t^2}{12}+\frac{5t}{12}-2$
$t \equiv 8 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}-\frac{7}{3}$	$\frac{t^2}{12}+\frac{t}{3}$	$\frac{t^2}{12}+\frac{5t}{12}+\frac{1}{3}$
$t \equiv 9 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}$	$\frac{t^2}{12}+\frac{t}{3}-\frac{7}{4}$	$\frac{t^2}{12}+\frac{5t}{12}-\frac{1}{2}$
$t \equiv 10 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}-\frac{5}{6}$	$\frac{t^2}{12}+\frac{t}{3}+\frac{1}{3}$	$\frac{t^2}{12}+\frac{5t}{12}-\frac{1}{2}$
$t \equiv 11 \mod 12$	$\frac{t^2}{12}+\frac{t}{4}+\frac{5}{6}$	$\frac{t^2}{12}+\frac{t}{3}-\frac{7}{4}$	$\frac{t^2}{12}+\frac{5t}{12}+\frac{1}{3}$

The constraints this puts on the values of t s.t. $t-2d \equiv 2 \mod 3$ will lead to the greatest values of $h(t)$ are then shown in Table 3.4. For $t \in \{2, 5, 7, 12\}$, t is not large enough for the case where $t-2d \equiv 2 \mod 3$ to be better than at least one of the other two cases. For example, for $t \equiv 2 \mod 12$, the value of $h(t)$ if $t-2d \equiv 0 \mod 3$ is $\frac{t^2}{12}+\frac{t}{4}+\frac{1}{6}$; and if $t-2d \equiv 2 \mod 3$, then it is $\frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6}$ and we have $\frac{t^2}{12}+\frac{t}{4}+\frac{1}{6} \leq \frac{t^2}{12}+\frac{5t}{12}-\frac{7}{6} \implies t \geq 8$. Similarly, the case where $t-2d \equiv 2 \mod 3$ is only at least as good as the case $t-2d \equiv 1 \mod 3$ if $t \geq 2$. This means that for $t = 2$ the case $t-2d \equiv 2 \mod 3$ will be at least as good as the case $t-2d \equiv 1 \mod 3$, but not as good as the case $t-2d \equiv 0 \mod 3$. Therefore to

get the value of $h(2)$, we should use the formula for $t - 2d \equiv 0 \mod 3$ for $t \equiv 2 \mod 12$ from Table 3.3 which gives $h(2) = \frac{2^2}{12} + \frac{2}{4} + \frac{1}{6} = 1$.

Table 3.4: Conditions on $t \in \mathbb{N}$, so that the case $t - 2d \equiv 2 \mod 3$ leads to at least as many vertices saved as the cases $t \equiv 0 \mod 3$ or $t \equiv 1 \mod 3$. It is derived from the equations in Table 3.3.

	At least as good as $t - 2d \equiv 0 \mod 3$	At least as good as $t - 2d \equiv 1 \mod 3$
$t \equiv 0 \mod 12$	$t \geq 12$	$t \geq 24$
$t \equiv 1 \mod 12$	$t \geq -17$	$t \geq -11$
$t \equiv 2 \mod 12$	$t \geq 8$	$t \geq 2$
$t \equiv 3 \mod 12$	$t \geq -9$	$t \geq 3$
$t \equiv 4 \mod 12$	$t \geq -2$	$t \geq -32$
$t \equiv 5 \mod 12$	$t \geq 5$	$t \geq 17$
$t \equiv 6 \mod 12$	$t \geq -12$	$t \geq -18$
$t \equiv 7 \mod 12$	$t \geq 13$	$t \geq 19$
$t \equiv 8 \mod 12$	$t \geq -16$	$t \geq -4$
$t \equiv 9 \mod 12$	$t \geq 3$	$t \geq -15$
$t \equiv 10 \mod 12$	$t \geq -2$	$t \geq 10$
$t \equiv 11 \mod 12$	$t \geq -7$	$t \geq -25$

We can now write down $h(t)$ for $t \in \mathbb{N}$. Equation (3.11) shows this in the form $h(t) = c \cdot d$. The d values are from Table 3.2; the c values are from substituting the d values into the relevant part of Equation (3.10); and the conditions are from Table 3.4.

$$h(t) = \begin{cases} 1 \cdot 1 & \text{if } t = 2 \\ 2 \cdot 2 & \text{if } t = 5 \\ 3 \cdot 2 & \text{if } t = 7 \\ 4 \cdot 4 & \text{if } t = 12 \\ \left(\frac{t}{3} - 1\right) \cdot \left(\frac{t}{4} + 2\right) & \text{if } t \equiv 0 \mod 12 \text{ and } t \geq 24 \\ \left(\frac{t}{3} + \frac{2}{3}\right) \cdot \left(\frac{t}{4} + \frac{3}{4}\right) & \text{if } t \equiv 1 \mod 12 \\ \left(\frac{t}{3} + \frac{7}{3}\right) \cdot \left(\frac{t}{4} - \frac{1}{2}\right) & \text{if } t \equiv 2 \mod 12 \text{ and } t \geq 8 \\ \frac{t}{3} \cdot \left(\frac{t}{4} + \frac{5}{4}\right) & \text{if } t \equiv 3 \mod 12 \\ \left(\frac{t}{3} + \frac{5}{3}\right) \cdot \frac{t}{4} & \text{if } t \equiv 4 \mod 12 \\ \left(\frac{t}{3} - \frac{2}{3}\right) \cdot \left(\frac{t}{4} + \frac{7}{4}\right) & \text{if } t \equiv 5 \mod 12 \text{ and } t \geq 17 \\ \left(\frac{t}{3} + 1\right) \cdot \left(\frac{t}{4} + \frac{1}{2}\right) & \text{if } t \equiv 6 \mod 12 \\ \left(\frac{t}{3} + \frac{8}{3}\right) \cdot \left(\frac{t}{4} - \frac{3}{4}\right) & \text{if } t \equiv 7 \mod 12 \text{ and } t \geq 19 \\ \left(\frac{t}{3} + \frac{1}{3}\right) \cdot \left(\frac{t}{4} + 1\right) & \text{if } t \equiv 8 \mod 12 \\ \left(\frac{t}{3} + 2\right) \cdot \left(\frac{t}{4} - \frac{1}{4}\right) & \text{if } t \equiv 9 \mod 12 \\ \left(\frac{t}{3} - \frac{1}{3}\right) \cdot \left(\frac{t}{4} + \frac{3}{2}\right) & \text{if } t \equiv 10 \mod 12 \\ \left(\frac{t}{3} + \frac{4}{3}\right) \cdot \left(\frac{t}{4} + \frac{1}{4}\right) & \text{if } t \equiv 11 \mod 12 \end{cases} \tag{3.11}$$

Note that these equations were derived to satisfy Condition (3.7); so to prove that these values of c and d are valid, it suffices to show that Condition (3.9) holds.

Consider the case $t \equiv 0 \mod 12$ and $t \geq 24$. Since $t \equiv 0 \mod 12$, $\frac{t}{3}$ is even, so $c = \frac{t}{3} - 1$ is odd. Hence $Man(v_f, s) = t - d - \left\lceil \frac{c}{2} \right\rceil = t - \frac{t}{4} - 2 - \frac{t}{6} = \frac{7}{12}t - 2$ from Equation (3.5). So the LHS is $c + 2d - 2Man(v_f, s) - 2 = \frac{t}{3} - 1 + \frac{t}{2} + 4 - \frac{7}{6}t - 4 - 2 = -\frac{t}{3} - 3 \leq 2$, hence Condition (3.9) holds and the values of c and d are valid. All other cases follow similarly.

The graph $P_{t,v_f} \square P_{t,v_f}$ has $n = 2t^2 + 2t + 1$ vertices. Note that for $t \in \{2, 5, 7, 12\}$, the fraction of vertices of $P_{t,v_f} \square P_{t,v_f}$ saved is always at least $\frac{1}{24}$. Furthermore, for all other cases of $h(t)$ in Equation (3.11) their limit as $t \to \infty$ is dominated by the $\frac{t^2}{12}$ term.

Hence overall, a limit for lower bound for the fraction of $P_{t,v_f} \square P_{t,v_f}$ that is saved by this strategy is given by:

$$\lim_{t \to \infty} \frac{h(t)}{2t^2 + 2t + 1} = \lim_{t \to \infty} \frac{\frac{t^2}{12}}{2t^2 + 2t + 1} = \lim_{t \to \infty} \frac{t^2}{24t^2 + 24t + 12} = \frac{1}{24}.$$

Input: $k, a, b, c, d \in \mathbb{N}$, with $a, b \leq \lfloor \frac{k-1}{2} \rfloor$, $a \leq b$, $k - c \leq a$ and $k - d > b$
Output: An ordered list D_2 of the edges to be defended for a single edge defender against a fire starting on $(a, b) \in P_k \square P_k$

$r = (k-1, k-d-1)$;
$s = (a, k-d-1)$;
split edge $= [((a, k-d-1), (a, k-d))]$;
$right =$Man(r, s);
$left = c - right - 1$;
L shape $= d + left$;
initialise an empty list *right defence*;
initialise an empty list *L defence*;
initialise *right counter* $= 0$;
while *right counter* $< right$ **do**
  append
  $((a + 1 +$ *right counter*$, k - d - 1), (a + 1 +$ *right counter*$, k - d))$ to *right defence* ;
  right counter $+= 1$;
end
initialise *left counter* $= 0$;
while *left counter* $< left$ **do**
  append $((a - 1 -$ *left counter*$, k - d - 1), (a - 1 -$ *left counter*$, k - d))$ to *L defence* ;
  left counter $+= 1$;
end
initialise *d counter* $= 0$;
while *d counter* $< d$ **do**
  append $((k - c - 1, k - d +$ *d counter*$), (k - c, k - d +$ *d counter*$))$ to *L defence* ;
  d counter $+= 1$;
end
if *right* $<$ *L shape* **then**
  $D_2 =$ *right defence* $+$ *split edge* $+$ *L defence*;
else
  $D_2 =$ *L defence* $+$ *split edge* $+$ *right defence*;
end
return D_2

Algorithm 5: Pseudocode for calculating D_2 under certain conditions. For other cases see Appendix B.

Input: $t, c, d \in \mathbb{N}$, with $t \geq 3$ and $c, d \in \mathbb{Z}$, calculated elsewhere.
Output: An ordered list D of the edges to be defended for a single edge defender against a fire starting on $f \in P_{t,v_f} \square P_{t,v_f}$.

initialise an empty list D;
initialise *defender counter, c one counter, c two counter, d counter* $= 0$;
if c is odd **then**
  $height = t - d - \lceil \frac{c}{2} \rceil$; $right = \lfloor \frac{c}{2} \rfloor$; $left = \lfloor \frac{c}{2} \rfloor$;
else
  $height = t - d - \lceil \frac{c+1}{2} \rceil$; $right = \frac{c}{2}$; $left = \frac{c}{2} - 1$;
end
while *c one counter* $< c$ **do**
  if *defender counter* is even **then**
    append $((-\lceil \frac{c \text{ one counter}}{2} \rceil, height), (-\lceil \frac{c \text{ one counter}}{2} \rceil, height + 1))$ to D ;
    c one counter, defender counter $+= 1$;
  else
    append $((\lceil \frac{c \text{ one counter}}{2} \rceil, height), (\lceil \frac{c \text{ one counter}}{2} \rceil, height + 1))$ to D ;
    c one counter, defender counter $+= 1$;
  end
end
while *d counter* $< 2d$ **do**
  if *defender counter* is even **then**
    append $((-left, height+1+\lfloor \frac{d \text{ counter}}{2} \rfloor), (-left-1, height+1+\lfloor \frac{d \text{ counter}}{2} \rfloor))$ to D ;
    d counter, defender counter $+= 1$;
  else
    append to D $((right, height+1+\lfloor \frac{d \text{ counter}}{2} \rfloor), (right+1, height+1+\lfloor \frac{d \text{ counter}}{2} \rfloor))$;
    d counter, defender counter $+= 1$;
  end
end
while *c two counter* $< c$ **do**
  if *defender counter* is even **then**
    append to D $((-left + \lfloor \frac{c \text{ two counter}}{2} \rfloor, height + d), (-left + \lfloor \frac{c \text{ two counter}}{2} \rfloor, height + d + 1))$;
    c two counter, defender counter $+= 1$;
  else
    append $((right - \lfloor \frac{c \text{ two counter}}{2} \rfloor, height + d), (right - \lfloor \frac{c \text{ two counter}}{2} \rfloor, height + d + 1))$ to D ;
    c two counter, defender counter $+= 1$;
  end
end

Algorithm 6: Pseudocode for calculating D for $P_{t,v_f} \square P_{t,v_f}$.

Chapter 4

Edge-Defence Computational Experiments

In this chapter, we focus on various heuristics for Edge-Defence Firefighter, adapted from the optimal vertex defence strategy for fire path equivalent games given in Theorem 16 of Section 2.6. To have optimal solutions with which to compare the heuristic solutions, we start by developing an integer program for Edge-Defence Firefighter and using various valid inequalities to improve its running time.

Our main motivation for investigating different variations of the Firefighter problem is to better model real world constraints. Optimising the run time for an integer program to find exact solutions and developing heuristic algorithms for instances that would take too long to solve exactly are important steps towards creating tools that could be used for policy decisions in real world epidemics.

An integer program for Classic Firefighter was first given in Develin and Hartke [15]. Subsequent papers have worked on modifying and improving the running times for this program, notably Finbow and MacGillivray [21], which modified it; García-Martínez et al. [24], which noted the importance of an upper limit for the number of time steps a game takes to complete for the running time and used an iterative approach to finding it; and Ramos et al. [47], which found improved upper bounds for this and added some valid inequalities, which we shall build upon.

An early result by Hartnell and Li [29] showed that for Classic Firefighter on trees with a single initially burning vertex and a single defender, defending the threshold vertex with the greatest number of descendants at each defender turn saves at least half the number of vertices saved by the optimal solution. There are further approximation algorithms for Classic Firefighter on trees, including in Cai, Verbin and Yang [6]; and Iwaikawa, Kamiyama and Matsui [32]. Since any vertex defence on a tree has an equivalent edge defence strategy which leads to the same amount of burning by Observation 25 above, these results instantly carry over to Edge-Defence Firefighter.

García-Martínez et al. [24] summarised various heuristics for Classic Firefighter on random graphs, based on vertex degree and the number of descendants of a vertex.

There has been considerable work on *metaheuristics* for Classic Firefighter. A

metaheuristic is a procedure which produces and applies heuristics in an attempt to find a near optimal solution. Blum et al. [3] used a metaheuristic purely based on ant colony optimisation, and another which started with ant colony optimisation, then used solution polishing. Hu, Windbichler and Raidl [31] used a variable neighbourhood search.

The paper presenting heuristic algorithms for a problem which arguably comes closest to Edge-Defence Firefighter is Michalak [42]. This presented an evolutionary algorithm for a problem based on Classic Firefighter, but with a few key differences. Firstly, edges could also be defended. This edge defence is different from the one we model, in that our defenders choose exactly which edges to defend, whereas in Michalak [42] a decision is made to defend edges of a given vertex, and which of these edges are defended is probabilistic. Furthermore, vertex defence was also allowed as part of the same strategies as this edge defence, and the spread of the fire was probabilistic, whereas we model it as deterministic. Michalak has written several other papers using evolutionary algorithms to find solutions for the multiobjective Firefighter and its variants, including in [40] and [41] as sole author and in [43] with Knowles.

For a more in-depth review, see Wagner [48]. As far as we know, there are no published papers giving heuristics for the Edge-Defence Firefighter Problem as we have defined it.

4.1 An Integer Program for Edge-Defence Firefighter

We present a minimal integer program for Edge-Defence Firefighter. For an integer program for Classic Firefighter, see Develin and Hartke [15]. While the problem definition of Edge-Defence Firefighter does not have any concept of edges burning (only of them being defended), this idea was useful for writing the integer program so the fire would spread in accordance with the rules. Assign $D(uv, t)$ to zero if the edge (u, v) is not defended at time t and assign it to one if uv is defended at time $0 \leq t \leq T$, where T is the final time step of the game, i.e. the length of our discrete time horizon. Similarly, assign $BE(uv, t)$ resp. $BV(v, t)$ to zero if the edge uv resp. the vertex v is not burning at t, and assign one if it is burning.

In the following integer program, if a vertex v catches fire in time step t, then all of its undefended edges catch fire in time $t + 1$. Once an edge has caught fire in time $t + 1$, any susceptible end points of that edge catch fire at time $t + 1$ too. This means that overall if vertex v catches fire at time t and it is connected to vertex w via an edge that is undefended at time $t + 1$, then the w will ignite at time $t + 1$, as the rules require.

minimise: $\sum_{v \in V} BV(v, T)$

subject to:

$$BE(uv,t) \geq BE(uv,t-1), \quad (u,v) \in E, 0 < t \leq T, \tag{4.1}$$
$$BV(v,t) \geq BV(v,t-1), \quad v \in V, 0 < t \leq T, \tag{4.2}$$
$$D(uv,t) \geq D(uv,t-1), \quad (u,v) \in E, 0 < t \leq T, \tag{4.3}$$
$$\sum_{u,v \in V} D(uv,t) - D(uv,t-1) \leq 2d, \quad (u,v) \in E, 0 < t \leq T, \tag{4.4}$$
$$D(uv,t) = D(vu,t), \quad (u,v) \in E, 0 \leq t \leq T, \tag{4.5}$$
$$BE(uv,t) = BE(vu,t), \quad (u,v) \in E, 0 \leq t \leq T, \tag{4.6}$$
$$BV(u,t) \geq BE(uv,t-1), \quad (u,v) \in E, 0 < t \leq T, \tag{4.7}$$
$$BE(uv,t) \geq BV(u,t) - D(uv,t), \quad (u,v) \in E, 0 < t \leq T, \tag{4.8}$$
$$D(uv,0) = 0, \quad (u,v) \in E, \tag{4.9}$$
$$BV(v,0) = 1, \quad v \in F, \tag{4.10}$$
$$BV(v,t) \in \{0,1\}, \quad v \in V, 0 \leq t \leq T. \tag{4.11}$$

The monotonic increase of edge resp. vertex burning is ensured by (4.1) resp. (4.2). Condition (4.3) ensures defence on an edge increases monotonically; while (4.4) ensures the defenders stick to the defence budget of d per time step. An edge $e = (u,v) = (v,u)$ has a uniquely determined amount of defence resp. burning per time step due to (4.5) resp. (4.6). Fires spread from newly burning edges to their vertices after one time step due to (4.7). Condition (4.8) has fires spread from burning vertices to their undefended edges in the same time step that the vertices start burning. No edges are initially defended due to (4.9); while the vertices F are forced to burn at time step 0 by (4.10). Finally, vertex burning is restricted to integer values by (4.11).

The times taken for various experiments to run can be seen in Tables 4.1 and 4.2, which will be discussed further below. In general, the average time taken for the above program to run on a single instance with $n = 100$ using the trivial upper bound for the number of time steps of $n - f$ varied between 3.2 and 26.5 seconds. The average time taken was longer for the denser graphs we tested and shorter for instances with more defenders or more initial fires. Further experiments using better upper bounds for the number of time steps showed that the time to find an optimal result was longer for larger graphs. Although these calculation times are very short, the graphs tested were much smaller than many epidemiological networks, which we conjecture would have a much larger run time. For this program to run as quickly as possible, it may be helpful to add extra inequalities that are valid either for optimal defence or for any game. These inequalities and their effect on experimental running times shall be the focus of the rest of this section.

Upper Bounds for T

Setting limits on the upper value of T could be very useful for reducing the running time, since it reduces the number of variables in the problem and was important for reducing the running times for Classic Firefighter - see García-Martínez et al.

[24] and Ramos et al. [47].

As with Classic Firefighter, the game ends in time step T if either

1. a vertex ignites at time T, but the fire cannot spread any further in the following time steps independent of what happens defence-wise, i.e. even if we stop defending all together, or

2. no vertex ignites at time T, but a vertex ignited at time $T-1$ and without defending in time T, the fire would continue to spread.

In the integer program above, all variables are fixed after time T.

For an optimal defence strategy, we denote by T^{opt} the time step when the game ends. We note that there may be more than one optimal defence strategy and that different optimal defence strategies may result in different values for T^{opt}. If this is the case, we define T^{opt} as the smallest among all values, i.e., we choose the optimal defence strategy that ends the game the quickest.

For the game to continue in time t there must either be a vertex which was not burning at the start of the game which can ignite in t or would ignite if we did not defend one of its edges in t. This instantly leads to a trivial upper bound on T:

$$T^{opt} \leq T^1 = n - f, \tag{4.12}$$

i.e. T^{opt} is at most the number of vertices that were not burning at the start of the game.

It was first observed by Ramos et al. in [47] that for Classic Firefighter, there always exists an optimal solution which uses the full defence budget at every time step t in $1 \leq t < T$, via a discussion of the new integer program for Classic Firefighter they derived. They then used this to derive the following upper bound for T^{opt}:

Lemma 57 (From Section 3.2 of Ramos et al. [47])**.** *For a game of Classic Firefighter on a graph with n vertices and with a defence budget d, $T^{opt} \leq \left\lceil \frac{n}{d} \right\rceil$.*

We first provide our own proof that for Classic or Edge-Defence Firefighter, there exists an optimal strategy which uses the full defence budget for all time steps t in $1 \leq t < T$.

Lemma 58. *Let D be an optimal defence strategy for an instance of Classic Firefighter or Edge-Defence Firefighter that ends the game after T^{opt} steps. If D does not use the maximum defence budget in every time step up to $T^{opt}-1$, then there also exists an optimal defence strategy D' which does use the full defence budget in every time step up to step $T^{opt}-1$, and ends the game after exactly the same number of time steps. Hence, calculating T for D also gives the value of T for D' and vice versa.*

Proof. Let D be an optimal defence strategy for an instance of Classic resp. Edge-Defence Firefighter that ends the game after T^{opt} time steps. Assume that D does not defend exactly d non-burning vertices resp. edges at time $t \leq T^{opt}-1$. As the game does not end in t and T^{opt} is minimal, there must be at least one

susceptible vertex resp. edge at time T^{opt} under defence D that is defended only in T^{opt}. Therefore, we can defend this vertex resp. edge already at time t and obtain an alternative defence D' which has the same burning and also ends after T^{opt} steps. □

Whereas Lemma 57 relies solely on using the full defence budget for each time step t in $1 \leq t < T$, a better bound can be easily derived from the observation that for the game to continue, at least one new vertex must catch fire every time step. Hence, the game must end after at most $\lceil \frac{n-f}{d+1} \rceil$ time steps for Classic Firefighter.

The limit $T^{opt} \leq \lceil \frac{n-f}{d+1} \rceil$ does not apply to Edge-Defence Firefighter, since it is not guaranteed that d vertices can be saved per time step. However, by considering that at least d edges will be defended in each time step t in $1 \leq t < T$ and that at least one new edge must burn, we reach the following limit on T^{opt} for Edge-Defence Firefighter:

$$T^{opt} \leq T^2 = \left\lceil \frac{m}{d+1} \right\rceil. \tag{4.13}$$

Note that defending all edges adjacent to a vertex will save the vertex. We denote by $\Delta(G)$ the maximum degree of any vertex in a graph G. If we have $d \geq \Delta(G)$, then we can improve on Lemma 58 and say that there exists an optimal strategy which not only uses the full defence budget in each time step t for $1 \leq t < T$, but that also does not defend any edges unnecessarily - i.e. if not defending that edge leads to exactly the same vertices burning.

Lemma 59. *Let D be an optimal defence strategy for an instance of Edge-Defence Firefighter that ends the game after T^{opt} steps and assume that $d \geq \Delta(G)$. If D does not use the maximum defence budget in every time step up to $T^{opt} - 1$ or defends an edge of a vertex that would be saved without any of its edges being defended in any time step up to T^{opt}, then there also exists an optimal defence strategy D' which does use the full defence budget in every time step up to step $T^{opt} - 1$, defending only edges of vertices which need to have at least one edge defended to be saved, and ends the game after exactly the same number of time steps.*

Proof. Let D be an optimal defence strategy for an instance of Edge-Defence Firefighter that ends the game after T^{opt} time steps. Assume that D either does not spend all of its budget at time $t \leq T^{opt} - 1$ or spends it on an edge of a vertex i that would not need any of its edges to be defended in order to be saved at time t. Let d^t be this unused or miss-used budget, so $d^t = \frac{1}{2}\left(2d - \Sigma_{(u,v)\in E} D(uv,t) - D(uv,t-1)\right) \geq 0$ if the budget was not used at all or $d^t = \Sigma_{(i,v)\in E} D(iv,t)$ for such a vertex i. As the game does not end in t and T^{opt} is minimal, there must be either at least one susceptible vertex $j \in V$ at time T^{opt} with at least one edge e_j that still needs to be defended to prevent j igniting and no new vertices catch fire, or a new vertex $j \in V$ is igniting at time T^{opt} but then the fire is fully contained.

Consider the first case, where at least one edge e_j of a susceptible vertex j is defended in T^{opt} and there is no further burning in that time step. The leftover

budget d^t could be spent on defending edges of j, resulting in an alternative defence D' which has the same burning and also ends after T^{opt} steps. Note that we can spend all of d^t on protecting j, because otherwise we could have completely defended j by defending all of its edges before T^{opt}, contradicting the assumption that T^{opt} is the minimum number of time steps for a game with optimal defence to end in.

In the second case, i.e. a new vertex j ignites in T^{opt}, then the left-over budget $d^{T^{opt}}$ in T^{opt} must be strictly less than $d(j)$, otherwise we could defend every edge of j and save it, contradicting the optimality of D. Let k be a vertex whose edges are defended in T^{opt}. We must have $d^t \leq d(k)$ as otherwise, you could defend every edge of k at time t, leaving the full budget of $d \geq \Delta(G)$ available at time T^{opt}, which you could then use to defend every edge of j and save it, contradicting the optimality of D. We therefore consider a defence strategy D'' which uses all of d^t to defend edges of k in time t. This would result in the same amount of burning and still end in T^{opt} time steps. Hence, the claim is proven. □

To derive our next improved upper bound, we define a function $c(v_i)$, which shows how many edges are sufficient to be defended to save a vertex v_i. We have:

$$c(v_i) \;=\; \begin{cases} d(v_i), & v_i \in V \setminus F, \\ 0, & v_i \in F. \end{cases}$$

Then, we consider a permutation σ of $\{1, \ldots, n\}$, such that, $c(v_{\sigma(1)}) \geq c(v_{\sigma(2)}) \geq \ldots \geq c(v_{\sigma(n)})$. Let r be the largest integer such that $\sum_{i=1}^{r} c(v_{\sigma(i)}) \leq d$. Then r is the minimum number of vertices one can defend per time step. We obtain the following Lemma:

Lemma 60. *For the Edge-Defence Firefighter, we have* $T^{opt} \leq T^3 = \left\lceil \frac{n-f}{r+1} \right\rceil$.

Proof. Assume there exists an instance where $T^{opt} > \left\lceil \frac{n-f}{r+1} \right\rceil$. After $\left\lceil \frac{n-f}{r+1} \right\rceil$ time steps, at least $r \left\lceil \frac{(n-f)}{r+1} \right\rceil$ vertices have been defended, leaving at most $(n-f) - r \left\lceil \frac{n-f}{r+1} \right\rceil$ vertices which could burn after time 0 and this number of burning vertices is bounded as follows:

$$(n-f) - r \left\lceil \frac{n-f}{r+1} \right\rceil \;\leq\; (n-f) - r\frac{n-f}{r+1} \;=\; \frac{n-f}{r+1} \;\leq\; \left\lceil \frac{n-f}{r+1} \right\rceil .$$

If at least one vertex ignites in time step T^{opt} then strictly more than $\left\lceil \frac{n-f}{r+1} \right\rceil$ vertices must burn after time step zero, a contradiction. If no new vertices ignite in time step T^{opt}, then at least one vertex must have ignited at time $T^{opt} - 1$ and at least one vertex needed to be defended at time T^{opt} which was not defended at time $T^{opt} - 1$. However, for this to happen we must have:

$$\left\lceil \frac{n-f}{r+1} \right\rceil \;\leq\; T^{opt} - 1 \;\leq\; (n-f) - r \left\lceil \frac{n-f}{r+1} \right\rceil \;\leq\; \left\lceil \frac{n-f}{r+1} \right\rceil ,$$

hence:

$$T^{opt} - 1 \;=\; \left\lceil \frac{n-f}{r+1} \right\rceil \;=\; \frac{n-f}{r+1}.$$

The second inequality comes from the fact that at each time step before T^{opt}, a vertex must burn and at most $(n-f) - r\left\lceil \frac{n-f}{r+1} \right\rceil$ vertices could burn after time 0.

But then, at the end of time $T^{opt}-1$, $\frac{n-f}{r+1}$ vertices are burning and at least $r\left\lceil \frac{n-f}{r+1} \right\rceil = r\frac{n-f}{r+1}$ are defended, leaving no vertex that would need to be defend at time T^{opt}, a contradiction. □

To derive our final improved upper bound, we relabel the vertices such that their degree is non-decreasing, i.e., $d(v'_1) \leq d(v'_2) \leq \ldots \leq d(v'_n)$.

Lemma 61. *For Edge-Defence Firefighter, we have $T^{opt} \leq T^4$ where*

$$T^4 \;=\; \min\left\{ t \mid \sum_{r=t+f}^{n} d(v'_r) \leq t \cdot d \right\}.$$

Proof. Assume that the game ends at time $T^{opt} > T^4$. By definition

$$\sum_{r=T^4+f}^{n} c'_r \leq T^4\, d\,,$$

that is, we have accumulated enough budget until time T^4 to defend the $(n-f) - T^4 + 1$ most expensive vertices to defend. As $T^{opt} > T^4$, at least one new vertex must ignite at every time step $1, \ldots, T^4 - 1$, resulting in at least $T^4 - 1$ burning vertices. Therefore, the game must end at time T^4 (at the latest), as we have sufficient defence budget to defend all non-burning vertices until T^4. □

Upper Bounds for T on Trees

In this section, we focus on T^{opt} for games on trees.

Observation 62. *Increasing d will always lead to optimal defence taking at most as many time steps as it leads to faster containment. In contrast, increasing f may or may not make the game last longer.*

Corollary 63. *If we want an upper bound of $T^{opt}(G, f, d)$ for any value of d, we only have to consider the case $d = 1$.*

Lemma 64. *There does not exist a tree G with $T^{opt} > 1$ such that $T^{opt}(G, f, 1) = \left\lceil \frac{n-f}{2} \right\rceil$ for Classic or Edge-Defence Firefighter.*

Proof. Assume that there exists a tree with $T^{opt} > 1$ and $T^{opt}(G, f, d) = \lceil \frac{n-f}{2} \rceil$ with a single defender. Note that for trees, for every vertex defence strategy there exists an edge defence strategy in which the same vertices burn in the same time steps by Observation 25, so we prove the claim for Classic Firefighter. Following the logic in the proof of Lemma 60, for the upper limit on T^{opt} to be met we must have exactly one new vertex igniting per time step if at least one vertex ignites in the final time step, or one vertex igniting per time step until $T^{opt} - 1$ if no new vertices ignite during T^{opt}; and exactly d vertices being defended with no vertices being saved without being themselves defended. This means that only one new

vertex f_1 can ignite at time step 1. We choose one of its neighbouring initial fires arbitrarily and label it f'. Any other fires either have no susceptible neighbours at time 0 other than f_1; or all such neighbours are defended in time step 1. We call such vertices $v_1, v_2 \dots v_d$.

Let f_1 be the single vertex that ignites in time step 1. After time step 1, at most one connected set of vertices F' can be spreading the fire, with any other burning vertices separated from the rest of the graph by a defender. Defence for the rest of the game then becomes equivalent to a game where F' is represented as a single, burning vertex. See Figures 4.1 and 4.3.

For such games, optimal defence defends only threshold vertices (see Costa et al. [10]), so we only consider defence strategies that defend threshold vertices for the rest of the game.

Defending a vertex saves all its descendants so the only way for defending only threshold vertices to save only the vertices defended is for those vertices to be leaves. Furthermore, for only one vertex to ignite per time step, each vertex that ignites after time step $0'$ must have degree three (one neighbour is the previously ignited vertex, one neighbour is the vertex that will be defended in that time step and the final neighbour is the next vertex the fire spreads to). This leaves only one possible structure for the graph and one possible defence strategy, shown in Figures 4.1 and 4.3, however this defence strategy can be improved upon significantly (see Figures 4.2 and 4.4). □

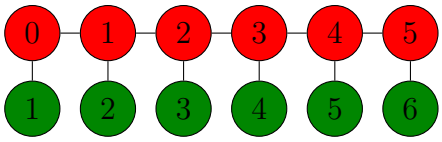


Figure 4.1: The only scenario where a game with a single fire on a tree could reach the upper limit for T^{opt}. Here $d = 1$. Red resp. green vertices are burn resp. are defended at the indicated time steps

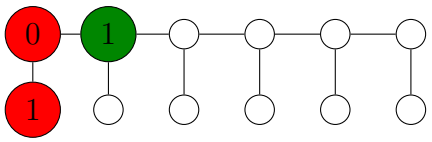


Figure 4.2: Optimal defence for the instance of Classic Firefighter shown in Figure 4.1. The game ends at $T = 1$.

Conjecture 65. *The balanced tree has the greatest value of T^{opt} amongst all trees with n vertices for both Classic and Edge-Defence Firefighter.*

Valid Inequalities

We shall investigate the effect of two closely related valid inequalities on the run times for the integer program. The first states that an edge cannot be simultaneously burning and defended at any time step:

$$BE(uv, t) + D(uv, t) \leq 1, \quad (u, v) \in E, 0 \leq t \leq T. \tag{4.14}$$

The second states that an edge cannot be simultaneously burning and defending at the final time step:

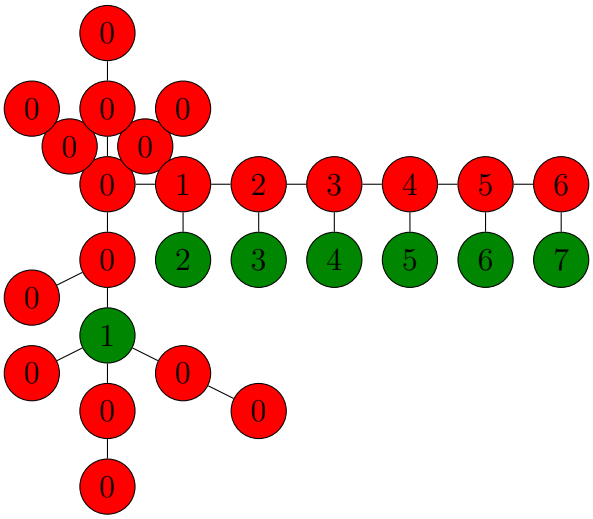


Figure 4.3: A game with multiple initial fires, using the defence strategy described in the proof of 64. Note that after time step 1, defence becomes equivalent to that shown in Figure 4.1.

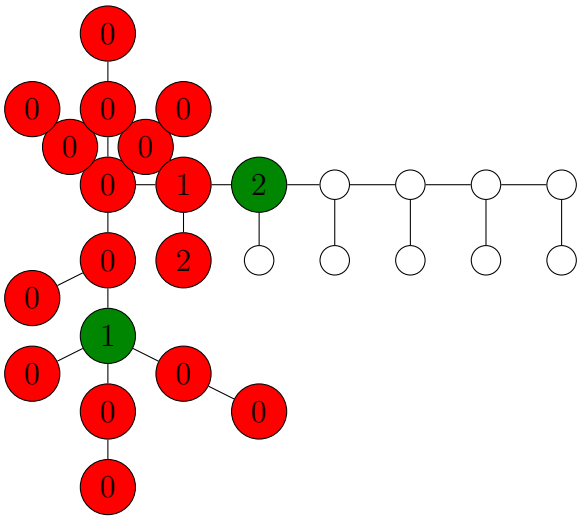


Figure 4.4: A better defence technique than that shown in Figure 4.3. Again, after time step 1 this is equivalent to the defence technique shown in Figure 4.2.

$$BE(uv,T) + D(uv,T) \leq 1, \quad (u,v) \in E. \tag{4.15}$$

Computational Results for Linear Program

We need to decide the combinations of parameters with which we run the computational experiments. Intuitively, the exact defence strategy used will matter less for instances with a low ratio of starting fires to defenders, than for game instances with a high ratio. This is because with enough defenders, even a sub-optimal defence strategy will defend some useful edges. To emphasise the differences between the optimal solution and the different heuristics we will later use, we therefore focus on game instances with a high ratio of fires to defenders. For dense graphs, this could again mask the differences between defence strategies if an extremely large number of vertices burn regardless of defence strategy is. We therefore focus on games on sparse graphs.

We start by optimising the time taken for the integer program to run. We ran experiments with one hundred instances each for each of the eight possible combinations of $n = 100$, $m \in \{100, 110\}$, $f \in \{5, 10\}$ and $d = \{1, 2\}$ and for the eight possible combinations of $n = 200$, $m \in \{205, 220\}$, $f \in \{10, 20\}$ and $d \in \{2, 4\}$.

Note that for $n = 100$, since $d \in \{1, 2\}$, the upper bounds for T provided by Lemmata 60 and 61 will almost always be much larger than $T^1 = n - f$ or $T^2 = \left\lceil \frac{m}{d+1} \right\rceil$. This is because the defence budget must be at least $\Delta(G) = \max\{d(v) : v \in G\}$, but it is not possible for a connected graph G to have $\Delta(G) = 1$ and it is only possible for a connected graph to have $\Delta(G) = 2$ if it is either the path or the cycle on n vertices. Since the graphs are so sparse we have $\left\lceil \frac{m}{d+1} \right\rceil < n - f$, so the former will always provide the lowest upper bound for T out of options T^1 to T^4 above. For $n = 200$, T^3 or T^4 may provide a lower bound than T^3, again dependant on the maximum degree of the graph being at most the defence budget d. For a given instance, if we had $\Delta(G) > d$, then the lower upper bound for T was given by $T^2 = \left\lceil \frac{m}{d+1} \right\rceil$; if $\Delta(G) \leq d$, then the lower upper bound for T was given by the minimum of T^2, T^3 and T^4.

To generate the instances, we first chose a graph uniformly at random from the list of trees on n vertices, then added the $m - (n-1)$ further edges required by choosing a random sample without replacement from the combinations of two vertices in the graph. This ensures the graphs are connected. Since the labelling of this graph was random, the starting fires were then selected at random by taking the vertices labeled $0, 1, \ldots, f-1$. For each combination of n and m, we used the same 100 graphs for every experiment in this chapter, with the same vertex labelling. All experiments were run on a HP Elitebook with an Intel Core i7-8665U processor. All linear programming experiments were implemented in Mosel and run on FICO Xpress IVE using optimiser version 34.01.05.

The results from the integer program optimisation are given in Tables 4.1 and 4.2. The first column shows which parameter combination is being considered and the remaining columns show the mean time (in seconds) it took to find the optimal solution. The second to fifth columns of Table 4.1 use the upper bound of T of $T = T^1 = n - f$; while the last three columns use the lowest appropriate

upper bound from T^1, T^2, T^3 and T^4. Since the lower upper bound made such a large difference on the run times for $n = 100$ and this agrees with the literature on Classic Firefighter (specifically García-Martínez et al. [24] and Ramos et al. [47]), only the lower bound of T was used for the $n = 200$ experiments. The valid inequalities used are also indicated at the top of each column.

For each combination of parameters, the fastest mean time is highlighted in bold. For $n = 100$, it is obvious that the lower bound for T improves performance, but the effect of adding valid inequalities (4.14) and/or (4.15), is mixed. For $n = 200$, overall not including either valid inequality generally provides the fastest mean run time. It is also worth noting that when the 'No valid inequalities' column has the fastest mean run time, it is often by a significant margin (e.g. for $m = 220$, $f = 20$, $d = 2$ it it less than 60% of the next fastest mean run time); whereas for the two parameter combinations where it was not the fastest (for $m = 220$, $f = 10$, $d = 4$; and $m = 220$, $f = 20$, $d = 4$), its mean run time was less than a second slower than the fastest mean. Overall, we recommend not using the inequalities (4.14) and (4.15), especially for larger graphs where the run time will be more significant. We hypothesise this is because the cuts to the feasible region provided by those valid inequalities are very shallow, so adding them as further constraints just increases the run time. In all cases, keeping the same graph density (and indeed graphs), but increasing the number of defenders significantly reduced run time; and instances with more edges with the same number of starting fires and defenders had significantly slower run times.

Table 4.1: Mean time taken in seconds for the integer program to run for $n = 100$, under various conditions.

	$T^{max} = T^1 = n - f$			$T^{max} = \min\{T^1, T^2, T^3, T^4\}$		
Parameters (m, f, d)	4.14	4.15	No valid inequalities	4.14	4.15	No valid inequalities
100, 5, 1	6.1	6.4	6.0	**4.3**	4.5	4.6
100, 5, 2	4.6	2.6	3.4	1.2	**1.2**	1.3
100, 10, 1	9.9	4.9	5.2	**3.4**	3.4	3.9
100, 10, 2	5.0	3.3	3.2	1.5	**1.4**	1.6
110, 5, 1	52.2	31.2	26.5	23.8	27.5	**21.1**
110, 5, 2	9.4	5.9	6.2	2.7	**2.4**	3.1
110, 10, 1	36.1	23.9	20.5	19.3	20.0	**17.5**
110, 10, 2	17.1	12.5	11.2	**6.0**	6.0	6.3

4.2 Heuristics for Edge-Defence Firefighter

GREEDY

All of the fire component based heuristics we will examine will only defend threshold edges. As before, an edge is in the *threshold* if one end is susceptible and the other end is burning. Preliminary experiments showed that a 'purely' greedy heuristic - which defended the most central edges, regardless of whether they were

Table 4.2: Mean time taken in seconds for the integer program to run for $n = 200$, under various conditions.

	$T^{max} = \min\{T^1, T^2, T^3, T^4\}$		
Parameters (m, f, d)	4.14	4.15	No valid inequalities
205, 10, 2	13.0	14.2	**9.9**
205, 10, 4	2.6	2.3	**2.1**
205, 20, 2	17.0	19.4	**11.2**
205, 20, 4	4.9	5.5	**3.3**
220, 10, 2	87.8	84.7	**50.8**
220, 10, 4	5.4	**3.9**	4.7
205, 20, 2	47.2	32.4	**31.7**
205, 20, 4	10.2	**7.8**	8.1

in the threshold or not - performed very badly, so to allow for fair comparison, we will only compare the fire component based heuristics with the 'greedy' heuristic for edge defence given in Algorithm 7. Note it does not specify which centrality measure we shall used, this will be further discussed later.

Input: A graph G; the set of burning vertices F; an integer value of number of defenders d; and a centrality measure f
Output: A vertex resp. edge defence strategy, D
for *defender move in FIREFIGHTER* **do**
  create a list *defence priorities* of all threshold edges in descending order of their image in f;
  defend the first d edges in *defence priorities*;
end

Algorithm 7: Heuristic for edge defence which uses a centrality measure and knowledge of which edges are in the threshold at a given time step. In this section it is known as 'GREEDY'

COMPONENT

All further heuristics in this section require the graph to be divided into *fire components*. This is done using Algorithm 2 on page 28, Section 2.6. They are arranged in this section loosely in order of increasing complexity.

The basic idea of the first heuristic is to divide the graph into fire components, then defend in the threshold, going component by component. A component gets higher priority if it has a large ratio of vertices within it to initially burning vertices - a possible indicator that defending a few edges could save a large number of vertices inside that component. This is shown in Algorithm 8. Note that the threshold will change in each consecutive time step.

Input: A list of fire components $C = C_1, C_2, \ldots$ for an instance of FIREFIGHTER on a graph G; the set of burning vertices F
Output: An edge defence strategy, D
create an empty mapping *component weights*;
for *component* C_i *in* C **do**
  $weight = \frac{|C_i|}{|\{v \in C_i | v \in F\}|} n^2$;
  while *weight is in image of* component weights **do**
    $weight += 1$
  end
  component weights$(C_i) = weight$;
end
create a list *ordered components* of components ordered by their image in *component weights*;
start playing FIREFIGHTER;
for *defender move in FIREFIGHTER* **do**
  create an empty list *defence priorities*;
  for *component in* ordered components **do**
    create a list *thresh* of threshold edges;
    append *thresh* to *defence priorities*;
    defend the first d edges in *defence priorities*;
  end
end

Algorithm 8: Heuristic for edge defence which uses only fire components and no centrality measures. For the rest of this section, it is known as 'COMPONENT'.

COMPONENT HIGH

This can be further refined by ordering the threshold edges of each fire component according to some centrality measure. This is shown in Algorithm 9. The exact nature of the centrality measures used will be discussed later. All further heuristics in the section will make use of centrality measures.

Input: A list of fire components $C = C_1, C_2, \ldots$ for an instance of FIREFIGHTER on a graph G; the set of burning vertices F; and a centrality measure f
Output: An edge defence strategy, DE
create an empty map *component weights*;
for *component* C_i *in* C **do**
  $weight = \frac{|C_i|}{|\{v \in C_i | v \in F\}|} n^2$;
  while *weight is in image of* component weights **do**
    $weight += 1$
  end
  component weights$(C_i) = weight$;
end
create a list *ordered components* of components ordered by their image in *component weights*;
start playing FIREFIGHTER;
for *defender move in FIREFIGHTER* **do**
  create an empty list *defence priorities*;
  for *component in* ordered components **do**
    create a list *thresh* of threshold edges, in descending order of their image in f;
    append *thresh* to *defence priorities*;
    defend the first d edges in *defence priorities*;
  end
end

Algorithm 9: Heuristic for edge defence that uses both fire components and a centrality measure. For the rest of this section, it will be known as 'COMPONENT HIGH'

COMPONENT JUMP

Assigning the components weights as a function of the size of the component divided by the number of starting fires inside it often leads to tiebreaks. Whereas Algorithm 8 and 9 broke these arbitrarily, our next two heuristics will focus on different ways to deal with tiebreaks. They both build on Algorithm 9 in that they incorporate a centrality measure. The first of the two we will consider, Algorithm 10, avoids tiebreaks entirely by treating all components with the same ratio vertices to starting fires equally, and defending the most central edge from any component with a given weight.

Input: A list of fire components $C = C_1, C_2, \ldots$ for an instance of FIREFIGHTER on a graph G; the set of burning vertices F; and a vertex resp. edge centrality measure f
Output: An edge defence strategy, DE.
create an empty map *component weights*;
for *component* C_i *in* C **do**
    $weight = \frac{|C_i|}{|\{v \in C_i | v \in F\}|}$;
    $component\ weights(C_i) = weight$;
end
create a list *ordered component weights* of the component weights in descending order;
start playing FIREFIGHTER;
for *defender move in FIREFIGHTER* **do**
    create an empty list *defence priorities*;
    for component weight *in* ordered component weights **do**
        create a list *thresh* of threshold edges e in any component with the weight *component weight*, ordered by the value of the sum $f(e) = f(u) + f(v)$ resp. $g(e)$;
        append *thresh* to *defence priorities*;
        defend the first d edges in *defence priorities*;
    end
end

Algorithm 10: Heuristic for edge defence that uses both fire components and a centrality measure. If two components have the same weight then the algorithm can 'jump' between them - defending the most central threshold edge from any component with a given weight. For the rest of this chapter, it will be known as 'COMPONENT JUMP'

COMPONENT DENSITY TIEBREAK

Our next attempt at dealing with the question of tiebreaks between is to prioritise denser components, as the fire would on average spread faster within them. This approach is shown in Algorithm 11.

Input: A list of fire components $C = C_1, C_2, \ldots$ for an instance of FIREFIGHTER on a graph G; the set of burning vertices F; and a centrality measure $f : V \to \mathbb{R}$ for vertex defence or $g : E \to \mathbb{R}$ for edge defence.
Output: An edge defence strategy, DV resp. DE.
for *component* C_i *in* C **do**
    $weight = \frac{|C_i|}{|\{v \in C_i | v \in F\}|}$;
    component weights$(C_i) = weight$;
end
create a list *ordered component weights* of the component weights in descending order;
for component weight *in* ordered component weights **do**
    create an ordered list *components by density* of the components with weight *component weight* in descending order by their density;
    append *components by density* to *ordered components*
end
create a list *ordered component weights* of the component weights in descending order;
start playing FIREFIGHTER;
for *defender move in FIREFIGHTER* **do**
    create an empty list *defence priorities*;
    for component weight *in* ordered component weights **do**
        create a list *thresh* of threshold edges e in any component with the weight *component weight*, ordered by the value of the sum $f(v)$ resp. $g(e)$;
        append *thresh* to *defence priorities*;
        defend the first d edges in *defence priorities*;
    end
end

Algorithm 11: Heuristic for edge defence that uses both fire components and a centrality measure. If two components have the same weight based on their length and number of burning ends, then the tie is broken by giving the denser component a higher weight. For the rest of this chapter, it will be known as 'COMP DENSITY TIEBREAK'

RECALCULATED COMP'S

Our final algorithm builds on Algorithm 11, breaking ties between components based on their density but then recalculates the components every time step after the fire spreads and before deciding which edges to defend in that time step. An example showing the difference between COMPONENT HIGH and RECALCULATED COMP'S is shown in Figures 4.5 and 4.6.

Input: A graph G; the set of burning vertices F; and a centrality measure $f : V \to \mathbb{R}$ for vertex defence or $g : E \to \mathbb{R}$ for edge defence.
Output: An edge defence strategy, DE.
start playing FIREFIGHTER;
for *defender move in FIREFIGHTER* **do**
  calculate the components $C_1, C_2, \ldots$ using Algorithm 2;
  for *component* C_i *in* C **do**
    $weight = \frac{|C_i|}{|\{v \in C_i | v \in F\}|}$;
    component weights$(C_i) = weight$;
  end
  create a list *ordered component weights* of the component weights in descending order;
  for component weight *in* ordered component weights **do**
    create an ordered list *components by density* of the components with weight *component weight* in descending order by their density;
    append *components by density* to *ordered components*
  end
  create a list *ordered component weights* of the component weights in descending order;
  create an empty list *defence priorities*;
  for component weight *in* ordered component weights **do**
    create a list *thresh* of threshold edges e in any component with the weight *component weight*, ordered by the value of the sum $f(v)$ resp. $g(e)$;
    append *thresh* to *defence priorities*;
    defend the first d edges in *defence priorities*;
  end
end

Algorithm 12: Heuristic for edge defence that uses both fire components and a centrality measure. If two components have the same weight based on their length and number of burning ends, then the tie is broken by giving the denser component a higher weight. The components and their weights are recalculated again after every step. For the rest of this chapter, it will be known as 'RECALCULATED COMP'S'

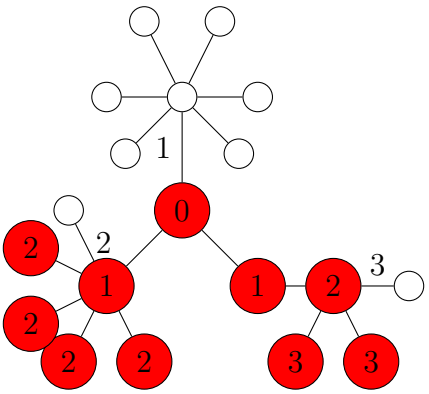


Figure 4.5: An example of a game of Edge-Defence Firefighter using COMPONENT HIGH with vertex degree as the centrality measure and a single defender. Red vertices burn and edges are defended during the indicated time steps. A total of 10 vertices burn.

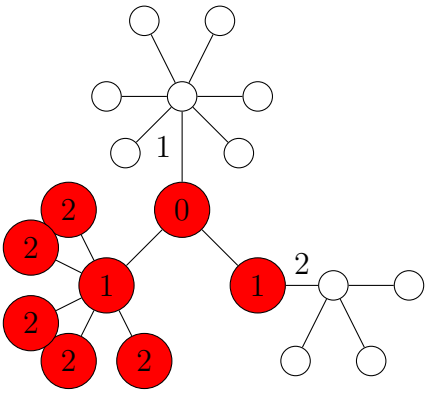


Figure 4.6: The same instance of Edge-Defence Firefighter as Figure 4.5 again with a single defender, but now using RECALCULATED COMP'S with vertex degree as the centrality measure. Only 8 vertices burn.

Centrality Measures

Having defined the heurstic algorithms, we now look at which centrality measures f to use in Algorithms 7, 9, 10, 11 and 12. The only edge centrality measure we considered was *edge betweenness*. To get more centrality measures, we created a edge centrality measure from various vertex centrality measures using the following definition:

$$f((u,v)) := g(u) + g(v), \text{where } g \text{ is a vertex centrality measure.} \tag{4.16}$$

The vertex centrality measures we considered were Katz centrality, Eigenvector centrality, closeness centrality and degree centrality.

By *degree centrality*, we simply mean the degree of a vertex, $d(v)$.

Closeness centrality is defined as follows:

$$Closeness(v) = \frac{n-1}{\Sigma_{u \in G, u \neq v} d(u,v)}$$

To define *betweenness*, we first define $\sigma(u, w)$ to be the number of distinct shortest paths between u and w; and define $\sigma(u, w|v)$ to be the number of these paths which contain vertex v. We then have:

$$Betweenness(v) = \Sigma_{u,w \in G} \frac{\sigma(u, w|v)}{\sigma(u, w)}.$$

Edge betweenness is the only centrality measure we use which is specifically designed for edges. We again use the notation $\sigma(u, w)$ for the number of distinct shortest paths between u and w; and now define $\sigma(u, w|e)$ to be the number of these paths which contain an edge e. We then have:

$$Edge\ betweenness(e) = \Sigma_{u,w \in G} \frac{\sigma(u, w|e)}{\sigma(u, w)}.$$

The essential idea of both *eigenvector centrality* and *Katz centrality* is that vertices have a high centrality value if they have many neighbours with high centrality values. To properly define them, it is first necessary to cover some linear algebra. We order and label the vertices with natural numbers arbitrarily $v_1, v_2, \ldots, v_n$. The *adjacency matrix* of a graph is the matrix A with entries $a_{i,j}$ defined as follows:

$$a_{i,j} := \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{else.} \end{cases}$$

A *left eigenvector* $\mathbf{x}$ and *eigenvalue* λ of a matrix A are a vector resp. real value which satisfy the equation $\mathbf{x}^T A = \lambda \mathbf{x}^T$. It is possible for a given matrix to have multiple eigenvectors and eigenvalues. For eigenvector centrality, we compute the eigenvector $\mathbf{x}$ which satisfies $\mathbf{x}^T A = \lambda \mathbf{x}^T$ for the largest positive eigenvalue λ_{max}. To find this, we use *power iteration*. This can converge slowly, which was a recurrent issue in the computational experiments. The *eigenvector centrality* for the i^{th} vertex is then defined as the i^{th} entry of the left eigenvector $\mathbf{x}$. Or

$$Eigenvector\ centrality(v_i) = \Sigma_j a_{i,j} \mathbf{x_j}.$$

By adding two extra parameters α and $\beta < \frac{1}{\lambda_{max}}$, we get *Katz centrality*:

$$Katz\ centrality(v_i) = \alpha \Sigma_j a_{i,j} \mathbf{x_j} + \beta.$$

The heuristic most likely to be significantly affected by the centrality measure used is GREEDY, since it relies only on knowledge of where the threshold is and which edges have the highest centrality. The heuristic that regularly performed best in preliminary experiments was RECALCULATED COMP'S - so these are the two heuristics we focus on for choosing the centrality measures to use when comparing the heuristics to each other, and later to the optimal solutions. We ran these heuristics for the 100 instances of each parameter combination, using

all the centrality measures discussed above. This included testing eigenvector centrality, but for all combinations of parameters it failed to converge within 100 iterations in over 70% of instances. Since the purpose of a heuristic is to be fast and the game instances where eigenvector centrality did converge may not be statistically representative, these results were excluded from the computational results. The only other centrality measure used that converges iteratively was Katz centrality, which converged within 100 iterations in every single instance. The results for $n = 100$ (in terms of mean number of vertices burned) are shown in Tables 4.3 and 4.5, with the mean time taken shown below the respective results, i.e. in Tables 4.4 and 4.6. The equivalent results for $n = 200$ are in Appendix C, Tables C.1,C.3, C.2 and C.4. Note that here and in the rest of the chapter, the times for the heuristic algorithms are measures in milliseconds, whereas the times for the integer program were measured in seconds. The centrality measures that lead to the lowest mean amount of burning for each heuristic and parameter combination are summarised in Table 4.7 for $n = 100$ and in Appendix C, Table C.5 for $n = 200$. Interestingly, degree centrality was rarely the best centrality measure, even though it is the only one we know of being used in a heuristic for a version of Firefighter - specifically for Classic Firefighter in García-Martínez et al. [24].

Table 4.3: $n = 100$, Mean amount of burning with different centrality measures fed into GREEDY

Parameters (m, f, d)	Edge Betweenness	Katz	Closeness	Betweenness	Degree
100, 5, 1	51.2	49.7	47.2	47.6	51.0
100, 5, 2	29.3	25.2	27.3	26.7	26.7
100, 10, 1	78.1	77.6	75.2	76.5	77.1
100, 10, 2	62.1	58.8	58.6	60.0	59.8
110, 5, 1	80.0	83.3	77.5	81.6	82.7
110, 5, 2	48.2	40.6	39.7	42.3	44.0
110, 10, 1	91.2	91.6	92.4	90.8	91.6
110, 10, 2	78.9	79.9	79.3	79.4	79.7

Table 4.4: $n = 100$, Mean amount of time with different centrality measures fed into GREEDY

Parameters (m, f, d)	Edge Betweenness (ms)	Katz (ms)	Closeness (ms)	Betweenness (ms)	Degree (ms)
100, 5, 1	26.9	5.5	11.7	24.9	2.2
100, 5, 2	26.8	5.1	11.2	24.4	1.8
100, 10, 1	32.7	6.5	13.8	29.6	2.2
100, 10, 2	30.0	5.5	12.4	27.2	1.9
110, 5, 1	30.6	7.0	13.0	27.9	2.6
110, 5, 2	29.6	6.3	12.5	26.7	2.3
110, 10, 1	29.5	6.0	12.1	26.5	2.0
110, 10, 2	29.6	6.1	12.2	26.6	2.0

Table 4.5: $n = 100$, Mean amount of burning with different centrality measures fed into RECALCULATED COMP'S

Parameters (m, f, d)	Edge Betweenness	Katz	Closeness	Betweenness	Degree
100, 5, 1	45.2	42.8	42.5	42.8	42.9
100, 5, 2	26.1	23.8	24.1	24.0	23.6
100, 10, 1	69.9	68.2	68.6	68.8	68.4
100, 10, 2	53.9	51.2	51.2	52.4	51.9
110, 5, 1	74.7	78.1	74.2	76.4	78.1
110, 5, 2	44.3	38.2	38.0	40.0	39.8
110, 10, 1	85.3	85.6	86.3	85.1	85.5
110, 10, 2	72.4	72.3	73.1	73.0	72.5

Table 4.6: $n = 100$, Mean amount of time with different centrality measures fed into RECALCULATED COMP'S

Parameters (m, f, d)	Edge Betweenness (ms)	Katz (ms)	Closeness (ms)	Betweenness (ms)	Degree (ms)
100, 5, 1	38.0	14.4	21.1	34.8	11.0
100, 5, 2	44.5	14.6	22.9	40.7	10.4
100, 10, 1	45.4	17.2	26.0	42.7	13.2
100, 10, 2	40.8	14.3	21.5	37.2	10.4
110, 5, 1	45.3	20.2	26.5	42.5	15.5
110, 5, 2	39.3	15.2	21.3	35.9	11.3
110, 10, 1	41.4	16.5	23.1	38.3	11.9
110, 10, 2	38.8	15.0	21.6	36.2	10.9

Table 4.7: Centrality measures that lead to the lowest mean amount of burning for $n = 100$.

Parameters (m, f, d)	Best centrality for GREEDY	Best centrality for RECALCULATED COMP'S
100, 5, 1	Closeness	Closeness
100, 5, 2	Katz	Degree Centrality
100, 10, 1	Closeness	Katz
100, 10, 2	Closeness	Closeness
110, 5, 1	Closeness	Closeness
110, 5, 2	Closeness	Closeness
110, 10, 1	Betweenness	Betweenness
110, 10, 2	Edge Betweenness	Katz

4.3 Heuristic Results

The results and times taken for the heuristic experiments are presented in Appendix C in Tables C.6 – C.51 below. The tables alternate between showing the results (in terms of number of vertices burned) and the times taken to run the heuristics (in milliseconds). The tables are grouped by the parameters used (sorted by n, then m, then f, then d in ascending order). The centrality measure used is whichever centrality measure gave the best results in the previous section, as summarised in Tables 4.7 and C.5. In cases where different centrality measures gave the best results for GREEDY as for RECALCULATED COMP'S, then the experiments were run using for each of those two centrality measures; with the results for the centrality that was best for GREEDY appearing first.

The best performing heuristic in terms of mean number of vertices burned was almost always RECALCULATED COMP'S - even when using a centrality measure which was optimal for GREEDY but not for RECALCULATED COMP'S. The worst performing heuristic by the same measure was always COMPONENT - although it did tie with GREEDY for the most vertices burned on average for the parameter combination $n = 200$, $m = 205$, $f = 20$ and $d = 2$. GREEDY was the second worst performing heuristic overall.

The time performance for the heuristics exactly mirrors this for those three heuristics, with COMPONENT almost always having the fastest mean run time; GREEDY usually having the second fastest mean run time; and RECALCULATED COMP'S always having the slowest mean run time. The distribution of the run times was far more consistent for all the heuristics than for the integer program.

4.4 Tiebreaks Between Components

It is obvious from the results in Tables C.6 – C.49 that COMPONENT HIGH, COMPONENT JUMP and COMP. DENSITY TIEBREAK very often produce identical or very similar results. The reason for this becomes apparent after analysing the number of components and the number of components with the

same weight for each combination of n, m and f. To do this, we used the following component weight definition for a component C_i:

$weight = \frac{|C_i|}{|\{v \in C_i | v \in F\}|}$.

This differs from the definitions of weights in COMPONENT HIGH and COMP. DENSITY TIEBREAK in that it does not attempt to tiebreak in any way between components with the same ratio of initially burning vertices to total vertices.

All of the component-based heuristics only differ from GREEDY if there are multiple components. The average number of components for the parameter combinations we used in the computational computation experiments can be seen in the second column of Table 4.8. COMPONENT HIGH, COMPONENT JUMP and COMP. DENSITY TIEBREAK can only differ if there are multiple components with the same weight. As can be seen from the third column of Table 4.8, this occurred fairly commonly. However, they are only likely to differ if the shared weight is fairly high, so those components would be defended before they burned. As we can see from the final column of Table 4.8, the average highest weight shared by at least two different components was very low. This explains the similarities between COMPONENT HIGH, COMPONENT JUMP and COMP. DENSITY TIEBREAK - any components with the same ratio of initially burning vertices to total vertices as another component were mostly completely burned before a defender could reach them.

Table 4.8: Number of components (with the same weight) for various parameters.

Parameters (n, m, f)	Mean Number of Components	Mean Number of Components with Most Common Weight	Mean Highest Shared Component Weight
$100, 100, 5$	5.3	2.1	1.8
$100, 100, 10$	9.3	3.8	2.2
$100, 110, 5$	3.4	1.8	1.3
$100, 110, 10$	6.1	3.2	1.6
$200, 205, 10$	8.5	3.6	2.1
$200, 205, 20$	15.6	7.0	3.0
$200, 220, 10$	6.2	3.2	1.7
$200, 220, 20$	11.5	6.2	2.3

4.5 Comparing Heuristics to Optimal Solutions and Conclusions

Since RECALCULATED COMP'S was by far the best-performing heuristic, we shall focus on this for comparison with the optimal solutions. Tables 4.9 and 4.10 summarise the gap between the optimum solution (calculated using the integer program) and the solution given by RECALCULATED COMP'S, using whichever centrality measure performed best for RECALCULATED COMP'S, as summarised in the right hand column of Tables 4.7 and C.5. For an optimal

solution x and heuristic solution y, the gap was calculated as the percentage increase: $\frac{y-x}{x} \cdot 100$.

The minimum gap between the optimal solution and that given by RECALCULATED COMP'S was 0% for many parameter combinations - meaning that RECALCULATED COMP'S gave the optimal solution at least once for those parameter combinations. This occurred in far more parameter combinations with $n = 100$ than with $n = 200$. The mean gap was usually smaller with instances with more starting fires; while the maximum gap was always much smaller for instances with more starting fires, which had higher optimum solutions to begin with, although the maximum burn for RECALCULATED CALC'S was always significantly below n. All heuristic algorithms were much faster than the integer program.

As stated above, we are not aware of any heuristic algorithms for Edge-Defence Firefighter as we have defined it. This means there is currently nothing with which to compare these heuristic algorithms. In future, we propose adapting the metaheuristic approaches used in Blum et al. [3] and Hu, Windbichler and Raidl [31] for Classic Firefighter to Edge-Defence Firefighter.

Table 4.9: Gap between optimum amount of burning and burning with best heuristic for $n = 100$.

Parameters (m, f, d)	Mean Optimum Value	Mean for RECALCULATED COMP'S	Min gap %	Max gap %	Mean gap %	Gap S.D.
$100, 5, 1$	34.9	42.5	0.0	78.9	22.2	17.8
$100, 5, 2$	19.9	23.6	0.0	78.9	18.7	16.7
$100, 10, 1$	60.1	68.3	0.0	39.66	14.1	10.3
$100, 10, 2$	44.3	51.2	0.0	65.9	16.1	11.9
$110, 5, 1$	51.2	74.2	0.0	151.4	46.6	29.7
$110, 5, 2$	26.4	38.0	0.0	178.3	43.1	34.8
$110, 10, 1$	73.0	85.1	0.0	53.4	16.9	8.4
$110, 10, 2$	54.7	72.3	2.6	80.5	32.9	14.5

Table 4.10: Gap between optimum amount of burning and burning with best heuristic for $n = 200$

Parameters (m, f, d)	Mean Optimum Value	Mean for RECALCULATED COMP'S	Min gap %	Max gap %	Mean gap %	Gap S.D.
$205, 10, 2$	69.4	90.6	1.3	93.5	31.2	18.4
$205, 10, 4$	39.5	48.1	0.0	51.4	21.0	13.5
$205, 20, 2$	121.6	141.3	0.8	41.0	16.4	8.4
$205, 20, 4$	89.4	107.9	3.0	47.0	21.1	9.7
$220, 10, 2$	96.8	148.3	22.6	175.9	55.9	24.6
$220, 10, 4$	51.4	77.5	3.2	117.6	50.1	27.1
$220, 20, 2$	140.0	167.6	0.6	48.7	20.3	10.1
$220, 20, 4$	105.1	140.9	4.8	91.5	34.8	13.1

Chapter 5

Limited Defence Firefighter

This chapter addresses two variants of this problem: Cost-Value Firefighter (CVF) and Distance-Limited Defence Firefighter (DLF).

- In the Cost-Value Firefighter, every vertex has both a cost - a minimum amount of defence budget that must be spent to protect it; and a value - the utility gained by protecting that vertex. The objective is then to maximise the total value of the vertices saved. Moreover, instead of a fixed number of defenders, we are given a defence budget that we can spend each time step for defending vertices. Differing from the classic model, in this variant, the defence of a vertex can be built up gradually, i.e., over multiple time steps. There has been some work assigning a value to saved vertices, see Duffy and MacGillivray [16]; and the idea of assigning only a fraction of a defender to a vertex in Coupechoux et al. [11] (discussed at length in Chapter 3, Section 3.3) bears some similarities to CVF in that vertices may not be fully defended in one time step - another important difference is that Coupechoux et al. [11] focused on *online* games where the available defence budget is not know for future time steps. We believe this is the first work to combine the cost and value of saving each vertex.

- We also deal with the Distance-Limited Firefighter, which is based on CF but adds restrictions to how far defenders can travel between time steps. In Chen et al. [8], a version with the distance limit set to one was termed *continuous firefighting*, and they studied how many defenders are needed to contain fires on an infinite square grid. DLF has attracted some attention recently, see Burgess et al. [4], Burgess et al. [5] (which introduced a different IP from the one we propose), and Days-Merill [13].

In this chapter, we analyse both problems: we derive mixed integer formulations, valid inequalities that strengthen them, and theoretical results that provide game-length bounds. The applications of these models are discussed below.

5.1 Introduction

The defence strategy in the Classic Firefighter Problem - first introduced by Hartnell in 1995 in [30] - can model various interventions including a vaccine that

prevents the vertex from being infected or from spreading the disease to other vertices; or an intervention that simply removes individuals from the area and thus from the path of the disease. Such interventions have been used to save endangered species from the Chytridiomycosis pandemic [26].

The vertices themselves could model individuals, with edges connecting individuals between whom disease could spread (for instance if they regularly come within a certain physical distance for airborne transmission). Alternatively, the vertices could be modelling population centres, with edges joining centres that in real life are linked by transportation and regularly have individuals travelling between them, potentially spreading the disease.

Although contact tracing and ring vaccination (vaccinating everyone who came into contact with an infected individual and everyone who came into contact with them etc.) within population centres are frequently used, sometimes it is preferable to vaccinate entire populations. This could be due to lack of information about contacts; a policy choice based on the cost of the vaccines and the risk to the public; or the logistical difficulty of having to visit the same population centres multiple times to administer further vaccines.

The variant CVF proposed in this chapter can better represent some real-word applications by setting:

- the cost proportional to the population size that must be vaccinated, and any extra costs related to that location; and
- the value proportional to the population size who would be protected.

This can model vaccinating population centres of varying sizes. There has been some interest in modelling the cost of vaccinating different vertices, notably in Michalak [42] - which was based on CF but uses probabilistic disease spread and control interventions beyond vaccination.

Of course, varying population sizes are not the only logistical constraint policy makers need to consider. In particular, in the second variant that we address (DLF), we focus on relaxing the assumption in CF that vaccination teams can travel arbitrarily large distances. The terrain covered by vaccination teams may be tough or simply very large; there may be extra constraints imposed by the need to maintain a cold chain for temperature sensitive vaccines; or when the health care workers require a security detail. The latter was notably the case in the $2018-2020$ Ebola outbreak in Kivu in the Democratic Republic of Congo [46]; and in polio vaccination campaigns in areas where the terrorist group Boko Haram is present [45].

5.2 Cost-Value Firefighter

We begin with the first extension. We denote the cost of defending vertex $i \in V$ by $c_i \in \mathbb{R}^+$ and its value for being saved by $p_i \in \mathbb{R}^+$. The defence budget for each time step is denoted by $B \in \mathbb{R}^+$ and we assume that $c_i \leq B$ for all $i \in V$. The game adheres to the following rules:

- All vertices in $F \subseteq V$ with $|F| = f$ are *burning* at time 0.

- In each subsequent time step $t > 0$, we first defend vertices and afterwards the fire spreads. Concerning the latter, the fire will spread from every burning vertex to all adjacent susceptible vertices, i.e., vertices that are neither burning nor defended.

  Once burning, a vertex will remain burning until the end of the game.

- In every time step $t > 0$, we can spend any amount between 0 and B on improving the defence of a vertex, as long as the combined spending per time step across all vertices does not exceed B.

  The defence of a vertex can be built up gradually, i.e., over multiple time steps. A vertex counts as *defended* if and only if the total amount of defence budget spent on the vertex so far is at least its defence cost c_i.

  Once defended, a vertex will remain defended until the end of the game.

- The game ends in time step T if either

  1. a vertex ignites at time T, but the fire cannot spread any further in the following time steps independent of what happens defence-wise, i.e. even if we stop defending vertices all together, or
  2. no vertex ignites at time T, but a vertex ignited at time $T - 1$ and without defending in time T, the fire would continue to spread.

The goal of the game is to spend the available defence budget across vertices in each time step so as to maximize the value of non-burning vertices at the end of the game.

An (optimal) solution to the game is called an *(optimal) defence strategy*. For an optimal defence strategy, we denote by T^{opt} the time step when the game ends. We note that there may be more than one optimal defence strategy and that different optimal defence strategies may result in different values for T^{opt}. If this is the case, we define T^{opt} as the smallest among all values, i.e., we choose the defence strategy that ends the game the quickest. An example for this is shown in Figure 5.3. Let $B = 1$, $c_i = 1$, $i \in V$, and $f = 1$ with the initial fire starting in Vertex 1. On the left hand-side is depicted an optimal defence strategy with duration 2, while on the right hand-side there is an optimal defence strategy with duration 1. Observe that in both strategies the profit of burning vertices is 2. In the first case, at time $t = 1$, an optimal defence strategy would spend all budget on defending Vertex 3, with a value of 2. Afterwards, the fire will spread to Vertex 2. In the next time step, it would be optimal to defend Vertex 5, with a value of 2, meaning that the fire will spread to Vertex 4. Then the game ends and the total value of the burning vertices would be 2. In the second case, at time $t = 1$, an optimal defence strategy would spend all budget on defending Vertex 2, with a value of 1. This prevents the fire from spreading to Vertices 4 and 5, with a total value of 3. Afterward, the fire will spread to Vertex 3. Then the game ends and the total value of the burning vertices would be 2. Note that the notation used in the figure to indicate whether vertices are burned or defended is described in the next paragraph and corresponds to the variables used in the formulation of the problem.

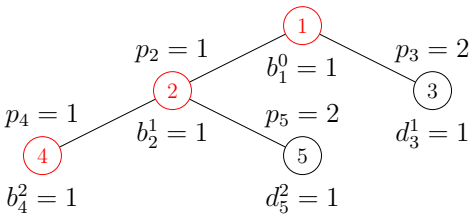


Figure 5.1: *
Two time steps

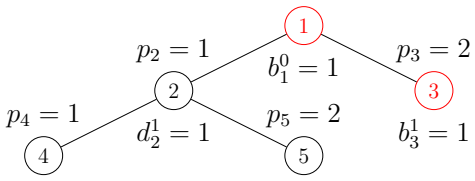


Figure 5.2: *
One time step

Figure 5.3: Example of different game durations for optimal defence strategies.

In the following, we present a mixed-integer linear programming formulation for the Cost-Value Firefighter game which is based on the one for Classic Firefighter presented in [15]. Let $T^{max} \in \mathbb{Z}^+$ be an upper bound for T^{opt}; we will discuss suitable values for T^{max} later, with a trivial example given by $T^{opt} \leq n - f$. Furthermore, we define the following decision variables

$$b_i^t = \begin{cases} 1, & \text{if vertex } i \text{ is burning at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

$$d_i^t = \begin{cases} 1, & \text{if vertex } i \text{ is defended at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

$$s_i^t = \text{ the total amount of defence budget spent on vertex } i \text{ up to and including time } t,$$

for all $i \in V$ and $0 \leq t \leq T^{max}$. Let $N(i) := \{j \in V : (i,j) \in E\}$ the set of vertices adjacent to $i \in V$. With these variables, we can formulate the problem as

$$\begin{aligned}
\min \quad & \sum_{i \in V} p_i \, b_i^{T^{max}} & & \\
\text{s.t.} \quad & b_i^t \geq b_i^{t-1}, & i \in V,\ 0 < t \leq T^{max}, & \quad (5.1) \\
& d_i^t \geq d_i^{t-1}, & i \in V,\ 0 < t \leq T^{max}, & \quad (5.2) \\
\sum_{i \in V} & \left(s_i^t - s_i^{t-1}\right) \leq B, & 0 < t \leq T^{max}, & \quad (5.3) \\
& b_i^{t-1} \leq b_j^t + d_j^t, & i \in V,\ j \in N(i),\ 0 < t \leq T^{max}, & \quad (5.4) \\
& c_i \cdot d_i^t \leq s_i^t, & i \in V,\ 0 \leq t \leq T^{max}, & \quad (5.5) \\
& s_i^0 = 0, & i \in V, & \quad (5.6) \\
& b_i^0 = 1, & i \in F, & \quad (5.7) \\
& b_i^t,\ d_i^t \in \{0,1\}, & i \in V,\ 0 \leq t \leq T^{max}, & \quad (5.8) \\
& s_i^t \geq 0, & i \in V,\ 0 \leq t \leq T^{max}. & \quad (5.9)
\end{aligned}$$

The objective function minimizes the profit of burning vertices, thereby maximizing the profit of all saved vertices. The monotonic increase of burning resp.

defence is ensured by Constraints (5.1) resp. (5.2). Constraint (5.3) ensures the defenders stick to the defence budget of B per time step. A vertex j neighbouring a burning vertex i must either burn or be defended the time step after its neighbour first burned thanks to Constraint (5.4). Condition (5.5) ensures that enough defence budget is spent on a vertex for it to be defended. No vertices start off with any defence budget spent towards them by Constraint (5.6). The fires start in the vertices of F at time 0 by Constraint (5.7) and burning and defence is binary by Condition (5.8). The defence budget spent on a vertex must be non-negative by (5.9).

For the Classic Firefighter problem, Ramos et al. [47] suggested two improvements to the formulation. The first fixes all b_i^t to 0 for $t < \min_{j \in F} d(i,j)$, as a fire cannot possibly reach vertex i by time t. However, in computational tests for graphs with up to 100 vertices (see Section 5.2 for more details), this had no effect on the run times. The second improvement is aggregating the budget constraints over time. For the Cost-Value Firefighter problem, this would mean replacing Constraint (5.3) by

$$\sum_{i \in V} s_i^t \leq t \cdot B, \quad 0 < t \leq T^{max}. \tag{5.10}$$

The equivalence of the two sets of constraints is based on the following result:

Lemma 66. *Let S be an optimal defence strategy for an instance of Cost-Value Firefighter that ends the game after T^{opt} steps and assume that $c_i \leq B$, $i \in V$. If S does not use the maximum defence budget in every time step up to $T^{opt} - 1$ or defends a vertex that would be saved without being defended itself in any time step up to T^{opt}, then there also exists an optimal defence strategy S' which does use the full defence budget in every time step up to step $T^{opt} - 1$, defending only vertices that absolutely need to be defended, and ends the game after exactly the same number of time steps.*

Proof. The proof is the same as the proof of Lemma 58 for the Edge-Defence Firefighter Problem, substituting defending edges of a given vertex with (partially) defending the vertex itself. □

As a consequence of Lemma 66, we can assume for the remainder of this section that any optimal defence strategy spends all defence budget on only necessary-to-defend vertices in every time step $t < T^{opt}$.

We defined both b_i^t and d_i^t to be binary, but it is easy to see that we can relax integrality of one of the two variables without loosing correctness of the formulation. If we relax the b_i^t variables, then by (5.4) b_j^t will be forced to 1 if i burns, unless $d_j^t = 1$. If j is defended, then b_j^t will remain 0 as we are minimizing the value of burned vertices. This always increased the run times, see Tables 5.1 and 5.2. On the other hand, if we relax the d_i^t variables, then again by (5.4) to prevent j catching fire, b_j^t will be forced to 1 if i burns at $t - 1$. Relaxing the integrality of d_i^t had a more mixed effect on the run times of the computational experiments (Tables 5.1 and 5.2), which we shall discuss later. In contrast to the previous case, we now may have fractional d_i^t variables in an optimal solution.

However, this must be for vertices which are either burning or have been saved without being fully defended, in which case setting them to 0 also yields an optimal solution. We will come back to this for the numerical experiments in Section 5.2.

Finally, we observe that increasing the defence budget B will always lead to an optimal defence that takes at most as many time steps as without that increase, since it leads to faster containment. In contrast, increasing f may or may not make the game last longer, which can be easily seen in the degenerate cases. If $f = n$, then $T^{opt} = 0$, and if $f = 0$, then $T^{opt} = 0$.

Values for T^{max}

As discussed in Section 4.1, a trivial upper bound for T^{opt} for every version of Firefighter considered in this thesis is given by $T^{max} = n - f$, the number of non-burning vertices at the start of the game. A better bound of $\lceil \frac{n}{D} \rceil$, where D is the number of available defenders was given for CF by Ramos et al. in [47] since there exists an optimal defence where the defenders will defend at least one vertex each in every time step t with $1 \leq t < T$. Taking into account the fact that a new vertex must ignite in every time step $1 \leq t < T$ for the game to continue, this can be improved to $\lceil \frac{n-f}{D+1} \rceil$, since we assume $c_i \leq B$ for all vertices. In the following, we derive two other improved upper bounds for T^{opt}, which are both based on Lemma 59.

The first one is a more accurate extension of $\lceil \frac{n-f}{D+1} \rceil$ to CVF. We start by defining revised defence costs where all initially burning vertices have a defence cost of 0.

$$c'_i = \begin{cases} c_i, & i \in V \setminus F, \\ 0, & i \in F. \end{cases}$$

Lemma 67. *Let σ be a permutation of $\{1, \ldots, n\}$, such that, $c'_{\sigma(1)} \geq c'_{\sigma(2)} \geq \ldots \geq c'_{\sigma(n)}$. Let r be the largest integer such that $\sum_{i=1}^{r} c'_{\sigma(i)} \leq B$. Then for the Cost-Value Firefighter with $c_i \leq B$, $i \in V$, we have $T^{opt} \leq T_1^{max} = \lceil \frac{n-f}{r+1} \rceil$.*

Proof. The proof is the same as the proof of Lemma 60 in Section 4.1. □

Note that this bound is tight, as the graph in Figure 5.4 shows. Let $B = 1$, $c_i = 1$, $i \in V$, and $f = 1$ with the initial fire starting on Vertex 1. We have $r = 1$ and, thus, $T^{max} = \lceil (n - f)/(r + 1) \rceil = 2$. At time $t = 1$, an optimal defence strategy would spend all the budget on defending Vertex 3, with a value of 5. Afterwards, the fire will spread to Vertex 2. In the next time step, it would be optimal to defend Vertex 5, with a value of 2, meaning that the fire will spread to Vertex 4. Then the game ends and the total value of saved vertices would be 7.

To derive our second improved upper bound, we now consider a permutation β that sorts the vertices in the opposite way, i.e., such that the revised costs of defending vertices are non-decreasing, i.e., $c'_{\beta(1)} \leq c'_{\beta(2)} \leq \ldots \leq c'_{\beta(n)}$. Note that the first f costs are 0, belonging to the initially burning vertices.

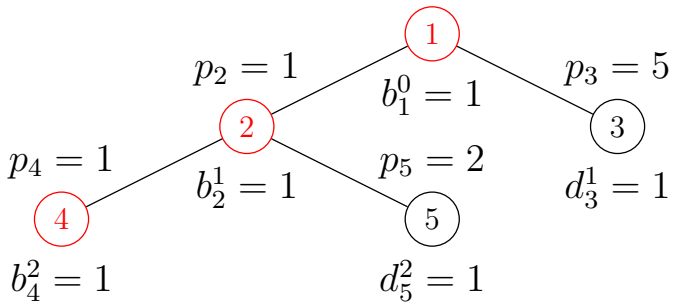


Figure 5.4: Example of tightness of T_1^{max}.

Let $1 \leq t \leq n - f$. If

$$\sum_{r=t+f}^{n} c'_{\beta(r)} \leq t \cdot B,$$

then we have enough budget to defend the $n-(t+f)+1$ most expensive vertices within t periods. Finding the smallest t for which this holds gives us our second bound:

Lemma 68. *For Cost-Value Firefighter, we have $T^{opt} \leq T_2^{max}$ where*

$$T_2^{max} = \min \left\{ t \,\middle|\, \sum_{r=t+f}^{n} c'_{\beta(r)} \leq t \cdot B \right\}.$$

Proof. The proof is the same as the proof of Lemma 61 in Section 4.1. □

Corollary 69. *For Cost-Value Firefighter, we have $T^{opt} \leq \min\{T_1^{max}, T_2^{max}\}$.*

We will numerically compare the bounds and assess their computational impact later in the section .

Valid Inequalities

In this section, we derive several valid inequalities, which will be tested in the next section for their efficiency.

The first one states that a vertex cannot be both burning and defended:

$$b_i^t + d_i^t \leq 1, \qquad i \in V, 0 \leq t \leq T^{max} \tag{5.11}$$

We note that this inequality is not necessary for the correctness of the formulation for the Cost-Value Firefighter game. If a vertex is already burning, then as we are minimizing the profit of burning vertices, there is no point objective-wise to defend the vertex. On the other hand, if the vertex is already defended, then by Constraint (5.4) it cannot catch fire. The constraints can also just be formulated for the last period T^{max}, but in our computational results, this version was inferior to (5.11).

The next valid inequality states that the accumulated defence budget spent on a vertex is monotonic:

$$s_i^{t-1} \leq s_i^t \qquad i \in V,\, 0 < t \leq T^{max}. \tag{5.12}$$

The next valid equality states that we do not spend anything on a vertex that will not eventually be defended and that we do not spend more than is necessary on a given vertex:

$$s_i^{T^{max}} = c_i \cdot d_i^{T^{max}}, \qquad i \in V. \tag{5.13}$$

Computational Experiments

We coded the mixed-integer programming formulation in C++ using CPLEX 20.1 and ran the experiments on a Dell PowerEdge R740 server running Scientific Linux 7 using up to 8 threads.

To generate the graphs, we drew coordinates for the vertices uniformly at random over the square $[0, 10]^2$. We then computed the Gabriel graph and the Delauney triangulation for the point sets. Both graphs are connected and planar, with the Gabriel graph being a proper subgraph of the Delauney graph. The average vertex degree over all generated instances for the Gabriel and Delauney graphs is 3.5 and 5.1, respectively. For the number n of vertices, we considered values of 20, 30, 40, 50, 60, 80, 100, 150, 200, 250, and 300. In the following, we label graphs with 20-50 vertices as small, with 60-100 vertices as medium, and with 150 vertices and more as large. The number of starting fires for small and medium graphs was set to 2 and 3, respectively. For large graphs, f is 4 for up to 200 vertices and 5 for the largest graphs. The vertex values p_i, $i \in V$, were drawn uniformly at random from the interval $[5, 25]$. The defence cost for a vertex i was drawn uniformly at random from the interval $[0.5p_i, 1.5p_i]$, i.e., proportional to the profit. The defence budget was computed based on the average defence cost α of vertices and the number of starting fires f. For each type of graph and each combination of n we generated ten random instances, the first five with budget $B = \alpha \cdot f$ and the second five with $B = \alpha \cdot (f - 1)$, i.e., a slightly tighter budget.

Analysis

We solved all instances to optimality and report the runtimes in seconds in Tables 5.1 and 5.2 for the MIP formulation for the Cost-Value Firefighter game with both variables b_i^t and d_i^t being binary or just one of them. Moreover, we assess the effect of using the trivial upper bound $T^{max} = n - f$ or our two improved bounds. With respect to the latter, on average the second bound derived in Lemma 68 is 10% smaller than the first one derived in Lemma 67. But for some graphs, especially for the small ones, the first bound is smaller than the second, so we decided to compute both for each instance and use the better of the two. The run times are averaged over the respective instances. The first column of the table specifies which type of graph we are considering, the second column the graph size and the third column the set of instances, where "Total" simply means the instances 1-10. In the next three columns, we report the run times for the MIP formulation

with both variables b_i^t and d_i^t being binary or just one of them using the trivial upper bound. In the next three columns, we do the same except for using the improved T^{max} this time.

The effect of the improved upper bound is quite striking for both types of graphs. Looking at the totals, the run times for the improved upper bound are almost one order of magnitude smaller than for the trivial one. Looking at the binary conditions, it becomes apparent that relaxing the integrality of the b_i^t variables always increases the run times (columns "d binary"). There is, however, no clear indication whether to keep the d_i^t variables binary or not (columns "b&d binary" and "b binary"). For the Gabriel graphs, relaxing the d_i^t variables leads to increased run times, whereas it is the opposite way around for the Delauney instances. It is not clear to us exactly why, but it might be due to the increased edge density of the Delauney graphs.

Moreover, the run times for the Delauney instances are almost one order of magnitude larger than the ones for the Gabriel graphs. Again, we suspect that this is due to the increased edge density. For the Delauney instances, on average 42% of vertices burn, whereas for the Gabriel graph only 25.3% of the vertices catch fire. So, seemingly, the latter graphs are easier to defend. What is also striking is that the instances 6-10 with a tighter defense budget are much harder to solve.

Table 5.1: Computational results for Cost-Value Firefighter for the Gabriel instances.

		Trivial T^{max}			Improved T^{max}		
Graph size	Instances	b & d binary	d binary	b binary	b & d binary	d binary	b binary
Small	1–5	2.3	2.5	2.3	0.5	0.7	0.5
	6–10	2.9	3.1	2.4	1.0	1.1	1.1
	Total	2.6	2.8	2.4	0.8	0.9	0.8
Medium	1–5	35.3	34.7	36.7	5.1	5.2	4.6
	6–10	51.1	58.6	53.4	11.9	13.4	14.0
	Total	43.2	46.6	45.0	8.5	9.3	9.3
Large	1–5	-	-	-	111.4	117.4	107.8
	6–10	-	-	-	205.3	403.8	230.2
	Total	-	-	-	158.4	260.6	169.0

Given the results, we will only use the improved upper bound from now on. Moreover, as the differences in run times for the small instances are marginal, we will focus in the remainder on the medium and large instances.

Next, we analyse the effectiveness of the three valid inequalities (5.11), (5.12), and (5.13). We will first consider the case that the variables for burning and defence are binary. The results are presented in Tables 5.3 and 5.4. The first three columns are the same as for the previous table. In the next column we repeat the results for the model without any valid inequalities, before presenting the results for all combinations of the three valid inequalities in the remaining columns. For the Gabriel data sets, adding valid inequality (5.11) reduces the

Table 5.2: Computational results for Cost-Value Firefighter for the Delauney instances.

		Trivial T^{max}			Improved T^{max}		
Graph size	Instances	b & d binary	d binary	b binary	b & d binary	d binary	b binary
Small	1–5	3.8	3.9	3.0	0.8	0.9	0.7
	6–10	3.8	4.0	4.0	1.2	1.3	1.1
	Total	3.8	3.9	3.5	1.0	1.1	0.9
Medium	1–5	81.4	88.4	87.4	10.2	11.0	8.2
	6–10	114.0	145.2	106.9	27.0	32.5	21.9
	Total	97.7	116.8	97.1	18.6	21.8	15.5
Large	1–5	-	-	-	437.6	512.2	327.2
	6–10	-	-	-	1980.1	1770.1	1813.8
	Total	-	-	-	1208.8	1141.2	1070.5

average run times across all instances, and adding any of the other two leads to worse results. Unfortunately, the message for the Delauney data sets is not so clear. While adding (5.11) is again beneficial compared to having no valid inequalities, this time the run times decrease if we also add (5.13) for the medium instances and (5.12) for the larger instances with tight budgets (although the improvement for the former is marginal). The latter is especially striking, as adding (5.12) for all other instances results in the worst combinations (whether alone or in combination with other valid inequalities). Looking closer, it is in fact only for the instances 6-10 with 300 vertices that (5.11)+(5.12) outperforms just (5.11) with an average run time of 3611.51 seconds over 5149.86 seconds.

Table 5.3: Computational results for Cost-Value Firefighter for the Gabriel instances if both burning and defence are binary.

Graph size	Instances	No VI's	(5.11)	(5.12)	(5.13)	(5.11) (5.12)	(5.11) (5.13)	(5.12) (5.13)	(5.11) (5.12) (5.13)
Medium	1–5	5.1	4.6	6.0	5.2	5.7	4.7	6.1	6.1
	6–10	11.9	12.1	13.9	12.1	16.6	12.7	15.6	19.5
	Total	8.5	8.3	10.0	8.7	11.3	8.7	10.9	13.0
Large	1–5	111.4	106.1	135.1	109.9	127.1	108.1	125.7	115.7
	6–10	205.3	154.9	225.3	205.9	221.8	158.9	232.1	209.3
	Total	158.4	130.5	180.2	157.9	174.4	133.5	178.9	162.5

Next we consider the case where only the b_j^t variables are binary. The results are presented in Tables 5.5 and 5.6, which has exactly the same structure as Tables 5.3 and 5.4. For the Gabriel data sets, the picture is very similar. While adding valid inequality (5.11) reduces the average run times across all instances, adding any of the other two leads to similar or worse results. For the Delauney data sets, the message is again mixed and, surprisingly, different from Tables 5.3 and 5.4. This time, adding valid inequality (5.11) increases the run times

Table 5.4: Computational results for Cost-Value Firefighter for the Delauney instances if both burning and defence are binary.

Graph size	Instances	No VI's	(5.11)	(5.12)	(5.13)	(5.11) (5.12)	(5.11) (5.13)	(5.12) (5.13)	(5.11) (5.12) (5.13)
Medium	1–5	10.2	9.0	14.6	10.2	12.3	9.3	11.9	14.0
	6–10	27.0	24.2	37.2	26.8	33.2	24.7	39.6	39.0
	Total	18.6	16.6	25.9	18.5	22.7	17.0	25.7	26.5
Large	1–5	437.6	336.2	574.7	396.2	480.5	340.1	490.5	513.9
	6–10	1980.1	1713.5	1778.5	1649.2	1424.9	1717.7	2175.3	1681.2
	Total	1208.8	1024.8	1176.6	1022.7	952.7	1029.0	1332.9	1097.5

compared to having no inequalities at all. Instead, adding (5.13) has the biggest impact for the medium data sets. For the large data sets, while none of the other two valid inequalities alone seem to be very strong, adding all three gives by far the best results for the large data sets; although only for the instances 6-10. For the remaining instances, having no valid inequalities at all gives the best results.

Table 5.5: Computational results for Cost-Value Firefighter for the Gabriel instances if only the burning variables are binary.

Graph size	Instances	No VI's	(5.11)	(5.12)	(5.13)	(5.11) (5.12)	(5.11) (5.13)	(5.12) (5.13)	(5.11) (5.12) (5.13)
Medium	1–5	4.6	4.6	5.5	4.2	5.1	4.6	5.2	5.4
	6–10	14.0	12.6	15.4	14.8	17.2	12.6	14.7	13.9
	Total	9.3	8.6	10.5	9.5	11.2	8.6	10.0	9.7
Large	1–5	107.8	97.5	125.3	114.2	145.7	97.9	125.9	102.2
	6–10	230.2	196.2	228.8	230.2	219.1	198.1	228.1	190.3
	Total	169.0	146.9	177.1	172.2	182.4	148.0	177.1	146.3

5.3 Distance-Limited Firefighter

As mentioned in the introduction, it may not always be realistic to assume that defenders can move arbitrarily far on the network from one time step to the next, so we will now relax that assumption. As this variant has not yet been introduced in the literature, we will analyse this first for the Classic Firefighter game (each vertex has the same value and cost for defending and we are given a fixed number $d \in \mathbb{Z}^+$ of defenders, i.e., $p_i = c_i = 1$).

Each defender can start protecting vertices anywhere on the graph at time $t = 1$, but afterwards they can only travel a distance of at most $l_d \in \mathbb{Z}^+$ from one time step to the next (each edge has a length of one). Because of this we have to name defenders and explicitly keep track of their movement across the graph.

Table 5.6: Computational results for Cost-Value Firefighter for the Delauney instances if only the burning variables are binary.

Graph size	Instances	No VI's	(5.11)	(5.12)	(5.13)	(5.11) (5.12)	(5.11) (5.13)	(5.12) (5.13)	(5.11) (5.12) (5.13)
Medium	1–5	8.2	8.5	11.9	8.1	10.6	8.2	11.9	9.7
	6–10	21.9	24.2	35.6	21.8	32.3	23.5	34.8	25.0
	Total	15.2	16.4	23.7	14.9	21.5	15.8	23.3	17.4
Large	1–5	327.2	366.2	474.6	337.1	396.3	359.0	496.7	459.4
	6–10	1813.8	1818.3	1677.8	1587.8	1737.0	1805.7	1675.3	1286.9
	Total	1070.5	1092.2	1076.2	962.4	1066.7	1082.4	1086.0	873.2

In the following, we label them as $x_1, x_2, \ldots, x_d$. In the classical game, as long as there are susceptible vertices left, a defender can always protect a vertex. This is no longer the case in this variant, as the following example shows where we assume that $l_d = 1$. The defender was initially positioned at Vertex 3, then moved to Vertex 7. Due to $l_d = 1$, they could not defend more vertices (all susceptible vertices were out of their reach). Without the distance limitation, they would have defended Vertex 4.

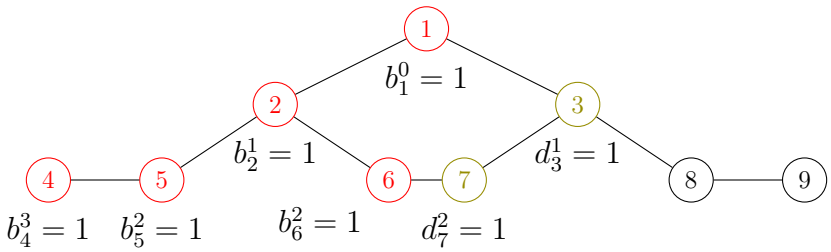


Figure 5.5: Example in which a single defender cannot defend in some time step

To facilitate this, we introduce the notion of a defender being *positioned* at vertex i in time step t. A defender positioned at i will always protect i if it is susceptible. Otherwise, as a burning vertex can never be made "un-burning" or a defended vertex cannot be defended again, the defender just "sits" on the vertex without doing anything. Obviously, we would like to avoid such situations, but as we have seen in the example above, it is not always be possible to find an optimal defence strategy where each defender protects a susceptible vertex in every time step. A defender must be positioned at exactly one vertex in each time step and can only defend the vertex they are positioned at. To avoid ambiguity, we assume that the movement of defenders happens between time steps. A defender can move less than l_d between time steps, and can even stay put on a vertex. Moreover, multiple defenders can be positioned at the same vertex.

Summing up, the Distance-Limited Firefighter (DFF) game adheres to the following rules:

- All vertices in $F \subset V$ are burning at time 0.

- In each time step $t > 0$, we first defend vertices and afterwards the fire spreads. Concerning the latter, the fire will spread from every burning vertex to all adjacent susceptible vertices, i.e., vertices that are neither burning nor defended.

  Once burning, a vertex will remain burning until the end of the game.

- At time step $t = 1$, each defender can be positioned anywhere on the graph and defend the corresponding vertex if it is susceptible. In every subsequent time step $t > 1$, a defender can be positioned at exactly one vertex i which must be a distance of at most l_d away from the vertex j the defender was positioned at in time $t - 1$.

  Once defended, a vertex will remain defended until the end of the game.

- The game ends if no more vertices catch fire.

The goal is to find a defence strategy that minimizes the number of burning vertices or, equivalently, maximizes the number of non-burning vertices at the end of the game. The end of the game and the final time step T^{opt} is defined analogously to Section 5.2. Again, there may be more than one optimal defence strategy and different optimal defence strategies may result in different values for T^{opt}. If this is the case, we define T^{opt} as the smallest among all values, i.e., we chose the defence strategy that ends the game the quickest.

In the following, we present a mixed-integer linear programming formulation for the problem. Let $T^{max} \in \mathbb{Z}^+$ be an upper bound for T^{opt}. A trivial bound is again given by $T^{max} = n - f$. Unfortunately, none of the upper bounds derived in Section 5.2 extend to Distance-Limited Firefighter. In addition to the variables d_i^t and b_i^t previously introduced, we define:

$$y_{il}^t = \begin{cases} 1, & \text{if the } l\text{-th defender is positioned at vertex } i \text{ at time } t. \\ 0, & \text{otherwise.} \end{cases}$$

for $i \in V$, $1 \leq l \leq d$, and $0 \leq t \leq T^{max}$. With these definitions, we can formulate the problem as:

$$\begin{aligned}
\min \quad & \sum_{i \in V} b_i^{T^{max}} & & \\
\text{s.t.} \quad & b_i^{t-1} \leq b_i^t, & i \in V, 0 < t \leq T^{max}, & (5.14)\\
& d_i^{t-1} \leq d_i^t, & i \in V, 0 < t \leq T^{max}, & (5.15)\\
& d_i^t - d_i^{t-1} \leq \sum_{1 \leq l \leq d} y_{il}^t, & i \in V, 0 < t \leq T^{max}, & (5.16)\\
& \sum_{i \in V} y_{il}^t = 1, & 1 \leq l \leq d, 0 < t \leq T^{max}, & (5.17)\\
y_{il}^t + & \sum_{j \in V : dist(i,j) > l_d} y_{jl}^{t-1} \leq 1, & i \in V, 1 \leq l \leq d, 1 < t \leq T^{max}, & (5.18)\\
& b_j^{t-1} \leq b_i^t + d_i^t, & j \in V, i \in N(j), 0 < t \leq T^{max}, & (5.19)\\
& d_i^0 = 0, & i \in V, & (5.20)\\
& b_i^0 = 1, & i \in F, & (5.21)\\
& b_i^t,\, d_i^t,\, y_{il}^t \in \{0,1\}, & i \in V, 1 \leq l \leq d, 0 \leq t \leq T^{max}, & (5.22)
\end{aligned}$$

The monotonic increase of burning resp. defence is ensured by (5.14) resp. (5.15). We ensure that each vertex i can only be newly defended in time step t if there is at least one defender positioned at i in time t using (5.16). This is an inequality because it is possible for a defender to be positioned at a vertex without defending it. Conditions (5.17) and (5.18) ensure that each individual defender is positioned at exactly one vertex and cannot move a distance of more than l_d between time steps, respectively. A vertex neighbouring a burning vertex must either burn or be defended in the following time step thanks to condition (5.19). No vertices are initially defended by (5.20) and the vertices in F are initially burning by (5.21). Burning, defence, and positioning are made binary by (5.22).

Values for T^{max} on Trees

In general, the arguments we used in Sections 4.1 for Edge-Defence Firefighter and in 5.2 for Cost-Value Firefighter about upper bounds for T^{opt} do not hold for Distance-Limited Firefighter because we cannot guarantee that it will be possible for a defender to defend a vertex in a given time step. However, we can provide an upper bound for T^{max} if the game takes place on a tree.

Lemma 70. *For Distance-Limited Firefighter, on a tree G:*

$$T^{opt}(G, 1, d) \leq \left\lceil \frac{n-1}{2} \right\rceil.$$

Proof. Assume that there exists a tree G, with $T^{opt} > 1$ and $T^{opt}(G, d, 1) = \lceil \frac{n-1}{2} \rceil$.

As in Lemma 64, following the logic in the proof of Lemma 60, for the upper limit on T^{opt} to be met we must have exactly one new vertex igniting per time step and exactly d vertices being defended with no vertices being saved without

being themselves defended. This means that only one new vertex f_1 can ignite at time step 1. We choose one of its neighbouring initial fires arbitrarily and label it f'. Any other fires either have no susceptible neighbours at time 0 other than f_1; or all such neighbours are defended in time time step 1.

Let M be the maximum distance on G between f' and any other vertex. Let the furthest vertex from f' be w. If $M < \lceil \frac{n-1}{2} \rceil$ then the game cannot possibly last $\lceil \frac{n-1}{2} \rceil$ time steps, a contradiction. Consider defence strategy D_1 with a single defender, where we defend $M - \lceil \frac{n-1}{2} \rceil + 1$ away from the fire on the unique path from the initially burning vertex to w, called P. For the first $\lceil \frac{n-1}{2} \rceil$ time steps, we can defend the next vertex on P and it will not already be burning because the unique path between it and the initially burning vertex has already been defended. At time step $\lceil \frac{n-1}{2} \rceil$ we then have $\lceil \frac{n-f}{2} \rceil$ vertices already defended, leaving only $n - 1 - \lceil \frac{n-1}{2} \rceil$ which can potentially ignite in the game, so the game using D_1 cannot last longer than $\lceil \frac{n-1}{2} \rceil$ time steps. If D_1 were optimal then the claim is proven. Assume now that D_1 is not optimal defence. Then the optimal defence must burn strictly fewer vertices than D_1, which itself burned at most $\lceil \frac{n-1}{2} \rceil$ vertices hence any optimal defence also cannot last strictly more than $\lceil \frac{n-1}{2} \rceil$ time steps, a contradiction. Hence overall the claim is proven. □

This bound is tight (at least for $f = d = l_d = 1$), see Figure 5.6.

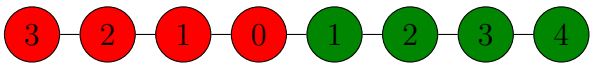


Figure 5.6: An example of a game of Distance-Limited Firefighter on a tree for which $T^{opt} = \lceil \frac{n-f}{2} \rceil$ for $l_d = 1$. Red resp. green vertices burn resp. are defended at the indicated time steps.

Valid Inequalities

The first valid inequality is identical to (5.11):

$$b_i^t + d_i^t \leq 1, \qquad i \in V, 0 \leq t \leq T^{max} \tag{5.23}$$

Alternatively to Constraint (5.18), or in addition, we can formulate the distance constraints from the point of view of a single positioning variable at time $t - 1$:

$$y_{il}^{t-1} + \sum_{j \in V : dist(i,j) > l_d} y_{jl}^t \leq 1, \qquad i \in V, 1 \leq l \leq d, 1 < t \leq T^{max}. \tag{5.24}$$

This constraint can be generalised to:

$$y_{il}^{t-r} + \sum_{j \in V : dist(i,j) > r \cdot l_d} y_{jl}^t \leq 1, \qquad i \in V, 2 \leq r < t \leq T^{max}. \tag{5.25}$$

As defenders can be positioned on any vertex and staying put will never decrease the number of burning vertices, we can force a defender to move each

time step without loss of generality:

$$y_{il}^{t} + y_{il}^{t-1} \leq 1, \qquad i \in V, 0 < t \leq T^{max}, 1 \leq l \leq d. \tag{5.26}$$

Computational Experiments

We coded the mixed-integer programming formulation in C++ using CPLEX 20.1 and ran the experiments on a Dell PowerEdge R740 server running Scientific Linux 7 using up to 8 threads. We will use the same data sets as in Section 5.2, but this time we only consider data sets with $n = 40, 50$, and 60 vertices, as the distance constrained Firefighter problem is much harder to solve. As a reminder, for $n = 40, 50$, we have 2 starting fires, i.e., $f = 2$, and for $n = 60$, there are three starting fires, i.e., $f = 3$. Concerning the number of defenders, instances 1-5 have $d = f$ defenders and instances 6-10 have $d = f - 1$ defenders. Finally, for the distance value we chose $l_d = 2, 3$, and 5.

Analysis

We solved all instances to optimality and start with reporting the results for the Gabriel data sets in Table 5.7 for the MIP formulation with both variables b_i^t and d_i^t being binary, just one of them, or none. All results are averaged over the respective instances. The first column of the table specifies the distance limit, the second column the number of vertices and the third column the set of instances, where "Total" simply means the instances 1-10. In the next two columns, we report the average number of time steps a game takes and the average number of burning vertices. In the following four columns, we report the run times in seconds for the MIP formulation with both variables b_i^t and d_i^t being binary or just one of them, or neither of them.

Starting with the first two columns, with fewer defenders (instances 6-10), not unsurprisingly the games last longer and we have more burning. Next, looking at the run times in the $b\&d$ column, we can first observe that games with fewer defenders are easier to solve. This is most likely because the solution space is smaller. Not unsurprisingly, the run times increase with the size of the graph, and decrease with an increasing distance limit l_d. The latter is most likely because with a larger l_d, the distance limit is becoming less restrictive and the DFF game starts to morph into the Classic Firefighter game without distance limits, which is much easier to solve for comparable problem sizes, see Section 5.2.

With respect to the integrality conditions on the b_i^t and d_i^t variables, there is no clear picture. For easy to solve instances with 40 and 50 vertices, relaxing the integrality on both variables seems to be beneficial. For the difficult instances with 60 vertices, keeping the integrality on both is the best option for $l_d = 2$ and $l_d = 3$, and keeping only the b_i^t variables binary the best for $l_d = 5$, although the difference in run times is much smaller than compared to the other two distance values. In summary, it is better to keep both variables binary.

Table 5.8 shows the same analysis for the Delauney data sets. We can make the same observations as for the Gabriel data sets. The only new and quite striking observation is that in contrast to the Cost-Value FF, the Delauney data sets are

					Binary conditions on			
	Vertices n	Instances	T^{opt}	Burn	b & d	d	b	none
$l_d = 2$	40	1-5	3.0	14.2	14.3	15.3	10.3	9.6
		6-10	4.4	25.6	6.7	6.7	6.8	6.6
		Total	3.7	19.9	10.5	11.0	8.5	8.1
	50	1-5	3.2	15.0	70.0	114.7	36.6	75.1
		6-10	4.8	30.6	32.3	25.8	20.6	24.8
		Total	4.0	22.8	51.1	70.2	28.6	49.9
	60	1-5	3.4	19.6	6710.7	10618.0	9797.1	10500.6
		6-10	4.0	28.8	1310.0	1055.6	817.6	958.1
		Total	3.7	24.2	4010.3	5836.8	5307.3	5729.4
$l_d = 3$	40	1-5	3.0	12.2	12.1	7.4	7.3	6.2
		6-10	4.2	23.2	5.3	5.1	4.6	4.6
		Total	3.6	17.7	8.7	6.3	6.0	5.4
	50	1-5	2.6	13.0	36.7	15.3	16.7	36.5
		6-10	4.2	27.6	15.4	12.3	13.0	13.7
		Total	3.4	20.3	26.0	13.8	14.8	25.1
	60	1-5	3.0	16.2	1238.0	2524.6	2010.3	2141.3
		6-10	3.8	25.4	226.6	445.4	263.5	695.7
		Total	3.4	20.8	732.3	1485.0	1136.9	1418.5
$l_d = 5$	40	1-5	2.8	11.8	4.2	3.6	3.3	3.5
		6-10	4.0	22.0	4.2	3.7	3.6	3.7
		Total	3.4	16.9	4.2	3.6	3.4	3.6
	50	1-5	2.6	12.4	9.0	8.2	7.3	9.3
		6-10	4.6	26.0	10.0	11.0	9.0	9.2
		Total	3.6	19.2	9.5	9.6	8.2	9.3
	60	1-5	2.4	14.6	26.6	70.0	20.9	36.3
		6-10	3.4	22.8	26.2	27.7	20.8	27.2
		Total	2.9	18.7	26.4	48.9	20.9	31.7

Table 5.7: Computational results for the Gabriel instances.

easier to solve than the Gabriel data sets for the difficult parameter combinations ($n = 60$ and $l_d = 2, 3$) and roughly equally difficult for the other combinations. A possible explanation is that with more edges, defenders can move further on the graph from one time step to the next, which has a similar effect to increasing the distance limit.

Computational Complexity

We focus on the decision question:

DISTANCE-LIMITED FIREFIGHTER (DLF)

Instance: A connected graph $G = (V, E)$, a set $F \subset V$ and three natural numbers $d, k, l_d \in \mathbb{N}$.

Question: If the fire starts at F in G, is there a strategy for d defenders moving at most distance l_d per time step to save at least k vertices?

	Vertices n	Instances	T^{opt}	Burn	Binary b & d	Binary d	Binary b	Binary none
$l_d = 2$	40	1-5	2.6	20.0	9.8	14.5	7.3	7.5
		6-10	4.0	30.0	5.9	6.1	5.2	5.8
		Total	3.3	25.0	7.8	10.3	6.2	6.6
	50	1-5	3.4	22.6	53.9	54.1	38.1	78.8
		6-10	3.8	33.6	17.4	15.0	15.7	12.9
		Total	3.6	28.1	35.7	34.6	26.9	45.8
	60	1-5	3.0	26.0	4843.4	8523.0	4382.1	7002.7
		6-10	3.4	32.4	489.3	251.1	1059.7	184.2
		Total	3.2	29.2	2666.3	4387.1	2720.9	3593.5
$l_d = 3$	40	1-5	2.6	19.4	6.9	5.0	5.4	6.3
		6-10	4.0	29.4	5.0	4.6	4.0	4.9
		Total	3.3	24.4	5.9	4.8	4.7	5.6
	50	1-5	3.2	22.2	47.5	65.7	22.9	26.7
		6-10	4.0	33.2	12.7	12.4	10.2	11.6
		Total	3.6	27.7	30.1	39.0	16.6	19.2
	60	1-5	2.6	25.0	918.6	2006.8	1696.0	2749.8
		6-10	3.2	31.6	41.3	65.1	32.4	45.2
		Total	2.9	28.3	479.9	1036.0	864.2	1397.5
$l_d = 5$	40	1-5	2.4	19.4	4.6	3.6	3.4	4.0
		6-10	4.2	29.4	4.7	4.2	3.7	4.0
		Total	3.3	24.4	4.6	3.9	3.6	4.0
	50	1-5	3.0	21.6	14.2	11.7	11.5	16.2
		6-10	4.0	33.0	11.2	10.4	9.0	11.6
		Total	3.5	27.3	12.7	11.0	10.3	13.9
	60	1-5	2.4	24.2	26.0	21.3	19.8	33.1
		6-10	3.0	31.0	23.9	19.1	18.4	23.8
		Total	2.7	27.6	25.0	20.2	19.1	28.5

Table 5.8: Computational results for the Delauney instances.

There is a proof in Burgess et al. [4] that DLF is NP-complete for general l_d or if l_d is at least as large as the diameter of the graph. This follows from the proof that Classic Firefighter with more than one defender is NP-hard in Bazgan, Chopin and Ries [2], which was reproduced above as Theorem 30. A proof that a related problem - in which the defenders cannot pass through burning vertices- in NP-hard is also provided in Burgess et al. [4].

Lemma 71. *DLF with a single fire and single defender with $l_d = 1$ on a tree rooted at the single fire on vertex v_f, is polynomial.*

Proof. Defending any vertex will save all of its descendants, so it is always optimal to begin by defending one of the neighbours of the fire (i.e. a member of $N(v_f)$) at time 1. After time step 1, this defender will not be able to overtake the fire to defend any other vertex in $N(v_f)$ or any of their descendants. Hence, finding optimal defence becomes a question of identifying the vertex in $N(v_f)$ with the most descendants and defending it in time 1, saving all the descendants and

leaving the rest of the graph to burn. This can be done in $\mathcal{O}(n+m)$ time using depth-first search, since we are considering trees, $m = n-1$, giving a complexity of $\mathcal{O}(n)$. □

An important result on Classic Firefighter transfers instantly to EDF. For Classic Firefighter, the decision question is stated as:

FIREFIGHTER

Instance: A connected graph $G = (V, E)$, a set $F \subset V$ and two natural numbers $d, k \in \mathbb{N}$.

Question: If the fire starts at F in G, is there a strategy for d defenders to save at least k vertices?

We focus on the following result on graphs with low maximum degree:

Theorem 72 (Theorem 4 of Finbow et al. [20])**.** *FIREFIGHTER with a single fire and a single defender on a graph with maximum degree three is polynomial if the fire stars on a vertex of degree at most two.*

For $l_d \geq 3$, this instantly applies to DLF:

Lemma 73. *DLF with a single fire and single defender with $l_d \geq 3$ on a graph with maximum degree three is polynomial if the fire stars on a vertex of degree at most two.*

Proof. This follows instantly from the proof of Theorem 72 in Finbow et al. [20], which offers a polynomial algorithm for optimal defence in such instances, and in which the defender never moves more than distance three per time step. We reproduce that strategy here, adding notes on the distances that the single defender has to move each time step.

As before, let $d(v)$ be the degree of vertex v and $dist(u, v)$ be the distance between vertices u and v. We begin by defining three important sets: $V_1 := \{v \in G : d(v) = 1\}$; $V_2 := \{v \in G : d(v) = 2\}$ and $V_c := \{v \in G : v \text{ is in a cycle}\}$. We also define $C(v)$ to be the length of the shortest cycle containing v for some $v \in G$. Let v_f be the single initially burning vertex. For our last definition:

$$f(v) := \begin{cases} dist(v, v_f) & \text{if } v \in V_1 \cup V_2, \\ dist(v, v_f) + C(v) - 1 & \text{if } v \in V_c \setminus V_2. \end{cases}$$

To begin the defence strategy, we first find $u \in G$ s.t. $f(u) = \min\{f(v) : v \in V_1 \cup V_2 \cup V_c\}$. Let P_u be the shortest path from v_f to u. If either of u or P_u are not unique, then chose arbitrarily. There are then two possible cases. Consider the case $u \in V_1 \cup V_2$. We use Strategy 1. At each turn, there will be at most one susceptible threshold vertex which is not on P_u. We defend this, moving the defender distance three per time step. If $u \in V_2$, at time step $f(u)$, we defend the susceptible neighbour of u. See Figure 5.7 for an example of this strategy.

For the second case, we have $u \in V_2 \setminus V_c$. We use Strategy 2. Let C denote the shortest cycle containing u. At each time step from 1 to $dist(v_f, u)$, defend the susceptible threshold vertex which is not on P_u, moving the defender distance three each time. At time step $dist(v_f, u)$, defend either of the susceptible

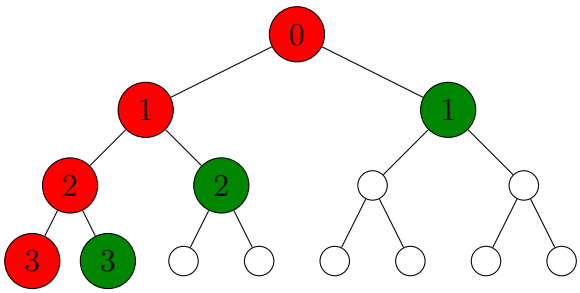


Figure 5.7: Strategy 1 of the polynomial time defence strategy given in Finbow et al. [20]. Note that defenders are always distance three away from the last defender move.

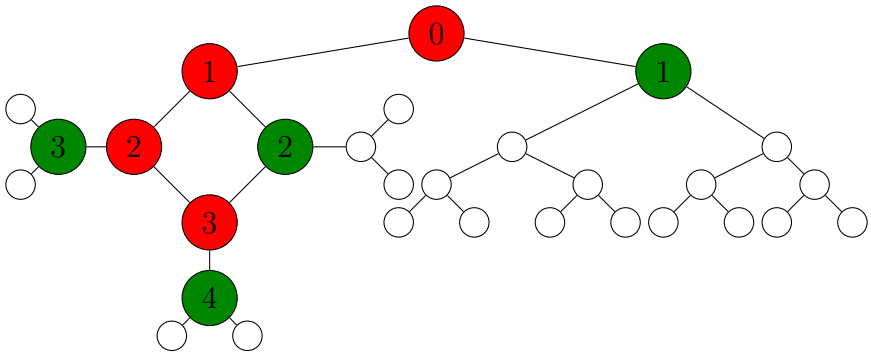


Figure 5.8: Strategy 2 of the polynomial time defence strategy given in Finbow et al. [20]. Note that defenders are always distance three away from the last defender move.

threshold vertices in C. After that you follow the fire around the cycle, defending threshold vertices that are not on C, moving the defender distance three each time, until the fire reaches the vertex defended at time step $dist(v_f, u)$ and the game finishes. For an example of Strategy 2, see Figure 5.8. □

5.4 Distance-Limited Firefighter on Paths

In this section, we will focus on DLF on paths. We will once again be using the idea of fire components, first introduced above in Section 2.6. The distance constraints on how far a defender can move between time steps means that we need a slightly altered definition and algorithm for finding the components. Fire components as found using Algorithm 2 on page 28 effectively identified all neighbouring fires as one burning vertex. However, for Distance-Limited Firefighter, the number of neighbouring vertices that are burning is important - especially if the defender has to travel through them to reach and defend a susceptible vertex. We shall therefore be using Algorithm 13 (which is an altered version of Algorithm 2) throughout our discussion of the Distance-Limited Firefighter game on paths.

Input: A graph $G = (V, E)$; a subset of its vertices $F \subset V$
Output: The list of fire components of (G, F)
create an empty list *components*;
for *vertex* $v \in F$ **do**
  delete v;
end
call the remaining graph $G' = C_1 \cup C_2 \cup \ldots$, where each C_i is a connected component
for *connected component* $C_i \in G'$ **do**
  for $v \in F$ **do**
    for $w \in C_i$ *st.* $(v, w) \in E$ **do**
      add (v, w) to C_i
    end
  end
end
add C_i to *components*;
for *vertex* $v \in F$ **do**
  for *vertex* $u \in N(v)$ **do**
    if $u \in F$ **then**
      append (u, v) to *components*
    end
  end
end
return *components*

Algorithm 13: Algorithm for finding the fire components for Distance-Limited Firefighter.

$l_d = 1$ on Paths

In this section we will focus on Distance-Limited Firefighter on a single path with $l_d = 1$ and any number of defenders and initial fires.

Theorem 74. *Algorithm 14 is a polynomial-time algorithm for finding the optimal defence on a single path with $l_d = 1$ and any number of defenders and any number of fires.*

Proof. Since $l_d = 1$, each defender can only defend in one component as it could never 'overtake' the fire to defend in a component it did not start in. Finding an optimal defence strategy therefore becomes a question of greedily placing the defenders in the components where they can save the most vertices. Since the defenders will never reach another component, it follows that optimal defence will have them placed initially in a vertex directly neighbouring a fire and then moving away from it to maximise how many vertices they can save in the component they start in. In a component C_i with a single fire, one defender would then save l_i vertices and placing any more defenders would not save more vertices. In a component C_i with two fires, a single defender would save $\lceil \frac{l_i}{2} \rceil$ vertices; adding a second defender would save all l_i vertices and adding further defenders would

Input: A list of fire components $C = C_1, C_2, \ldots$ for an instance of FIREFIGHTER on a path P; the set of burning vertices F
Output: A vertex defence strategy
create an empty mapping *component weights*;
for *component* C_i *in* C **do**
  map C_i to *weight* $= \frac{|C_i|}{|\{v \in C_i | v \in F\}}$ in *component weights*;
end
create a list *ordered components* of components in non-increasing order by their image in *component weights*;
start playing FIREFIGHTER;
for *component in* ordered components **do**
  create a list *thresh* of threshold vertices;
  append *thresh* to *defence priorities*;
end
place defenders on the first d vertices in *defence priorities*;
for *defender move in FIREFIGHTER* **do**
  for *defender in* D **do**
    move defender to the next vertex further away from the fire the defender started next to;
  end
end

Algorithm 14: Algorithm for defence against multiple fires on a path with multiple defenders and $l_d = 1$.

save no further vertices. Algorithm 14 then assigns weights to the components exactly in the order in which adding (another) defender to them leads to the most vertices saved. □

Higher values of l_d on Paths with a Single Defender

In this section we consider $l_d \geq 2$ on paths with any number of fires and with particular emphasis on $l_d = 2$. We start by considering the case with a single defender but with any number of initial fires. The case $l_d = 1$ was special since each defender could only ever defend vertices in one component. For large enough values of l_d (say $l_d \geq n$ for a trivial upper bound), the Distance-Limited game is exactly equivalent to the classic version of Firefighter with no distance limits. Since we are focusing on paths, all games are trivially *fire path equivalent* from the very beginning of the game (see Section 2.6), so Algorithm 3 on page 28 is a polynomial time algorithm which gives optimal defence for sufficiently large values of l_d.

However, 'medium' values of l_d which are large enough for individual defenders to be able to defend vertices in more than one component over the course of the game; but not yet so large that a defender can jump between components as Algorithm 3 would require, display some interesting behaviour - which shall be the focus of this section. We shall particularly focus on the case $l_d = 2$ since the computational results in Tables 5.7 and 5.8 in Section 5.3 demonstrated that this

is the case that takes longest to find exact solutions for using linear optimisation.

Throughout this section we shall arbitrarily fix 'left' and 'right' on the path so that one endpoint is the leftmost vertex and the other is the rightmost vertex. We then label the vertices $1, 2 \dots n$ so that the leftmost vertex has label 1 and the rightmost has label n. We will only consider defence strategies where the defender only moves in one direction, unless otherwise stated (in Lemma 76 we shall prove that this holds for optimal defence for $l_d = 2$). The two most important elements in defining a defence strategy are therefore the vertex that is initially defended and the direction the defender travels in, so we focus on initial (v, dir) pairs of starting vertices and directions they travel in, where $v \in V$ and $dir \in \{left, right\}$. We call a component or vertex *in front of* a (v, dir) pair if the direction is right and it is to the right of the vertex or if the direction is left and it is to the left of the vertex, with *behind* taking the opposite meaning. We call a vertex w *directly in front of* resp. *directly behind* a (v, dir) pair if w is in front of resp. behind (v, dir) and is adjacent to v. For a strategy starting with (v, dir), label the component we initially defend in C_1 with length l_1 after the fires are removed, the component in front of that C_2 with length l_2 after the fires are removed, and so on up to the final component in front of (v, dir) which we shall call C_r. Note that two fires next to each other count as one component C_i with $l_i = 0$ and three fires in a row count as two components C_i, C_{i+1} with $l_i = 0 = l_{i+1}$ etc. Note that this slightly differs from the definition of components in Section 2.6.

We will start by considering how many vertices a single defender can save in component C_1 (where the defender starts) with two fires and length (excluding initially burning vertices) $l_1 = x$ if it starts on a threshold vertex and then travels towards the opposite end of the component (rather than immediately leaving it). A defender can save up to l_d vertices per turn while one burns, so how the game progresses will follow a certain pattern depending on the value modulo $l_d + 1$ of l_1. An important variable is whether or not the defender moves forwards by less than l_d, in order to save more vertices in that component. We call this *shortening its step.*

Fix l_d and consider the game in a component of length $(l_d + 1)k$ for some k a non negative integer. In the first time step, one vertex (the threshold vertex the defender starts on) is defended and one other burns (the opposite threshold vertex). For the next $k - 1$ times steps, the defender will move forward, saving an extra l_d vertices per time step and one more vertex will burn at the opposite end of the component per time step. Subtracting the saved and burned vertices from the length of the component, at time step $k + 1$ there are then $(l_d + 1)k - 1 - 1 - (k - 1)l_d - (k - 1) = l_d - 1$ vertices between the defender and the nearest burning vertex. If the defender moves forward by only $l_d - 1$ vertices then the fire can progress no further in that component. The total number of vertices saved in C_1 is given by $1 + l_d(k - 1) + (l_d - 1) = k \cdot l_d$.

If $l_1 = k \cdot (l_d + 1) + 1$ for some k a non negative integer, then the game progresses as before until time step $k+1$ with $1 + l_d(k-1)$ vertices saved and k vertices burned at the opposite end, leaving a gap of $k \cdot (l_d + 1) + 1 - 1 - 1 - (k - 1)l_d - (k - 1) = l_d$ vertices between the defender and the nearest burning vertex. The defender moves forward l_d vertices and the fire can progress no further in that component. The total number of vertices saved in C_1 is given by $1 + l_d(k - 1) + (l_d) = k \cdot l_d + 1$.

For $l_d = k \cdot (l_d + 1) + z$ for some non negative integers k, z with $2 \leq z \leq l_d$, time step $k + 1$ has an important difference. There are now $k \cdot (l_d + 1) + z - 1 - 1 - (k - 1)l_d - (k - 1) = l_d + z - 1$ vertices between the defender and the nearest burning vertex. At step $k + 1$ the defender moves forward l_d vertices and one more vertex is burned leaving a gap of $z - 2$ between the defender and the nearest burning vertex. If $z = 2$, the fire can spread no further in C_1. For $2 < z \leq l_d$, if the defender moves forward by $z - 2$ steps in time $k + 2$, the fire can spread no further. The total number of vertices saved in C_1 is given by $1 + l_d(k - 1) + l_d + z - 2 = k \cdot l_d + z - 1$.

Let us assume now that the defender only moves forward by exactly l_d vertices per time step. If $l_1 = k \cdot (l_d + 1) + z$ for some non negatives integer k and $2 \leq z \leq l_d + 1$, in time step 1, the defender saves one vertex and one vertex ignites. In the next k moves, the defender moves forward by l_d each step, saving a further $k \cdot l_d$ vertices and k vertices burn. In time step $k + 2$, this leaves $k \cdot (l_d + 1) + z - 1 - 1 - k \cdot l_d - k = z - 2 < l_d$ susceptible vertices between the last defended vertex and the nearest burning vertex. Moving the defended vertex forward by l_d therefore moves it further than the furthest susceptible vertex, onto one that is already burning or out of C_1 entirely. This means no more vertices will be saved in C_1 after time step $k + 1$ and the total saved is $1 + k \cdot l_d$.

Finally, consider the case where we only move the defender forward by exactly l_d and $l_d = k \cdot (l_d + 1) + 1$ for some non negative integer k. If $k = 0$, then we defend the one susceptible vertex. If $k > 1$, then in time step 1, one vertex is defended and one burns, for the next $k - 1$ time steps l_d vertices are defended and one burns and in time step $k + 1$ there are $k \cdot (l_d + 1) + 1 - 1 - 1 - (k - 1)l_d - (k - 1) = l_d$ vertices between the previously defended vertex and the nearest burning vertex, so the defender moves forward by l_d and no more vertices burn in C_1, saving a total of $1 + (k - 1)l_d + l_d = 1 + k \cdot l_d$ vertices.

Since whether the defender shortens its step to defend more vertices is so important, we shall include it as a variable in the function. Summarising the above calculations, if C_1 with length x and whether the defender shortens its step is given by the Boolean variable s, the number of vertices saved is given by:

$$f(x, l_d, s) := \begin{cases} \left\lceil \frac{x \cdot l_d}{l_d + 1} \right\rceil & \text{if } s \text{ is true and } x \equiv 0, 1 \pmod{l_d + 1} \\ \left\lceil \frac{x \cdot l_d}{l_d + 1} \right\rceil - 1 & \text{if } s \text{ is true and } x \not\equiv 0, 1 \pmod{l_d + 1} \\ 1 + \left(\left\lceil \frac{x}{l_d + 1} \right\rceil - 1 \right) \cdot l_d & \text{if } s \text{ is false} \end{cases} \tag{5.27}$$

Note from the above discussion that shortening the defender's step by less than we proposed in time step $k + 1$ if $l_1 = k \cdot (l_d + 1)$ or if $l_1 = k \cdot (l_d + 1) + 1$ and in time step $k + 2$ else, will not result in the defender saving any more vertices in C_1, but will result in it not being able to reach further components until a later time step than had it not shortened and hence, save fewer vertices over the course of the game. If we move the defender forwards by less than l_d in earlier

time steps, then it will result in both saving fewer vertices in C_1 and reaching further components later. If we move the defender forwards by more than the recommended amount in the final time step it can defend a vertex in C_1 in, then it will lead to less vertices being saved in C_1 and we shall later see that this difference would not be outweighed by reaching further components sooner. Hence, the above cases are the only ones we need to consider for optimal defence.

This function has very regular behaviour. Importantly for all non negative values of x and l_d, we have:

$$f(x, l_d, true) \geq f(x, l_d, false).$$

We use the integer variable σ to keep track of by how much the defender's step was shortened before defending a given component - i.e. when the defender moved forward by strictly less than l_d vertices between moves. Although this section only considers games with a single defender, we shall reuse this variable for games with multiple defenders, so we define is for a defender d_i that started on vertex v_i and is located on vertex v_j at the end of time step $t-1$, as:

$$\sigma(d_i, t) = (t-2) \cdot l_d - d(v_i, v_j),$$

the difference between how far the defender would have travelled, had it always moved forward by l_d and how far forward it is has actually moved. Where the defender in question and the time step are obvious, we simply write σ.

For defending a component C_i that the defender did not start in, the next thing it is important to know is how many time steps pass before we can defend a susceptible vertex in C_i after defending the previous components $C_1, C_2 \dots C_{i-1}$ and 'catching up' with how far the fire has spread along C_i. This gives an upper limit for how far the fire can spread from each initially burning vertex in C_i (this upper limit is only not achieved if the component in question was too short).

Let $d(v, C_i)$ denote the distance between vertex v and the nearest vertex in C_i which is not burning in time step 0. The first thing to note is that shortening the defender's steps by x in total over previous moves effectively increases this distance by the same amount as if it had started distance x further away and not shortened its step. Note that $d(v, C_i)$ will always be at least two if the defender did not start in C_i. Write $\sigma + d(v, C_i) = k \cdot (l_d - 1) + z$ for some non negative integers k, z with $0 \leq z \leq l_d - 2$. Then in time step one the defender does not move beyond its initial position (as it was just placed there) and the fire spread by one, so the gap between the defender and the nearest susceptible vertex in C_i becomes $k \cdot (l_d - 1) + z + 1$. If $k \geq 1$, for the next $k-1$ time steps, the defender will move forwards by l_d and the fire will burn another vertex in C_i on the side closest to the defender, so the gap decreases by $l_d - 1$. At the end of time step k, this leaves a gap of $k \cdot (l_d - 1) + z + 1 - (k-1)(l_d - 1) = l_d + z$. If $z = 0$, then we move the defender forward by l_d onto the nearest susceptible vertex in C_i. If $z \neq 0$, then the defender moves forwards by l_d, the fire progresses by one and the gap at the end of time step k is $l_d + z - (l_d - 1) = z + 1 < l_d$. If $l_d = 2$ or if $l_d > 2$ and we assume the defender shortens its step to defend the nearest susceptible vertex in C_i, the defender moves forward by $z + 1$ and defends in C_i at time step

$k+1$.

Summarising the above discussion, if the defender started at vertex $v \in C_1$; travelled in direction dir; has shortened its step by a total of σ, then the following equation states how many time steps will have elapsed before the defender can start defending in C_i:

$$g(v, dir, i, \sigma, l_d) := \begin{cases} 0 & \text{if } v \in C_i \\ \left\lceil \frac{\sigma + d(v, C_i)}{l_d - 1} \right\rceil & \text{else.} \end{cases} \tag{5.28}$$

For a component C_i with two fires, if $l_d = 2$, not being able to defend in that component until time step t leads to as many vertices being saved as if we were defending a component that was $2t$ shorter than C_i as the fire has already spread t vertices from the right and t vertices from the left before we can start defending that component. Hence the number of vertices saved is given by $\max(0, f(l_i - 2g(v, dir, i, \sigma, l_d), l_d, s))$ for $s \in \{true, false\}$. For a component C_i with only one fire, the defender will reach it at time step $g(v, dir, i, \sigma, l_d)$ and then defend, saving the rest of the component. Hence the number of vertices saved is given by $\max(0, l_i - g(v, dir, i, \sigma, l_d))$.

Note that for $l_d > 2$, it is possible that when a defender overtakes a fire to defend in a component it did not start in, always moving forward by l_d could lead to overshooting the fire. This leads to another set of conditions under which the defender could or should shorten its step and goes beyond the scope of this thesis.

Overall, for $l_d = 2$, the number of vertices saved in component C_i by a defender that starts at vertex v for $v \in C_1$, if $i = 1$ and v is a threshold vertex or if $i \neq 1$ and v is any vertex, and travels in direction dir; and shortens its step σ number of times before reaching C_i is given by:

$$w(v, dir, i, \sigma) := \begin{cases} \max\{0, l_i - g(v, dir, i, \sigma)\} & \text{if } C_i \text{ has a single fire,} \\ 1 & \text{if } v \in C_i,\ v \text{ is directly} \\ & \text{behind a fire and } C_i \\ & \text{has two fires,} \\ \max\{0, f(l_i - 2g(v, dir, i, \sigma))\} & \text{else.} \end{cases} \tag{5.29}$$

We now consider which vertex the defender should start on.

For a single defender which shortens its step in the component it starts in (i.e., fixing s to be true), we have: $f(k \cdot (l_d+1) + m, l_d, true) + 1 = f(k \cdot (l_d+1) + m + 1, l_d, true)$ for all non-negative k, l_d and $m \neq 1$ a non-negative integer; and $f(k \cdot (l_d+1) + 1, l_d, true) = f(k \cdot (l_d+1) + 2, l_d, true)$ for all non-negative integers k, l_d. In particular, this means that starting the defence in the susceptible vertex next to the threshold (thereby effectively reducing the length of the component by one) leads to exactly as much defence as starting from the threshold vertex if and only if $l_1 \equiv 2 \mod (l_d + 1)$ and s is true.

For a single defender, fixing $s = false$, and considering what happens as the length of the component increases, we have:

$$\begin{aligned} f(k\cdot(l_d+1)+1,l_d,false) &= f(k\cdot(l_d+1)+2,l_d,false)\\ &= f(k\cdot(l_d+1)+3,l_d,false)\\ &= \dots\\ &= f(k\cdot(l_d+1)+l_d,l_d,false)\\ &= f((k+1)\cdot(l_d+1),l_d,false). \end{aligned}$$

We also have $f(k\cdot(l_d+1),l_d,false)+l_d = f(k\cdot(l_d+1)+1,l_d,false)$. This means that for $1 \leq a \leq l_d+1$, $f(k\cdot(l_d+1)+a,l_d,false) = f(k\cdot(l_d+1)+a+(a-1),l_d,false)$, so starting the defence in the susceptible vertex distance $a-1$ away from the threshold, i.e. distance a from the fire - thereby effectively reducing the length of the component by $a-1$ - leads to exactly as much defence as starting from the threshold vertex in that component.

If C_1 only has one initially burning vertex; or if the defender will travel left if it starts at the left side of C_1 or right if it starts at the right side, then starting away from the threshold vertex will only lead to fewer vertices being saved in C_1 since it will not save more vertices in C_1 and will reach other components later.

Starting a defender away from the threshold vertex can have the advantage that it can reach further components sooner and potentially save more vertices in them - see Figures 5.9 and 5.10.

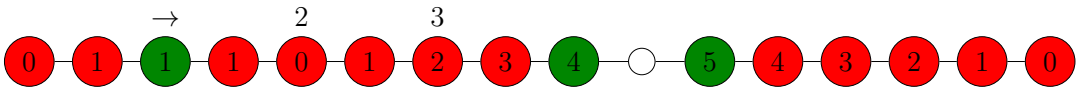


Figure 5.9: If the fires break out on the red vertices labelled 0, $l_d = 2$ and $d = 1$ and the defender starts distance one away from the leftmost threshold vertex, then it can save four vertices in total

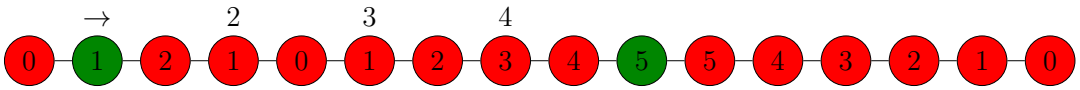


Figure 5.10: If there is the same initial game setup as in Figure 5.9, again with $l_d = 2$, $d = 1$, then if the defender starts on the leftmost threshold vertex instead, it can only save two vertices.

We use the notation $x_1^r =<$ resp. $x_1^l =<$ if we want a defender to start on the right resp. left hand side of component C_1 and then travel left and use $x_1^r =>$ resp. $x_1^l =>$ if it then travels right. We define $x_1^r = 0$ resp. $x_1^l = 0$ if the defender does not start on the left resp. right side of C_1. The vertex coordinates of the left resp. right fire in C_1 are given by f_1^l resp. f_1^r if they exist. If the left- resp. rightmost vertex of C_1 is not initially burning we define f_1^l resp. f_1^r to be null. We want to start the defender as far away from the threshold vertex as is possible, without reducing the number of vertices it can save in the component it starts

in. This is achieved with Algorithm 15, which follows directly from the above discussion. We will later prove that the vertices it gives are optimal to start defending from in Lemma 77. See Figures 5.4 and 5.4 for an examples showing some of the starting points given by Algorithm 15.

Input: l_d, l_j, s, f_j^l, f_j^r, x_1^r, x_1^l, all as defined above.
Output: A vertex coordinate v_d for the defender to be placed on at time step 1

if $f_j^r = null$ **then**
  Return $v_d = f_j^l + 1$;
else if $f_j^l = null$ **then**
  Return $v_d = f_j^r - 1$;
else if $x_1^l =<$ **then**
  Return $v_d = f_j^l + 1$;
else if $x_1^r =>$ **then**
  Return $v_d = f_j^r - 1$;
else if $x_1^l =>$ **then**
  if $s = false$ **then**
    Define a to be the unique integer s.t $l_j = k \cdot (l_d + 1) + a$ with $0 < a \leq l_d + 1$, $0 \leq k$, an integer.;
    Return $v_d = f_j^l + \min\{a, l_j\}$;
  else
    if $l_j \equiv 2 \mod (l_d + 1)$ **then**
      Return $v_d = f_j^l + 2$;
    else
      Return $v_d = f_j^l + 1$;
    end
  end
else
  if $s = false$ **then**
    Define a to be the unique integer s.t $l_j = k \cdot (l_d + 1) + a$ with $0 < a \leq l_d + 1$, $0 \leq k$, an integer;
    Return $v_d = f_j^r - \min\{a, l_j\}$;
  else
    if $l_j \equiv 2 \mod (l_d + 1)$ **then**
      Return $v_d = f_j^l + 2$;
    else
      Return $v_d = f_j^r - 1$
    end
  end
end

Algorithm 15: Algorithm for finding the starting positions for a single defender that maximise the number of vertices they save in the component in which they start.

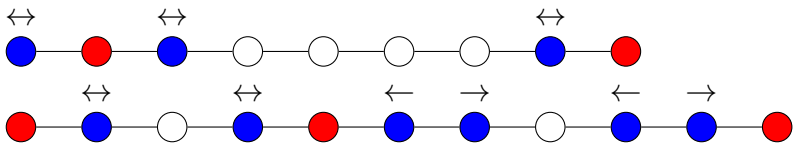


Figure 5.11: If the fires break out on the red vertices on the two paths shown at time 0 with $l_d = 2$, $d = 1$ and we should shorten in the first component, then all the starting points given by Algorithm 15 consist of the blue vertices paired with the direction indicated above them. A blue vertex v with $\leftrightarrow$ above it indicates that both $(v, left)$ and $(v, right)$ are valid starting points.

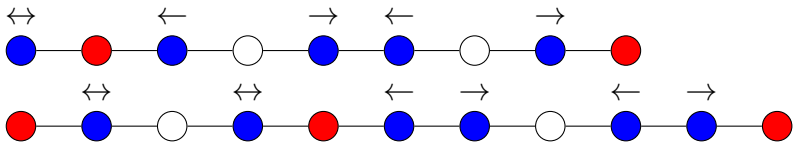


Figure 5.12: If all other conditions are as in Figure 5.4 but we should shorten in the first component, then all the starting points given by Algorithm 15 consist of the blue vertices paired with the direction indicated above them.

We will take this into account when describing the optimal defence algorithm, which we can now start putting together. The first thing we shall prove is which components we should shorten the defender's step in.

Lemma 75. *For a single defender with $l_d = 2$ starting in C_1 on a path with any number of initial fires, if at least one vertex can be saved in any further component C_j, then not shortening in C_i with $i < j$ leads to at most as much burning than shortening in C_i.*

Proof. By the discussion of $w(v, dir, j, \sigma)$ above, shortening in C_i will only lead to at most one extra vertex being saved in C_j. Meanwhile, not shortening has the effect of ensuring that the defender will reach further components C_j with $j > i$ at an earlier time step - i.e. their value of the function $g(v, dir, j, \sigma)$ will be reduced by one. For further components where you can save at least one vertex if you do not shorten, reducing $g(v, dir, C_j, \sigma)$ by one will increase the number of vertices saved by exactly one if the further component has only one fire or by either one or two if the component has two fires. □

Note that the last component with a vertex that can be saved given the defender's starting point is not necessarily the last component in front of the defender.

We shall now prove that once the defender has started moving in a given direction, they should continue in the same direction for the rest of the game.

Lemma 76. *For Distance-Limited Firefighter on a path P with a single defender with $l_d = 2$, there always exists an optimal defence strategy where the defender does not 'turn' i.e. they first move closer to one endpoint of P, then to another.*

Proof. It suffices to show that for any strategy D where the defender turns at least once, there exists a related strategy D' where the defender makes at least one fewer turn; and that leads to at most as much burning.

Without loss of generality, assume that using defence strategy D the defender d starts on vertex v_s in component C_1, travels left until component C_j (where C_j may be equal to C_1), until it defends vertex v_t in time step $t-1$, then travels right, starting to move towards v_s again in time step t_t. Whether d later turns again is not relevant for our argument.

We consider two main cases. In the first case, the defence strategy D does not save any vertices to the right of v_s. Consider a defence strategy D_1 which differs from D in that the defender does not turn at time t_t and instead keeps travelling left. Note that for $l_d = 2$, using defence strategy D, the defender could not save any further vertices between v_s and v_t on its second pass after it turned at v_t and went right, so the burning in this part of the graph is unaffected. If the original defence strategy D involved the defender eventually defending anything to the left of v_t after further turns, then in D_1, the defender shall defend exactly the same vertices to the left of v_s as in D, in the same order but at the earliest possible time step. Since the defender did not save any vertices to the right of v_s, the burning in that part of the path will be completely unchanged. To the left of v_t, the defender will reach each vertex in an earlier time step if it reached them at all in D, leading to at most as much burning as in D. If the original defence D did not involve ever defending vertices to the right of v_t, then assume we move the defender two steps further left per time step starting at t_t. Again, the burning to the right of v_s is unchanged, but at least as many vertices will be saved to the left of v_t.

For the second case, which forms the rest of the proof, we assume that D saves at least one vertex v^* in component C^* which is to the right of v_s.

We compare D to D_2, which is derived from D by having the defender start on v_t and then defend the same vertices that it does in D after initially reaching v_t, in the same order but at earlier time steps. D_2 will save at least as many vertices as D as long as any reduction in the number of vertices saved between v_t and v_s caused by defending from left to right in D_2 - rather than from right to left in D - is compensated by the extra vertices that the defender can save to the right of v_s by reaching them at an earlier time step.

Note that for component C^*, the number of vertices saved in it - given by $w(v, dir, *, \sigma)$ - decreases by one with each increase by one of $g(v, dir, *, \sigma)$ for components with a single fire but can decrease by one or two for each increase by one of $g(v, dir, *, \sigma)$ for components with two fires. So it is sufficient to prove the claim for the case where the only part of P to the left of v_s is a component with a single fire and the rightmost component of P as this will lead to the least difference in burning between D and D_2.

Since we assume the the defender can defend in C^*, we can assume no shortening before then by Lemma 75. If v_t is an odd distance away from v_s, then the defender must have shortened its step in D and we use D' and D'_2, where the turning resp. starting point is $v_t - 1$ and the defender does not shorten their step before reaching C^*.

Starting from v_t and travelling right (in either D or D_2), the defender will

travel forward by two vertices and the left fire in C^* will move right by one vertex per time step, so overall the distance between the defender and the nearest susceptible vertex in C^* is reduced by one per time step.

This means that using defence strategy D_2, where the defender starts on v_t, it will defend a susceptible vertex v_2^* in C^* at time step $d(v_t, C^*) + 1$ (since it does not move forward in time step one).

Using defence strategy D, the defender will need $\frac{d(v_s,v_t)}{2}+1$ time steps to travel from v_s to v_t; it will then take another $d(v_t, C^*)$ time steps for it to travel from v_t to v_2^* but the fire will have spread further in the extra time the defender took to reach v_2^*, specifically by $\frac{d(v_s,v_t)}{2}$. The defender needs to make up this distance and will do that in $\frac{d(v_s,v_t)}{2}$ time steps. Adding everything up, then using D, the defender will only be able to defend v^* in time step $\frac{d(v_s,v_t)}{2}+1+d(v_t, C^*)+\frac{d(v_s,v_t)}{2} \geq d(v_s, v_t) + d(v_t, C^*) + 1$.

So the defender will save at least $d(v_s, v_t)$ fewer vertices in C^* in strategy D than in D_2. This means even if D saved every single vertex between v_s and v_t, while D_2 saved none, D_2 would lead to at most as much burning as D and have one fewer turn, as required.

□

This unfortunately does not generally hold for larger values of l_d - for an example, see Figure 5.13. If $l_d \geq n$, then the game is equivalent to a game with no upper limit on how far the defender can move and moreover is firepath equivalent (since there are no vertices with degree strictly greater than to), so an optimal defence is given by Algorithm 3 on page 28.

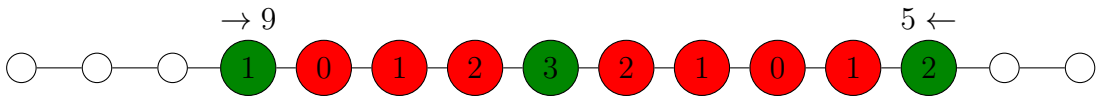


Figure 5.13: For a single defender with $l_d = 9$, this shows the optimal defence. The arrows and numbers above defended vertices show which direction and how far the defender will move in the next defence step.

The final claim we need to prove is that we should indeed initially defend one of the vertex, direction pairs returned by Algorithm 15.

Lemma 77. *For $l_d = 2$ and $d = 1$, initially defending the vertices given by Algorithm 15 always leads to at most as much burning as initially defending any other vertex in the same component and travelling in the same direction.*

Proof. Without loss of generality, we consider a defence strategy where a defender travels right and with a Boolean variable of whether it does or does not shorten of s. Assume the defender starts at a vertex v_s in component C_1 such that v_s is not returned by Algorithm 15 with inputs l_d, l_1, s, f_1^l, f_1^r and either $x_1^r = 0$ and $x_1^l =>$; or $x_1^r =>$ and $x_1^l = 0$.

If there are no further components to the right of C_1 then it is clear that defending instead starting from a, the vertex given by Algorithm 15 with inputs

l_d, l_1, s, f_1^l, f_1^r, $x_1^r = 0$ and $x_1^l =>$ (which may or may not be the left hand threshold vertex), will lead to a reduction in the amount burning of $f(l_1, l_d, s) - f(l_1 - dist(v_s, a), l_d, s) \geq 0$ if C_1 has two fires and v_s is to the right of a. If v_s is to the left of a then there will be no difference in burning in C_1 since Algorithm 15 was designed to output the furthest vertex to the right that would lead to the same amount of burning as if the defender started on the left threshold vertex. Hence there would be no difference in burning overall, since the defender will not defend any other components. If C_1 has a single fire to the left of v_s then starting on a rather than v_s will lead to a reduction in burning of $dist(v_s, a)$.

If there is a single fire in C_1 and it is to the right of v_s, then we initially defend b, the vertex given by Algorithm 15 with inputs l_d, l_1, s, f_1^l, f_1^r, $x_1^r =>$ and $x_1^l = 0$, which will be the rightmost threshold vertex in C_1. Initially defending b rather than v_s leads to a decrease in burning of $dist(v_s, b) \geq 0$, so the claim is proven.

This leaves the case where there is at least one component to the right of C_1. We will prove the claim by contradiction. For the rest of this proof, we denote by D_v the defence technique that starts at v and moves the defender two steps right each time step unless the defender is in a component with two starting fires, the defender is directly behind a susceptible vertex which is itself directly behind a fire and it is not possible to save at least one vertex in a component in front of the defender. In that case, we shorten the defender's step and move it forwards by one. There is an optimal defence strategy like this by Lemma 75.

Assume that the optimal defence D_{v_s} initially defends a vertex v_s in C_1 which as before is not returned by Algorithm 15 with the relevant inputs and that there is at least one component to the right of C_1. Assume further that the optimal defence saves strictly more vertices than any defence technique which initially defends a vertex returned by Algorithm 15 for a given direction and end of a component. In particular, it is strictly better than D_a and D_b, where a and b are defined as above.

If D_{v_s} does not save any vertices in any component to the right of C_1, then D_a would save at least as many vertices - with the difference between the number of vertices saved by D_a vs. D_{v_s} given by $f(l_1, l_d, s) - f(l_1 - dist(a, v_s), l_d, s) \geq 0$ - a contradiction.

Assume now that D_{v_s} saves at least one vertex before the fire reaches it in at least one component C_i which is to the right of C_1. D_{v_s} saves $f(dist(v_s, b) + 1, l_d, s)$ vertices in C_1, while D_b saves no vertices in C_1 other than b itself. Note that using D_b rather than D_{v_s} means reaching susceptible vertices and defending them in components to the right of C_1 faster - specifically by $g(b, right, i, \sigma) - g(v_s, right, i, \sigma) = dist(v_s, b)$ time steps for all components C_i to the right of C_1. Every reduction in the number of time steps it takes to reach a component will lead to one or two extra vertices saved in it (one if the component has a single fire, one or two if it has two fires and the defender shortens in that component as necessary). We consider the final component C_f that the defender can save any vertices in using D_{v_s}. Then it will always be optimal for the defender to shorten as necessary in this component, so at least $dist(b, v_s) \geq f(dist(v_s, b) + 1, l_d, s) - 1$ more vertices will be saved in C_f using D_b than by using D_{v_s}.

Hence, overall D_{v_s} cannot be strictly better than both D_a and D_b, so the claim is proven.

□

The previous lemmas and the calculations of how to calculate how many vertices will be saved if the defender starts at a given vertex and travels in a given direction can now be combined into an algorithm for optimal defence.

Theorem 78. *Algorithms 16 and 17 together give an algorithm for optimal defence with one defender which can travel at most distance two per time step, defending on a path, which runs in $\mathcal{O}(f^2)$.*

Proof. By Lemma 77, we should initially defend a vertex returned by Algorithm 15. There are at most $4f$ of these and we check all of them. By Lemma 76 we should then only move the defender in one direction. Finding the best defence technique then becomes a matter of calculating how many vertices would be saved starting the defender at one of the a valid (vertex, direction) combinations given by Algorithm 15. Calculating how many vertices would be saved using Algorithm 16 on page 141 involves some constant time calculations for at most $2f$ components for each pair due to Lemma 75, giving a worst case time of $\mathcal{O}(f^2)$. □

Multiple Defenders on Paths with $l_d = 2$

In this section, we focus on the case where $l_d = 2$ and there are multiple defenders; and we work towards a dynamic program for calculating the best defence strategy, assuming that there exists an optimal defence where defenders do not turn, and where they start on certain vertices.

We first need to consider how many vertices can be saved in a component C_j if the defenders start on threshold vertices, always travel forwards and shorten their step at most once per component. We use the notation $L(C_j)$ for this.

Again, we arbitrarily fix 'left' and 'right' on the path. In this section, we label the leftmost component C_1, its neighbour C_2 and so on. From left to right the vertices are given coordinates $1, 2, \ldots n$. The length of a component C_j is denoted by l_j and excludes initially burning vertices, as before. For a given component C_j, the vertex coordinates of the left resp. right fire are given by f_j^l resp. f_j^r if they exist. If the left- resp. rightmost vertex of C_j is not initially burning we define f_j^l resp. f_j^r to be null. We use the notation $x_j^r =<$ resp. $x_j^l =<$ if we want a defender to start on the right resp. left hand side of component C_j and then travel left and use $x_j^r =>$ resp. $x_j^l =>$ if it then travels right. We define $x_j^r = 0$ resp. $x_j^l = 0$ if the defender does not start on the left right resp. side of C_j.

If $x_j^l =>$, we define v_j^l to be the vertex the defender starting on the left side of C_j and traveling right starts on. Similarly, if $x_j^r =<$, then we define v_j^r to be the vertex the defender starting on the right side of C_j and traveling left starts on. If $x_j^l \neq>$ resp. $x_j^r \neq<$, then we define v_j^l resp. v_j^r to be the starting point for the nearest defender to the left resp. right of C_j (and not inside C_j) which is moving towards it. We define d_j^l resp. d_j^r to be the distance to such defenders. If there is no defender to the left resp. right of C_j which is moving towards it, we define d_j^l resp. d_j^r to be infinite. If $x_j^l =>$ resp. $x_j^r =<$, we also define d_j^l resp. d_j^r to be infinite, since those will not make any difference to the burning in C_j.

As before, it is important to know whether certain defenders will shorten their step in C_j. We use s_j^l resp. s_j^r to indicate whether the nearest defenders travelling right resp. left into C_j shorten their step inside C_j. Note that if $x_j^l =>$, then s_j^l refers to whether the defender that starts in C_j at the left hand side of C_j shortens its step in C_j; for any other value of x_j^l, s_j^l indicates whether the defender that initially starts on v_j^l shortens its step in C_j. The variable s_j^r is defined similarly. We also need to know by how much (if any) certain defenders have shortened their step before reaching C_j. We use the integer variables σ_j^l resp. σ_j^r to show the total amount that the same defenders have shortened their step before they reach C_j. If it is obvious which defender we are talking about, we just write σ.

The functions $f(x, l_d, s)$ and $g(v, dir, i, \sigma, l_d)$ are as defined in (5.27) and (5.28) respectively. Since we only focus on $l_d = 2$ in this section, we frequently write $g(v, dir, i, \sigma)$ to mean $g(v, dir, i, \sigma, 2)$.

We first consider the case where C_j only has one initially burning vertex. If a defender is placed in the susceptible vertex in C_j next to the fire at time 1, then all the vertices of C_j which were not initially burning will be saved. If there is no defender placed in C_j at time step 1, then the number of vertices that burn in C_j will equal the number of time steps the fire can spread before the nearest defender travelling towards C_j can reach a susceptible vertex - i.e. $g(v_j^l, right, j, \sigma)$ if C_j if the rightmost component or $g(v_j^l, right, j, \sigma)$ if C_j is the leftmost component. This is summarised in Algorithm 18 on page 142.

Now consider the case where C_j has two fires and no defenders are initially placed inside it. If there are no defenders moving towards C_j (i.e. $d_j^r = \infty = d_j^l$), then nothing will be saved in C_j. If there are only defenders moving towards C_j from one side (i.e. $d_j^r = \infty$ XOR $d_j^l = \infty$), then the amount of burning in C_j is equal to the amount of burning if the closest defender travelling into C_j were the only defender and is given by $\max(0, f(l_j - 2g(v, dir, j, \sigma)))$ where v is the start point of the closest defender which travels towards C_j, following from the discussion of $w(v, dir, j, \sigma)$ with $d = 1$ above. Finally, if there are defenders travelling into C_j from both directions, then we must consider the midpoint where the closest such defenders will meet, given by $Midpoint = \frac{v_j^r + v_j^l - \sigma_j^r + \sigma_j^l}{2}$, which may be non-integer and thus part way between two vertices. If the midpoint is entirely outside C_j, then the fire will have finished spreading in C_j before the further defender reaches it, so again burning is as in the $d = 1$ case if the nearest defender travelling towards C_j were the only defender. If C_j does contain the midpoint, then the number of vertices saved will be given by the maximum of the vertices that would be saved by the nearest defenders travelling into C_j if they were the only defender; and the number of vertices saved if both the nearest defender travelling into C_j from the left and the nearest defender travelling in from the right contributed to the defence of C_j. The later is given by l_j minus the number of vertices that burn on the left and right side of C_j before those two defenders reach C_j - i.e. $l_j - g(v_j^l, right, j, \sigma) - g(v_j^r, left, j, \sigma)$. This is summarised in Algorithm 19 on page 143.

Finally, we consider the case where at least one defender is initially placed in a component C_j which has two initially burning vertices. If there are defenders initially placed in both of the vertices in C_j which neighbour burning vertices in

time step one, then all the vertices of C_j which were not initially burning will be saved. Suppose now that no defenders are placed on the rightmost susceptible vertex and there is a defender initially placed on the leftmost susceptible vertex travelling left, i.e. leaving C_j in time step 2.

Suppose that the nearest defender travelling towards C_j from the right is too far away to contribute to any vertices being saved in C_j but the nearest defender travelling towards C_j from the left does contribute. It must reach the vertex f_j^l+2 in C_j distance two away from f_j^l before the fire spreading from the right reaches it.

If the distance between v_j^l and f_j^l+2 plus σ_j^l is even (recall from above arguments that increasing σ_j^l by one has the same effect as increasing the distance by one), then d_j^l will land on f_j^l+2. Since the nearest defender travelling from the left to C_j travels two vertices forward per time step after time step 1, it will take it $1+\frac{d(v_l^j,f_j^l)+2+\sigma_j^l}{2}$ time steps to reach the vertex f_j^l+2, by which time the fire will have spread $\frac{d(v_l^j,f_j^l+2)+\sigma_j^l}{2}$ along C_j from f_j^r and one vertex in C_j will already have been defended. This is equivalent to starting a defender travelling right on the left threshold vertex of a component of length $l_j-\left(\frac{d(v_l^j,f_j^l)+2+\sigma_j^l}{2}-1\right)$, so the number of vertices saved is given by $1+f\left(l_j-1-\frac{d(v_l^j,f_j^l)+2+\sigma_j^l}{2},2,s_j^l\right)$.

If $d(v_j^l,f_j^l+2)+\sigma_j^l$ is odd, then the nearest defender travelling from the left to C_j will land on f_r^l+1, the vertex which has already been protected. It will arrive there at time step $\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}$, during which time the fire will have travelled $\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}-1$ along C_j from f_j^r. If $l_j<2+\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}$ then it will not be able to save anything in C_j and only f_j^l+1 (which was protected in time step 1 by another defender) will be saved. If $l_j=2+\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}$ then this will leave a gap of one susceptible vertex between the defender and the fire at the end of that time step. If the defender shortens its step in the next turn, then it will save one vertex in addition to the left threshold vertex which was defended in time step 1 by another defender. If it does not shorten its step, then only the left threshold vertex will be saved. If $l_j>2+\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}$ then the defender will reach f_j^l+3 in time step $1+\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}$, ensuring f_j^l+1 and f_j^l+2 are saved, during which time the fire will have spread $\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}$, so the number of vertices saved is given by $2+f\left(l_j-2-\frac{d(v_j^l,f_j^l+3)+\sigma_j^l}{2}\right)$.

Thus far for the case where there is no defender on the rightmost susceptible vertex at time step 1 and that the defender on the leftmost vertex is travelling left, we have only considered the case where the nearest defender travelling into C_j from the left contributes to saving vertices in C_j and the nearest defender travelling into C_j from the right does not. If the nearest defender travelling into C_j from the right does contribute to saving vertices in C_j - i.e. if it overtakes the fire spreading from f_j^r - then $l_j-g(v_j^r,left,j,\sigma_j^r)$ vertices will be saved. If neither of the closest defenders travelling into C_j from the left and right save a vertex in C_j, then only the left threshold vertex that was defended by another defender

in time step 1 will be saved. Choosing the maximum of these values gives the actual number of vertices saved in C_j.

Finally, consider the case where there is no defender on the rightmost susceptible vertex at time step 1; the defender on the left side of C_j vertex is travelling right; and there is a defender to the right of C_j which is travelling left. The midpoint between v_j^r the defender on the left side of C_j and v_j^r will be at $\frac{v_j^l+v_j^r-s_j^l+\sigma_j^r}{2}$. If this midpoint is in C_j but the nearest defender travelling into C_j from the right will still not reach C_j in time to contribute to saving any vertices in it then $f(l_j, 2, s_j^l)$ vertices will be saved. If the nearest defender travelling into C_j from the right does contribute to saving at least one vertex in C_j then $g(v^r, left, j, \sigma)$ vertices will burn before it reaches C_j and the rest will be saved. If the midpoint is to the left of f_j^l (it cannot possibly be to the right), then the fire will have finished spreading by the time the nearest defender travelling into C_j from the right arrives in C_j and the number of vertices saved is given by $f(l_j, 2, s_j^l)$. This is summarised in Algorithm 20 on page 144. The case where there is a defender initially placed on the rightmost susceptible vertex and no defender initially placed on the leftmost susceptible vertex follows analogously.

Note from the above discussion that it never leads to a reduction in burning to shorten other than when a defender is distance two away from a burning vertex, which will happen at most once per component. Furthermore, it will never happen in a component where two different defenders contribute to defending it.

If there is only one defender starting in a given component, then the logic for which exact vertex a defender should start on is similar to the case where $d = 1$, discussed above. We again assume that we want to maximise the number of vertices saved by the defenders in the components in which they start, and only then try to maximise how far along in the direction the defender travels in that it starts. Unlike for the case where $d = 1$, proving this holds for optimal defence goes beyond the scope of this thesis.

Note that for games with multiple defenders, the exact same logic as for the $d = 1$ case does not always hold, and indeed it is strictly better to defend the threshold vertices of a component C_i if there are two defenders placed in C_i even if $l_i \equiv 2 \mod (l_d + 1)$. An example of this is given in Figures 5.14 and 5.15.

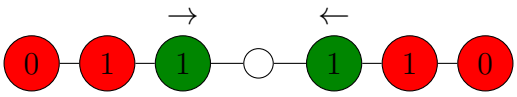


Figure 5.14: If the defenders are placed on the susceptible vertices next to the thresholds and $l_d = 2$, only three vertices are saved.

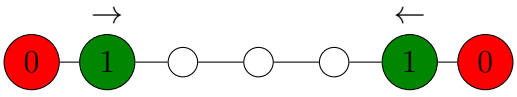


Figure 5.15: If the defenders are placed on the threshold vertices $l_d = 2$, five vertices are saved.

Another caveat arises if one defender a starts in C_j and moves away from the fire on a component with two fires and another defender b is moving towards

it. In this case it may be optimal for the first defender to start as far from the threshold as possible while still maximising $f(l_j, 2, s)$ if defender b does not reach C_j in time to contribute to the defence; or it may be optimal to start a on the threshold vertex if b does also contribute to the defence of C_j. For an example, see Figures 5.16–5.18 Since knowing which option would be better for a depends on what b does and how much it shortens its step, working it out could involve an infinite loop. To get around this, we suggest the two possible vertices a should possibly start on, and in the later algorithm for a defence strategy, we try both.

The vertices we propose the defenders should start on are given by Algorithm 21 on page 145.

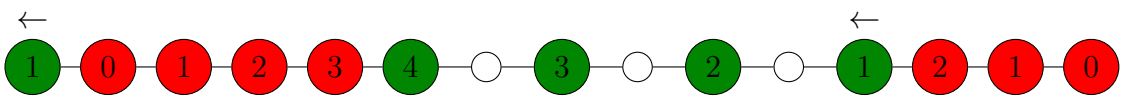


Figure 5.16: If the defender starting further left moves left and does not shorten, then the vertex distance two away from the threshold is an optimal starting place for the defender on the right.

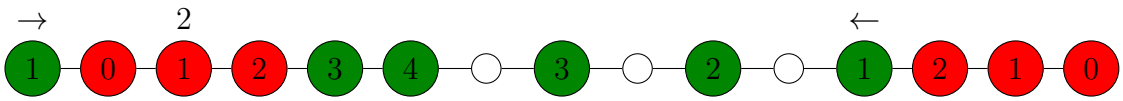


Figure 5.17: If the defender starting further left moves right and thus contributes to defending the component the defender starting further right start in, then starting the later defender distance two away from the threshold only leads to 8 vertices being saved in that component.

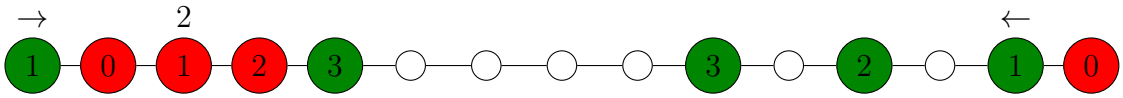


Figure 5.18: If the defender further right starts on the threshold vertex instead, then 10 vertices are saved in that component.

Since Algorithm 15 suggests sometime starting a defender a away from a threshold vertex, it could be possible for a defender b from another component to defend a vertex between the first defender and the nearest threshold vertex to it, which would change the calculations for $L(C_j)$ above. However, for $l_d = 2$ this does not occur, since the fire would spread all the way to the vertex a started on before b could possibly reach it, see Figure 5.19.

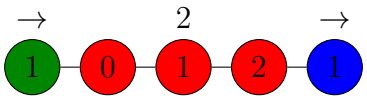


Figure 5.19: Even if the blue defender starts distance two from the threshold vertex and the green defender starts as close to that component as possible, it still cannot defend between the fire and the starting point for the blue defender before the fire spreads.

Note that unlike for the case with a single defender, it is possible for the optimal defence given the start points and directions for the defenders to involve a given defender shortening its step in multiple components, and for this to be strictly better than not shortening its step, or shortening its step only once. An example of this is given in Figure 5.20.

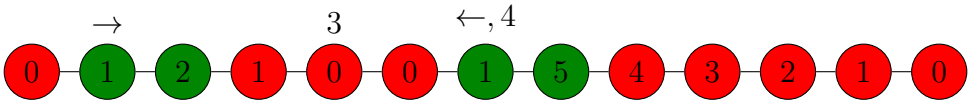


Figure 5.20: Assuming that the two defenders start on the vertices marked 1 and travel in the indicated directions, it is optimal for the defender starting further left to shorten its step twice. Numbers above vertices indicate the time step that it reached a given vertex in.

While there is a polynomial time algorithm for best defence against multiple fires with multiple fires and $l_d = 1$ it does not work for $l_d \geq 2$. Moreover, we can rule out certain greedy algorithms which optimise defenders separately for the case with multiple fires and defenders on a path with $l_d \geq 2$, see Figures 5.21–5.25.

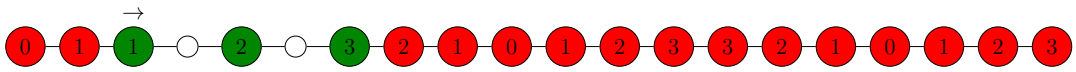


Figure 5.21: The best defence with a single defender starts on the third vertex from the left and travels right.

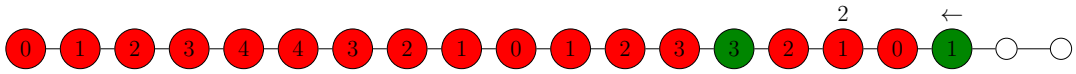


Figure 5.22: Starting defence on the third vertex from the right and travelling left only saves four vertices.

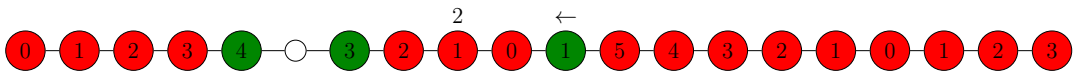


Figure 5.23: Starting defence on the tenth vertex from the right and travelling left also only saves four vertices.

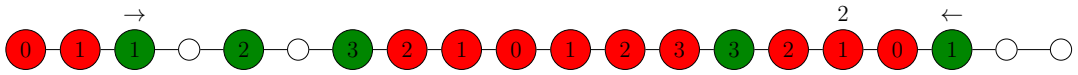


Figure 5.24: Having decided that one defender will start on the third vertex from the left and travel right, the best additional defender starts on the third vertex from the right and travels left, together saving nine vertices.

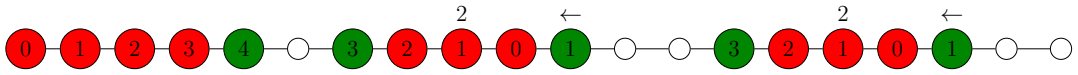


Figure 5.25: The best defence strategy for two defenders has one defender starting on the tenth vertex from the right and travelling left; and the other starting on the third vertex from the right and travelling left. Together, they save ten vertices. Note that neither (v, dir) pair was optimal as a lone defender.

Before we can develop a dynamic program to find optimal defence for $l_d = 2$ and $d > 1$, we need to address the questions of whether the defenders turn and whether they do indeed start on the vertices given by Algorithm 21 on page 145. Note that the proofs of the Lemmas of the equivalent claims for $l_d = 2$ and $d = 1$ (Lemmata 76 and 77 respectively) both rely on the fact that for $l_d = 2$ and $d = 1$, reaching a component where you could defend at least one vertex one step earlier always results in at least one more vertex being saved in that component. However, for $l_d = 2$ and $d \geq 2$, this does not hold - specifically for the case where the defender which may or may not arrive earlier can only defend in a component that had a defender placed on the side closest to the first defender and moving towards it. See Figures 5.26 and 5.27. While we hypothesise that these claims are still true for $l_d = 2$ and $d \geq 2$, proving them goes beyond the scope of this thesis. As such, we shall use the following assumptions for the dynamic program:

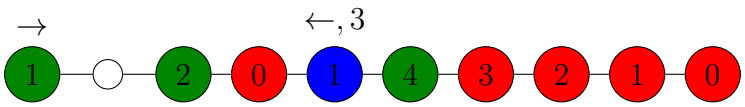


Figure 5.26: If the green defender starts distance four away from the component with the blue defender, it still only manages to save one vertex in that component, in addition to the one vertex saved by the blue defender.

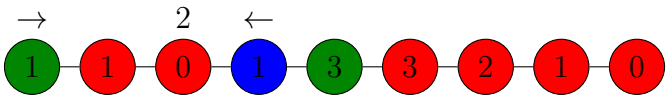


Figure 5.27: If the green defender starts distance only 3 away from the component with the blue defender, it still only manages to save one vertex in that component.

Assumptions

- No defender turns.

- The defenders only start on vertices returned by Algorithm 21, with the appropriate inputs.

Based on these assumptions, we can make one last deduction about optimal defence.

Lemma 79. *If $l_d = 2$ and $d \geq 2$ on a path with any number of fires, and given the above assumptions, as long as d is at most the number of threshold vertices in the game at time 0, there exists an optimal defence where at most one defender is placed on each end of each component.*

Proof. It suffices to show that for any defence strategy D where more than one defender is initially placed on the same end of a component C_j, there exists a related defence strategy D' where one of those defenders d_m is placed on a component side which did not have a defender initially placed on it in D and which leads to at most as much burning as D. Without loss of generality, assume that d_m was initially placed on the left side of C_j in strategy D.

Assume C_j only has defenders on one side and without loss of generality, assume it is the left side. We first assume at least one of the defenders on the left side of C_j in D was moving right. We define D' by moving this defender to the right threshold vertex of C_j and keeping it moving right; and moving all other defenders on the left hand side of C_j to the left threshold vertex (if they were not there already), keeping their direction. All vertices in C_j will be saved and the defender now on the right threshold vertex of C_j will reach any components further to the right at earlier time steps, leading to at most as much burning in components to the right of C_j. For components to the left of C_j, either there is still a defender from the left threshold vertex of C_j moving left, that will reach them at the same time step as in D, saving the same number of vertices; or there were no defenders in D in C_j moving left, in which case the burning is also unchanged.

Now assume all the defenders on the left hand side of C_j in D are moving left. Move one to the right threshold vertex (all others will already be on the left threshold vertex since we assume they were returned by Algorithm 21). This will save every vertex in C_j, not affect burning to the right of C_j since those components were not protected by any of the affected defenders in D and not affect burning to the left of C_j since there will still be a defender on the left threshold vertex of C_j moving left, which will reach those vertices in the same times steps.

If C_j has defenders at both sides, then they will be located on the two threshold vertices of C_j, since we assume they were returned by Algorithm 21. First, we try moving a defender d_m forwards in the direction it was moving in in D until it reaches a side of a component that does not have a defender on it at time 0. without loss of generality, assume d_m is moving right. If there is a component side to the right of C_j with no defenders, we move d_m to the closest one. The amount burning to the left of C_j is unaffected since d_m did not reach them in D. The amount of burning in C_j is unaffected since all non-initially burning vertices in it are also saved in D'. To the right of C_j, there will be at least one more saved vertex in the component d_m has been moved to.

If there is no such component side to the right of C_j with no defenders, then all threshold vertices to the right of C_j are defended and all vertices they contain that were not initially burning, will be saved, even if d_m is moved to a component side to the left of C_j, which we now do to define D', this leads to at least one more vertex being defended to the left of C_j than using D, completing the proof. □

We can now start on the dynamic program. To keep all the notation together, we summarise it here:

Notation for parameter values

$j :=$ The index of the current component, $1 \leq j \leq m$.

$C_j :=$ The j-th component of $G \setminus F$.

$c_j^l :=$ The left-most vertex of the j-th component.

$c_j^r :=$ The right-most vertex of the j-th component.

$f_j^l :=$ The index of the initially burning vertex immediately to the left of c_j^l, undefined if there is no initially burning vertex to the left of C_j.

$f_j^r :=$ The index of the initially burning vertex immediately to the right of c_j^r, undefined if there is no initially burning vertex to the right of C_j.

$S_j :=$ The sub-path starting at the left-most vertex c_j^l of component j and extending to the right-most vertex of the path.

$p_j :=$ The number of defenders we can use for sub-path S_j, $0 \leq p_j \leq d$.

$d_j^l :=$ The distance to the closest defender strictly to the left of C_j which is moving towards it, $d_j^l \in \{1, \ldots, c_j^l - 1, \infty\}$, ∞ if there is no such defender.

$\sigma_j^l :=$ How often the closest defender strictly to the left of C_j which is moving towards it has shortened its step before it reaches C_j. It is bounded above by the number of components the defender must pass through to reach C_j. If there is no defender to the left of C_j moving right, then σ_j^l is undefined and we write $\sigma_j^l = -$.

$\delta_j^r :=$ Indicates whether we must place a defender in C_j that is moving left, $\delta_j^r \in \{0, l^0, r^0, r^1, rr^0\}$; a value of 0 means that no such defender is placed in C_j, the remaining values indicate whether the defender is placed on the left threshold vertex (l), the right threshold vertex (r), or – if returned by Algorithm 21 for the case "$x_j^r =<$" – the second vertex on the right end of C_j (indicated by rr, and whether or not the defender shortens their step in C_j (^{0} or ^{1}).

$d_j^r :=$ The distance to the closest defender strictly to the right of C_j which is moving towards it, $d_j^r \in \{1, \ldots, n - c_j^r, \infty\}$, ∞ if there is no such defender.

$\sigma_j^r :=$ How much the closest defender strictly to the right of C_j which is moving towards it has shortened its step before it reaches C_j. It is bounded above by the number of components the defender must pass through to reach C_j. If there is no defender to the right of C_j moving left, then σ_j^r is undefined and we write $\sigma_j^r = -$.

Notation for decision variables

The decision to be taken for component C_j is given by the four-tuple $x_j = (x_j^l, s_j^l, x_j^r, s_j^r)$, where:

x_j^l : Indicates whether a defender is located closer to the left threshold vertex of C_j than to the right. The respective values are:

- 0 No defender is placed
- $<$ A defender moving left is placed on the left threshold vertex.
- $>$ A defender moving right is placed on the left threshold vertex.
- $\gg$ If Algorithm 21 with input $x_j^l =>$ returned two vertices, then a defender is placed on the vertex further to the right.

s_j^l : If $x_j^l \notin \{>, \gg\}$, then s_j^l indicates whether the nearest defender travelling from the left into C_j shortens their step inside C_j. If $s_j^l = 0$ then it does not shorten its step in C_j; if $s_j^l = 1$, then it does; if there is no such defender then s_j^l is undefined and in the tables we write $s_j^l = -$. If $x_j^l \in \{>, \gg\}$, i.e., we are placing a defender at the left end of C_j moving right, then this defender will supersede any other defender to the left of C_j moving towards it. Therefore, in this case s_j^l indicates whether the defender that starts in C_j at the left hand side of C_j shortens its step inside C_j.

x_j^r : Indicates whether a defender is located closer to the right threshold vertex of C_j than to the left. The respective values are:

- 0 No defender is placed
- $>$ A defender moving right is placed on the right threshold vertex.
- $<$ A defender moving left is placed on the right threshold vertex.
- $\ll$ If Algorithm 21 with input $x_j^r =<$ returned two vertices, then a defender is placed on the vertex further to the left.

s_j^r : If $x_j^r \notin \{<, \ll\}$, then s_j^r indicates whether the nearest defender travelling from the right into C_j shortens their step inside C_j. It has the same possible values as s_j^l. If $x_j^r \in \{<, \ll\}$, i.e., we are placing a defender at the right end of C_j moving left, then this defender will supersede any other defender to the right of C_j moving towards it. Therefore, in this case s_j^r indicates whether the defender that starts in C_j at the right hand side of C_j shortens its step inside C_j.

Notation for number of saved vertices

Finally, the number of saved vertices are given by:

$L(C_j) :=$ The number of saved vertices in the component C_j for decisions $x_j = (x_j^l, s_j^l, x_j^r, s_j^r)$ and the closest defender to the left resp. right moving towards C_j being d_j^l resp. d_j^r distance away from c_j^l resp. c_j^r having shortened their step σ_j^l resp. σ_j^r times. For notational ease, we omit the latter parameters in the function definition.

$L(S_j) :=$ The number of saved vertices in the sub-path S_j given p_j defenders, the closest defender to the left of C_j moving towards C_j being d_j^l distance away from c_j^l having shortened their step σ_j^l times.

Remarks on dominated cases

We can rule out a priori some cases of parameter combinations or decisions that are clearly inferior:

- If we place two defenders in a component C_j, then they should be at opposite ends of the component and move away from it, i.e. the defender on the left (right) moves left (right). Any other combination is inferior to that.
- If a defender is placed on a threshold vertex of C_j moving towards the adjacent burning fire, e.g. on $f_j^l + 1$ moving left, then it will never be worse for the defender not to shorten their step in C_j.
- For the calculations of the possible values of σ_j^r in the tables below, if $x_j^r \in \{0, >\}$ we only consider the possibility of the nearest defender to the right which moves into C_j shortening in a given component if shortening in that component was not dominated by not shortening in it. This is because increasing how much a defender has shortened its step can never lead to less burning in a later component, as long as the defender still has the option of shortening its step in that component. For the calculations of the possible values of σ_j^l in the table, we cannot say how much shortening would lead to less burning in the components to the left of C_j, so we consider all values of σ_j^l from 0 to the total number of components between v_j^l and C_j (including the component the defender starts in, and excluding C_j).

We shall now calculate the optimal defence strategies for the path with 19 vertices; if initial fires start on vertices $2, 15$ and 19 (as shown in Figure 5.28), with a defence budget of two and $l_d = 2$.

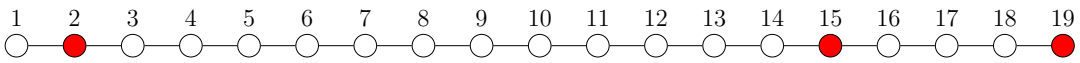


Figure 5.28: The example game we shall calculate optimal defence for, if $l_d = 2$ and $d = 2$. Vertex coordinates are shown above the vertices.

We shall start by calculating the possible starting points for the defenders using Algorithm 21. The results are shown in Table 5.9.

The tables for the example are given in Tables 5.10–5.13 on pages 146–149. We will often have the situation that multiple values for a parameter or decision variable result in exactly the same outcome. We collate these into one case by writing a, b. For consecutive values, we write $a+$ (all integer values between a and n) or $a\infty$ ($a+$ plus ∞). The final column marked 'Def's' gives an example of a defence strategy with that state, The notation $1-14, 18_2^l$ means that one defender is placed on any vertex in Table 5.9 between 1 and 14, moves in any direction and can shorten any appropriate number of times, as described above. The second defender starts on Vertex 18, moves left (^{l}) and shortens its step a total of two times moving into C_j, including inside C_j.

Throughout the example, we have the values $l_d = 2$, $l_1 = 1$, $l_2 = 12$, $l_3 = 3$, $f_1^l =$ undefined, $f_1^r = 2$, $f_2^l = 2$, $f_2^r = 15$, $f_3^l = 15$ and $f_3^r = 19$.

For our dynamic program, we begin with the right-most component C_3 and then move left until we reach C_1. The calculations for C_3 are given in Table 5.10. For $p_3 = 0$, even a defender placed on Vertex 14 moving right and not shortening their step is not going to be able to save any vertex in C_3. Hence, every value for d_j^l gives the same results. In addition, there can be no defender to the right of C_3, i.e. $d_j^r = \infty$, and δ_j^r must be 0, as $p_3 = 0$.

For $p_3 = 1$, again no defender placed to the left of C_3 can save a vertex in C_3, irrespective of where we place a defender in C_3. For $\delta_j^r = 0$, we must have a defender in C_3 moving right who may or may not shorten their step. For $\delta_j^r \neq 0$, Algorithm 21 returns just one vertex each on either side of C_3. moreover, the only reasonable placements are $x_j^l =<$ and $x_j^r =>$.

For $p_3 = 2$, the cases $\delta_j^r = 0$ resp. $\delta_j^r = l^1$ have been omitted, as the two defenders would both be moving right resp. the defender on the left moving towards the adjacent fire should not shorten its step in the component.

Next, we move to component C_2, the calculations for which are shown in Tables 5.11 – 5.13. We start by observing that the case $p_2 = 0$ is not possible, as C_1 has just one vertex. Thus, we begin with the case $p_2 = 1$, i.e. a defender must be placed on vertex 1, and the resulting calculations are given in Table 5.11. The case $d_j^l = \infty$ is dominated by $d_j^l = 2$ as the former corresponds to the defender on vertex 1 moving left. Moreover, for $d_j^l = 2$ the case $\sigma_j^l = 1$ is dominated by $\sigma_j^l = 0$, as shortening in C_1 does not save any additional vertex in C_1. Thus, we omit the columns for p_j, d_j^l, and $\sigma_j^l = 0$ in Table 5.11.

After having cleared the case $p_2 = 1$ in Table 5.11, we move to $p_2 = 2$, i.e. no defender can be placed on Vertex 1. Hence, we must have $d_j^l = \infty$ and $\sigma_j^l = -$, so we again omit the three columns in the Table 5.12.

Input: A list of fire components for an instance of FIREFIGHTER on a path P; the set of burning vertices F; possible starting positions and directions for the defender

Output: A vertex to start the defender on; a direction for the defender to travel in and the component in which it should shorten its step (if any).

set *best weight* to zero;
create an empty list *best strategy*;
for *all possible (vertex,dir) triples* **do**
  find the starting point v using Algorithm 15, assuming $s = false$;
  set *tally* $= 0$, $\sigma = 0$, *shortening comp* $=$ *null*;
  initialise a list C' of components in front of (v, dir) in order starting with the furthest and ending with the component containing v;
  for *component* C_i *in* C' **do**
    calculate $w(v, dir, C_i, \sigma)$;
    if C_i *has two fires and* $l_i - 2g(v, dir, C_i, \sigma) \equiv 0 \mod 3$ *and* $w(v, dir, C_i, \sigma) > 0$ **then**
      `we might have to shorten the defender's step;`
      if *tally* $= 0$ **then**
        `we do have to shorten the defender's step;`
        $\sigma = 1$;
        shortening comp$= C_i$;
      else
        `we should not shorten the defender's step, so we need to adjust the weight for` C_i;
        subtract one from *tally*;
      end
    end
    if $C_i = C_1$ *and tally* $= 0$ **then**
      recalculate v using using Algorithm 15, assuming $s = true$;
      calculate $w(v, dir, C_i, \sigma)$;
      if C_1 *has two fires and* $l_1 \equiv 0 \mod 3$ *and* $w(v, dir, C_1, \sigma) > 0$ **then**
        `we do have to shorten the defender's step;`
        set $\sigma = 1$ and *shortening comp* $= C_1$;
      end
    end
    add $w(v, dir, C_i, \sigma)$ to *tally*;
  end
  if *tally* $>$ best weight **then**
    set *best weight* equal to *tally*;
    set *best strategy* equal to (v, dir),*shortening comp*
  end
end
return *best strategy*

Algorithm 16: A polynomial time algorithm for finding best defence with a single defender on a path, which is limited to travelling at most distance two per time step.

Input: A vertex v to start the defender on; a direction *dir* for it to travel in and list of at most one component *component to shorten* it should shorten its step in for optimal defence, calculated using Algorithm 16.
Output: An optimal defence strategy.
place the defender at the vertex v at time step 1;
for *defender move* **do**
  if *the defender is in a component in* component to shorten *and is behind a susceptible vertex which is behind a burning vertex* **then**
    move the defender forwards one vertex;
  else
    move the defender forwards two vertices;
  end
end

Algorithm 17: An algorithm for optimal defence with with a single defender on a path, which is limited to travelling at most distance two per time step. This algorithm uses the output of Algorithm 16 as its input.

Input: C_j, l_j, x_j^l, x_j^r, v_j^l, v_j^r, σ_j^l, σ_j^r as defined above.
Output: The number of vertices that can be saved $L(C_j)$, if C_j only contains one fire
if C_j *is the rightmost component* **then**
  if $x_j^l \neq 0$ **then**
    $L(C_j) = l_j$
  else
    $L(C_j) = \max\{0, l_j - g(v_j^l, right, j, \sigma_j^l)\}$
  end
else
  if $x_j^r \neq 0$ **then**
    $L(C_j) = l_j$
  else
    $L(C_j) = \max\{0, l_j - g(v_j^r, left, j, \sigma_j^r)\}$
  end
end

Algorithm 18: Algorithm for finding number of vertices saved in a component C_j, for $l_d = 2$, $d \geq 1$ if there is only one fire in C_j.

Input: l_j, C_j, x_j^l, x_j^r f_j^r, f_j^l, v_j^l, v_j^r, σ_j^l, σ_j^r, s_j^l, s_j^r as defined above
Output: The number of vertices that can be saved $L(C_j)$ if $x_j^r = 0 = x_j^l$
if $v_j^l = null = v_j^r$ **then**
  $L(C_j) = 0$
else if $v_j^l = null$ **then**
  $L(C_j) = \max\{0, f(l_j - 2g(v_j^r, left, j, \sigma_j^r), 2, s_j^r)\}$
else if $v_j^r = null$ **then**
  $L(C_j) = \max\{0, f(l_j - 2g(v_j^l, right, j, \sigma_j^l), 2, s_j^l)\}$
else
  $Midpoint = \frac{v_j^r + v_j^l - \sigma_j^l + \sigma_j^r}{2}$;
  if $f_j^r < Midpoint$ **then**
    $L(C_j) = \max\{0, f(l_j - 2g(v_j^l, right, j, \sigma_j^l), 2, s_j^l)\}$
  else if $Midpoint < f_j^l$ **then**
    $L(C_j) = \max\{0, f(l_j - 2g(v_j^r, left, j, \sigma_j^r), 2, s_j^r)\}$
  else
    $L(C_j) = \max\{0, f(l_j - 2g(v_j^l, right, j, \sigma_j^l), 2, s_j^l), f(l_j - 2g(v_j^r, left, j, \sigma_j^r), 2, s_j^r), l_j - g(v_j^l, right, j, \sigma_j^l) - g(v_j^r, left, j, \sigma_j^r)\}$
  end
end

Algorithm 19: Algorithm for finding number of vertices saved in a component C_j, for $l_d = 2$, $d \geq 1$ if no defenders are initially placed in C_j.

Input: l_j, C_j, x_j^l, x_j^r f_j^r, f_j^l, v_j^l, v_j^r, σ_j^l, σ_j^r, s_j^l, s_j^r as defined above.
Output: The number of vertices that can be saved $L(C_j)$
if $x_j^l \neq 0$ *and* $x_j^r \neq 0$ **then**
  $L(C_j) = l_j$;
end
if $x_j^l =<$ *and* $x_j^r = 0$ **then**
  if $d(v_j^l, f_j^l) + \sigma_j^l \equiv 0 \mod 2$ **then**
    $\alpha = 1 + f\left(l_j - 1 - \frac{d(v_j^l, f_j^l+2)+\sigma_j^l}{2}, 2, s_j^l, \right)$
  else if $l_j < 2 + \frac{d(v_j^l, f_j^l+3)+\sigma_j^l}{2}$ **then**
    $\alpha = 1$
  else if $l_j = 2 + \frac{d(v_j^l, f_j^l+3)+\sigma_j^l}{2}$ *and* $s_j^l = False$ **then**
    $\alpha = 1$
  else if $l_j = 2 + \frac{d(v_j^l, f_j^l+3)+\sigma_j^l}{2}$ *and* $s_j^l = True$ **then**
    $\alpha = 2$
  else
    $\alpha = 2 + f\left(l_j - 2 - \frac{d(v_j^l, f_j^l+3)+\sigma_j^l}{2}, 2, s_j^l, \right)$
  end
  $L(C_j) = \max\{\alpha, 1, l_j - g(v_j^r, left, j, \sigma_j^r)\}$;
end
if $x_j^l =>$ *and* $x_j^r = 0$ **then**
  if $f_j^l < \frac{v_j^l + v_j^r - s_j^l + \sigma_j^r}{2} < f_j^r$ **then**
    $L(C_j) = \max\{f(l_j, 2, s_j^l), l_j - g(v_j^r, left, j, \sigma_j^r)\}$
  else
    $L(C_j) = f(l_j, 2, s_j^l)$
  end
end

Algorithm 20: Algorithm for finding number of vertices saved in a component C_j, for $l_d = 2$, $d \geq 1$ if there are two fires in the component one defender is initially placed in C_j on the leftmost susceptible vertex.

Input: l_d, l_j, f_j^l, x_j^l, s_j^l, f_j^r, x_j^r, s_j^r, all as defined above.
Output: Vertex coordinates for the defenders to start on.
if $x_j^l \in \{<, >\}$ *and* $x_j^r \in \{<, >\}$ **then**
  Return $v_j^l = f_j^l + 1$, $v_j^r = f_j^r - 1$;
else if $f_j^r = null$ **then**
  Return $v_j^l = f_j^l + 1$;
else if $f_j^l = null$ **then**
  Return $v_j^r = f_j^r - 1$;
else if $x_j^l =<$ **then**
  Return $v_j^l = f_j^l + 1$;
else if $x_j^r =>$ **then**
  Return $v_j^r = f_j^r - 1$;
else if $x_j^l =>$ **then**
  if $l_j \leq l_d + 1$ **then**
    Return $v_j^l = f_j^l + 1$
  else if $s_j^l = false$ **then**
    Define a to be the unique integer s.t $l_j = k \cdot (l_d + 1) + a$ with $0 < a \leq l_d + 1$, $0 \leq k$, an integer.;
    Return $v_j^l \in \{f_j^l + \min\{a, l_j\}, f_j^l + 1\}$;
  else
    if $l_j \equiv 2 \mod (l_d + 1)$ **then**
      Return $v_j^l \in \{f_j^l + 2, f_j^l + 1\}$;
    else
      Return $v_j^l = f_j^l + 1$;
    end
  end
else
  if $l_j \leq l_d + 1$ **then**
    Return $v_j^r = f_j^r - 1$
  else if $s = false$ **then**
    Define a to be the unique integer s.t $l_j = k \cdot (l_d + 1) + a$ with $0 < a \leq l_d + 1$, $0 \leq k$, an integer;
    Return $v_j^r \in \{f_j^r - \min\{a, l_j\}, f_j^r - 1\}$;
  else
    if $l_j \equiv 2 \mod (l_d + 1)$ **then**
      Return $v_j^r \in \{f_j^r - 2, f_j^r - 1\}$;
    else
      Return $v_j^r = f_j^r - 1$;
    end
  end
end

Algorithm 21: Algorithm for finding the starting points for defenders if $l_d \geq 2$ and $d = 1$.

Table 5.9: The starting points for defenders in the game shown in Figure 5.28, calculated using Algorithm 21. If x_j^l, s_j^l, x_j^r or s_j^l for some $j \in [1,3]$ are not stated, then their value does not affect the output. The other inputs are $l_d = 2$, $l_1 = 1$, $l_2 = 12$, $l_3 = 3$, $f_1^l = null$, $f_1^r = 2$, $f_2^l = 2$, $f_2^r = 15$, $f_3^l = 15$ and $f_3^r = 19$.

Input	Output
$x_1^r \in \{<, >\}$	$v_1^r = 1$
$x_2^l \in \{<, >\}$ and $x_2^r \in \{<, >\}$	$v_2^l = 3, d_2^r = 14$
$x_2^l =<$	$v_2^l = 3$
$x_2^l =>, x_2^r = -, s_2^l = true$	$v_2^l = 3$
$x_2^l =>, x_2^r = -, s_2^l = false$	$v_2^l \in \{3, 5\}$
$x_2^r =>$	$v_2^r = 14$
$x_2^r =<, x_2^l = -, s_2^r = true$	$v_2^r = 14$
$x_2^r =<, x_2^l = -, s_2^r = false$	$v_2^r \in \{12, 14\}$
$x_3^l \in \{<, >\}$	$v_3^l = 16$
$x_3^r \in \{<, >\}$	$v_3^r = 18$

Table 5.10: $j = 3$

p_j	d_j^l	σ_j^l	δ_j^r	d_j^r	σ_j^r	x_j^l	s_j^l	x_j^r	s_r^l	$L(C_j)$	$L(P_j)$	Def's
0	2∞	$0+$	0	∞	$-$	0	$0, 1$	0	$-$	0	0	$1-14, 1-14$
1	2∞	$0+$	0	∞	$-$	$>$	0	0	$-$	1	1	$1-14, 16_0^r$
						$>$	1	0	$-$	2	2	$1-14, 16_1^r$
						0	$0, 1$	$>$	0	1	1	$1-14, 18_0^r$
			l^0	∞	$-$	$<$	0	0	$-$	1	1	$1-14, 16_0^l$
			r^0	∞	$-$	0	$0, 1$	$<$	0	1	1	$1-14, 18_0^l$
			r^1	∞	$-$	0	$0, 1$	$<$	1	2	2	$1-14, 18_1^l$
2	∞	$-$	0	∞	$-$	$>$	0	$>$	0	1	1	$16_0^r, 18_0^r$
						$>$	1	$>$	0	2	2	$16_1^r, 18_0^r$
2	∞	$-$	l^0	∞	$-$	$<$	0	$>$	0	3	3	$16_0^l, 18_0^r$

Table 5.11: $j = 2$: $p_j = 1$, $d_j^l = 2$ and $\sigma_j^l = 0$

δ_j^r	d_j^r	σ_j^r	x_j^l	s_j^l	x_j^r	s_j^r	$L(C_j)$	p_{j+1}	d_{j+1}^l	σ_{j+1}^l	δ_{j+1}^r	d_{j+1}^r	σ_{j+1}^r	$L(P_{j+1})$	$L(P_j)$	Def's
0	2	0	0	0	0	0	8	1	15	0	l^0	∞	—	1	9	$1_0^r, 16_0^l$
			0	1	0	0	7	1	15	1	l^0	∞	—	1	8	$1_1^r, 16_0^l$
			0	0	0	1	7	1	15	0	l^0	∞	—	1	8	$1_0^r, 16_1^l$
			0	1	0	1	6	1	15	1	l^0	∞	—	1	7	$1_1^r, 16_1^l$
	4	0	0	0	0	0	6	1	15	0	r^0	∞	—	1	7	$1_0^r, 18_0^r$
			0	1	0	0	5	1	15	1	r^0	∞	—	1	6	$1_1^r, 18_0^l$
			0	0	0	1	5	1	15	0	r^0	∞	—	1	6	$1_0^r, 18_1^l$
			0	1	0	1	4	1	15	1	r^0	∞	—	1	5	$1_1^r, 18_1^l$
	1		0	0	0	0	5	1	15	0	r^1	∞	—	2	7	$1_0^r, 18_1^r$
			0	1	0	0	4	1	15	1	r^1	∞	—	1	6	$1_1^r, 18_1^l$
			0	0	0	1	4	1	15	0	r^1	∞	—	1	6	$1_0^r, 18_2^l$
			0	1	0	1	3	1	15	1	r^1	∞	—	1	5	$1_1^r, 18_2^l$
	∞	—	$>$	0	0	—	7	0	13	0	0	∞	—	0	7	$1_0^r, 3_0^r$
			$>$	1	0	—	8	0	13	1	0	∞	—	0	8	$1_0^r, 3_1^r$
			$\gg$	0	0	—	7	0	11	0	0	∞	—	0	7	$1_0^r, 5_0^r$
			0	0	$>$	0	10	0	2	0	0	∞	—	0	10	$1_0^r, 14_0^r$
			0	1	$>$	0	9	0	2	0	0	∞	—	0	9	$1_1^r, 14_0^r$
l^0	∞	—	$<$	0	0	—	7	0	15	0	0	∞	—	0	7	$1_0^r, 3_0^l$
r^0	∞	—	0	0	$<$	0	10	0	15	0	0	∞	—	0	10	$1_0^r, 14_0^l$
			0	1	$<$	0	9	0	15	1	0	∞	—	0	9	$1_0^r, 14_1^l$
r^1			0	0	$<$	1	10	0	15	0	0	∞	—	0	10	$1_1^r, 14_0^l$
			0	1	$<$	1	9	0	15	1	0	∞	—	0	9	$1_1^r, 14_0^l$
rr^0	∞	—	0	0	$\lll$	0	8	0	15	0	0	∞	—	0	8	$1_0^r, 12_0^l$
			0	1	$\lll$	0	7	0	15	1	0	∞	—	0	7	$1_1^r, 12_0^l$

Table 5.12: $j = 2$: $p_j = 2$, $d_j^l = \infty$ and $\sigma_j^l = -$, Part 1

δ_j^r	d_j^r	σ_j^r	x_j^l	s_j^l	x_j^r	s_j^r	$L(C_j)$	p_{j+1}	d_{j+1}^l	σ_{j+1}^l	δ_{j+1}^r	d_{j+1}^r	σ_{j+1}^r	$L(P_{j+1})$	$L(P_j)$	Def's
0	2	0	0	$-$	$>$	0	7	1	2	0	l^0	∞	$-$	1	8	$14_0^r, 16_0^l$
			0	$-$	$>$	1	6	1	2	0	l^0	∞	$-$	1	7	$14_0^r, 16_1^l$
			$>$	0	0	0	10	1	13	0	l^0	∞	$-$	1	11	$3_0^r, 16_0^l$
			$>$	0	0	1	9	1	13	0	l^0	∞	$-$	1	10	$3_0^r, 16_1^l$
			$>$	1	0	0	10	1	13	1	l^0	∞	$-$	1	11	$3_1^r, 16_0^l$
			$>$	1	0	1	9	1	13	1	l^0	∞	$-$	1	10	$3_1^r, 16_1^l$
			$\gg$	0	0	0	8	1	11	0	l^0	∞	$-$	1	9	$5_0^r, 16_0^l$
			$\gg$	0	0	1	7	1	11	0	l^0	∞	$-$	1	8	$5_0^r, 16_1^l$
4	0		0	$-$	$>$	0	7	1	2	0	r^0	∞	$-$	1	8	$14_0^r, 18_0^l$
			0	$-$	$>$	1	6	1	2	0	r^0	∞	$-$	1	7	$14_0^r, 18_1^l$
			$>$	0	0	0	8	1	13	0	r^0	∞	$-$	1	9	$3_0^r, 18_0^l$
			$>$	0	0	1	7	1	13	0	r^0	∞	$-$	1	8	$3_0^r, 18_1^l$
			$>$	1	0	0,1	8	1	13	1	r^0	∞	$-$	1	9	$3_1^r, 18_{0,1}^l$
			$\gg$	0	0	0,1	7	1	11	0	r^0	∞	$-$	1	8	$5_0^r, 18_{0,1}^l$
1	0		0	$-$	$>$	0	6	1	2	0	r^1	∞	$-$	2	8	$14_0^r, 18_1^l$
			0	$-$	$>$	1	5	1	2	0	r^1	∞	$-$	2	7	$14_0^r, 18_2^l$
			$>$	0	0	0,1	7	1	13	0	r^1	∞	$-$	2	9	$3_0^r, 18_{1,2}^l$
			$>$	1	0	0,1	8	1	13	1	r^1	∞	$-$	2	10	$3_1^r, 18_{0,1}^l$
			$\gg$	0	0	0,1	7	1	11	0	r^1	∞	$-$	2	9	$5_0^r, 18_{0,1}^l$
∞	$-$		0	$-$	0	$-$	0	2	∞	$-$	0	∞	$-$	2	2	$16_1^r, 18_0^r$
			0	$-$	$>$	0	1	1	2	0	0	∞	$-$	2	3	$14_0^r, 16_1^r$
			$>$	0	0	$-$	7	1	13	0	0	∞	$-$	2	9	$3_0^r, 16_1^r$
			$>$	1	0	$-$	8	1	13	1	0	∞	$-$	2	10	$3_1^r, 16_1^r$
			$\gg$	0	0	$-$	7	1	11	0	0	∞	$-$	2	9	$5_0^r, 16_1^r$
			$>$	0,1	$>$	0	12	0	2	0	0	∞	$-$	0	12	$3_{0,1}^r, 14_0^r$
			$\gg$	0	$>$	0	10	0	2	0	0	∞	$-$	0	10	$5_0^r, 14_0^r$

Table 5.13: $j = 2$: $p_j = 2$, $d_j^l = \infty$ and $\sigma_j^l = -$, Part 2

δ_j^r	d_j^r	σ_j^r	x_j^l	s_j^l	x_j^r	s_j^r	$L(C_j)$	p_{j+1}	d_{j+1}^l	σ_{j+1}^l	δ_{j+1}^r	d_{j+1}^r	σ_{j+1}^r	$L(P_{j+1})$	$L(P_j)$	Def's
l^0	2	0	$<$	0	0	0	10	1	∞	$-$	l^0	∞	$-$	1	11	$3_0^l, 16_0^l$
			$<$	0	0	1	9	1	∞	$-$	l^0	∞	$-$	1	10	$3_0^l, 16_1^l$
	4	0	$<$	0	0	0	8	1	∞	$-$	r^0	∞	$-$	1	9	$3_0^l, 18_0^l$
			$<$	0	0	1	7	1	∞	$-$	r^0	∞	$-$	1	8	$3_0^l, 18_1^l$
		1	$<$	0	0	0	7	1	∞	$-$	r^1	∞	$-$	2	9	$3_0^l, 18_1^l$
			$<$	0	0	1	6	1	∞	$-$	r^1	∞	$-$	2	8	$3_0^l, 18_2^l$
	∞	$-$	$<$	0	0	$-$	1	1	∞	$-$	0	∞	$-$	2	3	$3_0^l, 16_1^l$
			$<$	0	$<$	0,1	12	0	∞	$-$	0	∞	$-$	0	12	$3_0^l, 14_{0,1}^l$
			$<$	0	$>$	0,1	12	0	2	0,1	0	∞	$-$	0	12	$3_0^l, 14_{0,1}^r$
r^0	2	0	0	$-$	$<$	0	7	1	∞	$-$	l^0	∞	$-$	1	8	$14_0^l, 16_0^l$
	4	0	0	$-$	$<$	0	7	1	∞	$-$	r^0	∞	$-$	1	8	$14_0^l, 18_0^l$
		1	0	$-$	$<$	0	7	1	∞	$-$	r^1	∞	$-$	2	9	$14_0^l, 18_1^l$
	∞	$-$	0	$-$	$<$	0	7	1	∞	$-$	0	∞	$-$	2	9	$14_0^l, 16_1^r$
r^1	2	0	0	$-$	$<$	1	8	1	∞	$-$	l^0	∞	$-$	1	9	$14_1^l, 16_0^l$
	4	0	0	$-$	$<$	1	8	1	∞	$-$	r^0	∞	$-$	1	9	$14_1^l, 18_0^l$
		1	0	$-$	$<$	1	8	1	∞	$-$	r^1	∞	$-$	2	10	$14_1^l, 18_1^l$
	∞	$-$	0	$-$	$<$	1	8	1	∞	$-$	0	∞	$-$	2	10	$14_1^l, 16_1^r$
rr^0	2	0	0	$-$	$\lll$	0	7	1	∞	$-$	l^0	∞	$-$	1	8	$12_0^l, 16_0^l$
	4	0	0	$-$	$\lll$	0	7	1	∞	$-$	r^0	∞	$-$	1	8	$12_0^l, 18_0^l$
		1	0	$-$	$\lll$	0	7	1	∞	$-$	r^1	∞	$-$	2	9	$12_0^l, 18_1^l$
	∞	$-$	0	$-$	$\lll$	0	7	1	∞	$-$	0	∞	$-$	2	9	$12_0^l, 16_1^r$

Table 5.13 completes the calculations for $j = 2$. Finally, for $j = 1$ there are only two relevant cases: we put a defender on Vertex 1 or not. Starting with the former case, we can skip the case of the defender moving right, as moving left will save at least as many vertices. With $x_1^r =>$ and $s_1^r = 0$, we obtain for $j = 2$ the parameter values: $p_2 = 1$, $d_2^l = 2$, and $\sigma_2^l = 0$. For δ_2^r, all relevant values are possible. Consulting Table 5.11, the maximal number of saved vertices in P_2 is 10 (by putting the second defender on Vertex 14), hence we save a total of 11 vertices. Now looking at the latter case, i.e., $p_2 = 2$, $d_2^l = \infty$, and $\sigma_2^l = -$, from Tables 5.12 and 5.13 we can see that there a several different strategies that can save a total of 12 vertices in P_2, which is then also the number of saved vertices for P. All of them put defenders on Vertices 3 and 14.

Theorem 80. *Assuming that there exists an optimal defence strategy where no defender turns and the starting points for each of the defenders are returned by Algorithm 21 with the appropriate inputs, then the above dynamic program returns an optimum defence strategy for a game on a path with any number of fires, any number of defenders and $l_d = 2$. Furthermore it does this in polynomial time in f and d.*

Proof. The correctness of the dynamic program follows from the above discussion, so we focus on the time analysis.

For each of the at most $2f$ components, we have to consider at most $3 \cdot 2 \cdot 3 \cdot 2 = 36$ possible combinations for the decision variable x_j^l, s_j^l, x_j^r and s_j^r respectively (we also have to calculate the starting points using Algorithm 21, but this is constant time for each combination of parameters). We can have used at most d defenders to the left of p_j. We also have to consider the possible values of d_j^l, σ_j^l, d_j^r and σ_j^r. These variables can have at most $2f$ different values. Finally δ_j^r can have three different values. This means considering at most $2f \cdot 36 \cdot d \cdot 2f \cdot 2f \cdot 2f \cdot 2f \cdot 3 = 3456 \cdot d \cdot f^5$ different component, variable combinations. For each of these, we calculate $L(C_j)$ using Algorithms 18 – 20, which are constant time. Hence the final time analysis is given by $\mathcal{O}(d \cdot f^5)$ □

Chapter 6

Conclusions and Directions for Future Research

6.1 Conclusions

In Chapter 2 we found new $K(n, f)$-optimal graphs and showed that some of them are also *mmd*, using the following new lemma about how the two graph classes relate:

Lemma (Lemma 5 of Section 2.4)**.** *For $K(n, f) \leq 2f + d$ all $K(n, f)$ – optimal graphs are $mmd(n)$–graphs.*

We also presented the following new theories about the structure of *mmd* and $K(n, f)$-optimal graphs:

Lemma (Lemma 7 of Section 2.5)**.** *For any $n, f \in \mathbb{N}$ all spanning subgraphs of a $K(n, f)$–optimal graph are themselves $K(n, f)$–optimal.*

Lemma (Lemma 8 of Section 2.5)**.** *For any $n, f \in \mathbb{N}$, if all spanning subtrees of a graph G are $K(n, f)$–optimal then G is $K(n, f)$–optimal.*

Lemma (Lemma 9 of Section 2.5)**.** *For any $n, f, d \in \mathbb{N}$ all spanning subgraphs of an $mmd(n, f, d)$ graph G are themselves $mmd(n, f, d)$.*

The main result of Chapter 2 came from our work on fire path equivalent instances of the Firefighter problem. We provided a polynomial time algorithm to find optimal defence:

Theorem (Theorem 16 of Section 2.6)**.** *Let $S = (G = (V, E), F, D)$ be fire path equivalent. Then the polynomial time Algorithm 3 (used together with Algorithm 2) will give an optimal defence strategy.*

We later adapted this for the heuristics in Chapter 4 and reused it with some modifications for Distance-Limited Firefighter games on paths with a distance limit of 1 in Chapter 5.

Finally, in Chapter 2, we provided future researchers with some more pointers to continue to search for more *mmd* graphs:

Lemma (Lemma 18 of Section 2.7)**.** *If any graph for which there exists an M-set which includes all high degree vertices is mmd then all mmd graphs with fewer than f high degree vertices are optimal fire path equivalent graphs.*

In Chapter 3 we introduced Edge-Defence Firefighter (EDF). The quirk of EDF that defending an edge containing a vertex does not necessarily protect that vertex for the remainder of the game means that there will always be at least as much burning as in the original version of the game. However, we have shown that certain minimal bounds on survival rates remain the same - adapting work on Classic Firefighter for Fractional Online Firefighting on trees; and on planar graphs with large girth. We showed Edge-Defence Firefighter can be solved in the same amount of time as Classic Firefighter for P_k-free graphs. Surprisingly, we also found that a single fire can be contained on the infinite hexagonal grid using the same number of edge defenders as the lowest known bound for the number of vertex defenders it takes to contain the fire. This was especially notable since Edge-Defence Firefighter has a much worse survival rate than Classic Firefighter for single defenders on a square grid.

We calculated the best possible defence strategy for edge defence on a finite square grid; assuming first that we 'defend in a straight line' - i.e. all defended edges share either x or y-coordinates; then assuming that the defenders save a rectangular shape in a corner of the grid. We then compared the two strategies, established when one out performs the other and derived an expected survival rate:

Theorem (From Section 3.6)**.** *The edge survival rate for the finite square grid $P_k \square P_k$ with a single fire and a single defender tends to at least one half as k tends to infinity.*

For infinite square grids, it does not seem possible either to contain the fire or to save a constant positive fraction of vertices distance t away from the fire, if we consider all time steps. We developed a new metric, analysing what fraction of the vertices we could save if a 'snapshot' of the game was taken at a known time step. We then developed an algorithm for defence and a function $h(t)$ which gives the number of vertices saved by this algorithm. We derived the following limit for the proportion of vertices saved using our algorithm as t tends towards infinity:

Theorem (From Section 3.7)**.** *As t tends towards infinity, at least $\frac{1}{24}$ of the vertices distance t away from the fire on the infinite square grid can be saved.*

In Chapter 4, we first developed an integer program for EDF, then optimised the run times for it to be solved to optimality using valid inequalities. The most important of these focused on the following new results about T^{opt}, the number of time steps the game takes to finish, given optimal defence. The first new upper bound was given by Equation (4.13) in Section 4.1:

$$T^{opt} \leq T^2 = \left\lceil \frac{m}{d+1} \right\rceil .$$

The other new upper bounds were given in the following lemmas, rephrased from above:

Lemma 81 (Lemma 60 of Section 4.1)**.** *Define:*

$$c(v_i) \;=\; \begin{cases} d(v_i), & v_i \in V \setminus F, \\ 0, & v_i \in F. \end{cases}$$

and consider a permutation σ of $\{1, \ldots, n\}$, such that, $c(v_{\sigma(1)}) \geq c(v_{\sigma(2)}) \geq \ldots \geq c(v_{\sigma(n)})$. Let r be the largest integer such that $\sum_{i=1}^{r} c(v_{\sigma(i)}) \leq d$. Then, for the Edge-Defence Firefighter, we have $T^{opt} \leq T^3 = \left\lceil \frac{n-f}{r+1} \right\rceil$.

Lemma (Lemma 61 of Section 4.1)**.** *For Edge-Defence Firefighter, if we relabel the vertices such that their degree is non-decreasing, i.e., $d(v'_1) \leq d(v'_2) \leq \ldots \leq d(v'_n)$, we have $T^{opt} \leq T^4$ where*

$$T^4 \;=\; \min \left\{ t \mid \sum_{r=t+f}^{n} d(v'_r) \leq t \cdot d \right\}.$$

Finally, we presented a result on the upper bounds of T on trees:

Lemma (Lemma 64 of Section 4.1)**.** *There does not exist a tree G with $T^{opt} > 1$ such that $T^{opt}(G, f, 1) = \left\lceil \frac{n-f}{2} \right\rceil$ for Classic or Edge-Defence Firefighter.*

The improved upper bounds for T^{opt} lead to significant reductions in the computational times, as can be seen in Tables (4.1) and (4.2) in Section 4.1 and in agreement with the published literature on other variants of the Firefighter Game. We then developed five new heuristics, based on the fpe algorithm from Chapter 2. We ran computational experiments to find the best centrality measure to use with these heuristics; which heuristic performed best; and how the best-performing heuristic compared to the optimal solution.

The best-performing heuristic sometimes produced an optimal defence strategy, and was generally better for instances with more initially burning vertices All heuristics were significantly faster than the IP.

In Chapter 5, we introduced Cost-Value Firefighter (CVF) and Distance-Limited Firefighter (DLF). We presented mixed integer programs for both, which we again optimised the run times of using valid inequalities, again with a focus on the maximum number of time steps for the game to finish, given optimal defence. For CVF, many of the upper bounds derived for Classic or Edge-Defence Firefighter could be adapted, but the quirk of DLF that it may not be optimal - or even possible - to use the full defence budget in every time step meant that many of these upper bounds on the number of times steps were not applicable. We did manage to derive the following upper bound for games on trees:

Lemma (Lemma 70 of Section 5.3)**.** *For Distance-Limited Firefighter, on a tree G:*

$$T^{opt}(G, 1, d) \leq \left\lceil \frac{n-1}{2} \right\rceil.$$

Our computational experiments on DLF demonstrated that finding optimal solutions to DLF tends to take longer for lower distance limits, so we then focused

on DLF on paths, with low distance limits. We adapted the fpe algorithm from Chapter 2 to provide a polynomial time algorithm for best defence on a path with any number of starting fires, any number of defenders and a distance limit for the defenders of one.

Theorem (Theorem 74 of Section 5.4)**.** *Algorithm 14 is a polynomial-time algorithm for finding the optimal defence on a single path with $l_d = 1$ and any number of defenders and any number of fires.*

We also provided a polynomial time algorithm for optimal defence with any number of starting fires and a single defender with a distance limit of two.

Theorem (Theorem 78 of Section 5.4)**.** *Algorithms 16 and 17 together give an algorithm for optimal defence with one defender which can travel at most distance two per time step, defending on a path, which runs in $\mathcal{O}(f^2)$.*

Finally, for games on paths with multiple starting fires and multiple defenders with a distance limit of two, we provided a dynamic program to find a defence strategy which is optimal, given certain assumptions.

Theorem (Theorem 80 of Section 5.4)**.** *Assuming that there exists an optimal defence strategy where no defender turns and the starting points for each of the defenders are returned by Algorithm 21 with the appropriate inputs, then the above dynamic program returns an optimum defence strategy for a game on a path with any number of fires, any number of defenders and $l_d = 2$. Furthermore it does this in polynomial time in f and d.*

6.2 Directions for Future Research

It is clear from both the results of García-Martínez et al. [24] and Ramos et al. [47]; and from our own results in Chapters 4 and 5 that the most important step in optimising the run times for mixed integer programs to find exact solutions for variations of the Firefighter Problem is in finding better upper bounds for T^{opt}, the number of time steps it takes for the game to finish with optimal defence. Our computational results for Edge-Defence, Cost-Value and Distance-Limited Firefighter agreed with those of García-Martínez et al. [24] for Classic Firefighter that the current upper bounds discussed earlier in the thesis are often much larger than the actual value.

I believe that if variants of the Firefighter Problem are to have a practical use in advising policy during a disease outbreak then it is essential that an efficient way to calculate better upper bounds is found, so optimal solutions can be found in a practical amount of time. I furthermore propose that this could pose some interesting results, regardless of the real life usage.

A bilevel optimisation approach could prove useful for finding *mmd* graphs, especially with lower upper bounds for T to improve run times. In Chapter 2, we provided various conjectures that future researches may be able to use to complete a categorisation of *mmd* graphs. These include proving that supernova graphs are optimal fire path equivalent graphs; that connected graphs with strictly more

that f vertices with degree strictly greater than two are not *mmd* for any values of n or d; that for connected graphs with exactly f vertices of degree strictly greater than two that the M-set of size f consists of exactly these vertices. We also leave the open question of classifying all optimal fire path equivalent paths, which could also provide a lead in classifying all *mmd* graphs.

For the edge defence on the infinite square grid, we leave the following conjectures as open questions to prove or disprove:

Conjecture 82. *It is not possible to contain a single fire breaking out on the infinite square grid with a single edge defender.*

Finally, for Edge-Defence Firefighter, we propose that the metaheuristic methods used in Blum et al. [3] and Hu, Windbichler and Raidl [31] for Classic Firefighter could provide the next step in improving on our heuristics for Edge-Defence Firefighter.

Bibliography

[1] K. Appel and W. Haken. Every planar map is four colorable. *Bulletin of the American Mathematical Society*, 82(5):711–712, 1976.

[2] C. Bazgan, M. Chopin, and B. Ries. The firefighter problem with more than one firefighter on trees. *Discrete Applied Mathematics*, 161(7-8):899–908, 2013.

[3] C. Blum, M. J. Blesa, C. García-Martínez, F. J. Rodríguez, and M. Lozano. The firefighter problem: application of hybrid ant colony optimization algorithms. In *Evolutionary Computation in Combinatorial Optimisation: 14th European Conference, EvoCOP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers 14*, pages 218–229. Springer, 2014.

[4] A. Burgess, J. Hawkin, A. Howse, J. Marcoux, and D. A. Pike. Distance-restricted firefighting on finite graphs. *arXiv preprint arXiv:2306.12575*, 2023.

[5] A. Burgess, J. Marcoux, and D. Pike. Firefighting with a distance-based restriction. *arXiv preprint arXiv:2204.01908*, 2022.

[6] L. Cai, Y. Cheng, E. Verbin, and Y. Zhou. Surviving rates of graphs with bounded treewidth for the firefighter problem. *SIAM Journal on Discrete Mathematics*, 24(4):1322–1335, 2010.

[7] L. Cai and W. Wang. The surviving rate of a graph for the firefighter problem. *SIAM Journal on Discrete Mathematics*, 23(4):1814–1826, 2009.

[8] X. Chen, X. Hu, C. Wang, and Y. Zhang. Continuous firefighting on infinite square grids. In *International Conference on Theory and Applications of Models of Computation*, pages 158–171. Springer, 2017.

[9] F. Comellas, M. Mitjana, and J. Peters. Broadcasting in small-world communication networks. *TIC*, 1997:0963, 1997.

[10] V. Costa, S. Dantas, M. C. Dourado, L. Penso, and D. Rautenbach. More fires and more fighters. *Discrete Applied Mathematics*, 161(16-17):2410–2419, 2013.

[11] P. Coupechoux, M. Demange, D. Ellison, and B. Jouve. Firefighting on trees. *Theoretical Computer Science*, 794:69–84, 2019. Special Issue on Theory and Applications of Graph Searching.

[12] S. Crosby, A. Finbow, B. Hartnell, R. Moussi, K. Patterson, and D. Wattar. Designing fire resistant graphs. *Congr. Numerantium*, 173:207–222, 2005.

[13] S. Days-Merrill. Firefighter Problem Played on Infinite Graphs. Bachelor's thesis, Bridgewater State University, USA, 2019.

[14] A. Dean, S. English, T. Huang, R. A. Krueger, A. Lee, M. Mizrahi, and C. Wheaton-Werle. Firefighting on the hexagonal grid and on infinite trees. 2020.

[15] M. Develin and S. G. Hartke. Fire containment in grids of dimension three and higher. *Discrete Applied Mathematics*, 155(17):2257–2268, 2007.

[16] C. Duffy and G. MacGillivray. An analysis of the weighted firefighter problem. *J. Combin. Math. Combin. Comput*, 94:167–175, 2015.

[17] L. Esperet, J. Van den Heuvel, F. Maffray, and F. Sipma. Fire containment in planar graphs. *Journal of Graph Theory*, 73(3):267–279, 2013.

[18] S. Finbow. Personal communication. 2019.

[19] S. Finbow, B. Hartnell, Q. Li, and K. Schmeisser. On minimizing the effects of fire or a virus on a network. *JCMCC. The Journal of Combinatorial Mathematics and Combinatorial Computing*, 33, 01 2000.

[20] S. Finbow, A. King, G. MacGillivray, and R. Rizzi. The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307:2094–2105, 07 2007.

[21] S. Finbow and G. MacGillivray. The firefighter problem: a survey of results, directions and questions. *Australas. J Comb.*, 43:57–78, 2009.

[22] Finbow, S. The firefighter problem on graphs. Presentation, 2019. Scottish Combinatorics Meeting.

[23] F. V. Fomin, P. Heggernes, and E. J. van Leeuwen. The firefighter problem on graph classes. *Theoretical Computer Science*, 613:38–50, 2016.

[24] C. García-Martínez, C. Blum, F. J. Rodríguez, and M. Lozano. The firefighter problem: Empirical results on random graphs. *Computers & Operations Research*, 60:55–66, 2015.

[25] T. Gavenčiak, J. Kratochvíl, and P. Prałat. Firefighting on square, hexagonal, and triangular grids. *Discrete Mathematics*, 337:142 – 155, 2014.

[26] B. Goodman. To stem widespread extinction, scientists airlift frogs in carry-on bags. *The New York Times*, 2006.

[27] P. Gordinowicz. Planar graph is on fire. *Theoretical Computer Science*, 593:160 – 164, 2015.

[28] G. Gunther and B. Hartnell. On minimizing the effects of betrayals in a resistance movement. In *Proceedings of Eighth Manitoba Conference on Numerical Math. and Computing*, pages 285–306, 1978.

[29] B. Hartnell and Q. Li. Firefighting on trees: How bad is the greedy algorithm? *Congress Numerantium*, 145:187–192, 2000.

[30] Hartnell, B. Firefighter! An application of domination. Presentation, 1995. 24th Manitoba Conference on Combinatorial Mathematics and Computing.

[31] B. Hu, A. Windbichler, and G. R. Raidl. A new solution representation for the firefighter problem. In *Evolutionary Computation in Combinatorial Optimization: 15th European Conference, EvoCOP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings 15*, pages 25–35. Springer, 2015.

[32] Y. Iwaikawa, N. Kamiyama, and T. Matsui. Improved approximation algorithms for firefighter problem on trees. *IEICE Transactions on Information and Systems*, E94-D(2):196–199, Feb. 2011. Copyright: Copyright 2018 Elsevier B.V., All rights reserved.

[33] R. R. Kao, L. Danon, D. M. Green, and I. Z. Kiss. Demographic structure and pathogen dynamics on the network of livestock movements in great britain. *Proceedings of the Royal Society B: Biological Sciences*, 273(1597):1999–2007, 2006.

[34] J. Kong, L. Zhang, and W. Wang. Structural properties and surviving rate of planar graphs. *Discrete Mathematics, Algorithms and Applications*, 6(04):1450052, 2014.

[35] C. Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 15(1):271–283, 1930.

[36] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[37] G. MacGillivray and P. Wang. On the firefighter problem. *JCMCC. The Journal of Combinatorial Mathematics and Combinatorial Computing*, 47:83–96, 2003.

[38] C. Marzouk. Fires on large recursive trees. *Stochastic Processes and their Applications*, 126(1):265–289, 2016.

[39] B. McKay. Combinatorial data, https://users.cecs.anu.edu.au/ bdm/data/, accessed 23\11\2023.

[40] K. Michalak. Auto-adaptation of genetic operators for multi-objective optimization in the firefighter problem. In *Intelligent Data Engineering and Automated Learning–IDEAL 2014: 15th International Conference, Salamanca, Spain, September 10-12, 2014. Proceedings 15*, pages 484–491. Springer, 2014.

[41] K. Michalak. Estimation of distribution algorithms for the firefighter problem. In *Evolutionary Computation in Combinatorial Optimization: 17th European Conference, EvoCOP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings 17*, pages 108–123. Springer, 2017.

[42] K. Michalak. Evolutionary graph-based v+ e optimization for protection against epidemics. In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II 16*, pages 399–412. Springer, 2020.

[43] K. Michalak and J. D. Knowles. Simheuristics for the multiobjective nondeterministic firefighter problem in a time-constrained setting. In *European Conference on the Applications of Evolutionary Computation*, pages 248–265. Springer, 2016.

[44] M. Newman. *Networks: An Introduction*. OUP Oxford, 2010.

[45] L. Nkwogu, F. Shuaib, F. Braka, P. Mkanda, R. Banda, C. Korir, S. Bawa, S. Mele, M. Saidu, H. Mshelia, et al. Impact of engaging security personnel on access and polio immunization outcomes in security-inaccessible areas in borno state, nigeria. *BMC Public Health*, 18:67–73, 2018.

[46] B. Oppenheim, N. Lidow, P. Ayscue, K. Saylors, P. Mbala, C. Kumakamba, and M. Kleinman. Knowledge and beliefs about ebola virus in a conflict-affected area: Early evidence from the north kivu outbreak. *Journal of global health*, 9(2), 2019.

[47] N. Ramos, C. C. de Souza, and P. J. de Rezende. A matheuristic for the firefighter problem on graphs. *International Transactions in Operational Research*, 27(2):739–766, 2020.

[48] C. Wagner. A new survey on the firefighter problem. Master's thesis, University of Victoria, 2021.

[49] W. Wang, S. Finbow, and P. Wang. The surviving rate of an infected network. *Theoretical Computer Science*, 411(40):3651 – 3660, 2010.

[50] W. Wang, S. Finbow, and P. Wang. A lower bound of the surviving rate of a planar graph with girth at least seven. *Journal of Combinatorial Optimization*, 27(4):621–642, May 2014.

[51] W. Wang, T. Wu, X. Hu, and Y. Wang. Planar graphs without chordal 5-cycles are 2-good. *Journal of Combinatorial Optimization*, 35(3):980–996, Apr 2018.

Appendix A

New $K(n, f)$-optimal Graphs

All graphs were found using Algorithm 1 from Chapter 2 and complete lists of graphs from certain categories from McKay's combinatorial data set [39] and are presented in g6 format.

All $K(11, 2)$-optimal Chordal Graphs

J?AAD?oEAO?

All $K(11, 3)$-optimal Trees

JkE?K?@_??_ JkE?K?@_??_
J?AA@?OAFo? J?AA@?OaF_?
J?AA@?OaF__

All $K(11, 4)$-optimal Chordal Graphs

J?AA@?OAFo?

All $K(12, 2)$-optimal Chordal Graphs

K??CB@OI?gP?

All $K(12, 2)$-optimal Trees

K??CB@OI?gP?	K???EA_E?gQO	K???EA_E?gAW
K???EA_E?KI_	K?AA@?OAF?B_	K??CA?_C?Oˆ_
K??CA?_CCO]_	K??CA?_C?Wˆ?	K??CA?_C?W]@
K??CA?_CEOU_	K??CA?_CEOE‘	K??CA?_CCW]?
K??CA?_CCWN?	K??CA?_CCWM@	K??CA?_cAO[_
K??CA?_c?W\?	K??CA?_c?W[@	K??CA?_cBOW_
K??CA?_E?g]?	K??CA?_E?g\?	K??CA?_E?g[A
K??CA?_EE_Ca	K??CA?_ECgL?	K??CA?_ECgKA
K??CAA_K?WL?	K??CAA_K?WJ?	K??CA?oICG[?
K??CA?oICGY?	K??CA?oICGK_	K??CA?oICGWC
K??CA?oI?gWA	K???C@_SCOWo	K???C@_S?WXO
K???C@_S?WWP	K???C@_SCWWO	K???C@_E?KY_

Appendix B

Pseudocode for D_2

This appendix contains pseudocode for calculating D_2 as used in 3.6 for all relevant cases except $k - c \leq a$ and $k - d > b$, which is given in Algorithm 5 of that section.

Input: $k, a, b, c, d \in \mathbb{N}$, with $a, b \leq \lfloor \frac{k-1}{2} \rfloor$, $a \leq b$, $k - c > a$ and $k - d \leq b$
Output: An ordered list D_2 of the edges to be defended for a single edge defender against a fire starting on $(a, b) \in P_k \square P_k$

$t = (k - c - 1, k - 1)$;
$s = (k - c - 1, b)$;
split edge $= [((k - c - 1, b), (k - c, b))]$;
above $= \text{Man}(s, t)$;
below $= d - above - 1$;
L shape $= below + c$;
initialise an empty list *above defence*;
initialise an empty list *L defence*;
initialise *right counter* $= 0$;
while *above counter* $<$ *above* **do**
    append
    $((k - c - 1, b + 1 + above\ counter), (k - c, b + 1 + above\ counter))$ to *above defence* ;
    above counter $+= 1$;
end
initialise *below counter* $= 0$;
while *below counter* $<$ *below* **do**
    append
    $((k - c - 1, b - 1 - below\ counter), (k - c, b - 1 - below\ counter))$ to *L defence* ;
    below counter $+= 1$;
end
initialise *right counter* $= 0$;
while *right counter* $<$ *right* **do**
    append
    $((k - c + right\ counter, k - d - 1), (k - c + right\ counter, k - d))$ to *L defence* ;
    right counter $+= 1$;
end
if *above* $<$ *L shape* **then**
    $D_2 =$ *above defence* $+$ *split edge* $+$ *L defence*;
else
    $D_2 =$ *L defence* $+$ *split edge* $+$ *above defence*;
end
return D_2

Input: $k, a, b, c, d \in \mathbb{N}$, with $a, b \leq \lfloor \frac{k-1}{2} \rfloor$, $a \leq b$, $k - c > a$ and $k - d > b$
Output: An ordered list D_2 of the edges to be defended for a single edge defender against a fire starting on $(a, b) \in P_k \square P_k$

initialise an empty list *c defence*;
initialise an empty list *d defence*;
initialise *c counter* $= 0$;
while *c counter* $<$ *c* **do**
  append $((k - c + c\ counter, k - d - 1), (k - c + c\ counter, k - d))$ to *c defence* ;
  c counter $+= 1$;
end
initialise *d counter* $= 0$;
while *d counter* $<$ *d* **do**
  append $((k - c - 1, k - d + d\ counter), (k - c, k - d + d\ counter))$ to *d defence* ;
  d counter $+= 1$;
end
if $c < d$ **then**
  $D_2 = c\ defence + d\ defence$;
else
  $D_2 = d\ defence + c\ defence$;
end
return D_2

Appendix C

Results of Heuristic EDF Experiments

Table C.1: $n = 200$, Mean amount of burning with different centrality measures fed into GREEDY

Parameters (m, f, d)	Edge Betweenness	Katz	Closeness	Betweenness	Degree
205, 10, 2	115.8	103.5	104.7	105.6	108.0
205, 10, 4	62.5	52.7	57.8	55.1	55.0
205, 20, 2	163.1	161.4	160.0	163.1	160.8
205, 20, 4	130.0	124.9	123.5	124.7	126.8
220, 10, 2	164.8	165.4	159.1	161.4	165.2
220, 10, 4	95.1	86.0	83.8	82.5	89.4
220, 20, 2	182.3	181.6	184.0	183.4	182.4
220, 20, 4	158.6	153.1	152.4	155.0	154.8

Table C.2: $n = 200$, Mean amount of time with different centrality measures fed into GREEDY

Parameters (m, f, d)	Edge Betweenness (ms)	Katz (ms)	Closeness (ms)	Betweenness (ms)	Degree (ms)
205, 10, 2	107.0	12.4	42.5	97.1	5.1
205, 10, 4	110.9	11.4	42.8	98.4	4.0
205, 20, 2	112.5	12.0	43.6	100.5	4.2
205, 20, 4	111.2	11.7	42.9	99.6	3.9
220, 10, 2	113.5	13.7	44.3	102.5	5.5
220, 10, 4	114.4	13.2	43.8	101.5	5.1
220, 20, 2	117.4	13.0	44.4$	104.4	4.2
220, 20, 4	113.5	12.6	43.1	101.7	4.4

Table C.3: $n = 200$, Mean amount of burning with different centrality measures fed into RECALC

Parameters (m, f, d)	Edge Betweenness	Katz	Closeness	Betweenness	Degree
205, 10, 2	98.3	90.6	92.3	93.7	92.1
205, 10, 4	54.9	48.1	49.5	50.1	48.4
205, 20, 2	142.8	141.3	143.9	142.6	142.0
205, 20, 4	110.8	107.9	107.9	107.9	108.2
220, 10, 2	151.9	149.9	151.6	152.2	148.3
220, 10, 4	85.9	77.7	78.6	77.5	79.6
220, 20, 2	168.3	167.6	169.9	169.2	167.7
220, 20, 4	142.8	140.9	141.1	141.2	141.3

Table C.4: $n = 200$, Mean amount of time with different centrality measures fed into RECALC CENTRALITY

Parameters (m, f, d)	Edge Betweenness (ms)	Katz (ms)	Closeness (ms)	Betweenness (ms)	Degree (ms)
205, 10, 2	133.9	37.2	68.5	123.0	30.5
205, 10, 4	133.3	30.4	63.1	119.9	23.1
205, 20, 2	135.2	36.6	69.3	124.2	29.6
205, 20, 4	131.1	32.8	64.5	119.7	25.7
220, 10, 2	143.8	44.7	76.1	132.8	37.4
220, 10, 4	142.9	38.2	70.9	130.2	30.9
220, 20, 2	139.0	38.6	69.9	127.2	31.0
220, 20, 4	136.0	36.5	67.8	125.0	28.8

Table C.5: Centrality measures that lead to the lowest mean amount of burning for $n = 200$.

Parameters	Best centrality for GREEDY	Best centrality for RECALCULATED COMP'S
205, 10, 2	Katz	Katz
205, 10, 4	Katz	Katz
205, 20, 2	Closeness	Katz
205, 20, 4	Closeness	Katz
220, 10, 2	Closeness	Degree Centrality
220, 10, 4	Betweenness	Betweenness
220, 20, 2	Katz	Katz
220, 20, 4	Closeness	Katz

Table C.6: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 100$, $m = 100$, $f = 5$, $d = 1$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT. Mean Burn and standard deviation rounded to one decimal place.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	17	92	60.9	18.1
COMPONENT HIGH	19	77	44.9	13.3
COMPONENT JUMP	19	77	44.9	13.3
RECALCULATED COMP'S	19	70	42.5	11.8
COMP. DENSITY TIEBREAK	19	77	44.9	13.3
GREEDY	19	89	47.2	15.1

Table C.7: Time taken for various heuristics to run with closeness centrality and $n = 100$, $m = 100$, $f = 5$, $d = 1$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S. All values rounded to one decimal place.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	4.0	3.0	0.6
COMPONENT HIGH	12.9	26.9	15.7	1.7
COMPONENT JUMP	12.0	20.0	14.7	1.4
RECALCULATED COMP'S	16.0	24.0	19.8	1.6
COMPONENT DENSITY TIEBREAK	13.9	23.9	16.2	1.5
GREEDY	9.0	19.0	11.7	1.5

Table C.8: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 100$, $m = 100$, $f = 5$, $d = 2$. The best performing heuristic in terms of mean amount of burning was COMP. DENSITY TIEBREAK; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	11	66	31.5	12.2
COMPONENT HIGH	10	43	23.8	8.0
COMPONENT JUMP	10	43	23.8	8.0
RECALCULATED COMP'S	10	45	23.8	8.0
COMP. DENSITY TIEBREAK	10	43	23.7	8.0
GREEDY	10	54	25.2	9.6

Table C.9: Time taken for various heuristics to run with Katz centrality and $n = 100$, $m = 100$, $f = 5$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	1.0	5.0	2.7	0.7
COMPONENT HIGH	6.0	17.0	8.3	1.3
COMPONENT JUMP	6.0	10.0	7.4	0.9
RECALCULATED COMP'S	8.0	17.0	11.1	1.5
COMPONENT DENSITY TIEBREAK	7.0	16.9	8.8	1.4
GREEDY	4.0	10.0	5.1	0.8

Table C.10: Mean amount of burning using various heuristic algorithms with degree centrality for $n = 100$, $m = 100$, $f = 5$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	11	66	31.5	12.2
COMPONENT HIGH	10	43	23.9	7.8
COMPONENT JUMP	10	43	23.9	7.8
RECALCULATED COMP'S	10	45	23.6	7.7
COMP. DENSITY TIEBREAK	10	43	23.8	7.7
GREEDY	10	57	26.7	10.4

Table C.11: Time taken for various heuristics to run with degree centrality and $n = 100$, $m = 100$, $f = 5$, $d = 2$. The fastest heuristic on average was GREEDY; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	4.0	2.7	0.6
COMPONENT HIGH	3.0	7.0	5.0	0.8
COMPONENT JUMP	3.0	7.0	4.1	0.8
RECALCULATED COMP'S	3.9	14.0	8.0	1.6
COMPONENT DENSITY TIEBREAK	3.0	8.9	5.4	1.0
GREEDY	0.5	4.0	1.8	0.6

Table C.12: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 100$, $m = 100$, $f = 10$, $d = 1$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	51	96	78.8	9.8
COMPONENT HIGH	45	94	71.7	9.6
COMPONENT JUMP	45	94	71.8	9.7
RECALCULATED COMP'S	45	88	68.6	8.4
COMP. DENSITY TIEBREAK	45	94	71.7	9.6
GREEDY	46	96	75.2	9.6

Table C.13: Time taken for various heuristics to run with closeness centrality and $n = 100$, $m = 100$, $f = 10$, $d = 1$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	4.6	2.7	0.6
COMPONENT HIGH	13.9	21.0	15.4	1.1
COMPONENT JUMP	12.9	18.0	14.5	1.1
RECALCULATED COMP'S	16.9	24.9	19.8	1.3
COMPONENT DENSITY TIEBREAK	14.0	21.5	16.0	1.1
GREEDY	10.0	14.0	11.3	0.7

Table C.14: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 100$, $m = 100$, $f = 10$, $d = 1$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	51	96	78.8	9.8
COMPONENT HIGH	45	93	71.3	9.3
COMPONENT JUMP	45	93	71.2	9.3
RECALCULATED COMP'S	45	86	68.3	7.9
COMP. DENSITY TIEBREAK	45	93	71.3	9.3
GREEDY	56	94	77.6	7.9

Table C.15: Time taken for various heuristics to run with Katz centrality and $n = 100$, $m = 100$, $f = 10$, $d = 1$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	4.0	2.5	0.5
COMPONENT HIGH	6.9	12.9	9.1	1.1
COMPONENT JUMP	7.0	14.0	8.4	1.2
RECALCULATED COMP'S	11.0	21.9	13.6	1.6
COMPONENT DENSITY TIEBREAK	7.0	15.0	9.7	1.2
GREEDY	4.0	6.0	5.1	0.6

Table C.16: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 100$, $m = 100$, $f = 10$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	40	78	59.4	9.6
COMPONENT HIGH	39	73	53.3	8.5
COMPONENT JUMP	39	73	53.3	8.5
RECALCULATED COMP'S	35	71	51.2	7.9
COMP. DENSITY TIEBREAK	39	73	53.4	8.6
GREEDY	34	87	58.6	10.3

Table C.17: Time taken for various heuristics to run with closeness centrality and $n = 100$, $m = 100$, $f = 10$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	7.0	2.6	0.7
COMPONENT HIGH	13.0	20.9	14.9	1.1
COMPONENT JUMP	12.0	21.9	14.1	1.5
RECALCULATED COMP'S	14.9	26.1	18.4	1.6
COMPONENT DENSITY TIEBREAK	14.0	23.9	15.3	1.2
GREEDY	10.0	16.0	11.3	0.9

Table C.18: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 100$, $m = 110$, $f = 5$, $d = 1$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	59	96	88.4	6.6
COMPONENT HIGH	23	98	77.5	17.2
COMPONENT JUMP	23	98	77.5	17.2
RECALCULATED COMP'S	23	98	74.2	15.3
COMP. DENSITY TIEBREAK	23	98	77.5	17.2
GREEDY	23	98	77.5	17.2

Table C.19: Time taken for various heuristics to run with closeness centrality and $n = 100$, $m = 110$, $f = 5$, $d = 1$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	5.0	3.1	0.5
COMPONENT HIGH	15.0	22.0	18.2	1.6
COMPONENT JUMP	14.9	24.0	17.7	1.6
RECALCULATED COMP'S	18.9	30.9	23.1	2.2
COMPONENT DENSITY TIEBREAK	15.9	23.9	18.6	1.7
GREEDY	9.9	15.0	12.4	1.2

Table C.20: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 100$, $m = 110$, $f = 5$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	19	90	66.1	17.1
COMPONENT HIGH	14	85	39.2	15.8
COMPONENT JUMP	14	85	39.2	15.8
RECALCULATED COMP'S	14	80	38.0	14.9
COMP. DENSITY TIEBREAK	14	85	39.2	15.8
GREEDY	14	85	39.7	16.0

Table C.21: Time taken for various heuristics to run with closeness centrality and $n = 100$, $m = 110$, $f = 5$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	5.0	3.1	0.6
COMPONENT HIGH	13.9	19.9	16.3	1.4
COMPONENT JUMP	12.9	26.9	15.9	1.8
RECALCULATED COMP'S	15.9	25.9	20.0	2.1
COMPONENT DENSITY TIEBREAK	13.9	21.0	16.6	1.6
GREEDY	9.0	16.8	11.9	1.3

Table C.22: Mean amount of burning using various heuristic algorithms with betweenness centrality for $n = 100$, $m = 110$, $f = 10$, $d = 1$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	73	98	91.4	5.0
COMPONENT HIGH	67	99	89.7	6.9
COMPONENT JUMP	67	99	89.7	6.9
RECALCULATED COMP'S	67	97	85.1	6.3
COMP. DENSITY TIEBREAK	67	99	89.7	6.9
GREEDY	67	99	90.8	6.8

Table C.23: Time taken for various heuristics to run with betweenness centrality and $n = 100$, $m = 110$, $f = 10$, $d = 1$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	4.0	2.6	0.5
COMPONENT HIGH	27.9	34.9	30.8	1.7
COMPONENT JUMP	26.9	33.9	30.0	1.6
RECALCULATED COMP'S	30.9	43.4	34.6	2.1
COMPONENT DENSITY TIEBREAK	27.9	42.4	31.1	2.0
GREEDY	22.9	59.8	26.0	3.7

Table C.24: Mean amount of burning using various heuristic algorithms with edge betweenness centrality for$n = 100$, $m = 110$, $f = 10$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	57	95	81.7	7.7
COMPONENT HIGH	48	97	78.4	9.8
COMPONENT JUMP	48	97	78.4	9.8
RECALCULATED COMP'S	43	90	72.4	8.7
COMP. DENSITY TIEBREAK	48	97	78.9	9.8
GREEDY	48	97	98.9	9.8

Table C.25: Time taken for various heuristics to run with edge betweenness centrality and $n = 100$, $m = 110$, $f = 10$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	4.0	2.7	0.5
COMPONENT HIGH	28.0	42.9	33.7	2.2
COMPONENT JUMP	28.9	40.0	33.0	2.1
RECALCULATED COMP'S	31.0	46.8	37.2	2.5
COMPONENT DENSITY TIEBREAK	30.9	41.9	34.4	2.2
GREEDY	25.9	32.9	29.0	1.6

Table C.26: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 100$, $m = 110$, $f = 10$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	57	95	81.7	7.7
COMPONENT HIGH	49	96	78.6	10.2
COMPONENT JUMP	49	96	78.6	10.2
RECALCULATED COMP'S	39	90	72.3	9.3
COMP. DENSITY TIEBREAK	49	96	78.6	10.2
GREEDY	57	96	79.9	9.7

Table C.27: Time taken for various heuristics to run with Katz centrality and $n = 100$, $m = 110$, $f = 10$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	2.0	6.0	2.8	0.7
COMPONENT HIGH	9.0	19.0	11.3	1.5
COMPONENT JUMP	8.0	14.0	10.3	1.1
RECALCULATED COMP'S	12.0	17.0	14.6	1.1
COMPONENT DENSITY TIEBREAK	10.0	14.0	11.5	0.9
GREEDY	4.9	8.0	6.0	0.6

Table C.28: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 200$, $m = 205$, $f = 10$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	71	188	136.4	27.6
COMPONENT HIGH	52	154	97.0	24.3
COMPONENT JUMP	52	154	97.0	24.3
RECALCULATED COMP'S	52	142	90.6	20.6
COMP. DENSITY TIEBREAK	52	154	97.0	24.3
GREEDY	54	154	103.5	23.7

Table C.29: Time taken for various heuristics to run with Katz centrality and $n = 200$, $m = 205$, $f = 10$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	5.0	12.0	6.5	1.0
COMPONENT HIGH	20.0	44.9	26.8	4.1
COMPONENT JUMP	16.9	35.9	24.3	3.6
RECALCULATED COMP'S	29.0	59.8	37.7	5.1
COMPONENT DENSITY TIEBREAK	20.1	57.4	28.4	4.3
GREEDY	9.0	20.9	12.3	1.4

Table C.30: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 200$, $m = 205$, $f = 10$, $d = 4$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	25	130	67.5	23.9
COMPONENT HIGH	23	92	48.9	14.0
COMPONENT JUMP	23	92	48.9	14.0
RECALCULATED COMP'S	23	82	48.1	13.2
COMP. DENSITY TIEBREAK	23	92	48.9	14.0
GREEDY	23	107	52.7	16.0

Table C.31: Time taken for various heuristics to run with Katz centrality and $n = 200$, $m = 205$, $f = 10$, $d = 4$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	4.0	8.0	5.6	0.7
COMPONENT HIGH	16.9	30.0	20.9	2.3
COMPONENT JUMP	14.0	22.9	18.5	2.0
RECALCULATED COMP'S	21.9	36.9	28.9	3.2
COMPONENT DENSITY TIEBREAK	18.0	32.9	22.3	2.3
GREEDY	8.0	15.0	11.1	1.1

Table C.32: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 200$, $m = 205$, $f = 20$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	126	189	161.4	13.2
COMPONENT HIGH	112	185	152.9	14.2
COMPONENT JUMP	112	185	153.0	14.2
RECALCULATED COMP'S	112	173	143.9	12.2
COMP. DENSITY TIEBREAK	112	185	152.9	14.2
GREEDY	113	188	159.9	15.6

Table C.33: Time taken for various heuristics to run with closeness centrality and $n = 200$, $m = 205$, $f = 20$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	4.9	7.0	5.7	0.6
COMPONENT HIGH	50.9	75.8	58.3	4.3
COMPONENT JUMP	47.9	76.8	54.8	4.7
RECALCULATED COMP'S	60.9	90.8	69.4	4.7
COMPONENT DENSITY TIEBREAK	53.8	87.8	60.6	4.7
GREEDY	37.9	51.8	44.1	2.5

Table C.34: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 200$, $m = 205$, $f = 20$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT, tied exactly with GREEDY.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	126	189	161.4	13.2
COMPONENT HIGH	116	178	150.9	13.7
COMPONENT JUMP	116	178	151.0	13.7
RECALCULATED COMP'S	114	168	141.3	11.9
COMP. DENSITY TIEBREAK	116	178	150.9	13.7
GREEDY	128	181	161.4	12.3

Table C.35: Time taken for various heuristics to run with Katz centrality and $n = 200$, $m = 205$, $f = 20$, $d = 2$. FThe fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	4.0	7.0	5.5	0.5
COMPONENT HIGH	18.9	31.9	25.7	3.0
COMPONENT JUMP	17.0	40.9	22.6	3.3
RECALCULATED COMP'S	31.9	45.9	37.3	2.9
COMPONENT DENSITY TIEBREAK	22.9	36.9	27.8	2.8
GREEDY	9.9	13.0	11.6	0.7

Table C.36: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 200$, $m = 205$, $f = 20$, $d = 4$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	77	169	126.7	17.0
COMPONENT HIGH	86	153	115.2	13.8
COMPONENT JUMP	86	153	115.2	13.9
RECALCULATED COMP'S	81	135	107.9	11.5
COMP. DENSITY TIEBREAK	86	153	115.2	13.8
GREEDY	90	168	123.5	17.4

Table C.37: Time taken for various heuristics to run with closeness centrality and $n = 200$, $m = 205$, $f = 20$, $d = 4$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	4.0	7.0	5.4	0.6
COMPONENT HIGH	47.9	62.8	53.7	2.6
COMPONENT JUMP	44.7	71.8	50.7	3.2
RECALCULATED COMP'S	55.9	70.8	63.1	3.4
COMPONENT DENSITY TIEBREAK	50.8	62.8	55.8	2.7
GREEDY	37.9	46.9	42.7	1.8

Table C.38: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 200$, $m = 205$, $f = 20$, $d = 4$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	77	169	126.7	17.0
COMPONENT HIGH	77	152	113.9	14.6
COMPONENT JUMP	77	152	113.9	14.6
RECALCULATED COMP'S	77	139	107.9	11.7
COMP. DENSITY TIEBREAK	77	152	113.9	14.5
GREEDY	93	153	124.9	14.5

Table C.39: Time taken for various heuristics to run with closeness centrality and $n = 200$, $m = 205$, $f = 20$, $d = 4$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	4.0	7.0	5.4	0.6
COMPONENT HIGH	47.9	62.8	53.7	2.6
COMPONENT JUMP	44.7	71.8	50.7	3.2
RECALCULATED COMP'S	55.9	70.8	63.1	3.4
COMPONENT DENSITY TIEBREAK	50.8	62.8	55.8	2.7
GREEDY	37.9	46.9	42.7	1.8

Table C.40: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 200$, $m = 220$, $f = 10$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	136	192	175.8	9.2
COMPONENT HIGH	62	193	158.9	25.9
COMPONENT JUMP	62	193	158.9	25.9
RECALCULATED COMP'S	62	187	151.6	23.0
COMP. DENSITY TIEBREAK	62	193	158.9	25.9
GREEDY	62	193	159.1	26.0

Table C.41: Time taken for various heuristics to run with closeness centrality and $n = 200$, $m = 220$, $f = 10$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	4.9	9.0	6.3	0.8
COMPONENT HIGH	55.8	74.8	65.0	3.9
COMPONENT JUMP	53.8	70.8	62.7	3.5
RECALCULATED COMP'S	67.9	84.8	75.4	4.0
COMPONENT DENSITY TIEBREAK	55.9	76.8	66.1	4.1
GREEDY	38.9	51.9	43.3	2.2

Table C.42: Mean amount of burning using various heuristic algorithms with degree centrality for $n = 200$, $m = 220$, $f = 10$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	136	192	175.8	9.2
COMPONENT HIGH	84	190	164.3	17.8
COMPONENT JUMP	84	190	164.3	17.8
RECALCULATED COMP'S	68	175	148.3	19.1
COMP. DENSITY TIEBREAK	84	190	164.3	17.8
GREEDY	84	195	165.2	17.5

Table C.43: Time taken for various heuristics to run with degree centrality and $n = 200$, $m = 220$, $f = 10$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	5.0	8.0	5.9	0.6
COMPONENT HIGH	18.9	34.9	26.2	3.2
COMPONENT JUMP	18.0	31.9	24.2	3.2
RECALCULATED COMP'S	26.9	48.7	35.3	3.5
COMPONENT DENSITY TIEBREAK	21.9	36.9	27.5	3.1
GREEDY	4.0	10.0	5.3	1.1

Table C.44: Mean amount of burning using various heuristic algorithms with betweenness centrality for$n = 200$, $m = 220$, $f = 10$, $d = 4$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	45	175	132.7	27.7
COMPONENT HIGH	31	141	80.7	26.5
COMPONENT JUMP	31	141	80.7	26.5
RECALCULATED COMP'S	31	130	77.5	24.0
COMP. DENSITY TIEBREAK	31	141	80.7	26.5
GREEDY	31	154	82.5	27.6

Table C.45: Time taken for various heuristics to run with betweenness centrality and $n = 200$, $m = 220$, $f = 10$, $d = 4$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	6.0	10.0	6.9	0.8
COMPONENT HIGH	103.0	144.7	114.3	6.0
COMPONENT JUMP	103.7	129.7	112.3	5.2
RECALCULATED COMP'S	111.7	160.6	125.7	7.4
COMPONENT DENSITY TIEBREAK	105.7	142.7	115.7	5.9
GREEDY	92.7	125.7	102.0	5.3

Table C.46: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 200$, $m = 220$, $f = 20$, $d = 2$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	156	192	182.4	7.0
COMPONENT HIGH	152	196	179.7	9.1
COMPONENT JUMP	152	196	179.7	9.2
RECALCULATED COMP'S	144	189	167.6	8.7
COMP. DENSITY TIEBREAK	152	196	179.7	9.1
GREEDY	159	196	181.6	8.0

Table C.47: Time taken for various heuristics to run with Katz centrality and $n = 200$, $m = 220$, $f = 20$, $d = 2$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	3.9	8.0	5.1	0.7
COMPONENT HIGH	21.9	38.4	28.6	2.6
COMPONENT JUMP	19.9	32.9	26.0	2.2
RECALCULATED COMP'S	32.9	61.9	37.3	3.3
COMPONENT DENSITY TIEBREAK	23.9	37.9	30.4	2.4
GREEDY	10.0	14.0	11.7	0.8

Table C.48: Mean amount of burning using various heuristic algorithms with closeness centrality for $n = 200$, $m = 220$, $f = 20$, $d = 4$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	126	186	164.8	10.9
COMPONENT HIGH	79	178	150.1	17.3
COMPONENT JUMP	79	178	150.1	17.4
RECALCULATED COMP'S	79	175	141.1	14.9
COMP. DENSITY TIEBREAK	79	178	150.1	17.3
GREEDY	79	181	152.4	17.9

Table C.49: Time taken for various heuristics to run with closeness centrality and $n = 200$, $m = 220$, $f = 20$, $d = 4$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	5.0	7.0	5.6	0.5
COMPONENT HIGH	50.9	70.9	58.9	3.8
COMPONENT JUMP	47.8	74.8	56.4	4.0
RECALCULATED COMP'S	57.9	79.7	66.9	4.0
COMPONENT DENSITY TIEBREAK	54.9	69.8	60.6	3.1
GREEDY	38.9	54.9	43.1	2.5

Table C.50: Mean amount of burning using various heuristic algorithms with Katz centrality for $n = 200$, $m = 220$, $f = 20$, $d = 4$. The best performing heuristic in terms of mean amount of burning was RECALCULATED COMP'S; the worst was COMPONENT.

Heuristic	Min Burn	Max Burn	Mean Burn	S.D.
COMPONENT	126	186	164.8	10.9
COMPONENT HIGH	91	179	151.0	16.5
COMPONENT JUMP	91	179	151.1	16.5
RECALCULATED COMP'S	83	162	140.9	14.7
COMPONENT DENSITY TIEBREAK	91	179	151.0	16.5
GREEDY	191	179	153.1	16.6

Table C.51: Time taken for various heuristics to run with Katz centrality and $n = 200$, $m = 220$, $f = 20$, $d = 4$. The fastest heuristic on average was COMPONENT; the slowest was RECALCULATED COMP'S.

Heuristic	Min Time (ms)	Max Time (ms)	Mean Time (ms)	S.D. (ms)
COMPONENT	4.9	10.0	5.6	0.8
COMPONENT HIGH	22.9	50.9	28.1	3.3
COMPONENT JUMP	18.9	33.9	25.1	2.3
RECALCULATED COMP'S	30.9	48.9	36.5	3.0
COMPONENT DENSITY TIEBREAK	23.9	44.7	30.0	2.7
GREEDY	10.0	20.9	12.1	1.3