TOWARDS OPTIMAL APPROXIMATIONS FOR FIREFIGHTING AND RELATED PROBLEMS

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
ANDREA BAGGIO

Dott. mag. in Matematica, Università degli Studi di Padova
born on 27.01.1988
citizen of Italy

accepted on the recommendation of
Prof. Dr. Rico Zenklusen, examiner
Prof. Dr. Ola Svensson, co-examiner
Dr. David Adjiashvili, co-examiner

2016
Acknowledgements

First of all, I would like to express my greatest gratitude and sincere appreciation to my advisor Prof. Rico Zenklusen for the exceptional guidance during my Ph.D. studies. His assistance and patience have been a great help for conducting research and writing this thesis. I am grateful to Prof. Robert Weismantel for his support as head of the Institute for Operations Research and for his insightful criticism which has been very valuable for both my academic and personal growth. I am thankful to Prof. Ola Svensson for accepting to be a member of my thesis committee.

I wish to express my thanks to Prof. Andrea Lodi for his support during my secondment in Bologna, for his supervision in our joint research project, and for his invaluable role as the head of the EU grant FP7-PEOPLE-2012-ITN no. 316647 “Mixed-Integer Nonlinear Optimization”, which has funded most of my Ph.D. studies.

I thank Annette Ryter for the administrative support and for being available whenever I needed her help. I would also like to thank all my colleagues at the Institute for Operations Research for the pleasant and enjoyable time, especially Kevin Zemmer, Jörg Bader, and my office mate Simon Bruggmann, who supported me during the final stages of this thesis.

I thank all the people I met during my visit in Bologna for the great time spent together and for bringing me to many places in which we had delicious food. I am grateful to Margarida Carvalho and Andrea Tramontani, with whom I collaborated during and after the time spent in Bologna. Moreover, I owe special thanks to Dimitri Thomopulos for the help concerning the access to the cluster.

Big thanks also have to be acknowledged to my parents and my sister. They supported and encouraged my studies since I was a kid, especially my parents, who always took care of my well-being allowing me to concentrate solely on my education.

Finally, I owe huge thanks to David Adjiashvili. I cannot find words that can really honor his guidance. He has been an extraordinary mentor in the academic world and a supportive friend outside of it.
Abstract

This thesis addresses problems that arise in protection models for optimal inhibition of harmful diffusion processes in networks. Specifically, we study the Firefighter problem, introduced two decades ago by Hartnell [59], a variant of it, known as Resource Minimization for Fire Containment (RMFC), and the Multilevel Critical Node problem (MCN), which is a new model introduced in this thesis.

The Firefighter problem and the RMFC problem have attracted considerable attention in the last decades, but despite progresses on several fronts, the approximability of these problems is still badly understood. This is the case even when the underlying graph is a tree, which is one of the most-studied graph structures in this context and the focus of the first part of this thesis. Prior to this work, the best known approximation ratios were an $O(1)$-approximation for the Firefighter problem and an $O(\log^* n)$-approximation for RMFC, both being LP-based and essentially matching the integrality gaps of two natural LP relaxations. We improve on both approximations by presenting a PTAS for the Firefighter problem and an $O(1)$-approximation for RMFC, both qualitatively matching the known hardness results. Our results are obtained through a combination of the known LPs with several new techniques, which allow for efficiently enumerating over super-constant size sets of constraints to strengthen the natural LPs.

In the second part of this thesis, we define the MCN problem as a combination of two different paradigms in the field of network protection. The first perspective looks at network safety from the point of view of prevention: for a given network, the goal is to modify its structure, in order to minimize its capacity to propagate failures. The second perspective consider blocking models. These problems have been proposed for scenarios where the attack has already taken place, in the spirit of the Firefighter problem. In this case, the harmful diffusion process is assumed to propagate through the network with particular dynamics that allow for adopting defensive reactions. The MCN problem combines these two perspectives, following the framework of the Defender-Attacker-Defender model, as introduced by Brown et al. [19]. In this thesis, we define the MCN problem, we devise an exact algorithm for it and we perform an experimental study, both on the performance and
on the quality of the optimal solution in comparison to a simpler sequential model.


Riassunto

Questa tesi affronta problemi derivanti da modelli di protezione per l’inibizione ottimale di processi dannosi che si diffondono su reti. Nello specifico, studiamo il Firefighter problem, introdotto circa venti anni fa da Hartnell [59], una sua variante, conosciuta come Resource Minimization for Fire Containment (RMFC), e il Multilevel Critical Node problem (MCN), che è un nuovo modello introdotto in questa tesi.

Il Firefighter problem e il problema RMFC hanno attirato un’attenzione considerevole negli ultimi decenni. Tuttavia, nonostante i progressi su vari fronti, l’approssimabilità di questi problemi è ancora poco conosciuta. Questo corrisponde pure al caso in cui il grafo sottostante è un albero, che è una delle strutture di grafi più studiate in questo contesto e l’interesse della prima parte di questa tesi. Prima di questo lavoro, i migliori rapporti di approssimazione conosciuti erano una $O(1)$-approximation per il Firefighter problem e una $O(\log^* n)$-approximation per RMFC, entrambi LP-based e essenzialmente corrispondenti agli integrality gap dei due rilassamenti dei rispettivi LP conosciuti. Miglioriamo entrambi le approssimazioni, presentando un PTAS per il Firefighter problem e una $O(1)$-approximation per RMFC, entrambi qualitativamente coincidenti con le rispettive complessità conosciute. I nostri risultati sono ottenuti attraverso la combinazione degli LP conosciuti e di diverse nuove tecniche, che permettono di enumerare efficientemente su un più che costante numero di insiemi di vincoli in modo da rafforzare gli LP.

Nella seconda parte di questa tesi, definiamo il problema MCN come una combinazione di due paradigmi noti nel campo della protezione di reti. Il primo studia la sicurezza di reti da un punto di vista preventivo: data una rete, l’obiettivo è quello di modificarne la struttura in modo da minimizzare la sua capacità di propagare guasti. Il secondo invece prende in considerazione modelli di contenimento. Questi problemi sono stati proposti per la circostanza in cui il guasto ha già preso luogo, come ad esempio per il Firefighter problem. In questo caso, si assume che il processo dannoso di diffusione si propaga attraverso la rete con una dinamica tale da permettere una reazione di contenimento. Il problema MCN, mette assieme queste due prospettive e si classifica come un modello Defender-Attacker-Defender,
introdotto da Brown et al. [19]. In questa tesi, definiamo il problema MCN, ideiamo un algoritmo esatto per risolverlo e esibiamo studi computazionali sia sulle prestazioni dell’algoritmo che sulla qualità delle soluzioni ottimali rispetto a quelle che ne derivano da un modello sequenziale più semplice.
Contents

Acknowledgements i

Abstract iii

Riassunto v

Contents vi

1 Introduction 1

2 Firefighting on Trees 5

2.1 Introduction 6

2.1.1 Our results 8

2.1.2 Further related results 10

2.1.3 Organization of the chapter 11

2.2 Classic LP relaxations and preliminaries 12

2.3 Overview of PTAS for Firefighter problem 15

2.3.1 Basic rounding approach 15

2.3.2 Upper bounding the loss incurred by basic rounding 16

2.3.3 Limiting the loss through stronger constraints 17

2.3.4 Towards an efficient procedure 21

2.3.5 Putting everything together 23
## CONTENTS

2.4 Overview of $O(1)$-approximation for RMFC ........................................... 24

2.4.1 Partial rounding to save leaves largely covered on low levels ........................................... 26

2.4.2 Partial enumeration algorithm ................................................................. 28

2.4.3 Summary of our $O(1)$-approximation for RMFC .............................. 33

2.5 Details on compression results ................................................................. 34

2.6 Missing details for Firefighter PTAS ......................................................... 39

2.7 Missing details for $O(1)$-approx. for RMFC ........................................ 46

2.8 Transformations for the Firefighter problem ............................................ 52

2.9 Conclusions .......................................................................................... 57

3 The Multilevel Critical Node Problem ......................................................... 59

3.1 Introduction ............................................................................................ 60

3.1.1 Our contribution ................................................................................. 64

3.1.2 Further related work ......................................................................... 64

3.1.3 Measuring the effectiveness of the three-stage approach ......................... 66

3.1.4 Organization of the chapter .............................................................. 68

3.2 Solving the Multilevel Critical Node problem .......................................... 68

3.2.1 Solving the Attack-Protect problem ............................................... 73

3.3 Computational Results ........................................................................... 77

3.4 Conclusions .......................................................................................... 85

4 Conclusions .............................................................................................. 87

Bibliography ............................................................................................... 88
Chapter 1

Introduction

In the last decades the study of diffusion processes, such as epidemics, spread of information, virus propagation among computers, purchase recommendations, and more, has attracted a large interest in many different research communities. Although those processes come from completely different systems, they all share important similarities. These similarities allow for obtaining general models of infection for the underlying networks, which differ in their spreading mechanics, but are often removed from any application-specific information.

On the one hand, models differ according to the possible states that any individual can assume during the diffusion process. For example, in this thesis, we consider any individual to be either susceptible, infective or removed. Susceptible individuals are those that are not affected by the spreading agent yet. Infective individuals are those who contracted the infection and may infect susceptible individuals if on contact with them. Once an individual gets infected it stays so and it cannot heal. Removed individuals are immune to the infection and, thus, do not contract or propagate the virus.

On the other hand, different models can be defined according to the importance that is given to the system topology as an influencing factor for the diffusion of the process. With respect to this distinction, is it possible to distinguish two general paradigms: homogeneous-mixing and network-based models. The homogeneous approach does not take into account the topology of the system, but rather assumes every individual to be connected to any other individual. These models investigate global attributes of the diffusion process and assume that the infection rate only depends on the density of the infected population. See [10, 5, 64] for further information.
CHAPTER 1. INTRODUCTION

about those model applied to epidemiology. Equivalent assumptions can be found under the study of *mean-field* theory. This is a well studied topic in system’s science, e.g. see [13].

Network-based models, in contrast, consider the underlying topology of the system as an important aspect that determines the diffusion of viral processes. In [12], the authors present a qualitative investigation on both homogeneous-mixing and network-based models and stress the importance of taking into consideration the system topology for describing epidemics diffusion in heterogeneous scale-free network. Recent progresses in network science and the significant growth of the computational power of the last decades has attracted much interests to network-based modeling of propagation in several research communities [3, 40, 81, 90, 66, 31, 87, 25]. Network-based models are also the focus of this thesis.

As mentioned, the study of diffusion processes in networks has attracted attention of various communities, for example network sciences, epidemiology, physics, sociology and computer science. Often, these different communities are interested in different aspects of propagation. For example, in physics, epidemiology and network sciences, typically the network is assumed to be drawn from some distribution and thresholds for stability are studied [67, 86, 13, 64]. In computer science and optimization, which is the field to which this thesis belongs, the main investigations revolve around questions of maximization and minimization of the diffusion of the agent in the given (deterministic) network. Roughly speaking, models in the latter fields can be divided into *influence maximization* and *spread minimization* problems.

The interest for influence maximization is motivated by applications to marketing. Suppose that we want to put on the market a new product or innovation and we can try to convince individuals to adopt it. Based on principles of *viral marketing*, targeting a small influential subset of individuals may convince other individuals to try it, which may convince other ones as well, resulting in a cascade of additional adoptions. Ideas spread into a system of individuals through basic interactions, e.g. the “word of mouth” effect. The underlying network is hence chosen to represent these interactions, e.g. friendship on Facebook. The question that naturally
arises in this context is which subset of individuals should be targeted as to trigger a cascade of adoptions that spread through the biggest fraction of the network [40, 81]. The seminal work by Kempe et al. [66], addressed this question and inspired many more studies on this topic.

Spread minimization, instead, is based on the assumption that the diffusion process is an harmful agent from which the network has to be defended. In this scenario, models specifies not only the spreading dynamics but also the admissible defensive strategies for the network defender. Many real world applications motivate these problems, such as designing defensive strategy that limits the impact of terrorist attacks on networks, failures in power grids, viral epidemics in human societies, viruses in computer networks, advertisement by a competing company, and more. The defensive strategies that appear in these models can be generally subdivided into two categories, both of which are addressed in this thesis. The first class of defensive strategies consider preventive actions and assume the system yet to be affected by the harmful agent. We refer to them as **vaccination** or **prevention** strategies. The second class, instead, assumes the identification of an attack in a network. The network operator is allowed in these models to intervene with **blocking** strategies that aim at containing the harmful spread. Typically, in such models, the temporal dimension is introduced, and the goal is to devise an optimal action, one per time point, until a steady state is reached.

Models based on preventive strategies essentially study the resilience of a network against possible diffusion processes, which may be random failures or clever well planned attacks. Concretely, the network topology is typically modified, by removing **critical infrastructures** such as nodes or edges, so as to minimize its capacity to propagate failures. Thus, the goal is to reduce the effectiveness of a harmful agent’s diffusion by reducing the network overall capacity to transmit the harmful agent. Since the source location of the diffusion process cannot be foreseen in advance, network resilience is often modeled by a particular metric, defined in terms of connectivity properties or other graph attributes [73, 57], that capture the capacity of the process to spread through the network. Problems that ask for finding a subset of nodes whose vaccination optimize some given metric are called Critical Node
Problems (CNPs) [7]. This class of problems has been widely investigated in literature for many different graph metrics, see [37, 1, 88, 84, 85] and references therein.

Blocking models, instead, consider strategies that allow for protecting the network after a diffusion process has been detected. In this case, the diffusion is assumed to propagate according to particular dynamics that allow for an effective defensive reaction. The goal is to isolate the propagation so as to maximize the unharmed fraction of the network, or minimize the resources necessary to protect some given part of the network. The Firefighter problem, introduced by Hartnell [59], is the most studied problem in this class of models. Together with a related problem called the Resource Minimization for Fire Containment problem (RMFC), the Firefighter problem is the main focus of the second chapter of this thesis. Another example of blocking models lies in the topic called unbalanced cuts. More precisely, the Minimum-Size Bounded-Capacity Cut problem, as introduced by Hayrapetyan et al. [61], asks for finding a minimal size subset of nodes containing \( s \) whose cut capacity is bounded by some given size. The vertex-separator version of this problem is considered in the third chapter of this thesis in a more general framework that entails preventive strategies as well.

The contribution of thesis can be summarized in the following points. In Chapter 2, we devised an optimal approximation for the Firefighter problem on trees, which is based on a structural investigation of a Linear Program and on the introduction of new combinatorial techniques. In the second chapter, we also present an asymptotically optimal approximation for the RMFC problem on trees, based on similar results used for the Firefighter problem in addition to a new recursive enumerative technique. In Chapter 3, we introduce a model that combines protection and blocking strategies into a three stage sequential game. We devise an exact algorithm and we show numerical experiments on the performance and on the gain that this three level approach achieves with respect to a sequential approach. Finally, in Chapter 4, we draw some conclusions and we suggest directions for future work.
Chapter 2

Firefighting on Trees

The Firefighter problem and a variant of it, known as Resource Minimization for Fire Containment (RMFC), are natural models for optimal inhibition of harmful spreading processes. Despite considerable progress on several fronts, the approximability of these problems is still badly understood. This is the case even when the underlying graph is a tree, which is one of the most-studied graph structures in this context and the focus of this chapter. In their simplest version, a fire spreads from one fixed vertex step by step from burning to adjacent non-burning vertices, and at each time step $B$ many non-burning vertices can be protected from catching fire. The Firefighter problem asks, for a given $B$, to maximize the number of vertices that will not catch fire, whereas RMFC (on a tree) asks to find the smallest $B$ that allows for saving all leaves of the tree. Prior to this work, the best known approximation ratios were an $O(1)$-approximation for the Firefighter problem and an $O(\log^* n)$-approximation for RMFC, both being LP-based and essentially matching the integrality gaps of two natural LP relaxations.

We improve on both approximations by presenting a PTAS for the Firefighter problem and an $O(1)$-approximation for RMFC, both qualitatively matching the known hardness results. Our results are obtained through a combination of the known LPs with several new techniques, which allow for efficiently enumerating over super-constant size sets of constraints to strengthen the natural LPs.
2.1 Introduction

The Firefighter problem was introduced by Hartnell [59] as a natural model for optimal inhibition of harmful spreading phenomena on networks. Despite considerable interest in the problem and progress on several fronts, our understanding of how well this and related problems can be approximated is still very limited. Interestingly, this is even true when the underlying graph is a spanning tree, which is one of the most-studied graph structures in this context and also the focus of this chapter.

The Firefighter problem on trees is defined as follows. We are given a graph \( G = (V, E) \) which is a spanning tree and a vertex \( r \in V \), called root. The problem is defined over discretized time steps. At time 0, a fire starts at \( r \) and spreads step by step to neighboring vertices. During each time step \( 1, 2, \ldots \) an arbitrary non-burning vertex \( u \) can be protected, preventing \( u \) from burning in any future time step. In its original form, the goal is to find a protection strategy minimizing the number of vertices that will catch fire. A closely related problem, called Resource Minimization for Fire Containment (RMFC) on trees, was introduced by Chalermsook and Chuzhoy [26]. Here the task is to determine the smallest number \( B \in \mathbb{Z}_{>0} \) such that if one can protect \( B \) vertices at each time step (instead of just 1), then there is a protection strategy where none of the leaves of the tree catches fire. In this context, \( B \) is referred to as the number of firefighters.

The Firefighter problem as well as RMFC—both restricted to trees as defined above—are known to be computationally hard problems. More precisely, Finbow, King, MacGillivray and Rizzi [45] showed NP-hardness for the Firefighter problem on trees with maximum degree three. For RMFC on trees, it is NP-hard to decide whether one firefighter, i.e., \( B = 1 \), is sufficient [69]; thus, unless \( P = NP \), there is no (efficient) approximation algorithm with an approximation factor strictly better than 2.

On the positive side, several approximation algorithms have been suggested for the Firefighter problem and RMFC. Hartnell and Li [60] showed that a natural greedy algorithm is a \( \frac{1}{2} \)-approximation for the Firefighter problem. This approximation guarantee was later improved by Cai, Verbin and Yang [21] to \( 1 - \frac{1}{e} \), using a natural linear programming (LP) relaxation.
and dependent randomized rounding. It was later observed by Anshelevich, Chakrabarty, Hate and Swamy [6] that the Firefighter problem on trees can be interpreted as a monotone submodular function maximization (SFM) problem subject to a partition matroid constraint. This leads to alternative ways to obtain a \((1 - \frac{1}{e})\)-approximation by using a recent \((1 - \frac{1}{e})\)-approximation for monotone SFM subject to a matroid constraint [89, 33]. The factor \(1 - \frac{1}{e}\) was later only improved for various restricted tree topologies (see [65]) and hence, for arbitrary trees, this is the best known approximation factor to date. For RMFC on trees, Chalermsook and Chuzhoy [26] presented an \(O(\log^* n)\)-approximation, where \(n = |V|\) is the number of vertices.\(^1\) Their algorithm is based on a natural linear program which is a straightforward adaptation of the one used in [21] to get a \((1 - \frac{1}{e})\)-approximation for the Firefighter problem on trees.

Whereas there are still considerable gaps between current hardness results and approximation algorithms for both the Firefighter problem and RMFC on trees, the currently best approximations essentially match the integrality gaps of the underlying LPs. More precisely, Chalermsook and Vaz [27] showed that for any \(\epsilon > 0\), the canonical LP used for the Firefighter problem on trees has an integrality gap of \(1 - \frac{1}{e} + \epsilon\). This generalized a previous result by Cai, Verbin and Yang [21], who showed the same gap if the integral solution is required to lie in the support of an optimal LP solution. For RMFC on trees, the integrality gap of the underlying LP is \(\Theta(\log^* n)\) [26].

It remained open to what extent these integrality gaps may reflect the approximation hardnesses of the problems. This question is motivated by two related problems whose hardnesses of approximation indeed matches the above-mentioned integrality gaps for the Firefighter problem and RMFC. In particular, many versions of monotone SFM subject to a matroid constraint—which we recall was shown in [6] to capture the Firefigther problem on trees as a special case—are hard to approximate up to a factor of \(1 - 1/e + \epsilon\) for any constant \(\epsilon > 0\). This includes the problem of maximizing an explicitly given coverage function subject to a single cardinality constraint, as shown

\(^1\) \(\log^* n\) denotes the minimum number \(k\) of logs of base two that have to be nested such that \(\log \log \ldots \log n \leq 1\).
CHAPTER 2. FIREFIGHTING ON TREES

by Feige [43]. Moreover, as highlighted in [26], the Asymmetric \( k \)-center problem is similar in nature to RMFC, and has an approximation hardness of \( \Theta(\log^* n) \) [30].

The goal of this chapter is to fill the gap between current approximation ratios and hardness results for the Firefighter problem and RMFC on trees. To obtain our results we introduce several new techniques, which may be of independent interest.

This chapter is based on a joint work with David Ajiashvili and Rico Zenklusen [2]

2.1.1 Our results

Our main results are approximation algorithms for both Firefighting and RMFC that essentially match known hardness bounds, showing that approximation factors substantially stronger than the integrality gaps of the natural LPs can be achieved. In particular, we obtain the following result for RMFC.

**Theorem 1.** There is a 12-approximation for RMFC on trees.

Since RMFC is hard to approximate within any factor better than 2, the above result is optimal up to a constant factor, and improves on the \( O(\log^* n) \)-approximation of Chalermsook and Chuzhoy [26].

Moreover, our main result for the Firefighter problem is the following, which, in view of NP-hardness of the problem, is essentially best possible in terms of approximation guarantee.

**Theorem 2.** There is a PTAS for the Firefighter problem on trees.\(^2\)

This essentially closes the question of approximability of the Firefighter problem on trees. Notice that the Firefighter problem does not admit an FPTAS\(^3\) unless \( P = NP \), since the optimal value of any Firefighter problem

\(^2\)A polynomial time approximation scheme (PTAS) is an algorithm that, for any constant \( \epsilon > 0 \), returns in polynomial time a \( (1 - \epsilon) \)-approximate solution.

\(^3\)A fully polynomial time approximation scheme (FPTAS) is a PTAS with running time polynomial in the input size and \( \frac{1}{\epsilon} \).
on a tree of \( n \) vertices is bounded by \( O(n) \).\(^4\) We introduce several new techniques that allow us to obtain approximation factors well beyond the integrality gaps of the natural LPs, which have been a barrier for previous approaches. We start by providing an overview of these techniques.

Despite the fact that we obtain approximation factors beating the integrality gaps, the natural LPs play a central role in our approaches. In combination with several other techniques, we introduce new enumeration procedures to gain information about a \textit{super-constant} size subset of the optimal solution. This allows us to define a residual problem with small integrality gap.

Similar high-level approaches, like guessing an influential constant-size subset of an optimal solution are well-known in various contexts to decrease integrality gaps of natural LPs. The best-known example may be classic PTASs for the knapsack problem, where the integrality gap of the natural LP can be decreased to an arbitrarily small constant by first guessing a constant number of heaviest elements of an optimal solution. However, our approach differs substantially from this standard enumeration idea. Apart from several transformation techniques we introduce, which allow for obtaining better structured instances, we will introduce new combinatorial approaches to gain information about a \textit{super-constant} subset of an optimal solution. In particular, for the RMFC problem we define a recursive enumeration algorithm which, despite being very slow for enumerating all solutions, can be shown to reach a good subsolution within a small recursion depth that can be reached in polynomial time. This enumeration procedure explores the space step by step, and at each step we solve an LP that determines how to continue the enumeration in the next step. For the Firefighter problem, we use a well-chosen enumeration procedure that can be interpreted as considering only a polynomial number of faces of the original LP. Each of those faces can be shown to have a small integrality gap when restricting to solutions only on that face. Moreover, one face contains the optimal solution. Focussing on faces of the feasible region of the classic LP allows us to exploit sparsity properties that we can show to hold for all vertices of

\(^4\) The nonexistence of FPTASs unless \( P = NP \) can often be derived easily from strong NP-hardness. Notice that the Firefighter problem is indeed strongly NP-hard because its input size is \( O(n) \), in which case NP-hardness is equivalent to strong NP-hardness.
the polytope describing the feasible region of the classic LP. These sparsity properties are heavily exploited in our PTAS.

A further application of our LP-guided recursive enumeration technique for RMFC was very recently found by Goyal, Krishnawamy, and Chakrabarty [56], who discovered an elegant connection between the non-uniform $k$-center problem and RMFC.

### 2.1.2 Further related results

Iwaikawa, Kamiyama and Matsui [65] showed that the approximation guarantee of $1 - \frac{1}{e}$ can be improved for some restricted families of trees, in particular of low maximum degree. Anshelevich, Chakrabarty, Hate and Swamy [6] studied the approximability of the Firefighter problem in general graphs, which they prove admits no $n^{1-\epsilon}$-approximation for any $\epsilon > 0$, unless $P = \text{NP}$. In a different model, where the protection also spreads through the graph (the *Spreading Model*), the authors show that the problem admits a polynomial $(1 - \frac{1}{e})$-approximation on general graphs. Moreover, for RMFC, an $O(\sqrt{n})$-approximation for general graphs and an $O(\log n)$-approximation for directed layered graphs is presented. The latter result was obtained independently by Chalermsook and Chuzhoy [26]. Klein, Levopoulos and Lingas [70] introduced a geometric variant of the Firefighter problem, proved its NP-hardness and provided a constant-factor approximation algorithm. The Firefighter problem and RMFC are natural special cases of the Maximum Coverage Problem with Group Constraints (MCGC) [28] and the Multiple Set Cover problem (MSC) [41], respectively.

The input in MCGC is a set system consisting of a finite set $X$ of elements with nonnegative weights, a collection of subsets $S = \{S_1, \ldots, S_m\}$ of $X$ and an integer $k$. The sets in $S$ are partitioned into groups $G_1, \ldots, G_l \subseteq S$. The goal is to pick a subset $H \subseteq S$ of $k$ sets from $S$ whose union covers elements of total weight as large as possible with the additional constraint that $|H \cap G_j| \leq 1$ for all $j \in [l] := \{1, \ldots, l\}$. In MSC, instead of the fixed bounds for groups and the parameter $k$, the goal is to choose a subset $H \subseteq S$ that covers $X$ completely, while minimizing $\max_{j \in [l]} |H \cap G_j|$. The Firefighter problem and RMFC can naturally be interpreted as special cases
2.1. INTRODUCTION

of the latter problems with a laminar set system $S$.

The Firefighter problem admits polynomial time algorithms in some restricted classes of graphs. Finbow, King, MacGillivray and Rizzi [45] showed that, while the problem is NP-hard on trees with maximum degree three, when the fire starts at a vertex with degree two in a subcubic tree, the problem is solvable in polynomial time. Fomin, Heggernes and van Leeuwen [50] presented polynomial algorithms for interval graphs, split graphs, permutation graphs and $P_k$-free graphs.

Several sub-exponential exact algorithms were developed for the Firefighter problem on trees. Cai, Verbin and Yang [21] presented a $2^{O(\sqrt{n} \log n)}$-time algorithm. Floderus, Lingas and Persson [48] presented a simpler algorithm with a slightly better running time, as well as a sub-exponential algorithm for general graphs in the spreading model and an $O(1)$-approximation in planar graphs under some further conditions.

Additional directions of research on the Firefighter problem include parameterized complexity [21, 15, 34, 14], generalizations to the case of many initial fires and many firefighters [16, 32], and the study of potential strengthenings of the canonical LP for the Firefighter problem on trees [58, 27]. Computing the Survivability of a graph is a further problem closely related to Firefighting that has attracted considerable attention (see [22, 20, 79, 42, 55, 71]). For a graph $G$ and a parameter $k \in \mathbb{Z}_{\geq 0}$, the $k$-survivability of $G$ is the average fraction of nodes that one can save with $k$ firefighters in $G$, when the fire starts at a random node.

For further references we refer the reader to the survey of Finbow and MacGillivray [46].

2.1.3 Organization of the chapter

We start by introducing the classic linear programming relaxations for the Firefighter problem and RMFC in Section 2.2. Section 2.3 and Section 2.4 outline our main techniques and algorithms for the Firefighter problem and RMFC, respectively. Some proofs and additional discussions are deferred to later sections, namely Section 2.5, providing details for a compression tech-
unique that we need in both Firefighting and RMFC, Section 2.6, containing further proofs for results related to the Firefighter problem, Section 2.7, containing further proofs for results related to RMFC, and Section 2.8, providing some basic transformations showing how to reduce different variations of the Firefighter problem to each other. Finally, Section 2.9 summarizes our main results and presents new possible direction for future research.

2.2 Classic LP relaxations and preliminaries

Interestingly, despite the fact that we obtain approximation factors considerably stronger than the known integrality gaps of the natural LPs, these LPs still play a central role in our approaches. We thus start by introducing the natural LPs together with some basic notation and terminology.

Let $L \in \mathbb{Z}_{\geq 0}$ be the depth of the tree, i.e., the largest distance—in terms of number of edges—between $r$ and any other vertex in $G$. Hence, after at most $L$ time steps, the fire spreading process will halt. For $\ell \in [L] := \{1, \ldots, L\}$, let $V_\ell \subseteq V$ be the set of all vertices at distance $\ell$ from $r$, which we call the $\ell$-th level of the instance. For brevity, we use $V_{\leq \ell} = \bigcup_{k=1}^{\ell} V_k$, and we define in the same spirit $V_{\geq \ell}, V_{< \ell},$ and $V_{> \ell}$. Moreover, we denote by $\Gamma \subseteq V$ the set of all leaves of the tree, and for any $u \in V$, the set $P_u \subseteq V \setminus \{r\}$ denotes the set of all vertices on the unique $u$-$r$ path except for the root $r$.

The relaxation for RMFC used in [26] is the following:

$$\begin{align*}
\min & \quad B \\
& x(P_u) \geq 1 \quad \forall u \in \Gamma \\
& x(V_{\leq \ell}) \leq B \cdot \ell \quad \forall \ell \in [L] \\
& x \in \mathbb{R}_{\geq 0}^{V \setminus \{r\}}
\end{align*}$$

(LP_{RMFC})

where $x(U) := \sum_{u \in U} x(u)$ for any $U \subseteq V \setminus \{r\}$. LP_{RMFC} is indeed a relaxation due to the following. If one enforces $x \in \{0, 1\}^{V \setminus \{r\}}$ and $B \in \mathbb{Z}$ in LP_{RMFC}, an exact description of RMFC is obtained where $x$ is the characteristic vector of the vertices to be protected and $B$ is the number of firefighters: The constraints $x(P_u) \geq 1$ for $u \in \Gamma$ enforce that for each leaf $u$, a vertex between $u$ and $r$ is protected, which makes sure that $u$ will
not be reached by the fire; moreover, the constraints $x(V_{\leq \ell}) \leq B \cdot \ell$ for
$\ell \in [L]$ describe the vertex sets that can be protected given $B$ firefighters per
time step (see [26] for more details). Also, as already highlighted in [26],
there is an optimal solution to RMFC (and also to the Firefighter problem),
that protects with the firefighters available at time step $\ell$ only vertices in $V_\ell$.
Hence, the above relaxation can be transformed into one with same optimal
objective value by replacing the constraints $x(V_{\leq \ell}) \leq B \cdot \ell$ $\forall \ell \in [L]$ by the
constraints $x(V_\ell) \leq B$ $\forall \ell \in [L]$.

The natural LP relaxation for the Firefighter problem, which leads to the
previously best $(1 - 1/e)$-approximation presented in [21], is obtained analogously.
Due to higher generality, and even more importantly to obtain more
flexibility in reductions to be defined later, we work on a slight generalization
of the Firefighter problem on trees, extending it in two ways:

(i) Weighted version: vertices $u \in V$ have weights $w(u) \in \mathbb{Z}_{\geq 0}$ with
$w(r) = 0$. The goal is to maximize the total weight of vertices not
catching fire. The classical Firefighter problem corresponds to unit
budgets, i.e., $w(u) = 1$ $\forall u \in V \setminus \{r\}$, and $w(r) = 0$ as always.

(ii) General budgets/firefighters: We allow for having a different number
of firefighters at each time step, say $B_\ell \in \mathbb{Z}_{> 0}$ firefighters for time
step $\ell \in [L]$.

Indeed, the above generalizations are mostly for convenience of presentation,
since general budgets can be reduced to unit budgets (see Section 2.8 for a
proof):

**Lemma 3.** Any weighted Firefighter problem on a tree with $n$ vertices and
general budgets can be transformed efficiently into an equivalent weighted
Firefighter problem on a tree with unit budgets and $O(n^2)$ vertices.

We also show in Section 2.8 that up to an arbitrarily small error in terms
of objective, any weighted Firefighter instance can be reduced to a unit-
weighted one. In what follows, we always assume to deal with a weighted

---

5Without loss of generality we exclude $B_\ell = 0$, since a level with zero budget can be
eliminated through a simple contraction operation. For more details we refer to the proof of
Theorem 10 which, as a sub-step, eliminates zero-budget levels.
Firefighter instance if not specified otherwise. Regarding the budgets, we will be explicit about whether we work with unit or general budgets, since some techniques are easier to explain in the unit-budget case, even though it is equivalent to general budgets by Lemma 3.

An immediate extension of the LP relaxation for the unit-weighted unit-budget Firefighter problem used in [21]—which is based on an IP formulation presented in [72]—leads to the LP relaxation \( \text{LP}_{\text{FF}} \) shown below for the weighted Firefighter problem with general budgets. For \( u \in V \), we denote by \( T_u \subseteq V \) the set of all vertices in the subtree rooted at \( u \) and including \( u \), i.e., all vertices \( v \) such that the unique \( r-v \) path in \( G \) contains \( u \).

\[
\begin{align*}
\max \quad & \sum_{u \in V \setminus \{r\}} x_u w(T_u) \\
x(P_u) & \leq 1 \quad \forall u \in \Gamma \\
x(V_{\leq \ell}) & \leq \sum_{i=1}^{\ell} B_i \quad \forall \ell \in [L] \\
x & \in \mathbb{R}_{\geq 0}^{V \setminus \{r\}}.
\end{align*}
\]

The constraints \( x(P_u) \leq 1 \) exclude redundancies, i.e., a vertex \( u \) is forbidden of being protected if another vertex above it, on the \( r-u \) path, is already protected. This elimination of redundancies allows for writing the objective function as shown above.

We recall that the integrality gap of \( \text{LP}_{\text{RMFC}} \) was shown to be \( \Theta(\log^* n) \) [26], and the integrality gap of \( \text{LP}_{\text{FF}} \) is asymptotically \( 1 - 1/e \) (when \( n \to \infty \)) [27].

Throughout the chapter, all logarithms are of base 2 if not indicated otherwise. When using big-\( O \) and related notations (like \( \Omega, \Theta, \ldots \)), we will always be explicit about the dependence on small error terms \( \epsilon \)—as used when talking about \( (1 - \epsilon) \)-approximations—and not consider it to be part of the hidden constant. Moreover, whenever we refer to a Firefighter or RMFC problem in this chapter, we assume that the underlying graph is a tree.
2.3 Overview of PTAS for Firefighter problem

To best present our approach, we first discuss a basic rounding scheme that underlines our algorithm, but has to be refined later. We then show how the loss of this rounding scheme can be analyzed. Inspired by this analysis, we present an exponential time algorithm that enumerates over families of constraints that can be added to the LP, one of which makes our analysis work out and implies a $(1 - \epsilon)$-approximation. Finally, we show how to turn this exponential time $(1 - \epsilon)$-approximation into an efficient procedure through two preprocessing techniques that transform the original Firefighter instance into a well-structured one.

2.3.1 Basic rounding approach

We start by introducing a basic rounding idea that underlies our algorithm, and which we will refine later. Despite the fact that LP$_{FF}$ has a large integrality gap, it is a crucial tool in our PTAS. Consider a general-budget Firefighter instance, and let $x$ be an optimal vertex solution to LP$_{FF}$. We partition its support $\text{supp}(x) := \{u \in V \setminus \{r\} \mid x(u) > 0\}$ into what we call loose and tight vertices. A vertex $u \in \text{supp}(x)$ is $x$-loose, or simply loose, if $x(P_u) < 1$, and it is $x$-tight, or simply tight, if $x(P_u) = 1$. The sets of all $x$-loose and $x$-tight vertices are denoted by $V_L$ and $V_T$, respectively.

To transform $x$ into an integral solution $y$, we determine an optimal vertex solution $y$ to LP$_{FF}$ with the additional restriction that only variables corresponding to vertices in $V_T$ can be used, i.e., we set $x(u) = 0$ for $u \in V \setminus V_T$. We start by observing that $y$ will indeed be a $\{0, 1\}$-vector. For this, notice that the set $V_T$ is nicely structured; more precisely, any two distinct vertices $u, v \in V_T$ are incomparable, in the sense that $u \notin P_v$ and $v \notin P_u$. Indeed, for any distinct $u, v \in \text{supp}(x)$ with $u \in P_v$, we have $x(P_u) \leq x(P_v) - x(v) < x(P_v)$, because $x(v) > 0$. Hence, it is not possible that both $u$ and $v$ are tight. Thus, when restricting LP$_{FF}$ to $V_T$, the path constraints $x(P_u) \leq 1$ for $u \in \Gamma$ transform into trivial constraints requiring $x(u) \leq 1$ for $u \in V_T$, and one can easily observe that the resulting constraint system is totally unimodular because it describes a laminar
CHAPTER 2. FIREFIGHTING ON TREES

matroid constraint given by the budget constraints (see [83, Volume B] for more details on matroid optimization). Hence, the vertex solution $y$ to $\text{LP}_{\text{FF}}$ restricted to vertices in $V^T$ is a $\{0, 1\}$-vector as desired.

Clearly, the above rounding idea needs further refinement. In particular, being based on $\text{LP}_{\text{FF}}$, we still have to overcome the known $(1 - \frac{1}{e})$-integrality gap. We show next a particular way to upper bound the loss between $x$ and $y$ in terms of LP-value, which will inspire techniques to be introduced later.

2.3.2 Upper bounding the loss incurred by basic rounding

We are interested in bounding $\text{val}(x) - \text{val}(y)$, where $\text{val}(x)$, and analogously $\text{val}(y)$, denotes the LP-value of $x$, i.e., $\text{val}(x) = \sum_{u \in V \setminus \{r\}} x_u w(T_u)$. To bound this difference, we construct a new solution $z$ of $\text{LP}_{\text{FF}}$ such that $\text{supp}(z) \subseteq V^T$, by starting with $x$ and moving load, i.e., $x$-value, from loose vertices to tight ones. Since $\text{val}(y)$ is an optimal solution to $\text{LP}_{\text{FF}}$ with $\text{supp}(y) \subseteq V^T$, we have $\text{val}(y) \geq \text{val}(z)$ and we can thus bound the incurred loss between $x$ and $y$ in LP-value by $\text{val}(x) - \text{val}(y) \leq \text{val}(x) - \text{val}(z)$.

Starting with $z = x$, we construct $z$ as follows. For each loose node $u \in V^L$, if $T_u \cap V^T \neq \emptyset$, we move its load $z(u)$ from $u$ to a well-chosen tight node $v$ in its subtree $T_u$, i.e., we set $z(u) = 0$ and $z(v) = z(v) + z(u)$. If $T_u \cap V^T = \emptyset$, we simply discard the load on $u$ by setting $z(u) = 0$. First observe that $z$ will indeed be feasible for $\text{LP}_{\text{FF}}$ because we only shift loads down and remove loads. Moreover, notice that if the load $x(u)$ of some vertex $u \in V^L$ gets shifted down to a vertex $v \in V^T \cap T_u$, the LP-value gets reduced by

$$x(u) \cdot (w(T_u) - w(T_v)) \leq w(T_u) - w(T_v) = w(T_u \setminus T_v). \quad (2.1)$$

Furthermore, whenever a load on some vertex $u \in V^L$ gets set to zero, the LP-value reduces by

$$x(u) \cdot w(T_u) \leq w(T_u). \quad (2.2)$$

To upper bound the difference $\text{val}(x) - \text{val}(y)$, we will sum up over all loose vertices the loss bounds given by the right-hand sides of (2.1) and (2.2),
respectively. Ideally, to have a low total loss in LP-value, we want to be in a situation where the following holds.

(i) For any $u \in V^L$, either $w(T_u)$ is small, or there is a vertex $v \in T_u \cap V^T$ such that $w(T_u \setminus T_v)$ is small.
(ii) There is only a small number $|V^L|$ of loose vertices.

In the following, we show how these properties can be achieved through various techniques. To bound $|V^L|$, it is key that we work with a vertex solution $x$ of $LP_{FF}$, for which we show the following sparsity result.

**Lemma 4.** Let $x$ be a vertex solution to $LP_{FF}$ for a Firefighter problem with general budgets. Then the number $|V^L|$ of $x$-loose vertices is at most $L$, the depth of the tree.

### 2.3.3 Limiting the loss through stronger constraints

Loosely speaking, to limit the loss of our load reassignment analysis presented in the previous section, we aim at adding constraints to $LP_{FF}$ that will keep the impact of any single load reassignment local. More precisely, we will determine a well-chosen subset of vertices $Q \subseteq V \setminus \{r\}$ and want to “guess” which vertices in $Q$ will catch fire in an optimal solution. This is achieved by enumerating over all possible subsets of $Q$. As we will prove later, vertex sets $Q$ that have the following structure, which we call $\eta$-splits, allow us to limit the loss of our load reassignment analysis.

**Definition 5 (\(\eta\)-split).** Let $\eta > 0$. A subset of the vertices $Q \subseteq V$ of a Firefighter instance is an $\eta$-split if:

(i) Removing $Q \cup \{r\}$ from $G$ breaks $G$ into connected components each of total weight at most $\eta$.
(ii) The nearest common ancestor of any two vertices in $Q$ is part of $Q \cup \{r\}$.

Moreover, it is not hard to find small $\eta$-splits.

**Lemma 6.** Let $\eta > 0$. One can efficiently compute an $\eta$-split $Q$ with $|Q| \leq \frac{2w(V)}{\eta}$. 

Let \( \eta > 0 \) and consider an \( \eta \)-split \( Q \). We are interested in enumerating the different possibilities of which vertices in \( Q \) get saved from the fire, and which ones will burn. Consider one such possibility, represented by a vertex set \( S \subseteq Q \) designating the vertices of \( Q \) that will be saved. We incorporate this guess into \( \text{LP}_{\text{FF}} \) by including the constraints \( x(P_q) = 1 \) for \( q \in S \) and \( x(P_q) = 0 \) for \( q \in Q \setminus S \), leading to the following linear program \( \text{LP}_{\text{FF}}(S) \).

\[
\begin{align*}
\max \quad & \sum_{u \in V \setminus \{r\}} x_u w(T_u) \\
\text{s.t.} \quad & x(P_u) \leq 1 \quad \forall u \in \Gamma \\
& x(V_{\leq \ell}) \leq \sum_{i=1}^{\ell} B_i \quad \forall \ell \in [L] \quad (\text{LP}_{\text{FF}}(S)) \\
& x(P_q) = 1 \quad \forall q \in S \\
& x(P_q) = 0 \quad \forall q \in Q \setminus S \\
& x \in \mathbb{R}_{\geq 0}^{V \setminus \{r\}}.
\end{align*}
\]

There are \( 2^{|Q|} \) such linear programs, one for each set \( S \subseteq Q \). One such set corresponds to the vertices of \( Q \) that will not catch fire in an optimal solution. Notice that some guesses \( S \subseteq Q \) may be inconsistent, in the sense that \( S \) may impose that some vertex \( q_1 \in S \) gets saved, but some vertex \( q_2 \in T_{q_1} \) may not get saved, i.e., \( q_2 \in Q \setminus S \). Clearly, it suffices to only consider consistent guesses. However, for simplicity, and since our analysis does not depend on this refinement, we will consider all \( 2^{|Q|} \) many subsets \( S \subseteq Q \). This leads to the following algorithm, which is parameterized by \( \eta > 0 \), and is central to our PTAS.

A key property of \( \text{LP}_{\text{FF}}(S) \) is that it maintains the sparsity structure of \( \text{LP}_{\text{FF}} \) shown in Lemma 4. More precisely, the following lemma shows that enumerating over all \( \text{LP}_{\text{FF}}(S) \) for \( S \subseteq Q \), corresponds to enumerating over a subset of the faces of \( \text{LP}_{\text{FF}} \), and thus preserves the vertex structure of \( \text{LP}_{\text{FF}} \).

**Lemma 7.** For any \( S \subseteq Q \subseteq V \setminus \{r\} \), the polytope over which \( \text{LP}_{\text{FF}}(S) \) optimizes is a face of the polytope describing the feasible region of \( \text{LP}_{\text{FF}} \). Consequently, any vertex solution \( x \) of \( \text{LP}_{\text{FF}}(S) \) is a vertex solution of \( \text{LP}_{\text{FF}} \),

---

**CHAPTER 2. FIREFIGHTING ON TREES**

---
Algorithm 1:

1. Compute an $\eta$-split $Q$ of size $|Q| \leq \frac{2w(V)}{\eta}$ using Lemma 6.
2. For each $S \subseteq Q$, compute optimal value of $\text{LP}_{\text{FF}}(S)$. Let $S^* \subseteq Q$ be a set for which the optimal value of $\text{LP}_{\text{FF}}(S^*)$ is largest among all $S \subseteq Q$, and let $x^*$ be an optimal vertex solution to $\text{LP}_{\text{FF}}(S^*)$.
3. Let $V^T$ be the $x^*$-tight vertices. Obtain optimal vertex solution $y^*$ to $\text{LP}_{\text{FF}}$ restricted to variables that correspond to vertices in $V^T$. Return the set $U$ corresponding to $\{0, 1\}$-vector $y^*$.

and hence, the number of loose vertices with respect to any vertex solution of $\text{LP}_{\text{FF}}(S)$ is bounded by $L$, the depth of the tree.

**Proof.** The statement immediately follows by observing that for any $u \in V \setminus \{r\}$, the inequalities $x(P_u) \leq 1$ and $x(P_u) \geq 0$ are valid inequalities for $\text{LP}_{\text{FF}}$. Indeed, $x(P_u) \geq 0$ is valid since it is implied by the non-negativity constraints. Furthermore, $x(P_u) \leq 1$ is implied by the fact that the load on any leaf-root path is at most 1. More precisely, for an arbitrary $v \in \Gamma \cap T_u$, we have $x(P_u) \leq x(P_v) \leq 1$. \hfill \Box

The crucial property of an $\eta$-split $Q$ is that no matter what set $S \subseteq Q$ we choose, the resulting linear program $\text{LP}_{\text{FF}}(S)$ has an integrality gap that can be bounded in terms of $L$ and $\eta$, and this bound can be achieved with our basic rounding procedure.

**Lemma 8.** Let $\eta > 0$, $Q$ be an $\eta$-split, and $S \subseteq Q$ be such that $\text{LP}_{\text{FF}}(S)$ is feasible. Let $x$ be an optimal vertex solution to $\text{LP}_{\text{FF}}(S)$, and let $V^T \subseteq V \setminus \{r\}$ be all $x$-tight vertices. Let $y$ be an optimal solution to $\text{LP}_{\text{FF}}$ restricted to variables corresponding to $V^T$. Then $\text{val}(x) - \text{val}(y) \leq L \cdot \eta$.

**Proof.** To prove the lemma, we employ the analysis highlighted in Section 2.3.2, to get a feasible solution $z$ to $\text{LP}_{\text{FF}}(S)$ with $\text{supp}(z) \subseteq V^T$, obtained by shifting load from loose vertices in $x$ to tight ones below them and
deleting the load on some loose vertices. By Lemma 4, we have $|V^L| \leq L$. Hence, to prove the lemma, it suffices to show that each load reassignment will incur a loss in terms of LP-value of at most $\eta$.

We denote by $G' = G[V \setminus (Q \cup \{r\})]$ the graph obtained from $G$ by removing the vertices $Q \cup \{r\}$. Since $Q$ is an $\eta$-split, each connected component of $G'$ has total weight at most $\eta$. Let $u \in V^L$ be an $x$-loose vertex. Notice that $u \not\in Q$, since for each vertex $q \in Q$ we either have $x(P_q) = 1$, and hence $q$ must be a tight vertex if $x(q) > 0$, or we have $x(P_q) = 0$, implying that $q$ is not loose.

We distinguish between whether there is a $Q$-vertex below $u$, i.e., $T_u \cap Q \neq \emptyset$, or not. If $T_u \cap Q = \emptyset$, then $T_u$ is contained in a single connected component of $G'$, which has total weight at most $\eta$, and hence $w(T_u) \leq \eta$ bounds the LP-loss when deleting the $x$-load on $u$. If $T_u \cap Q \neq \emptyset$, observe that among all vertices in $T_u \cap Q$, there is a nearest one $q \in T_u \cap Q$ in the following sense: for any other $q' \in T_u \cap Q$ we have $q \in P_{q'}$. This follows from the fact that for any two vertices in $Q$, their nearest common ancestor is also part of $Q \cup \{r\}$. Since $x(u) > 0$ (because $u$ is loose), we must have $q \in S$ and hence $x(P_q) = 1$. Thus, there is an $x$-tight vertex $v \in P_q \setminus P_u$. We move the $x$-load of $u$ to $v$, thus incurring an LP-loss of at most $w(T_u \setminus T_v) \leq w(T_u \setminus T_q)$. It remains to observe that $T_u \setminus T_q$ is in the same connected component of $G'$, and hence $w(T_u \setminus T_q) \leq \eta$.

Hence, the LP-loss of each single reassignment of an $x$-value of a loose vertex is bounded by $\eta$. The result follows since there are at most $L$ loose vertices.

In summary, the above bound on the integrality gap implies the following.

**Theorem 9.** For any $\eta > 0$, Algorithm 1 returns a solution to a Firefighter instance of value at least $\text{val}(\text{OPT}) - L \cdot \eta$ and runs in time $O(2^{w(V)}/\eta \cdot \text{poly}(\langle \text{input} \rangle))$, where $\text{poly}(\langle \text{input} \rangle)$ is a polynomial in the input size.

Hence, for Theorem 9 to guarantee that Algorithm 1 returns a $(1 - \epsilon)$-approximate solution, we would have to choose $\eta = \epsilon \text{val}(\text{OPT})/L$. However, even if we knew $\text{val}(\text{OPT})$—which can be guessed approximately—and set $\eta = \epsilon \text{val}(\text{OPT})/L$, the running time bound obtained by Theorem 9
depends linearly on $2^{O(Lw(V)/\epsilon \text{val(OPT)})}$, which is exponential in the input size in general. In the following, we present techniques to preprocess a Firefighter instance to an almost equivalent instance satisfying

$$\frac{Lw(V)}{\epsilon \text{val(OPT)}} = O(\log N),$$

where $N$ is the number of vertices in the original instance. This will turn Algorithm 1 into an efficient procedure and completes our PTAS.

### 2.3.4 Towards an efficient procedure

The transformation of the original instance will be done in two steps. We first show how to compress a Firefighter instance to one with only logarithmic depth. In a second step, we show how a Firefighter instance can be pruned to obtain $\text{val(OPT)} = \Omega(w(V))$.

**Compression**

Consider a unit-budget Firefighter instance on a tree $G = (V, E)$. To reduce the depth of the tree, we will first do a sequence of what we call down-pushes. Each down-push acts on two levels $\ell_1, \ell_2 \in [L]$ with $\ell_1 < \ell_2$ of the tree, and moves the budget $B_{\ell_1}$ of level $\ell_1$ down to $\ell_2$, i.e., the new budget of level $\ell_2$ will be $B_{\ell_1} + B_{\ell_2}$, and the new budget of level $\ell_1$ will be 0. Clearly, down-pushes only restrict our options for protecting vertices. However, we can show that one can do a sequence of down-pushes such that first, the optimal objective value of the new instance is very close to the one of the original instance, and second, only $O(\log L)$ levels have non-zero budgets. Finally, levels with 0-budget can easily be removed through a simple contraction operation, thus leading to a new instance with only $O(\log L)$ depth.

Theorem 10 below formalizes our main compression result for the Firefighter problem, which we state for unit-budget Firefighter instances for simplicity.

**Theorem 10.** Let $I$ be a unit-budget Firefighter instance on a tree with depth $L$, and let $\delta \in (0, 1)$. Then one can efficiently construct a general-budget
Firefighter instance $\overline{I}$ with depth $L' = O\left(\frac{\log L}{\delta}\right)$, and such that the following holds, where $\text{val}(\text{OPT}(\overline{I}))$ and $\text{val}(\text{OPT}(I))$ are the optimal values of $\overline{I}$ and $I$, respectively.

(i) $\text{val}(\text{OPT}(\overline{I})) \geq (1 - \delta) \text{val}(\text{OPT}(I))$, and 
(ii) any solution to $\overline{I}$ can be transformed efficiently into a solution of $I$ with the same objective value.

Since Lemma 3 implies that every general-budget Firefighter instance with $n$ vertices can be transformed into a unit-budget Firefighter instance with $O(n^2)$ vertices—and thus $O(n^2)$ levels—Theorem 10 can also be used to reduce any Firefighter instance on $n$ vertices to one with $O\left(\frac{\log n}{\delta}\right)$ levels, by losing a factor of at most $1 - \delta$ in terms of objective.

Interestingly, the above compression result already allows for obtaining a QPTAS\(^6\) for the Firefighting problem on trees. More precisely, after compression, an instance with $L' = O(\log L)$ levels is obtained, on which one can use a dynamic programming approach from the leaves up, and save for each vertex $u \in V$ the following table. For each vector $q \in [n]^{L'}$, where $q_\ell$ describes the budget of level $\ell$ used to protect vertices in $T_u$, we want to save the maximum weight that can be saved within $T_u$ by using the budget described by $q$. It is not hard to see that this leads to a QPTAS. However, it seems highly non-trivial to transform this approach into an efficient algorithm.

**Pruning**

Given a general-budget Firefighter instance on a tree, we now aim at obtaining a nearly equivalent instance satisfying $\text{val}(\text{OPT}) = \Omega(w(V))$. To this end we use an operation that we call pruning, and which is heavily inspired by the $\frac{1}{2}$-approximate greedy algorithm for the Firefighter problem introduced in [60]\(^7\). More precisely, we fix an integer $\lambda = \Theta\left(\frac{1}{\epsilon}\right)$ and show

\(^6\)A QPTAS is an algorithm that, for any constant $\epsilon > 0$, returns a $(1 - \epsilon)$-approximation in quasipolynomial time, i.e., its running time is bounded by $O(2^{\text{polylog}(\langle \text{input} \rangle)})$, where $\langle \text{input} \rangle$ is the input size of the problem.

\(^7\)This greedy algorithm is actually a special case of the well-known $\frac{1}{2}$-approximate greedy algorithm for maximizing a monotone submodular function over a matroid constraint.
the following. When running the greedy algorithm with an extended budget, where we allow for using $\lambda$-times as many Firefighters than are actually available at each time step, we save a set of vertices such that if we restrict the original instance to only those vertices, we obtain a sub-instance with the desired property. The following theorem summarizes our pruning result.

**Theorem 11.** Let $I$ be a general-budget Firefighter instance on a tree $G = (V, E)$ with weights $w$. Then for any $\lambda \in \mathbb{Z}_{\geq 1}$, one can efficiently construct a new Firefighter instance $\overline{I}$ on a subtree $G' = (V', E')$ of $G$ with same budgets, by starting from $I$ and applying node deletions and weight reductions, such that

\[(i) \quad \text{val}(\text{OPT}(\overline{I})) \geq \left(1 - \frac{1}{\lambda}\right) \text{val}(\text{OPT}(I)), \quad \text{and}
\]

\[(ii) \quad \text{val}(\text{OPT}(\overline{I})) \geq \frac{1}{\lambda}w'(V'), \quad \text{where} \ w' \leq w \text{ are the vertex weights in instance } \overline{I}.
\]

The deletion of $u \in V$ corresponds to removing the whole subtree below $u$ from $G$, i.e., all vertices in $T_u$. Since the new instance is obtained using only node deletions and weight reductions, any solution to the new instance is also a solution to the original instance of at least the same objective value.

To show Theorem 11, we heavily exploit the tree-structure of the problem, in particular that Firefighting on trees can be interpreted as a coverage problem with a laminar set family.

### 2.3.5 Putting everything together

Finally, our PTAS is obtained by first preprocessing the instance using our compression and pruning techniques, and then apply Algorithm 1 on the resulting instance. For completeness, our PTAS is summarized in Algorithm 2. For simplicity, we present our PTAS for the unit-budget case. By Lemma 3, this immediately implies a PTAS also for general budgets.

The following theorem formally finishes our discussion and implies Theorem 2, our main result for the Firefighter problem on trees.

**Theorem 12.** Algorithm 2 is a PTAS for the unit-budget Firefighter problem on trees.
Algorithm 2: A PTAS for the unit-budget Firefighter problem on trees

1. Compress original instance $I$ through Theorem 10 with $\delta = \epsilon/3$ to obtain compressed instance $I'$.

2. Prune $I'$ using Lemma 11 with $\lambda = \lceil \frac{3}{\epsilon} \rceil$ to obtain a compressed and pruned instance $I''$. In the following, we denote by $G'' = (V'', E'')$ the graph of the instance $I''$, with weights $w''$ and depth $L''$.

3. Run Algorithm 1 with $\eta = \frac{1}{12} \frac{\epsilon^2}{L''} w''(V'')$ on instance $I''$, to obtain a solution $U''$ to $I''$.

4. By Theorem 11 and Theorem 10, the solution $U''$ for $I''$ can be transformed into a solution $U$ for $I$ of the same objective value. We return $U$.

2.4 Overview of $O(1)$-approximation for RMFC

Also our $O(1)$-approximation for RMFC uses the natural LP, i.e., $LP_{RMFC}$, as a crucial tool to guide the algorithm. Similar to the Firefighting problem, it will be key to work on an instance with small depth. Using a similar technique as for Firefighting, we can compress an RMFC instance to contain only a logarithmic number of levels, by losing a constant factor in terms of the objective.

Theorem 13. Let $G = (V, E)$ be a rooted tree of depth $L$. Then one can construct efficiently a rooted tree $G' = (V', E')$ with $|V'| \leq |V|$ and depth $L' = O(\log L)$, such that:

(i) If the RMFC problem on $G$ has a solution with budget $B \in \mathbb{Z}_{>0}$ at each level, then the RMFC problem on $G'$ with non-uniform budgets, where level $\ell \geq 1$ has a budget of $B'_{\ell} = 2^\ell \cdot B$, has a solution.

(ii) Any solution to the RMFC problem on $G'$, where level $\ell$ has budget $B'_{\ell} = 2^\ell \cdot B$, can be transformed efficiently into an RMFC solution for $G$ with budget $2B$.

Notice that the budgets of the compressed instance are not uniform anymore,
2.4. OVERVIEW OF $O(1)$-APPROXIMATION FOR RMFC

but are exponentially increasing as highlighted in the above theorem. The exponential increase in budget is a fact that we exploit in our algorithm. Slightly abusing terminology, we call the compressed instance still an RMFC instance, despite the fact that the budgets are not uniform.

Throughout this section we will work on a compressed instance $G = (V, E)$ of RMFC, obtained through Theorem 13. Hence, the number of levels is $L = O(\log N)$, where $N$ is the number of vertices of the original instance. Furthermore, the budget on level $\ell \in [L]$ is given by $B_\ell = 2^\ell B$. The advantage of working with a compressed instance for RMFC is twofold. First, we will again apply sparsity reasonings to limit in certain settings the number of loose (badly structured) vertices by the number of levels of the instance. Second, the fact that low levels—i.e., levels far away from the root—have high budget, will allow us to protect a large number of loose vertices by only increasing $B$ by a constant.

For simplicity, we work with a slight variation of $LP_{RMFC}$, where we replace, for $\ell \in [L]$, the budget constraints $x(V_{\leq \ell}) \leq \sum_{i=1}^{\ell} B_i$ by $x(V_\ell) \leq B_\ell$. For brevity, we define

$$PB = \left\{ x \in \mathbb{R}^V_{\geq 0} \mid x(V_\ell) \leq B \cdot 2^\ell \quad \forall \ell \in [L] \right\}.$$  

As previously mentioned (and shown in [26]), the resulting LP is equivalent to $LP_{RMFC}$. Furthermore, since the budget $B$ for a feasible RMFC solution has to be chosen integral, we require $B \geq 1$. Hence, the resulting linear relaxation asks to find the minimum $B \geq 1$ such that the following polytope is non-empty:

$$\bar{P}_B = PB \cap \left\{ x \in \mathbb{R}^V_{\geq 0} \mid x(P_u) \geq 1 \quad \forall u \in \Gamma \right\}.$$  

Our constant-factor approximation for RMFC consists of two main ingredients. First we will show how one can round an LP solution $x$ with only constant-factor loss such that a well-chosen subset of leaves is cut off from the root. Loosely speaking, we can cut off all leaves $u \in \Gamma$ such that $P_u$ contains a large $x$-load on low levels, i.e., levels far away from the root. This shows that on low levels, we can essentially replace the LP-solution by an integral solution. It also implies that the $\Theta(\log^*(n))$ integrality gap of the
natural LP relaxation for RMFC is only due to levels close to the root. As a second main ingredient, we present a recursive LP-guided enumeration algorithm that finds a strong solution on levels close to the root.

### 2.4.1 Partial rounding to save leaves largely covered on low levels

We start by discussing approaches to partially round a fractional point \( x \in \bar{P}_B \), for some fixed budget \( B \geq 1 \). Any leaf \( u \in \Gamma \) is fractionally cut off from the root through the \( x \)-values on \( P_u \). A crucial property we derive and exploit is that leaves that are (fractionally) cut off from \( r \) largely on low levels can be cut off from the root via a set of vertices to be protected that are budget-feasible when increasing \( B \) only by a constant factor. To exemplify the above statement, consider the level \( h = \lfloor \log L \rfloor \) as a threshold to define top levels \( V_\ell \) as those with indices \( \ell \leq h \) and bottom levels when \( \ell > h \). For any leaf \( u \in \Gamma \), we partition the path \( P_u \) into its top part \( P_u \cap V_{\leq h} \) and its bottom part \( P_u \cap V_{>h} \). Consider all leaves that are cut off in bottom levels by at least 0.5 units: \( W = \{ u \in \Gamma \mid x(P_u \cap V_{>h}) \geq 0.5 \} \). We will show that there is a subset of vertices \( R \subseteq V_{>h} \) on bottom levels to be protected that is feasible for budget \( \bar{B} = 2B + 1 \leq 3B \) and cuts off all leaves in \( W \) from the root. We provide a brief sketch why this result holds, and present a formal proof later. If we set all entries of \( x \) on top levels \( V_{\leq h} \) to zero, we get a vector \( y \) with \( \text{supp}(y) \subseteq V_{>h} \) such that \( y(P_u \cap V_{>h}) \geq 0.5 \) for \( u \in W \). Hence, \( 2y \) fractionally cuts off all vertices in \( W \) from the root and is feasible for budget \( 2B \). To increase sparsity, we can replace \( 2y \) by a vertex \( \tilde{z} \) of the polytope

\[
Q = \left\{ z \in \mathbb{R}^{V \setminus \{r\}} \geq 0 \mid \begin{array}{l}
z(V_\ell) \leq 2B \cdot 2^\ell \ \forall \ell \in [L], \\
z(V_{\leq h}) = 0, \\
z(P_u) \geq 1 \ \forall u \in W
\end{array} \right\},
\]

which describes possible ways to cut off \( W \) from \( r \) only using levels \( V_{>h} \), and \( Q \) is non-empty since \( 2y \in Q \). Exhibiting a sparsity reasoning analogous to the one used for the Firefighter problem, we can show that \( \tilde{z} \) has no more than \( L \) many \( \tilde{z} \)-loose vertices. Thus, we can first include all \( \tilde{z} \)-loose vertices
in the set $R$ of vertices to be protected by increasing the budget of each level $\ell > h$ by at most $L \leq 2^{h+1} \leq 2^\ell$. The remaining vertices in $\text{supp}(\bar{z})$ are well structured (no two of them lie on the same leaf-root path), and an integral solution can be obtained easily as we will show formally later. The new budget value is $\bar{B} = 2B + 1$, where the “+1” term pays for the loose vertices.

The following theorem formalizes the above reasoning and generalizes it in two ways. First, for a leaf $u \in \Gamma$ to be part of $W$, we required it to have a total $x$-value of at least 0.5 within the bottom levels; we will allow for replacing 0.5 by an arbitrary threshold $\mu \in (0, 1]$. Second, the level $h$ defining what is top and bottom can be chosen to be of the form $h = \lfloor \log^q L \rfloor$ for $q \in \mathbb{Z}_{\geq 0}$, where $\log^q L := \log \log \ldots \log L$ is the value obtained by taking $q$ many logs of $L$, and by convention we set $\log^0 L := L$. The generalization in terms of $h$ can be thought of as iterating the above procedure on the RMFC instance restricted to $V_{\leq h}$.

**Theorem 14.** Let $B \in \mathbb{R}_{\geq 1}$, $\mu \in (0, 1]$, $q \in \mathbb{Z}_{\geq 1}$, and $h = \lfloor \log^q L \rfloor$. Let $x \in \bar{P}_B$ with $\text{supp}(x) \subseteq V_{> h}$, and we define $W = \{ u \in \Gamma \mid x(P_u) \geq \mu \}$. Then one can efficiently compute a set $R \subseteq V_{> h}$ such that

(i) $R \cap P_u \neq \emptyset \quad \forall u \in W$, and

(ii) $\chi^R \in P'_{B'}$, where $B' = \frac{q}{\mu} B + 1$ and $\chi^R \in \{0, 1\}^{V \setminus \{r\}}$ is the characteristic vector of $R$.

Theorem 14 has several interesting consequences. It immediately implies an LP-based $O(\log^* N)$-approximation for RMFC, thus matching the currently best approximation result by Chalermsook and Chuzhoy [26]: It suffices to start with an optimal LP solution $B \geq 1$ and $x \in \bar{P}_B$ and invoke the above theorem with $\mu = 1$, $q = 1 + \log^* L$. Notice that by definition of $\log^*$ we have $\log^* L = \min\{\alpha \in \mathbb{Z}_{\geq 0} \mid \log^\alpha L \leq 1\}$; hence $h = \lfloor \log^{1+\log^* L} L \rfloor = 0$, implying that all levels are bottom levels. Since the integrality gap of the LP is $\Omega(\log^* N) = \Omega(\log^* L)$, Theorem 14 captures the limits of what can be achieved by techniques based on the standard LP.

Interestingly, Theorem 14 also implies that the $\Omega(\log^* N)$ integrality gap is only due to the top levels of the instance. More precisely, if, for any $q = O(1)$
and $h = \lfloor \log^{(q)} L \rfloor$, one would know what vertices an optimal solution $R^*$ protects within the levels $V_{\leq h}$, then a constant-factor approximation for RMFC follows easily by solving an LP on the bottom levels $V_{>h}$ and using Theorem 14 with $\mu = 1$ to round the obtained solution.

Also, using Theorem 14 it is not hard to find constant-factor approximation algorithms for RMFC if the optimal budget $B_{\text{OPT}}$ is large enough, say $B_{\text{OPT}} \geq \log L$. The main idea is to solve the LP and define $h = \lfloor \log L \rfloor$. Leaves that are largely cut off by $x$ on bottom levels can be handled using Theorem 14. For the remaining leaves, which are cut off mostly on top levels, we can re-solve an LP only on the top levels $V_{\leq h}$ to obtain a vertex solution that cuts them off. This LP solution is sparse and contains at most $h \leq \log L \leq B_{\text{OPT}}$ loose nodes. Hence, all loose vertices can be selected by increasing the budget by at most $h \leq B_{\text{OPT}}$, leading to a well-structured residual problem for which one can easily find an integral solution. The following theorem summarizes this discussion. A formal proof of Theorem 15 can be found in Section 2.7.

**Theorem 15.** There exists an efficient algorithm that computes a feasible solution to a (compressed) instance of RMFC with budget $B \leq 3 \cdot \max\{\log L, B_{\text{OPT}}\}$.

In what follows, we therefore assume $B_{\text{OPT}} < \log L$ and present an efficient way to partially enumerate vertices to be protected on top levels, leading to the claimed $O(1)$-approximation.

### 2.4.2 Partial enumeration algorithm

Throughout our algorithm, we set

$$h = \lfloor \log^{(2)} L \rfloor$$

to be the threshold level defining top vertices $V_{\leq h}$ and bottom vertices $V_{>h}$. Within our enumeration procedure we will solve LPs where we explicitly

---

8 Actually, the argument we present in the following works for any $B_{\text{OPT}} = \log^{(O(1))} L$. However, we later only need it for $B_{\text{OPT}} \geq \log L$ and thus focus on this case.
include some vertex set $A \subseteq V_{\leq h}$ to be part of the protected vertices, and also exclude some set $D \subseteq V_{\leq h}$ from being protected. Our enumeration works by growing the sets $A$ and $D$ throughout the algorithm. We thus define the following LP for two disjoint sets $A, D \subseteq V_{\leq h}$:

$$\begin{align*}
\min & \quad B \\
\text{s.t.} & \quad x \in \bar{P}_B \\
& \quad B \geq 1 \\
& \quad x(u) = 1 \quad \forall u \in A \\
& \quad x(u) = 0 \quad \forall u \in D.
\end{align*}$$

(LP($A, D$))

Notice that LP($A, D$) is indeed an LP even though the definition of $\bar{P}_B$ depends on $B$ (but it does so linearly).

Throughout our enumeration procedure, the disjoint sets $A, D \subseteq V_{\leq h}$ that we consider are always such that for any $u \in A \cup D$, we have $P_u \setminus \{u\} \subseteq D$. In other words, the vertices $A \cup D \cup \{r\}$ form the vertex set of a subtree of $G$ such that no root-leaf path contains two vertices in $A$. We call a disjoint pair of sets $A, D \subseteq V_{\leq h}$ with this property a clean pair.

Before formally stating our enumeration procedure, we briefly discuss the main idea behind it. Let $OPT \subseteq V \setminus \{r\}$ be an optimal solution to our (compressed) RMFC instance corresponding to some budget $B_{OPT} \in \mathbb{Z}_{\geq 1}$. We assume without loss of generality that $OPT$ does not contain redundancies, i.e., there is precisely one vertex of $OPT$ on each leaf-root path. Assume that we already guessed some clean pair $A, D \subseteq V_{\leq h}$ of vertex sets to be protected and not to be protected, respectively, and that this guess is compatible with $OPT$, i.e., $A \subseteq OPT$ and $D \cap OPT = \emptyset$. Let $(x, B)$ be an optimal solution to LP($A, D$). Because we assume that the sets $A$ and $D$ are compatible with $OPT$, we have $B \leq B_{OPT}$ since $(B_{OPT}, \chi_{OPT})$ is feasible for LP($A, D$). We define

$$W_x = \left\{ u \in \Gamma \mid x(P_u \cap V_{>h}) \geq \frac{2}{3} \right\}$$

to be the set of leaves cut off from the root by an $x$-load of at least $\mu = \frac{2}{3}$ within bottom levels. For each $u \in \Gamma \setminus W_x$, let $f_u \in V_{\leq h}$ be the vertex
closest to the root among all vertices in \((P_u \cap V_{\leq h}) \setminus D\), and we define

\[ F_x = \{ f_u \mid u \in \Gamma \setminus W_x \} \setminus A. \quad (2.3) \]

Notice that by definition, no two vertices of \(F_x\) lie on the same leaf-root path. Furthermore, every leaf \(u \in \Gamma \setminus W_x\) is part of the subtree \(T_f\) for precisely one \(f \in F_x\). The main motivation for considering \(F_x\) is that to guess vertices in top levels, we can show that it suffices to focus on vertices lying below some vertex in \(F_x\), i.e., vertices in the set \(Q_x = V_{\leq h} \cap (\bigcup_{f \in F_x} T_f)\). To exemplify this, we first consider the special case \(\text{OPT} \cap Q_x = \emptyset\), which will also play a central role later in the analysis of our algorithm. We show that for this case we can get an \(O(1)\)-approximation to RMFC, even though we may only have guessed a proper subset \(A \subsetneq \text{OPT} \cap V_{\leq h}\) of \(\text{OPT}\) within the top levels.

**Lemma 16.** Let \((A, D)\) be a clean pair of vertices that is compatible with \(\text{OPT}\), i.e., \(A \subseteq \text{OPT}\), \(D \cap \text{OPT} = \emptyset\), and let \(x\) be an optimal solution to LP\((A, D)\). Moreover, let \((y, \bar{B})\) be an optimal solution to LP\((A, V_{\leq h} \setminus A)\). Then, if \(\text{OPT} \cap Q_x = \emptyset\), we have \(\bar{B} \leq \frac{5}{2} B_{\text{OPT}}\).

Furthermore, if \(\text{OPT} \cap Q_x = \emptyset\), by applying Theorem 14 to \(y \wedge \chi^{V_{> h}}\) with \(\mu = 1\) and \(q = 2\), a set \(R \subseteq V_{> h}\) is obtained such that \(R \cup A\) is a feasible solution to RMFC with respect to the budget \(6 \cdot B_{\text{OPT}}\).

**Proof.** Notice that \(\text{OPT} \cap Q_x = \emptyset\) implies that for each \(u \in \Gamma \setminus W_x\), we either have \(A \cap P_u \neq \emptyset\) and thus a vertex of \(A\) cuts \(u\) off from the root, or the set \(\text{OPT}\) contains a vertex on \(P_u \cap V_{> h}\). Indeed, consider a leaf \(u \in \Gamma \setminus W_x\) such that \(A \cap P_u = \emptyset\). Then \(\text{OPT} \cap Q_x = \emptyset\) implies that no vertex of \(T_{f_u} \cap V_{\leq h}\) is part of \(\text{OPT}\). Furthermore, \(P_{f_u} \setminus T_{f_u} \subseteq D\) because \((A, D)\) is a clean pair and \(f_u\) is the topmost vertex on \(P_u\) that is not in \(D\). Therefore, \(\text{OPT} \cap P_u \cap V_{\leq h} = \emptyset\), and since \(\text{OPT}\) must contain a vertex in \(P_u\), we must have \(\text{OPT} \cap P_u \cap V_{> h} \neq \emptyset\).

However, this observation implies that \(z = \frac{3}{2} (x \wedge \chi_{V_{> h}}) + (\chi_{\text{OPT}} \wedge \chi_{V_{> h}}) + \chi^A\) satisfies \(z(P_u) \geq 1\) for all \(u \in \Gamma\). Moreover we have \(z \in P_{\frac{3}{2} B + B_{\text{OPT}}}\).

---

\(^9\)For two vectors \(a, b \in \mathbb{R}^n\) we denote by \(a \wedge b \in \mathbb{R}^n\) the component-wise minimum of \(a\) and \(b\).
due to the following. First, \( x \wedge \chi^{V > h} \in P_B \) and \( \chi^{OPT} \in P_{B_{OPT}} \), which implies \( z - \chi^A \in P_{\frac{3}{2}B + B_{OPT}} \). Furthermore, \( \chi^A \in P_B \), and the vertices in \( A \) are all on levels \( V \leq h \) which are disjoint from the levels on which vertices in \( \text{supp}(z - \chi^A) \subseteq V > h \) lie, and thus do not compete for the same budget. Hence, \((z, \frac{3}{2}B + B_{OPT})\) is feasible for \( \text{LP}(A, V \leq h \setminus A) \), and thus \( \bar{B} \leq \frac{3}{2}B + B_{OPT} \leq \frac{5}{2}B_{OPT} \), as claimed.

The second part of the lemma follows in a straightforward way from Theorem 14. Observe first that each leaf \( u \in \Gamma \) is either fully cut off from the root by \( y \) on only top levels or only bottom levels because \( y \) is a \( \{0, 1\} \)-solution on the top levels \( V \leq h \), since on top levels it was fixed to \( \chi^A \) because it is a solution to \( \text{LP}(A, V \leq h \setminus A) \). Reusing the notation in Theorem 14, let \( W = \{u \in \Gamma \mid (y \wedge \chi^{V > h})(P_u) \geq 1\} \) be all leaves cut off from the root by \( y \wedge \chi^{V > h} \). By the above discussion, every leaf is thus either part of \( W \) or it is cut off from the root by vertices in \( A \). Theorem 14 guarantees that \( R \subseteq V > h \) cuts off all leaves in \( W \) from the root, and hence, \( R \cup A \) indeed cuts off all leaves from the root. Moreover, by Theorem 14, the set \( R \subseteq V > h \) is feasible with respect to the budget \( 5B_{OPT} + 1 \leq 6B_{OPT} \). Furthermore, \( A \) is feasible for budget \( B_{OPT} \) because it is a subset of \( OPT \). Since \( A \subseteq V \leq h \) and \( R \subseteq V > h \) are on disjoint levels, the set \( R \cup A \) is feasible for the budget \( 6B_{OPT} \).

Our final algorithm is based on a recursive enumeration procedure that computes a polynomial collection of clean pairs \((A, D)\). We can show that there is one pair \((A, D)\) in the collection with a corresponding LP solution \( x \) of \( \text{LP}(A, D) \) satisfying that the triple \((A, D, x)\) fulfills the conditions of Lemma 16, and thus leading to a constant-factor approximation. Our enumeration algorithm \( \text{Enum}(A, D, \gamma) \) is described below. It contains a parameter \( \gamma \in \mathbb{Z}_{\geq 0} \) that bounds the recursion depth of the enumerations.
CHAPTER 2. FIREFIGHTING ON TREES

Enum\((A, D, \gamma)\): Enumerating triples \((A, D, x)\) to find one satisfying the conditions of Lemma 16.

1. Compute optimal solution \((x, B)\) to LP\((A, D)\).
2. **If** \(B > \log L\): **stop.** Otherwise, continue with step 3.
3. Add \((A, D, x)\) to the family of triples to be considered.
4. **If** \(\gamma \neq 0\):
   
   //recursion depth not yet reached
   
   *For* \(u \in F_x\):
   
   //\(F_x\) is defined as in (2.3)
   
   Recursive call to Enum\((A \cup \{u\}, D, \gamma - 1)\).
   
   Recursive call to Enum\((A, D \cup \{u\}, \gamma - 1)\).

Notice that for any clean pair \((A, D)\) and \(u \in F_x\), the two pairs \((A \cup \{u\}, D)\) and \((A, D \cup \{u\})\) are clean, too. Hence, if we start Enum\((A, D, \gamma)\) with a clean pair \((A, D)\), we will encounter only clean pairs during all recursive calls. Moreover, observe that the recursive calls to the enumeration algorithm depend on an LP solution computed at the beginning of the algorithm. Hence, the exploration of new clean pairs is guided by LP solutions.

The key property of the above enumeration procedure is that only a small recursion depth \(\gamma\) is needed for the enumeration algorithm to explore a good triple \((A, D, x)\), which satisfies the conditions of Lemma 16, if we start with the trivial clean pair \((\emptyset, \emptyset)\). Furthermore, due to step 2, we always have \(B \leq \log L\) whenever the algorithm is in step 4. As we will see later, this allows us to prove that \(|F_x|\) is small, which will limit the width of our recursive calls, and leads to an efficient procedure as highlighted in the following Lemma.

**Lemma 17.** Let \(\bar{\gamma} = 2(\log L)^2 \log^{(2)} L\). The enumeration procedure Enum\((\emptyset, \emptyset, \bar{\gamma})\) runs in polynomial time. Furthermore, if \(B_{\text{OPT}} \leq \log L\), then Enum\((\emptyset, \emptyset, \bar{\gamma})\) will encounter a triple \((A, D, x)\) satisfying the conditions of Lemma 16, i.e.,

(i) \((A, D)\) is a clean pair,

(ii) \(A \subseteq \text{OPT}\),

(iii) \(D \cap \text{OPT} = \emptyset\), and
(iv) $\text{OPT} \cap Q_x = \emptyset$. 

Hence, combining Lemma 17 and Lemma 16 completes our enumeration procedure and implies the following result.

**Corollary 18.** Let $I$ be an RMFC instance on $L$ levels on a graph $G = (V, E)$ with budgets $B_\ell = 2^\ell \cdot B$ for $\ell \in [L]$. Let $B_{\text{OPT}} \in \mathbb{Z}_{\geq 1}$ be the optimal budget value for $B$ for instance $I$—recall that $B = B_{\text{OPT}}$ in instance $I$ implies that level $\ell \in [L]$ has budget $2^\ell \cdot B_{\text{OPT}}$—and let $B_{\text{OPT}}^{\text{orig}}$ be the optimal budget for

**Proof.** It suffices to run $\text{Enum}(\emptyset, \emptyset, \bar{\gamma})$ to first efficiently obtain a family of triples $(A_i, D_i, x_i)$, where $(A_i, D_i)$ is a clean pair, and $x_i$ is an optimal solution to $\text{LP}(A_i, D_i)$. By Lemma 17, one of these triples satisfies the conditions of Lemma 16. (Notice that these conditions cannot be checked since it would require knowledge of OPT.) For each triple $(A_i, D_i, x_i)$ we obtain a corresponding solution for $I$ following the construction described in Lemma 16. More precisely, we first compute an optimal solution $(y_i, \bar{B}_i)$ to $\text{LP}(A_i, V_{\leq h} \setminus A_i)$. Then, by applying Theorem 14 to $y_i \wedge \chi^{V_{> h}}$ with $\mu = 1$ and $q = 2$, a set of vertices $R_i \subseteq V_{> h}$ is obtained such that $R_i \cup A_i$ is feasible for $I$ for some budget $B_i$. Among all such sets $R_i \cup A_i$, we return the one with minimum $B_i$. Because Lemma 17 guarantees that one of the triples $(A_i, D_i, x_i)$ satisfies the conditions of Lemma 16, we have by Lemma 16 that the best protection set $Q = R_j \cup A_j$ among all $R_i \cup A_i$ has a budget $B_j$ satisfying $B_j \leq 6B_{\text{OPT}}$. \qed

### 2.4.3 Summary of our $O(1)$-approximation for RMFC

Starting with an RMFC instance $I^{\text{orig}}$ on a tree with $N$ vertices, we first apply our compression result, Theorem 13, to obtain an RMFC instance $I$ on a graph $G = (V, E)$ with depth $L = O(\log N)$, and non-uniform budgets $B_\ell = 2^\ell \cdot B$ for $\ell \in [L]$. Let $B_{\text{OPT}} \in \mathbb{Z}_{\geq 1}$ be the optimal budget value for $B$ for instance $I$—recall that $B = B_{\text{OPT}}$ in instance $I$ implies that level $\ell \in [L]$ has budget $2^\ell \cdot B_{\text{OPT}}$—and let $B_{\text{OPT}}^{\text{orig}}$ be the optimal budget for
$\mathcal{I}^{\text{orig}}$. By Theorem 13, we have $B_{\text{OPT}} \leq B_{\text{OPT}}^{\text{orig}}$, and any solution to $\mathcal{I}$ using budget $B$ can efficiently be transformed into one of $\mathcal{I}^{\text{orig}}$ of budget $2B$.

We now invoke Theorem 15 and Corollary 18. Both guarantee that a solution to $\mathcal{I}$ with certain properties can be computed efficiently. Among the two solutions derived from Theorem 15 and Corollary 18, we consider the one $(Q, B)$ with lower budget $B$, where $Q \subseteq V \setminus \{r\}$ is a set of vertices to protect, feasible for budget $B$. If $B \geq \log L$, then Theorem 15 implies $B \leq 3B_{\text{OPT}}$, otherwise Corollary 18 implies $B \leq 6B_{\text{OPT}}$. Hence, in any case we have a $6$-approximation for $\mathcal{I}$. As mentioned before, Theorem 13 implies that the solution $Q$ can efficiently be transformed into a solution for the original instance $\mathcal{I}^{\text{orig}}$ that is feasible with respect to the budget $2B \leq 12B_{\text{OPT}} \leq 12B_{\text{OPT}}^{\text{orig}}$, thus implying Theorem 1.

### 2.5 Details on compression results

In this section, we present the proofs for our compression results, Theorem 10 and Theorem 13. We start by proving Theorem 10. The same ideas are used with a slight adaptation in the proof of Theorem 13.

We call an instance $\mathcal{I}$ obtained from an instance $\mathcal{I}$ by a sequence of down-push operations a *push-down of $\mathcal{I}$*. We prove Theorem 10 by proving the following result, of which Theorem 10 is an immediate consequence, as we will soon show.

**Theorem 19.** Let $\mathcal{I}$ be a unit-budget Firefighter instance with depth $L$, and let $\delta \in (0, 1)$. Then one can efficiently construct a push-down $\overline{\mathcal{I}}$ of $\mathcal{I}$ such that

1. $\text{val}(\text{OPT}(\overline{\mathcal{I}})) \geq (1 - \delta) \text{val}(\text{OPT}(\mathcal{I}))$, and
2. $\overline{\mathcal{I}}$ has nonzero budget on only $O\left(\frac{\log L}{\delta}\right)$ levels.

Before we prove Theorem 19, we show how it implies Theorem 10.

**Proof of Theorem 10.** We start by showing how levels of zero budget can be removed through the following *contraction operation*. Let $\ell \in \{2, \ldots, L\}$
be a level whose budget is zero. For each vertex \( u \in V_{\ell-1} \) we contract all edges from \( u \) to its children and increase the weight \( w(u) \) of \( u \) by the sum of the weights of all of its children. Formally, if \( u \) has children \( v_1, \ldots, v_k \in V_{\ell} \), the vertices \( u, v_1, \ldots, u_k \) are replaced by a single vertex \( z \) with weight \( w(z) = w(u) + \sum_{i=1}^{k} w(v_i) \), and \( z \) is adjacent to the parent of \( u \) and to all children of \( v_1, \ldots, v_k \). One can easily observe that this is an “exact” transformation in the sense that any solution before the contraction remains one after contraction and vice versa (when identifying the vertex \( z \) in the contracted version with \( v \)); moreover, solutions before and after contraction have the same value.

Now, by first applying Theorem 19 and then applying the latter contraction operations level by level to all levels \( \ell \in \{2, \ldots, L\} \) with zero budget (in an arbitrary order), we obtain an equivalent instance with the desired depth, thus satisfying the conditions of Theorem 10.

It remains to prove Theorem 19.

**Proof of Theorem 19.** Consider a unit-budget Firefighter instance on a tree \( G = (V, E) \) with depth \( L \). The push-down \( \overline{I} \) that we construct will have nonzero budgets precisely on the following levels \( \mathcal{L} \subseteq [L] \):

\[
\mathcal{L} = \left\{ \left\lceil (1 + \delta)^j \right\rceil \bigg| j \in \left\{ 0, \ldots, \left\lfloor \frac{\log L}{\log(1+\delta)} \right\rfloor \right\} \right\} \cup \{L\}.
\]

For simplicity, let \( \mathcal{L} = \{\ell_1, \ldots, \ell_k\} \) with \( 1 = \ell_1 < \ell_2 < \cdots < \ell_k = L \). Hence, \( k = O\left(\frac{\log L}{\log(1+\delta)}\right) = O\left(\frac{\log L}{\delta}\right) \). The push-down \( \overline{I} \) is obtained by pushing any budget on a level not in \( \mathcal{L} \) down to the next level in \( \mathcal{L} \). Formally, for \( i \in [k] \), the budget \( B_{\ell_i} \) at level \( \ell_i \) is given by \( B_{\ell_i} = \ell_i - \ell_{i-1} \), where we set \( \ell_0 = 0 \). Moreover, \( B_{\ell} = 0 \) for \( \ell \in [L] \setminus \mathcal{L} \). Clearly, the instance \( \overline{I} \) can be constructed efficiently. Furthermore, the number of levels with nonzero budget is equal to \( k = O\left(\frac{\log L}{\delta}\right) \) as desired. It remains to show point (i) of Theorem 19.

To show (i), consider an optimal redundancy-free solution \( S^* \subseteq V \) of \( I \); hence, \( \text{val}({\text{OPT}}(I)) = \sum_{u \in S^*} w(T_u) \) and no two vertices of \( S^* \) lie on the same leaf-root path. We will show that there is a feasible solution \( \overline{S} \) to \( \overline{I} \).
such that $\overline{S} \subseteq S^*$ and the value of $\overline{S}$ is at least $(1 - \delta) \text{val}(\text{OPT}(I))$. Notice that since $S^*$ is redundancy-free, any subset of $S^*$ is also redundancy-free. Hence, the value of the set $\overline{S}$ to construct will be equal to $\sum_{u \in \overline{S}} w(T_u)$. The set $S^*$ being (budget-)feasible for $I$ implies

$$|S^* \cap V_{\leq \ell}| \leq \ell \quad \forall \ell \in [L].$$

(2.4)

Analogously, a set $S \subseteq V$ is feasible for $\overline{I}$ if and only if

$$|S \cap V_{\leq \ell}| \leq \sum_{i=1}^{\ell} B_i \quad \forall \ell \in [L].$$

Hence, we want to show that there is a set $\overline{S}$ satisfying the above system and such that $\sum_{u \in \overline{S}} w(T_u) \geq (1 - \delta) \text{val}(\text{OPT}(I))$. Notice that in (2.5), the constraint for any $\ell \in [L - 1]$ such that $B_{\ell+1} = 0$ is redundant due to the constraint for level $\ell + 1$ which has the same right-hand side but a larger left-hand side. Thus, system (2.5) is equivalent to the following system

$$|S \cap V_{\leq \ell_{i+1}-1}| \leq \ell_i \quad \forall i \in [k - 1],$$

$$|S \cap V| \leq L.$$ 

(2.6)

To show that there is a good subset $\overline{S} \subseteq S^*$ that satisfies (2.6) we use a polyhedral approach. Observe that (2.5) is the constraint system of a laminar matroid (see [83, Volume B] for more information on matroids). Hence, the convex hull of all characteristic vectors $\chi^S \in \{0, 1\}^V$ of sets $S \subseteq S^*$ satisfying (2.6) is given by the following polytope

$$P = \left\{ x \in [0, 1]^V \left| \begin{array}{l} x(V_{\leq \ell_{i+1}-1}) \leq \ell_i \quad \forall i \in [k - 1], \\
 x(V) \leq L, \\
 x(V \setminus S^*) = 0 \end{array} \right. \right\}.$$ 

Alternatively, to see that $P$ indeed describes the correct polytope, without relying on matroids, one can observe that its constraint matrix is totally unimodular because it has the consecutive-ones property with respect to the columns.
Thus there exists a set $\overline{S} \subseteq S^*$ with $\sum_{u \in \overline{S}} w(T_u) \geq (1 - \delta) \text{val}(\text{OPT}(I))$ if and only if
\[
\max \left\{ \sum_{u \in S^*} x(u) \cdot w(T_u) \mid x \in P \right\} \geq (1 - \delta) \text{val}(\text{OPT}(I)). \tag{2.7}
\]
To show (2.7), and thus complete the proof, we show that $y = \frac{1}{1+\delta} \chi^{S^*} \in P$. This will indeed imply (2.7) since the objective value of $y$ satisfies
\[
\sum_{u \in S^*} y(u) \cdot w(T_u) = \frac{1}{1+\delta} \text{val}(\text{OPT}(I)) \geq (1 - \delta) \text{val}(\text{OPT}(I)).
\]
To see that $y \in P$, notice that $y(V \setminus S^*) = 0$ and $y(V) = \frac{1}{1+\delta}|S^*| \leq \frac{1}{1+\delta} L \leq L$, where the first inequality follows by $S^*$ satisfying (2.4) for $\ell = L$. Finally, for $i \in [k - 1]$, we have
\[
y(V_{\leq \ell_{i+1} - 1}) = \frac{1}{1+\delta} |S^* \cap V_{\leq \ell_{i+1} - 1}| \leq \frac{1}{1+\delta} (\ell_{i+1} - 1),
\]
where the inequality follows from $S^*$ satisfying (2.4) for $\ell = \ell_{i+1} - 1$. It remains to show $\ell_{i+1} - 1 \leq (1 + \delta) \ell_i$ to prove $y \in P$. Let $\alpha \in \mathbb{Z}_{\geq 0}$ be the smallest integer for which we have $\ell_{i+1} = \lceil (1 + \delta)^\alpha \rceil$. In particular, this implies $\ell_i = \lceil (1 + \delta)^{\alpha-1} \rceil$. We thus obtain
\[
\ell_{i+1} - 1 \leq (1 + \delta)^\alpha = (1 + \delta)(1 + \delta)^{\alpha-1} \leq (1 + \delta)\ell_i,
\]
as desired. \qed

We conclude with the proof of Theorem 13.

**Proof of Theorem 13.** We start by describing the construction of $G' = (V', E')$. As is the case in the proof of Theorem 10, we first change the budget assignment of the instance and then contract all levels with zero budgets. Notice that, for a given budget $B$ per layer, we can consider an RMFC instance as a Firefighter instance, where each leaf $u \in \Gamma$ has weight $w(u) = 1$, and all other weights are zero. Since our goal is to save all leaves, we want to save vertices of total weight $|\Gamma|$. 

For simplicity of presentation we assume that $L$ is two less than a power of 2, i.e., $L = 2^\alpha - 2$ for some $\alpha \in \mathbb{Z}_{\geq 2}$. This assumption does not compromise generality, as one can always augment the original tree with one path starting from the root and going down to level $2^{\lceil \log(L + 2) \rceil} - 2$.

The set of levels in which the transformed instance will have nonzero budget is

$$\mathcal{L} = \{2^j - 1 \mid j \in \{1, \ldots, \lceil \log(L + 2) \rceil - 1\}\}.$$

However, instead of down-pushes we will do up-pushes where budget is moved upwards. More precisely, the budget of any level $\ell \in [L] \setminus \mathcal{L}$ will be assigned to the first level in $\mathcal{L}$ that is above $\ell$, i.e., has a smaller index than $\ell$. As for the Firefighter case, we now remove all 0-budget levels using contraction, which will lead to a new weight function $w'$ on the vertices. Since our goal is to save the weight of the whole tree, we can remove for each vertex $u$ with $w'(u) > 0$, the subtree below $u$. This does not change the problem since we have to save $u$, and thus will anyway also save its subtree. This finishes our construction of $G' = (V', E')$, and the task is again to remove all leaves of $G'$. Notice that $G'$ has $L' \leq |\mathcal{L}| = O(\log L)$ many levels, and level $\ell \in [L']$ has a budget of $B2^\ell$ as desired. Analogous to the discussion for compression in the context of the Firefighter problem we have that if the original problem is feasible, then so is the RMFC problem on $G'$ with budgets $B2^\ell$; this is an immediate consequence of the fact that we did down-pushes. Indeed, before performing the contraction operations (which do not change the problem), the original RMFC problem was a push-down of the one we constructed.

Similarly, one can observe that before contraction, the instance we obtained is itself dominated by a push-down of the original instance with budgets $2B$ on each level, where with domination we mean that on each level, the budget of the push-down is at least as high as the one of the instance we constructed. Hence, analogously to the compression result for the Firefighter case, any solution to the RMFC problem on $G'$ can efficiently be transformed into a solution to the original RMFC problem on $G$ with budgets $2B$ on each level. \qed
2.6 Missing details for Firefighter PTAS

In this section we present the missing proofs for our PTAS for the Firefighter problem.

We start by proving Lemma 4, showing that any vertex solution $x$ to $\text{LP}_{\text{FF}}$ has few $x$-loose vertices. More precisely, the proof below shows that the number of $x$-loose vertices is upper bounded by the number of tight budget constraints. The precise same reasoning used in the proof of Lemma 4 can also be applied in further contexts, in particular for the RMFC problem.

Proof of Lemma 4

Let $x$ be a vertex of the polytope defining the feasible set of $\text{LP}_{\text{FF}}$. Hence, $x$ is uniquely defined by $|V \setminus \{r\}|$ many linearly independent and tight constraints of this polytope. Notice that the tight constraints can be partitioned into three groups:

(i) Tight nonnegativity constraints, one for each vertex in $F_1 = \{u \in V \setminus \{r\} \mid x(u) = 0\}$.

(ii) Tight budget constraints, one for each level in $F_2 = \{\ell \in [L] \mid x(V_{\leq \ell}) = \sum_{i=1}^{\ell} B_i\}$.

(iii) Tight leaf constraints, one for each vertex in $F_3 = \{u \in \Gamma \mid x(P_u) = 1\}$.

Due to potential degeneracies of the polytope describing the feasible set of $\text{LP}_{\text{FF}}$ there may be several options to describe $x$ as the unique solution to a full-rank linear subsystem of the constraints described by $F_1 \cup F_2 \cup F_3$. We consider a system that contains all tight nonnegativity constraints, i.e., constraints corresponding to $F_1$, and complement these constraints with arbitrary subsets $F_2' \subseteq F_2$ and $F_3' \subseteq F_3$ of budget and leaf constraints that lead to a full rank linear system corresponding to the constraints $F_1 \cup F_2' \cup F_3'$. Hence

$$|F_1| + |F_2'| + |F_3'| = |V| - 1. \quad (2.8)$$

Let $V^L \subseteq \text{supp}(x)$ and $V^T \subseteq \text{supp}(x)$ be the $x$-loose and $x$-tight vertices, respectively. We first show $|F_3'| \leq |V^T|$. For each leaf $u \in F_3'$, let $f_u \in V^T$
be the first vertex on the unique $u$-root path that is part of $\text{supp}(x)$. In particular, if $u \in \text{supp}(x)$ then $f_u = u$. Clearly, $f_u$ must be an $x$-tight vertex because the path constraint with respect to $u$ is tight. Notice that for any distinct vertices $u_1, u_2 \in F'_3$, we must have $f_{u_1} \neq f_{u_2}$. Assume by sake of contradiction that $f_{u_1} = f_{u_2}$. However, this implies $\chi_{P_{u_1}} - \chi_{P_{u_2}} \in \text{span}(\{\chi_v | v \in F_1\})$, since $P_{u_1} \Delta P_{u_2} := (P_{u_1} \setminus P_{u_2}) \cup (P_{u_2} \setminus P_{u_1}) \subseteq F_1$, and leads to a contradiction because we exhibited a linear dependence among the constraints corresponding to $F'_3$ and $F_1$. Hence, $f_{u_1} \neq f_{u_2}$ which implies that the map $u \mapsto f_u$ from $F'_3$ to $V_T$ is injective and thus

$$|F'_3| \leq |V_T|. \quad (2.9)$$

We thus obtain

$$|\text{supp}(x)| = |V| - 1 - |F_1| \quad (\text{supp}(x) = \{u \in V \setminus \{r\} | u \not\in F_1\})$$

$$= |F'_2| + |F'_3| \quad (\text{by } (2.8))$$

$$\leq |F'_2| + |V_T| \quad (\text{by } (2.9)),$$

which leads to the desired result since

$$|V^L| = |\text{supp}(x)| - |V^T| \leq |F'_2| \leq L.$$

\[\square\]

**Proof of Lemma 6**

We construct the $\eta$-split $Q$ in two phases as follows. First we construct a set $\overline{Q} \subseteq V \setminus \{r\}$ with $|\overline{Q}| \leq w(V)/\eta$ fulfilling only the first property, i.e., the graph $G[V \setminus \overline{Q} \cup \{r\}]$ obtained from $G$ by removing $\overline{Q} \cup \{r\}$ has connected components each of weight at most $\eta$. Finally, the set $Q$ consists of $\overline{Q}$ and all vertices $u \in V \setminus \{r\}$ such that $u$ is a nearest common ancestor of any two vertices in $\overline{Q}$.

We construct $\overline{Q}$ step by step, starting with $\overline{Q} = \emptyset$. During the construction of $\overline{Q}$, we maintain a subgraph $G' = (V', E')$ of $G$, which we set initially to $G' = G$. We say that a vertex $u \in V' \setminus \{r\}$ is $G'$-heavy, if the weight of its
subtree is strictly larger than \( \eta \), i.e., \( w(T'_u) > \eta \), where \( T'_u \) is the subtree of \( u \) in \( G' \). As long as there is a \( G' \)-heavy vertex, we do the following. We choose an arbitrary \( G' \)-heavy vertex \( u \) such that there is no \( G' \)-heavy vertex in \( T'_u \). We set \( \overline{Q} = \overline{Q} \cup \{ u \} \) and update \( G' \) by removing from \( G' \) all vertices in \( T'_u \). This procedure stops as soon as we end up with a graph \( G' \) that does not contain any \( G' \)-heavy vertex anymore. Clearly, this construction guarantees that all connected components of \( G[V \setminus (\overline{Q} \cup \{ r \})] \) have weight at most \( \eta \). Furthermore, since each vertex \( u \) that gets added to \( \overline{Q} \) removes from \( G' \) a subtree \( T'_u \) of weight strictly greater than \( \eta \), we have \( |Q| < \frac{w(V)}{\eta} \).

Consider now the set \( Q \), which, we recall, is obtained by adding to \( \overline{Q} \) all vertices of \( V \setminus \{ r \} \) that are a nearest common ancestor of two vertices in \( \overline{Q} \). It remains to show that there are at most \( \frac{w(V)}{\eta} \) such nearest common ancestors. To see this, consider all vertices \( W \subseteq V \) that lie on some path between the root and some vertex in \( Q \), i.e., \( W = \cup_{q \in Q} P_q \). Notice that each leaf of the subtree \( G[W] \) of \( G \) over the vertices \( W \) is contained in \( \overline{Q} \). Furthermore, any nearest common ancestor in \( V \setminus \{ r \} \) of two vertices in \( \overline{Q} \) is a vertex of degree at least 3 in \( G[W] \). However, the number of vertices of degree at least 3 in any tree is strictly less than the number of leaves, because the average degree of any tree is strictly less than 2. Hence, \( |Q \setminus \overline{Q}| < |\overline{Q}| \), which implies \( |Q| < 2|\overline{Q}| \leq \frac{2w(V)}{\eta} \), as desired.

To conclude the proof, it remains to note that the above construction of \( Q \) can easily be performed in polynomial time.

\( \square \)

**Proof of Theorem 9**

Let \( \text{OPT} \subseteq V \setminus \{ r \} \) be an optimal solution to the considered Firefighter instance with value \( \text{val(OPT)} \). Observe first that the optimal value \( \text{val}(x^*) \) of \( \text{LP}_{\text{FF}}(S^*) \) satisfies \( \text{val}(x^*) \geq \text{val(OPT)} \), because one of the sets \( S \subseteq Q \) corresponds to \( \text{OPT} \), namely \( S = \{ q \in Q \mid P_q \cap \text{OPT} \neq \emptyset \} \), and for this set \( S \) the characteristic vector \( \chi_{\text{OPT}} \) of \( \text{OPT} \) is feasible for \( \text{LP}_{\text{FF}}(S) \). It now follows from Lemma 8 that the set \( U \) returned by Algorithm 1 satisfies \( \text{val}(U) = \text{val}(y^*) \geq \text{val}(x^*) - L \cdot \eta \geq \text{val(OPT)} - L \cdot \eta \), as desired.

The running time bound for the algorithm follows immediately from the fact
that during the second step of the algorithm $2^{|Q|} \leq 2^{w(V)/\eta}$ many sets $S$ have to be considered, and all other operations are clearly efficient.  

Proof of Theorem 11

Within this proof we focus on protection sets where the budget available for any level is spent on the same level (and not a later one). As discussed, there is always an optimal protection set with this property.

Let $B_\ell \in \mathbb{Z}_{\geq 0}$ be the budget available at level $\ell \in [L]$ and let $\lambda_\ell = \lambda B_\ell$. We construct the tree $G'$ using the following greedy procedure. Process the levels of $G$ from the first one to the last one. At every level $\ell \in [L]$, pick $\lambda_\ell$ vertices $u_1^\ell, \ldots, u_{\lambda_\ell}^\ell$ at the $\ell$-th level of $G$ greedily, i.e., pick each next vertex such that the subtree corresponding to that vertex has largest weight among all remaining vertices in the level. After each selection of a vertex the greedy procedure can no longer select any vertex in the corresponding subtree in subsequent iterations.\footnote{For $\lambda = 1$ this procedure produces a set of vertices, which comprise a $\frac{1}{2}$-approximation for the Firefighter problem, as it coincides with the greedy algorithm of Hartnell and Li [60].}

Now, the tree $G'$ is constructed by deleting from $G$ any vertex that is both not contained in any subtree $T_{u_i}^\ell$, and not contained in any path $P_{u_i}^\ell$ for $\ell \in [L]$ and $i \in [\lambda_\ell]$. In other words, if $U \subseteq V$ is the set of all leaves of $G$ that were disconnected from the root by the greedy algorithm, then we consider the subtree of $G$ induced by the vertices $\cup_{u \in U} P_u$. Finally, the weights of vertices on the paths $P_{u_i}^\ell \setminus \{u_i^\ell\}$ for $\ell \in [L]$ and $i \in [\lambda_\ell]$ are reduced to zero. This concludes the construction of $G' = (V', E')$ and the new weight function $w'$. Denote by $D_\ell = \{u_1^\ell, \ldots, u_{\lambda_\ell}^\ell\}$ the set of vertices chosen by the greedy procedure in level $\ell$, and let $D = \cup_{\ell \in [L]} D_\ell$. Observe that by construction we have that each vertex with non-zero weight is in the subtree of a vertex in $D$, i.e.,

$$w'(V') = \sum_{u \in D} w'(T_u').$$

The latter immediately implies point (ii) of Theorem 11 because the vertices $D$ can be partitioned into $\lambda$ many vertex sets that are budget-feasible and
can thus be protected in a Firefighter solution. Hence an optimal solution to
the Firefighter problem on $G'$ covers at least a $\frac{1}{\lambda}$-fraction of the total weight
of $G'$.

It remains to prove point (i) of the theorem. Let $S^* = S_1^* \cup \cdots \cup S_L^*$
be the vertices protected in some optimal solution in $G$, where $S_{\ell}^* \subseteq V_{\ell}$
are the vertices protected in level $\ell$ (and hence $|S_{\ell}^*| \leq B_{\ell}$). Without loss
of generality, we assume that $S^*$ is redundancy-free. For distinct vertices
$u, v \in V$ we say that $u$ covers $v$ if $v \in T_u \setminus \{u\}$.

For $\ell \in [L]$, let $I_\ell = S_{\ell}^* \cap D_\ell$ be the set of vertices protected by the
optimal solution that are also chosen by the greedy algorithm in level $\ell$. Furthermore, let $J_\ell \subseteq S_{\ell}^*$ be the set of vertices of the optimal solution that
are covered by vertices chosen by the greedy algorithm in earlier iterations,
i.e., $J_\ell = S_{\ell}^* \cap \bigcup_{u \in D_1 \cup \cdots \cup D_{\ell-1}} T_u$. Finally, let $K_\ell = S_{\ell}^* \setminus (I_\ell \cup J_\ell)$ be all
other optimal vertices in level $\ell$. Clearly, $S_{\ell}^* = I_\ell \cup J_\ell \cup K_\ell$ is a partition of
$S_{\ell}^*$.

Consider a vertex $u \in K_\ell$ for some $\ell \in [L]$. From the guarantee of the
greedy algorithm it holds that for every vertex $v \in D_\ell$ we have $w'(T_v) = w(T_v) \geq w(T_u)$. The same does not necessarily hold for covered vertices.
On the other hand, covered vertices are contained in $G'$ with their original
weights. We exploit these two properties to prove the existence of a solution
in $G'$ of almost the same weight as $S^*$.

To prove the existence of a good solution we construct a solution $A = A_1 \cup \cdots \cup A_L$ with $A_\ell \subseteq V_{\ell}$ and $|A_\ell| \leq B_{\ell}$ randomly, and prove a bound
on its expected quality. We process the levels of the tree $G'$ top-down to
construct $A$ step by step. This clearly does not compromise generality.
Recall that we only need to prove the existence of a good solution, and not
compute it efficiently. We can hence assume the knowledge of $S^*$ in the
construction of $A$. To this end assume that all levels $\ell' < \ell$ were already
processed, and the corresponding sets $A_{\ell'}$ were constructed. The set $A_\ell$ is
constructed as follows:

1. Include in $A_\ell$ all vertices in $I_\ell$.

2. Include in $A_\ell$ all vertices in $J_\ell$ that are not covered by vertices in
A_1 \cup \cdots \cup A_{\ell-1} \text{ (vertices selected so far).}

3. Include in A_\ell a uniformly random subset of |K_\ell| vertices from D_\ell \setminus I_\ell.

It is easy to verify that the latter algorithm returns a redundancy-free solution, as no two chosen vertices in A lie on the same path to the root. Next, we show that the expected weight of vertices saved by A is at least \((1 - \frac{1}{\lambda}) \operatorname{val}(\operatorname{OPT}(I))\), which will prove our claim, since then at least one solution has the desired quality.

Since we only need a bound on the expectation we can focus on a single level \(\ell \in [L]\) and show that the contribution of vertices in A_\ell is in expectation at least \(1 - \frac{1}{\lambda}\) times the contribution of the vertices in \(S_\ell^*\). Observe that the vertices in I_\ell are contained both in \(S_\ell^*\) and in A_\ell, hence it suffices to show that the contribution of \(A_\ell \setminus I_\ell\) is at least \(1 - \frac{1}{\lambda}\) times the contribution of \(S_\ell^* \setminus I_\ell\), in expectation. Also, recall that every vertex in D_\ell contributes at least as much as any vertex in K_\ell, by the greedy selection rule. It follows that the \(|K_\ell|\) randomly selected vertices in A_\ell have at least as much contribution as the vertices in K_\ell. Consequently, to prove the claim is suffices to bound the expected contribution of vertices in \(A_\ell \cap J_\ell\) with respect to the contribution of \(J_\ell\). Since \(A_\ell \cap J_\ell \subseteq J_\ell\) it suffices to show that every vertex \(u \in J_\ell\) is also present in A_\ell with probability at least \(1 - \frac{1}{\lambda}\).

To bound the latter probability we make use of the random choices in the construction of A as follows. Let \(\ell' < \ell\) be the level at which for some \(w \in D_{\ell'}\) it holds that \(u \in T_w\). In other words, \(\ell'\) is the level that contains the ancestor of \(u\) that was chosen by the greedy construction of \(G'\). Now, since \(S^*\) is redundancy-free, and by the way that \(A\) is constructed, it holds that if \(u \not\in A_\ell\) then \(w \in A_{\ell'}\), namely if \(u\) is covered, it can only be covered by the unique ancestor \(w\) of \(u\) that was chosen in the greedy construction of \(G'\). Furthermore, in such a case the vertex \(w\) was selected randomly in the third step of the \(\ell'\)-th iteration. Put differently, the probability that the vertex \(u\) is covered is exactly the probability that its ancestor \(w\) is chosen randomly to be part of \(A_{\ell'}\). Since these vertices are chosen to be a random subset of \(|K_{\ell'}|\) vertices from the set \(D_{\ell'} \setminus I_{\ell'}\), this probability is at most

\[
\frac{|K_{\ell'}|}{|D_{\ell'}| - |I_{\ell'}|} = \frac{|K_{\ell'}|}{\lambda B_{\ell'} - |I_{\ell'}|} \leq \frac{1}{\lambda},
\]
where the last inequality follows from $|K_{\ell'}| + |I_{\ell'}| \leq B_{\ell'}$. This implies that $u \in A_{\ell}$ with probability of at least $1 - \frac{1}{\lambda}$, as required and concludes the proof of the theorem.

\section*{Proof of Theorem 12}

We start by showing that Algorithm 2 indeed returns a $(1 - \epsilon)$-approximation, before showing that it is efficient. Let $\nu$, $\nu'$, and $\nu''$ be the optimal values of the instances $\mathcal{I}$, $\mathcal{I}'$, and $\mathcal{I}''$, respectively. By Theorem 10 we have

$$\nu' \geq \left(1 - \frac{\epsilon}{3}\right) \nu,$$

and Theorem 11 implies

$$\nu'' \geq \left(1 - \frac{1}{\lambda}\right) \nu' \geq \left(1 - \frac{\epsilon}{3}\right) \nu'.$$

Hence,

$$\nu'' \geq \left(1 - \frac{2\epsilon}{3}\right) \nu. \quad (2.10)$$

Due to Theorem 9, the solution $U''$ returned in step 3 of the algorithm has value in $\mathcal{I}''$ of at least $\nu'' - \eta L''$. By Theorem 11 and Theorem 10, the value of $U$ in $\mathcal{I}$ is at least as high as the value of $U''$ in $\mathcal{I}''$, thus implying

$$\text{val}(U) \geq \nu'' - \eta L''.$$

Expanding this inequality, we get

$$\text{val}(U) \geq \nu'' - \eta L''$$

$$\begin{align*}
&= \nu'' - \frac{1}{12} \epsilon^2 w''(V'') \
&\geq \nu'' - \frac{\lambda}{12} \epsilon^2 \nu'' \
&\geq \left(1 - \frac{\epsilon}{3}\right) \nu'' \
&\geq (1 - \epsilon) \nu. \quad \text{(by (2.10)),}
\end{align*}$$
as desired. It remains to show that the algorithm is efficient.

Clearly, all steps of Algorithm 2 are efficient except for possibly the call to Algorithm 1. By Theorem 9, we know that the call to Algorithm 1 runs in polynomial time if \( w''(V'') / \eta \) is logarithmic in the input size of \( \mathcal{I} \). This is indeed the case since

\[
\frac{w''(V'')}{\eta} = \frac{12L''}{\epsilon^2} \quad \text{(because } \eta = \frac{1}{12} \frac{\epsilon^2}{L''} w''(V''))
\]

\[
= O \left( \frac{\log n}{\epsilon^3} \right),
\]

where \( n \) are the number of vertices in \( \mathcal{I} \), and the second equality holds because of the following. The compression results, Theorem 10, implies that the depth \( L' \) of the tree in \( \mathcal{I}' \) satisfies \( L' = O(\log n / \epsilon) \). Furthermore, pruning only does node deletions and weight reductions, thus not increasing the depth of the tree. Hence, \( L'' \leq L' = O(\log n / \epsilon) \). Thus, Algorithm 2 indeed runs in polynomial time. \( \square \)

2.7 Missing details for \( O(1) \)-approximation for RMFC

This section contains the missing proofs for our 12-approximation for RMFC.

Proof of Theorem 14

To prove Theorem 14 we first show the following result, based on which Theorem 14 follows quite directly.

Lemma 20. Let \( B \in \mathbb{R}_{\geq 1}, \eta \in (0, 1], k \in \mathbb{Z}_{\geq 1}, \text{ and } \ell_1 = \lfloor \log^k L \rfloor, \ell_2 = \lfloor \log^{k-1} L \rfloor \). Let \( x \in P_B \) with \( \text{supp}(x) \subseteq V(\ell_1, \ell_2) := V_{> \ell_1} \cap V_{\leq \ell_2} \), and we define \( Y = \{ u \in \Gamma \mid x(P_u) \geq \eta \} \). Then one can efficiently compute a set \( R \subseteq V(\ell_1, \ell_2) \) such that
2.7. **MISSING DETAILS FOR O(1)-APPROX. FOR RMFC**

(i) \( R \cap P_u \neq \emptyset \) \( \forall u \in Y \), and
(ii) \( \chi^R \in P_{\tilde{B}} \), where \( \tilde{B} = \frac{1}{\eta} B + 1 \).

We first observe that Lemma 20 indeed implies Theorem 14.

**Proof of Theorem 14.** For \( k = 1, \ldots, q \), let \( \ell^k_1 = \lfloor \log^{(k)} L \rfloor \) and \( \ell^k_2 = \lfloor \log^{(k-1)} L \rfloor \), and we define \( x^k \in P_B \) by \( x^k = x \wedge \chi^{V(\ell^k_1, \ell^k_2)} \). Hence, \( x = \sum_{k=1}^q x^k \). For each \( k \in [q] \), we apply Lemma 20 to \( x^k \) with \( \eta = \frac{\mu}{q} \) to obtain a set \( R^k \subseteq V(\ell^k_1, \ell^k_2) \) satisfying

(i) \( R^k \cap P_u \neq \emptyset \) \( \forall u \in Y^k = \{ u \in \Gamma \mid x^k(P_u) \geq \eta \} \), and
(ii) \( \chi^{R^k} \in P_{\tilde{B}} \), where \( \tilde{B} := \frac{1}{\eta} B + 1 = \frac{q}{\mu} B + 1 =: B' \).

We claim that \( R = \bigcup_{k=1}^q R^k \) is a set satisfying the conditions of Theorem 14. The set \( R \) clearly satisfies \( \chi^R \in P_{B'} \) since \( \chi^{R^k} \in P_{B'} \) for \( k \in [q] \) and the sets \( R^k \) are on disjoint levels. Furthermore, for each \( u \in W = \{ v \in \Gamma \mid x(P_v) \geq \mu \} \) we indeed have \( P_u \cap R \neq \emptyset \) due to the following. Since \( x = \sum_{k=1}^q x^k \) and \( x(P_u) \geq \mu \) there exists an index \( j \in [q] \) such that \( x^j(P_u) \geq \eta = \frac{\mu}{q} \), and hence \( P_u \cap R \supseteq P_u \cap R^j \neq \emptyset \). \( \square \)

Thus, it remains to prove Lemma 20.

**Proof of Lemma 20.** Let \( \tilde{B} = \frac{1}{\eta} B \). We start by determining an optimal vertex solution \( y \) to the linear program \( \min \{ z(V \setminus \{ r \}) \mid z \in Q \} \), where

\[
Q = \{ z \in P_{\tilde{B}} \mid z(u) = 0 \ \forall u \in V \setminus (V(\ell_1, \ell_2) \cup \{ r \}), \ z(P_u) \geq 1 \ \forall u \in Y \}.
\]

Notice that \( Q \neq \emptyset \) since \( \frac{1}{\eta} x \in Q \); hence, the above LP is feasible. Furthermore, notice that \( y(P_u) \leq 1 \) for \( u \in \Gamma \); for otherwise, there is a vertex \( v \in \text{supp}(y) \) such that \( y(P_v) > 1 \), and hence \( y - \epsilon \chi^v \in Q \) for a small enough \( \epsilon > 0 \), violating that \( y \) is an optimal vertex solution.

Let \( V^L \) be all \( y \)-loose vertices. We will show that the set

\[
R = V^L \cup \{ u \in V \setminus \{ r \} \mid y(u) = 1 \}
\]
fulfills the properties claimed by the lemma. Clearly, \( R \subseteq V_{(\ell_1, \ell_2]} \) since \( \text{supp}(y) \subseteq V_{(\ell_1, \ell_2]} \).

To see that condition (i) holds, let \( u \in Y \), and notice that we have \( y(P_u) = 1 \). Either \( |P_u \cap \text{supp}(y)| = 1 \), in which case the single vertex \( v \) in \( P_u \cap \text{supp}(y) \) satisfies \( y(u) = 1 \) and is thus contained in \( R \); or \( |P_u \cap \text{supp}(y)| > 1 \), in which case \( P_u \cap V^c \neq \emptyset \) which again implies \( R \cap P_u \neq \emptyset \).

To show that \( R \) satisfies (ii), we have to show that \( R \) does not exceed the budget \( \bar{B} \cdot 2^\ell = (\frac{1}{\eta}B + 1)2^\ell \) of any level \( \ell \in \{\ell_1 + 1, \ldots, \ell_2\} \). We have

\[
|R \cap V_\ell| \leq y(V_\ell) + |V^c| \leq \bar{B}2^\ell + |V^c| = \frac{1}{\eta}B2^\ell + |V^c|,
\]

where the second inequality follows from \( y \in Q \). To complete the proof it suffices to show \( |V^c| \leq 2^\ell \). This follows by a sparsity reasoning analogous to Lemma 4 implying that the number of \( y \)-loose vertices is bounded by the number of tight budget constraints, and thus

\[
|V^c| \leq \ell_2 - \ell_1 \leq \ell_2 = [\log^{(k-1)} L]. \quad (2.11)
\]

Furthermore,

\[
2^\ell \geq 2^{\ell_1+1} = 2^{[\log^{(k)} L]+1} \geq 2^{\log^{(k)} L} = \log^{(k-1)} L,
\]

which, together with (2.11), implies \( |V^c| \leq 2^\ell \) and thus completes the proof.

**Proof of Theorem 15**

Let \((y, B)\) be an optimal solution to the RMFC relaxation \( \min \{ B \mid x \in \bar{P}_B \} \) and let \( h = [\log L] \). Hence, \( B \leq B_{OPT} \). We invoke Theorem 14 with respect to the vector \( y \wedge \chi_{V > h} \) and \( \mu = 0.5 \) to obtain a set \( R_1 \subseteq V_{> h} \) satisfying

(i) \( R_1 \cap P_u \neq \emptyset \) \( \forall u \in W \), and

(ii) \( \chi^{R_1} \in P_{2B+1} \),
2.7. **MISSING DETAILS FOR O(1)-APPROX. FOR RMFC**

where \( W = \{ u \in \Gamma \mid y(P_u \cap V_{>h}) \geq 0.5 \} \). Hence, \( R_1 \) cuts off all leaves in \( W \) from the root by only protecting vertices on levels \( V_{>h} \) and using budget bounded by \( 2B + 1 \leq 3B \leq 3 \max\{\log L, B_{\text{OPT}}\} \).

We now focus on the leaves \( \Gamma \setminus W \), which we will cut off from the root by protecting a vertex set \( R_2 \subseteq V_{\leq h} \) feasible for budget \( 3 \max\{\log L, B_{\text{OPT}}\} \). Let \((z, \bar{B})\) be an optimal vertex solution to the following linear program

\[
\min \{ \bar{B} \mid x \in P_B, \ x(P_u) = 1 \ \forall u \in \Gamma \setminus W \} . \tag{2.12}
\]

First, notice that (2.12) is feasible for \( \bar{B} \leq 2B \). This follows by observing that the vector \( q = 2(y \land \chi_{V_{\leq h}}) \) satisfies \( q \in P_{2B} \) since \( y \in P_B \). Moreover, for \( u \in \Gamma \setminus W \), we have

\[
q(P_u) = 2y(P_u \cap V_{\leq h}) = 2(1 - y(P_u \cap V_{>h})) > 1,
\]

where the last inequality follows from \( y(P_u \cap V_{>h}) < 0.5 \) because \( u \in \Gamma \setminus W \). Finally, there exists a vector \( q' < q \) such that \( q'(P_u) = 1 \) for \( u \in \Gamma \setminus W \). The vector \( q' \) can be obtained from \( q \) by successively reducing values on vertices \( v \in \text{supp}(q) \) satisfying \( q(P_v) > 1 \). This shows that \((q', 2B)\) is a feasible solution to (2.12) and hence \( \bar{B} \leq 2B \).

Consider the set of all \( z \)-loose vertices \( V^L = \{ u \in \text{supp}(z) \mid z(P_u) < 1 \} \). We define

\[
R_2 = V^L \cup \{ u \in \text{supp}(z) \mid z(u) = 1 \}.
\]

Notice that for each \( u \in \Gamma \setminus W \), the set \( R_2 \) contains a vertex on the path from \( u \) to the root. Indeed, either \( |\text{supp}(z) \cap P_u| = 1 \) in which case there is a vertex \( v \in P_u \) with \( z(v) = 1 \), which is thus contained in \( R_2 \), or \( |\text{supp}(z) \cap P_u| > 1 \) in which case the vertex \( v \in \text{supp}(z) \cap P_u \) that is closest to the root among all vertices in \( \text{supp}(z) \cap P_u \) is a \( z \)-loose vertex. Hence, the set \( R = R_1 \cup R_2 \) cuts off all leaves from the root. It remains to show that it is feasible for budget \( 3 \max\{\log L, B_{\text{OPT}}\} \).

Using an analogous sparsity reasoning as in Lemma 4, we obtain that \( |V^L| \) is bounded by the number of tight budget constraints, which is at most...
$h = \lfloor \log L \rfloor \leq \log L$. Hence, for any level $\ell \in [h]$, we have

$$|R_2 \cap V_\ell| \leq |V^\ell| + z(V_\ell)$$

$$\leq \log L + 2^\ell \bar{B} \quad \text{((z, } \bar{B}) \text{ feasible for (2.12))}$$

$$\leq \log L + 2^\ell \cdot 2B \quad \text{(} \bar{B} \leq 2B \text{)}$$

$$\leq 2^\ell \cdot (3 \max\{\log L, B_{\text{OPT}}\}). \quad \text{(} B \leq B_{\text{OPT}} \text{)}$$

Thus, both $R_1$ and $R_2$ are budget-feasible for budget $3 \max\{\log L, B_{\text{OPT}}\}$, and since they contain vertices on disjoint levels, $R = R_1 \cup R_2$ is feasible for the same budget. \hfill \square

**Proof of Lemma 17**

To show that the running time of $\text{Enum}(\emptyset, \emptyset, \bar{\gamma})$ is polynomial, we show that there is only a polynomial number of recursive calls to $\text{Enum}(A, D, \gamma)$. Notice that the number of recursive calls done in one execution of step 4 of the algorithm is equal to $2 |F_x|$. We thus start by upper bounding $|F_x|$ for any solution $(x, B)$ to $\text{LP}(A, D)$ with $B < \log L$. Consider a vertex $f_u \in F_x$, where $u \in \Gamma \setminus W_x$. Since $u$ is a leaf not in $W_x$, we have $x(P_u \cap V_{\leq h}) > \frac{1}{3}$, and thus

$$x(T_{f_u} \cap V_{\leq h}) > \frac{1}{3} \quad \forall f_u \in F_x.$$  

Because no two vertices of $F_x$ lie on the same leaf-root path, the sets $T_{f_u} \cap V_{\leq h}$ are all disjoint for different $f_u \in F_x$, and hence

$$\frac{1}{3} |F_x| < \sum_{f \in F_x} x(T f \cap V_{\leq h})$$

$$\leq x(V_{\leq h}) \quad \text{(} T_f \cap V_{\leq h} \text{ are disjoint for different } f \in F_x \text{)}$$

$$\leq \sum_{\ell=1}^{h} 2^\ell B \quad \text{(} x \text{ satisfies budget constr. of } \text{LP}(A, D) \text{)}$$

$$< 2^{h+1} B$$

$$< 2(\log L)^2. \quad \text{(} h = \lfloor \log^{(2)} L \rfloor \text{ and } B < \log L \text{)}$$
2.7. MISSING DETAILS FOR $O(1)$-APPROX. FOR RMFC

Since the recursion depth is $\tilde{\gamma} = 2(\log L)^2 \log^2 L$, the number of recursive calls is bounded by

$$O\left((2|F_x|)^{\tilde{\gamma}}\right) = (\log L)^{O((\log L)^2 \log^2 L)} = 2^{o(L)} = o(N),$$

thus showing that $\text{Enum}(\emptyset, \emptyset, \tilde{\gamma})$ runs in polynomial time.

It remains to show that $\text{Enum}(\emptyset, \emptyset, \tilde{\gamma})$ finds a triple satisfying the conditions of Lemma 16. For this we identify a particular execution path of the recursive procedure $\text{Enum}(\emptyset, \emptyset, \tilde{\gamma})$ that, at any point in the algorithm, will maintain a clean pair $(A, D)$ that is compatible with OPT, i.e., $A \subseteq \text{OPT}$ and $D \cap \text{OPT} = \emptyset$. At the beginning of the algorithm we clearly have compatibility with OPT since $A = D = \emptyset$. To identify the execution path we are interested in, we highlight which recursive call we want to follow given that we are on the execution path. Hence, consider a clean pair $(A, D)$ that is compatible with OPT, we define $\Phi(A, D) \in \mathbb{Z}_{\geq 0}$ in the following way. For each $u \in \text{OPT} \cap V_{\leq h}$, let $d_u \in \mathbb{Z}_{\geq 0}$ be the distance of $u$ to the first vertex in $A \cup D \cup \{r\}$ when following the unique $u$-$r$ path. We define $\Phi(A, D) = \sum_{u \in \text{OPT} \cap V_{\leq h}} d_u$. Notice that as long as we have a triple $(A, D, x)$ on our execution path that does not satisfy the conditions of Lemma 16, then the next triple $(A', D', x')$ on our execution path satisfies $\Phi(A', D') < \Phi(A, D)$. Hence, either we will encounter a triple on our execution path satisfying the conditions of Lemma 16 while still having a strictly positive potential, or we will encounter a triple $(A, D, x)$ compatible with OPT and $\Phi(A, D) = 0$,
which implies $\text{OPT} \cap V_{\leq h} = A$, and we thus correctly guessed all vertices of $\text{OPT} \cap V_{\leq h}$ implying that the conditions of Lemma 16 are satisfied for the triple $(A, D, x)$. Since $\Phi(A, D) \geq 0$ for any compatible clean pair $(A, D)$, this implies that a triple satisfying the conditions of Lemma 16 will be encountered if the recursion depth $\tilde{\gamma}$ is at least $\Phi(\emptyset, \emptyset)$. To evaluate $\Phi(\emptyset, \emptyset)$ we have to compute the sum of the distances of all vertices $u \in \text{OPT} \cap V_{\leq h}$ to the root. The distance of $u$ to the root is at most $h$ since $u \in V_{\leq h}$. Moreover, $|\text{OPT} \cap V_{\leq h}| < 2^{h+1} \cdot B_{\text{OPT}}$ due to the budget constraints. Hence,

$$
\Phi(\emptyset, \emptyset) < h \cdot 2^{h+1} \cdot B_{\text{OPT}} \\
\leq 2 \log^2(L) \cdot (\log L)^2 \quad (h = \lfloor \log^2(L) \rfloor \text{ and } B_{\text{OPT}} \leq \log L) \\
= \tilde{\gamma},
$$

implying that a triple fulfilling the conditions of Lemma 16 is encountered by $\text{Enum}(\emptyset, \emptyset, \tilde{\gamma})$. \hfill \Box

### 2.8 Transformations for the Firefighter problem

In this section, we provide some basic transformations showing how different natural variations of the Firefighter problem can be reduced to each other. We start by proving Lemma 3.

**Proof of Lemma 3.** Consider an instance of the weighted Firefighter problem with general budgets consisting of a tree $G = (V, E)$ of depth $L$ rooted at the vertex $r \in V$, weights $w(u) \in \mathbb{Z}_{\geq 0}$ for all $u \in V \setminus \{r\}$ and budgets $B_\ell \in \mathbb{Z}_{> 0}$ for all $\ell \in [L]$. We transform the instance into an equivalent instance with unit budgets by performing the following simple steps for all levels $V_\ell$ for $\ell \in [L]$:

- For every $u \in V_\ell$, subdivide the edge connecting $u$ to its ancestor in $G$ into a path with $B_\ell$ edges, by introducing $B_\ell - 1$ new vertices. Denote the nodes on this path, excluding the ancestor of $u$ in $G$, by $Y_u$. 

• Set the weight of all new vertices to zero, while maintaining the weight $w(u)$ for the original vertex $u$.

Denote the resulting tree by $G' = (V', E')$. To conclude the construction it remains to allow one unit of budget in every level of the transformed tree. It is easy to verify that feasible solutions to the Firefighter problem for the two instances are in correspondence. A feasible solution for $G$ is transformed to a solution in $G'$ by replacing the $B_\ell$ vertices $S_\ell$ protected in any level $V_\ell$ of $G$ with any $B_\ell$ vertices on the corresponding paths $\{Y_u \mid u \in S_\ell\}$ in $G'$, one in each of the $B_\ell$ distinct levels of $G'$ that are in correspondence with $V_\ell$. The opposite transformation selects for every protected vertex $u \in V'$ in a feasible solution for $G'$ the vertex $u \in V$ such that $u' \in Y_u$. It is straightforward to verify that in both transformations the obtained solutions are feasible and that they have weights identical to the original solutions.

Finally, since $B_\ell \leq n$ can be assumed for every $\ell \in [L]$, each one of the $n - 1$ edges in $G$ is subdivided into a path of length at most $n$, thus the number of vertices in $G'$ is at most $O(n^2)$.

We remark that a construction analogous to the one used in the proof of Lemma 3 can be used to show that RMFC with non-uniform budgets can be reduced to the uniform budget case. In an RMFC instance with non-uniform budgets, the budget on level $\ell$ is equal to $B \cdot a_\ell$, where $a_\ell \in \mathbb{Z}_{>0}$ for $\ell \in [L]$ are given as input, and the goal is still to find the minimum $B$ to protect vertices that cut off all leaves from the root and fulfill the budget constraints.

Next, we show how a weighted instance of the Firefighter problem can be transformed into a unit-weight one with only an arbitrarily small loss in term of the objective function.

**Lemma 21.** Let $\delta > 0$ and $\alpha \in (0, 1]$. Any weighted unit-budget Firefighter problem on a tree $G = (V, E)$ and weights $w(u) \in \mathbb{Z}_{\geq 0}$ for $u \in V \setminus \{r\}$ can be transformed efficiently into a polynomial-size unit-weight unit-budget Firefighter problem on a tree $G' = (V', E')$ such that any $\alpha$-approximate feasible solution for $G'$ can be efficiently transformed into a $(1 - \delta)\alpha$-approximate solution for $G$. 
Proof. Assume \( w(V) > 0 \), i.e., not all weights are zero, since for otherwise the result trivially holds. Notice that this assumption also implies that the value \( \text{val(OPT)} \) of an optimal Firefighter solution in \( G \) satisfies \( \text{val(OPT)} \geq 1 \).

For simplicity we present the transformation in two steps, each losing at most a \( \frac{\delta}{2} \)-fraction in terms of objective. First we use a standard scaling and rounding technique to obtain a new weight function that is bounded by a polynomial in the size of the tree. Concretely, we construct weights \( w'(u) \in \mathbb{Z}_{\geq 0} \) for \( u \in V \setminus \{r\} \) such that \( w'(u) = O\left(\frac{n}{\delta}\right) \) for every \( u \in V \), and for a well-chosen parameter \( D \in \mathbb{R}_{>0} \) we have:

\[
D w'(S) \leq w(S) \leq D w'(S) + \frac{\delta}{2} \text{val(OPT)} \quad \forall S \subseteq V \setminus \{r\}. \tag{2.13}
\]

In a second phase discussed below we use the obtained instance to construct a unit-weight instance with the desired property.

Let \( w_{\text{max}} = \max_{u \in V \setminus \{r\}} w(u) \) be the maximum weight of any vertex in \( G \). Define \( D = \frac{\delta w_{\text{max}}}{2n} \), where \( n = |V| \), and for every \( u \in V \setminus \{r\} \) set \( w'(u) = \lfloor w(u)/D \rfloor \). Observe that \( \text{val(OPT)} \geq w_{\text{max}} \) since any single vertex can be protected. The latter scaling indeed fulfills the desired properties as \( w'(u) \leq 2n/\delta = O(n/\delta) \), and for every \( S \subseteq V \setminus \{r\} \) we have

\[
D w'(S) \leq w(S) \leq D w'(S) + D |S| \leq D w'(S) + \frac{\delta}{2} \text{val(OPT)},
\]

where the first two inequalities follows from \( w'(u) = \lfloor w(u)/D \rfloor \ \forall u \in V \setminus \{r\} \), and the last one from \( D |S| \leq D n = \delta w_{\text{max}}/2 \leq \delta \text{val(OPT)}/2 \).

This shows (2.13).

We show next that the latter transformation loses at most a \( \frac{\delta}{2} \)-fraction in the objective function. More precisely, let \( S' \subseteq V \setminus \{r\} \) be a set of vertices that will not burn in an \( \alpha \)-approximate solution to the Firefighter problem with respect to the weights \( w' \). We will show that \( w(S') \geq (1 - \frac{\delta}{2}) \alpha \text{val(OPT)} \), implying that the same solution is \( (1 - \frac{\delta}{2}) \alpha \)-approximate with respect to the original weights \( w \). Let \( S^* \subseteq V \setminus \{r\} \) be the vertices that will not burn at the end of the process in an optimal solution for \( G \). By (2.13) we have
2.8. TRANSFORMATIONS FOR THE FIREFIGHTER PROBLEM

\[ Dw'(S^*) + \frac{\delta}{2} \text{val}(\text{OPT}) \geq w(S^*) = \text{val}(\text{OPT}), \]  
implies \[ Dw'(S^*) \geq (1 - \frac{\delta}{2}) \text{val}(\text{OPT}). \]

We conclude:

\[
\left(1 - \frac{\delta}{2}\right) \text{val}(\text{OPT}) \leq Dw'(S^*) 
\leq \frac{1}{\alpha} Dw'(S') \quad (S' \text{ corresponds to } \alpha\text{-approx. for } w') 
\leq \frac{1}{\alpha} w(S'), \quad (w'(u) = \lfloor w(u)/D \rfloor \text{ for } u \in V \setminus \{r\})
\]

which yields \[ w(S') \geq (1 - \frac{\delta}{2}) \alpha \text{val}(\text{OPT}), \] as desired.

Next we present the second transformation, which, given a weighted Fire-fighter problem with tree \( G = (V, E) \) and integer weights \( w(u) \in \mathbb{Z}_{\geq 0} \) bounded by \( O(n) \), transforms it into a unit-weight instance on a new tree \( G' = (V', E') \) by losing at most a \( \frac{\delta}{2} \)-fraction in terms of objective.

The tree \( G' \) is obtained from \( G \) by taking a copy of \( G \) and attaching \( \lfloor \frac{4n}{\alpha \delta} w(u) \rfloor \) new leaves to every vertex \( u \in V \setminus \{r\} \). For brevity, for a vertex set \( R \subseteq V \setminus \{r\} \), we denote by \( sv(R) \subseteq V \) the set of all vertices that will not burn in \( G \) if one protects the set \( R \), i.e., \( sv(R) = \cup_{u \in R} Tu \).

Similarly, for \( R' \subseteq V' \setminus \{r\} \), we denote by \( sv'(R') = \cup_{u \in R'} T'_u \subseteq V' \) all vertices in \( G' \) that will not burn if \( R' \) gets protected.

Consider a solution that protects a set \( R' \subseteq V' \setminus \{r\} \) of vertices in \( G' \). Observe that \( V \cap R' \) is a feasible set of vertices to protect in \( G \). We can upper bound the objective value of \( R' \) in \( G' \) as follows, where \( w' \) is the unit-weight function used in \( G' \):

\[
w'(sv'(R')) = |sv'(R')| \leq |R' \setminus V| + \sum_{u \in sv(R' \cap V)} \left(1 + \left\lfloor \frac{4n}{\alpha \delta} w(u) \right\rfloor \right)
\leq n + \sum_{u \in sv(R' \cap V)} \left(1 + \frac{4n}{\alpha \delta} w(u) \right)
\leq 2n + \frac{4n}{\alpha \delta} w(sv(R' \cap V)).
\]  
\[(2.14)\]
Moreover, for any set of vertices $R \subseteq V \setminus \{r\}$ in $G$ we have

$$w'(sv'(R)) = |sv'(R)| = \sum_{u \in sv(R)} \left( 1 + \left\lfloor \frac{4n}{\alpha \delta} w(u) \right\rfloor \right)$$

$$\geq \sum_{u \in sv(R)} \frac{4n}{\alpha \delta} w(u)$$

$$= \frac{4n}{\alpha \delta} w(sv(R)).$$

(2.15)

We complete the rest of the proof similar to the proof of the first transformation. Let $R^* \subseteq V \setminus \{r\}$ be an optimal set of vertices to protect in $G$, and let $R' \subseteq V' \setminus \{r\}$ be an $\alpha$-approximation for the unit-weight Firefighter instance on $G'$. Our goal is to show that $R' \cap V$ is a solution to the Firefighter problem on $G$ of value at least $(1 - \frac{\delta}{2}) \alpha \text{val}(\text{OPT})$. Indeed, we have

$$\text{val}(\text{OPT}) = w(sv(R^*)) \leq \frac{\alpha \delta}{4n} w'(sv'(R^*)) \quad \text{(by } 2.15 \text{)}$$

$$\leq \frac{\delta}{4n} w'(sv'(R')) \quad \text{(} R' \text{ is } \alpha \text{-approx. for } G' \text{)}$$

$$\leq \frac{\delta}{2} + \frac{1}{\alpha} w(sv(R' \cap V)) \quad \text{(by } 2.14 \text{)}$$

$$\leq \frac{\delta}{2} \text{val}(\text{OPT}) + \frac{1}{\alpha} w(sv(R' \cap V)), \quad \text{(because } \text{val}(\text{OPT}) \geq 1 \text{)}$$

which implies

$$w(sv(R' \cap V)) \geq \left( 1 - \frac{\delta}{2} \right) \alpha \text{val}(\text{OPT}),$$

as desired.

Finally, both transformations can be implemented in polynomial time. For the first transformation this is trivial, while for the second transformation one uses the fact that the input weights are polynomially bounded, and hence $G'$ has polynomial size.
2.9 Conclusions

In this chapter we developed new approximation algorithms for the Firefighter and the Resource Minimization for Fire Containment problems in trees that are asymptotically optimal with respect to the approximation factors, assuming that \( P \neq NP \).

Apart from the approximation results themselves, the techniques developed shed a new light on the structure of the Firefighter and RMFC problems. The most unusual aspect in both of our algorithms is the way we use the standard LP relaxations, which are known to have large integrality gaps (larger than the obtained approximation factors), to obtain good solutions. For each problem we perform a delicate enumeration over a super-constant number of nodes in the tree, thus reducing the integrality gaps to the desired constants. This contrasts standard such approaches that are only able to deal in polynomial time with enumerations of constant-size structures (e.g. like in the classical PTAS for the Knapsack problem). The latter techniques are also combined with new transformations of the input trees that, while not affecting the integrality gap, are crucial in reducing the size of on which run our enumeration algorithms.

We believe that some of these techniques can find further applications in similar problems. The natural potential generalizations of our results include considering other graph classes and other spreading dynamics. Since the first publication of these results, our techniques for RMFC have already found an application to the Non-Uniform \( k \)-Center Problem [56]. In simple terms, the result of Goyal et al. [56] relies on a transformation of the instance of their problem to an RMFC instance. This transformation relies on the a solution to the natural LP relaxation of the problem, which suffers from a \( O(\log^* n) \) integrality gap, so to obtain a constant factor approximation, the authors cannot use our results in a black box way. Instead, the authors follow our general scheme for obtaining a constant factor approximation for RMFC, while adapting it in an interesting way to their needs.
Chapter 3

The Multilevel Critical Node Problem

In recent years, a lot of effort has been dedicated to develop strategies to defend networks against possible cascade failures or malicious viral attacks. In particular, many results rely on two different viewpoints. On the one hand, network safety is investigated from a preventive perspective. In this paradigm, for a given network, the goal is to modify its structure, in order to minimize the propagation of failures. On the other hand, blocking models have been proposed for scenarios where the attack has already taken place. In this case, a harmful spreading process is assumed to propagate through the network with particular dynamics, allowing some time for an effective defensive reaction.

In this work we combine these two perspectives. More precisely, following the framework of the Defender-Attacker-Defender model, as introduced by Brown et al. [19], we consider a model of prevention, attack, and damage containment using a three-stage, sequential game. Thus, we assume the defender not only to be able to adopt preventive strategies but also to defend the network after an attack takes place. Assuming that the attacker will act optimally, we want to chose a defensive strategy for the first stage that would minimize the total damage to the network in the end of the third stage.

Our contribution consists of considering this problem as a trilevel Mixed-Integer Linear Program and design an exact algorithm for it based on tools developed for multilevel programming.
3.1 Introduction

Many important connectivity systems, such as communication or social networks, transportation systems, electrical grids, mobility networks, computer networks, etc., are susceptible to attacks, breakdowns or epidemic phenomena, which may trigger a cascade of failures that propagates through the entire system. In the last decades, a lot of effort has been dedicated to develop strategies to defend networks against possible cascade failures or malicious viral attacks. The problem we consider in this chapter is of the latter type. In particular, it aims at designing defensive strategies for networks that are threatened by well-planned attacks that trigger a spreading of failures. Moreover, our model assumes that the network operator can act preventively before an attack has taken place and also implement a protective reaction against the spread after the attack has been localized.

Specifically, the problem we tackle in this chapter can be formalized as follows. We are given a directed graph $G = (V,A)$, in which $V$ and $A$ denote the nodes and the arcs, respectively. Let $I \subseteq V$ be the subset of nodes that a malicious agent decides to attack. Let $D \subseteq V$ and $P \subseteq V$ be the sets of nodes that the network operator decides to vaccinate and protect, before and after the attack takes places, respectively. In the model we consider, vaccinated or protected nodes can neither be affected by the cascade of failures starting at $I$, nor can the spread propagate through them. Furthermore, the failure starting at $I$ reaches any node which is connected by a path to a node in $I$, that does not contain nodes in $D \cup P$. Thus, we call a node $v \in V$ saved if it is vaccinated, protected, or there is no path from any attacked node in $I$ to $v$ in the induced subgraph, $G[V \setminus \{D \cup P\}]$, obtained by removing vaccinated and protected nodes. Let $S(I, D \cup P) \subseteq V$ be the set of saved nodes, corresponding to the attack $I$ and to the defensive strategy $V \cup P$. Furthermore, we assume that the attacker cannot strike vaccinated nodes and that the defender cannot protect attacked nodes, namely $I \subseteq V \setminus D$ and $P \subseteq V \setminus I$. For any action, we consider budgets $\Omega$, $\Phi$, and $\Lambda$ in $\mathbb{Z}_{\geq 0}$, limiting the number of nodes which the respective agent can choose for the phases of vaccination, attack, and protection, respectively. The problem can
be then formalized as follows.

$$\max_{D \subseteq V} \min_{I \subseteq V \setminus D} \max_{P \subseteq V \setminus I} |S(I, D \cup P)|$$  \hspace{1cm} (MCN)

We call the latter problem the Multilevel Critical Node (MCN) problem.

MCN tries to model at the same time two well-know paradigms in the field of network defense. Basically, the difference between the two approaches we refer to lies in the moment in which we apply the defense strategy compared to the time of execution of the attack. More precisely, on the one hand, a defense action can take place preventively before a potential threat appears or, on the other hand, once an attack has been localized, a defensive reaction might be required to contain damages.

In the first paradigm, the network is assumed yet to be attacked and its resilience is investigated from a preventive perspective. Considering its graph topology, the goal is to modify this structure, using node or edge deletions, in order to minimize its capacity to propagate failures. The basic idea is indeed to identify those critical infrastructures, such as nodes or edges, whose removal can reduce the attack’s effectiveness. Since it is not possible to predict the specific location of a possible attack or failure, network vulnerability can often be linked to a particular metric that somehow describes the effectiveness of a potential attack, defined in terms of its connectivity properties. Concretely, those problems that look for a subset of nodes whose removal maximally decrease some given graph metric are called Critical Node Problems (CNPs) [7], and they are a special case of network interdiction problems. In literature, these problems have been widely investigated. The considered metrics are often based on graph properties, e.g. [57], or more complex measures, such as pair-wise connectivity [73, 37, 1], distance-based connectivity [88], number of disconnected components or size of the biggest connect component [84, 85], and more. Additionally, there have also been studies on how both nodes and edges should be deleted at the same time so as to decrease network vulnerability [62, 38].

The other paradigm we consider studies blocking models for the case an attack has already taken place and a protection strategy is required. In this case, a harmful spreading process is assumed to propagate through
the network with particular dynamics, allowing some time for an effective defensive reaction. The goal is to isolate the propagation in order to defend the most important portion of the network. The Firefighter problem, introduced by Hartnell [59], is an example of this class of problems for which we refer to the second chapter of this thesis for a more detailed exposition. Some applications for unbalanced cuts can be interpreted as another example that belongs to this paradigm. More precisely, the minimum-size bounded-capacity cut problem, as introduced by Hayrapetyan et al. [61], asks for finding a minimal size cut which contains a given source \( s \) and its capacity does not exceeds a prescribed bound. In this sense, assuming \( s \) to be the source of a harmful spreading phenomenon, the corresponding cut can be interpreted as the minimal infected area that can be isolated by deleting the outgoing edges from this cut. Since in this work we consider only actions (vaccination, attack, protection) on nodes and not on edges, we will be interested in a vertex-separator version of this problem which is defined in [61] and addressed also by [49]. Similar problems are also considered in [53, 44].

The problem we consider for the protection stage is equivalent in its general form to the minimum-size bounded-capacity vertex-cut problem. However, in order to make it compatible with our framework we need to rephrase it a bit. Moreover, for brevity, we refer to the following version of the minimum-size bounded-capacity vertex-cut problem with multiple sources of infection as the Fence problem. Recall that \( I \) represent the set of nodes that have been attacked. We assume the defender has budget \( \Lambda \in \mathbb{Z}_{\geq 0} \) for protecting not yet infected nodes. Recall that, for a given a set of protected nodes \( P \subseteq V \setminus I \), \( v \in V \) is saved by \( P \) if either \( v \in P \), or there is no path in \( G[V \setminus P] \) from \( v \) to any of the nodes in \( I \). We denote by \( S(I, P) \subseteq V \) the set of saved nodes for attacked nodes \( I \) and protected nodes \( P \). The Fence problem asks, for an initial set of attacked node \( I \), to solve the following problem.

\[
\max_{P \subseteq V \setminus I, \ |P| \leq \Lambda} |S(I, P)| \tag{Fe}
\]

Notice that the Fence problem can be seen as a variant of the Firefighter problem with multiple fires where the budget \( \Lambda \) is given fully at time 1,
and no protection is allowed during the next time steps. The dynamics of propagation will indeed make the saved nodes at the end of the process coincide with the saved set of nodes, leading to the same objective.

The MCN problem, which we define and study in this chapter, combines the two different paradigms we describe above. Concretely, it is easy to see that it extends the Critical Node problem and the Fence problem, at the same time. On the one hand, if we assume $\Lambda = 0$ and we look at the problem from the defender’s perspective, the derived max-min problem corresponds to the Critical Node problem for which the metric considered is the sum of the total weights of the $\Phi$ heaviest components. On the other hand, if we assume $\Omega = 0$, for a fixed attack $I \subseteq V$, the resulting problem is exactly the Fence problem.

Moreover, it is important to notice that (MCN) implicitly requires to solve optimally the two-stage problem corresponding to the optimal attack strategy. In this problem, attacker is required to strike nodes $I$ that mostly affect the network after the best blocking reaction is used. Hence, the attacker problem is modeled by the following problem.

$$\min_{I \subseteq V} \max_{P \subseteq V \setminus I} |S(I, P)| \quad \text{(AP)}$$

Following the framework of the Defender-Attacker-Defender model, as introduced by Brown et al. [19], MCN captures a model of prevention, attack, and damage containment, using a three-stage sequential game. Moreover, according to the definition of the problem, the model assumes for this three-stage game perfect information and rationality, i.e., both agents have perfect knowledge of the graph topology, of the virus dynamics, of the adversary’s actions, and they act optimally. Thus, MCN asks for identifying the best preventive strategy the defender should employ in light of the fact that it will still be possible to protect the network even after a worst-case attack has taken place.

This chapter is based on joint work with Margarida Carvalho, Andrea Lodi, and Andrea Tramontani [9].
3.1.1 Our contribution

In this chapter we define the Multilevel Critical Node problem combining two previously known paradigms from network defense into a three-stage Defender-Attacker-Defender game. The model is motivated by the real-world hypothesis that the network operator has two stages in which he can act to protect the network, namely before and after the harmful propagation starts.

We model this problem as a trilevel Mixed-Integer Linear Program and we design an exact algorithm for solving it. The algorithm is based on recent tools developed for bilevel programming and on a simultaneous column and row generation approach. Our algorithm invokes as subroutine an algorithm that we devise for solving the bilevel problem defined by the second and third stages. Moreover, both the algorithm and the subroutine invoke a solver for Mixed-Integer Linear Programs (MIPs) as a black box. In particular, for the implementation of our algorithm, we use IBM ILOG CPLEX in order to solve MIPs.

We present examples in which the optimal action in the vaccination stage that assumes the existence of the protection stage has a vastly better objective function value than the value obtained by sequentially optimizing the first and the third steps separately. Such gap, that we call relative gain, is also illustrated by our numerical experiments.

Finally, we perform computational experiments on the performances of the developed algorithm. Our test set is based on instances consisting of randomly generated trees and graphs.

3.1.2 Further related work

Since we focus on identifying critical infrastructure in a network determined by possible intentional attacks, this work falls into the framework of Network Interdiction problems. Moreover, both paradigms we combine in our model are special cases of this class of problems. Network interdiction problems have been vastly investigated since decades and found applications in many real-world problems, including controlling the diffusion of infections...
in hospitals [8], coordinating military strikes [54], combating the traffic of illegal drugs [91], protection of infrastructure system [82, 29], floods control [80], and contrasting nuclear smuggling [76]. Interdiction problems have been widely investigated for well-known combinatorial optimization problems as well. This includes natural interdiction versions of problems such as minimum spanning tree [52, 95], maximum s-t flow [91, 78, 93] (often known as network flow interdiction), maximum matching [92, 39], shortest path [11, 68], independent set and vertex cover [17], packing [39], connectivity of a graph [94], and facility location [29].

Interdiction problems are often modeled as Mixed-Integer Bilevel Linear Programs (MIBLPs). Analogously, we model the MCN problem as a three-level MILP. In particular, the algorithm we devise is based on tools developed for bilevel programming. Solving MIBLPs to optimality is generally a challenging task. Because of this reason, and especially because MIBLPs model many important real-world problems, a lot of effort has been dedicated to developing efficient algorithms for solving generic MIBLPs in recent years. Common approaches try to exploit consolidated techniques known for solving MILPs. Concretely, these algorithms use as subroutines MILP models and their implementations use modern MILP solvers. The first approach for solving MIBLPs has been presented in [75], where the authors applied a branch-and-bound algorithm to a relaxed formulation. Later, [36, 35] improved on these techniques by adding cuts based on integrality and feasibility properties. This approach has been further improved by [24], and very recently by [47], in which the authors provided significantly better performances by introducing stronger cuts. A lot of effort has also been dedicated to devising tailored approaches to solve particular special cases. For instance, [63] considered a bilevel influence interdiction problem in networks, and [35, 18, 23] investigated interdiction versions of the knapsack problem.

There have also been studies on trilevel programs. For example, a three-stage sequential game designed for modeling infrastructure defense has been presented [4]. In particular, the authors describe a Defend-Attack-Defend model for system resilience against an attack. They present a realistic formulation of this model for the case of a transportation network and design
a decomposition algorithm for solving the resulting model.

### 3.1.3 Measuring the effectiveness of the three-stage approach

In order to compare the effectiveness of our three-level model with respect to an approach that considers preventive and blocking strategies separately, we introduced the notion of *relative gain*. Concretely, for any instance $\mathcal{J}$ of MCN, we define the relative gain as

$$relGain(\mathcal{J}) = \frac{OPT - VA_{AP}}{OPT},$$

(3.1)

where $OPT$ is the optimal value determined for the MCN instance $\mathcal{J}$, and $VA_{AP}$ is the value of the solution obtained by separating the three stages into two steps (Vaccinate-Attack and Attack-Protect) and solving them to optimality sequentially. More precisely, for the evaluation of $VA_{AP}$, we first determine the best vaccination strategy for the three-stage problem in the case no budget for protection is given, i.e., $\Lambda = 0$, and then we look for an optimal solution for the resulting AP problem obtained by removing the vaccinated set of nodes found before. Notice that, for the computation of $VA_{AP}$, the infected sets in the two steps do not need to coincide. Thus, $relGain(\mathcal{J})$ represents the gain in relative terms obtained by the network’s operator when he prefers the three-level approach to the sequential one for the instance $\mathcal{J}$. Clearly, $OPT \geq VA_{AP}$ always holds, so for every MCN instance $\mathcal{J}$, $relGain(\mathcal{J})$ is a value in the interval $[0, 1]$. More precisely, the closer it is to 1, the the more effective is the three-level approach with respect to the sequential one. On the other hand, $relGain(\mathcal{J}) = 0$ implies that, for the instance $\mathcal{J}$, a sequential approach provides a solution optimal also for the MCN problem.

We observe that $relGain$ can be arbitrarily close to 1, as is demonstrated by the following instance. Let $m \in \mathbb{Z}_{\geq 5}$. We consider a directed graph $G = (V, A)$ built as follows. $V$ contains nodes $u_i$, $i \in \{1, \ldots, m\}$, $s$, and $a$. All the nodes $u_i$, $i \in \{1, \ldots, m\}$, form a wheel whose center is $s$. More precisely, we first connect in both directions every node $u_i$ to $u_{i+1}$, $i \in \{1, \ldots, m - 1\}$, and $u_1$ to $u_m$, so as to obtain a cycle. Then, we connect...
Figure 3.1: An example for constructing an instance whose $\text{relGain}$ is arbitrarily close to 1

in both directions every node $u_i, i \in \{1, \ldots, m\}$, to $s$. Finally, we connect in both direction $u_1$ to $a$. Figure 3.1 shows the obtained graph for $m = 12$. Furthermore, let us assume $\Omega = 1$, $\Phi = 1$, and $\Lambda = 2$. On the one hand, an optimal solution for MCN is to vaccinate $s$, and the corresponding value is $OPT = m$. Once $s$ is vaccinated, the most effective attack strikes the node $v_1$. The defender can then protect the pair $\{u_m, u_2\}$, saving, at the end of the process, $n$ nodes and loosing only $v_1$ and $a$. Moreover, it easy to verify that any vaccination strategy that differs from $s$ leads to a worse outcome for the defender. On the other hand, assuming $\Lambda = 0$, MCN is reduced to the problem that asks for finding a node whose removal minimizes the size of the biggest connected component. An optimal solution corresponds to selecting $v_1$, since it is the only option that disconnects the graph into two components. Once $v_1$ gets vaccinated, it is easy to verify that the best possible attack strategy for AP corresponds to striking $s$. Then, any selection of two nodes among $\{v_2, \ldots, v_m\}$ is equivalent, leading to a value $VA_{AP} = 4$. Finally, notice that $relGain = \frac{m - 4}{m}$, which can be made arbitrarily close to 1 for large enough $m$. 
3.1.4 Organization of the chapter

In Section 3.2, we describe an algorithm for solving the Multilevel Critical Node problem and a subroutine for solving the two-stage (AP) problem. In Section 3.3, we present some computational experiments showing the performance of the algorithm and some statistics on the relative gain. Experiments are performed on random tree networks and random graphs. Finally, in Section 3.4, we draw some conclusions and present new possible directions for future research.

3.2 Solving the Multilevel Critical Node problem

In this section we describe an approach to solve the MCN problem based on tools developed for bilevel programming. We start by modeling the problem as a three-level program. Let us consider the following indicator variables which model all the possible different states for a node $v \in V$.

$$z_v = \begin{cases} 1 & \text{if } v \text{ vaccinated} \\ 0 & \text{otherwise} \end{cases} \quad x_v = \begin{cases} 1 & \text{if } v \text{ protected} \\ 0 & \text{otherwise} \end{cases}$$

$$y_v = \begin{cases} 1 & \text{if } v \text{ attacked} \\ 0 & \text{otherwise} \end{cases} \quad \alpha_v = \begin{cases} 1 & \text{if } v \text{ saved} \\ 0 & \text{otherwise} \end{cases}$$

Notice that a node $v \in V \setminus I$ is saved if either it has been vaccinated, protected, or all of its adjacent nodes are saved. In this way, it is possible to determine if a node is saved by only restricting to constraints that take into account the state of adjacent nodes. We can then formulate the MCN
3.2. SOLVING THE MULTILEVEL CRITICAL NODE PROBLEM  69

problem as the following three-level MILP.

\[
\begin{align*}
\max_{z \in \{0,1\}^V} & \sum_{v \in V} \alpha_v \\
\sum_{v \in V} z_v & \leq \Omega
\end{align*}
\]

where \( \alpha_v \) solves the attacker’s problem

\[
\begin{align*}
\min_{y \in \{0,1\}^V} & \sum_{v \in V} \alpha_v \\
\sum_{v \in V} y_v & \leq \Phi
\end{align*}
\]

(3lvMIP)

where \( \alpha_v \) solves the defender’s problem

\[
\begin{align*}
\max_{x \in \{0,1\}^V} & \sum_{v \in V} \alpha_v \\
\sum_{v \in V} x_v & \leq \Lambda \\
\alpha_v & \leq 1 + z_v - y_v \quad \forall \, v \in V \\
\alpha_v & \leq \alpha_u + x_v + z_v \quad \forall \, (u, v) \in A \\
\alpha_v & \leq 1 \quad \forall \, v \in V.
\end{align*}
\]

It is not difficult to see that the latter three-level MILP models exactly the MCN problem. In particular, requiring \( \alpha \) to be binary is not necessary, since this is already enforced by the model.

Bilevel and three-level programs are in general very difficult objects. The feasible region for the optimization problem at the first level in (3lvMIP) is determined by a budget constraint and by a inner bilevel problem, which makes this region extremely difficult to describe. For the particular case we are studying, notice that the three levels share the same objective function, up to the direction of optimization. This allow (3lvMIP) to be equivalently reformulated as a single-level program as follows. Let \( \mathcal{U} = \{ y \in \{0,1\}^V \mid \sum_{v \in V} y_v \leq \Phi \} \) be the set of all possible feasible attack strategies. With respect to the definition of \( x \) and \( \alpha \) given above, we denote by \( x(y) \) and \( \alpha(y) \), respectively, the indicator variables for the protection level in response to the attack scenario \( y \in \mathcal{U} \). Concretely, we define two indicator vectors \( x(y), \alpha(y) \in \{0,1\}^V \) for any possible feasible attack
Thus, $(3lvMIP)$ is equivalent to the following MILP.

$$
\begin{align*}
\max & \quad \Delta \\
\Delta & \leq \sum_{v \in V} \alpha_v(y) \quad \forall y \in U \\
\sum_{v \in V} z_v & \leq \Omega \\
\sum_{v \in V} x_v(y) & \leq \Lambda \quad \forall y \in U \\
\alpha_v(y) & \leq 1 + z_v - y_v \quad \forall v \in V, \forall y \in U \\
\alpha_v(y) & \leq \alpha_u(y) + x_v(y) + z_v \quad \forall (u, v) \in A, \forall y \in U \\
\alpha_v(y) & \leq 1 \quad \forall v \in V, \forall y \in U \\
z, x(y) & \in \{0, 1\}^V.
\end{align*}
$$

Clearly, every element of $U$ contributes to the size of the model $\Theta(|V|)$ variables and $\Theta(|V| + |A|)$ constraints. Since the size of $U$ is $\Omega(|V|^{\Phi})$, the size of $(1lvMIP)$ gets prohibitively large for efficient computations on any nontrivial graph. Thus, any reasonable approach has to take into account fewer scenarios in $U$. More precisely, to follow this approach, it is necessary to find a good subset of attack strategies $Q \subseteq U$ for which $(1lvMIP_Q)$, i.e., $(1lvMIP)$ restricted to $Q$, still returns an optimal solution. In fact, a natural approach to arrive at an optimal solution, consists of repetitively solving $(1lvMIP_Q)$, starting from the empty set $Q = \emptyset$ and adding each time a new attack strategy $y \in U$ to $Q$, until a guarantee of optimality is reached. The selection of any $y \in U$ is done according to the impact that it potentially provides to the quality of the solution at the following step. Any new scenario added to $Q$ increases the size of $(1lvMIP_Q)$ by $2|A|$ variables and $2|V| + |A| + 2$ constraints. In other words, this can be interpreted as a simultaneous column and row generation approach. Column and row generation approaches have been extensively studied in the literature, see for example [51, 77]. This high-level approach can be formalized as follows and is guaranteed to return an optimal solution for $(1lvMIP)$.

1. Initialize an empty set of scenarios $Q = \emptyset$. 
2. Solve \((1lvMIP_\mathcal{Q})\) (this gives an upper bound). Let \(\text{best}\) and \(z_{\text{best}}\) be the optimal value and the \(z\)-part of an optimal solution, respectively.

3. If there is any \(y \in \mathcal{U}\) such that solving the third stage (protection), given the attack \(y\) and the initial vaccination \(z_{\text{best}}\), the number of nodes surviving is strictly less than \(\text{best}\), add \(y\) to \(\mathcal{Q}\) and go to step 2; otherwise, return \(\text{best}\) and \(z_{\text{best}}\).

The vaccination of a subset of nodes \(V^{z_{\text{best}}} = \{v \in V \mid z_{\text{best}}v = 1\}\) at the first level corresponds to the removal of these nodes from the original graph for the next levels. For instance, once a node is saved in the first stage, it represents a disconnection through which the attack cannot propagate. Thus, step 3 requires to solve the Attack-Protect problem \(\text{AP}\) restricted to the subgraph \(G[V \setminus V^{z_{\text{best}}}]\), which is a bilevel program consisting of second and third level in \((3lvMIP)\). By relaxing the third level variables, the corresponding problem can reformulated as a MILP by applying strong duality and exploiting the McCormick convex relaxation, see [74], to linearize the bilinear terms. An optimal value for this MILP provides an upper bound to the best value for the original Attack-Protect problem. We will see in the next section how to solve the Attack-Protect problem, and in particular how to obtain the aforementioned MILP, which we call \(rlx\text{AP}\).

At this point, it is important to mention that, since an optimal solution to \(rlx\text{AP}\) returns an upper bound to the \(\text{AP}\) problem, in the event that this upper bound is smaller than \(\text{best}\), the corresponding attack gives a scenario that can be added to \(\mathcal{Q}\). On the other hand, if this upper bound is greater than \(\text{best}\), no conclusion can be drawn and it is necessary to proceed further in the exploration of the possible attack strategies for \(\text{AP}\) restricted to \(G[V \setminus V^{z_{\text{best}}}]\). In particular, we are looking for an attack strategy for the \(\text{AP}\) problem such that either its value is smaller than \(\text{best}\), thus it can be added to \(\mathcal{Q}\), or it is optimal for \(\text{AP}\) and its value equals to \(\text{best}\), thus \(V^{z_{\text{best}}}\) is optimal for \(\text{MCN}\). This implies that at least once in the whole algorithm we use to solve \((1lvMIP)\), there exists an \(\text{AP}\) instance, restricted to some \(G[V \setminus V^{z_{\text{best}}}]\), that has to be solved to optimality and such \(V^{z_{\text{best}}}\) corresponds to the optimal strategy for \(\text{MCN}\). For all the other strategies that are not optimal for \(\text{MCN}\), we only require to find a strategy for \(\text{AP}\) whose value is smaller than \(\text{best}\).
We conclude this part by describing the full algorithm for solving the MCN Problem and we proceed in the following subsection by presenting the subroutine used for the AP problem, \( AP(V, A, \Phi, \Lambda, \text{goal}) \). We anticipate that \( AP(V, A, \Phi, \Lambda, \text{goal}) \) receives as input, together with the budget variables and the value \( \text{goal} \), the induced subgraph \( G[V \setminus D] \) in which the vaccinated nodes are removed, and returns an attacking strategy \( I \) and a status flag for that solution that depends on \( \text{goal} \). The value \( \text{goal} \) represents the quality that an attacking strategy should outmatch in order to be outputted without proof of optimality. In particular, \( \text{goal} \) represents the value \( \text{best} \). However, since \( \text{AP} \) works on the induced subgraph \( G[V \setminus D] \), \( \text{goal} \) has to not count towards the already saved nodes \( D \), namely \( \text{goal} = \text{best} - |D| \). In case \( \text{goal} \) is strictly smaller than optimal value for \( \text{AP} \), and hence \( \text{goal} \) is an underestimator of the optimal value, \( \text{AP}(V, A, \Phi, \Lambda, \text{goal}) \) runs until it finds an optimal strategy \( I \) and it guarantees optimality for it. The status flag stores the information about these two possible states for the returned solution \( I \). Concretely, in case the value of the strategy is smaller or equal than \( \text{goal} \), the corresponding status is “goal”. Otherwise, if the value of the strategy is strictly greater than \( \text{goal} \), the corresponding status is “opt”, since it has to be optimal as already discussed. In the following, let \( V_D = V \setminus D \) and \( A_D = A[G[V_D]] \) be the set of nodes and arcs of the subgraph induced by the removal of \( D \), respectively.

\[
\text{MCN}(V, A, \Omega, \Phi, \Lambda) : \text{Solving the Multilevel Critical Node problem}
\]

\[
Q \leftarrow \emptyset, \text{best} \leftarrow |V|, D \leftarrow \emptyset.
\]

While True:

\[
\text{goal} \leftarrow \text{best} - |D| .
\]

\[
(I, \text{status}) \leftarrow \text{AP}(V_D, A_D, \Phi, \Lambda, \text{goal}).
\]

If status = “opt”:

\[
\text{Return } (D, \text{best}).
\]

If status = “goal”:

\[
Q \leftarrow Q \cup \{I\}.
\]

\[
(D = V^{z_{\text{best}}}, \text{best}) \leftarrow \text{solve}(1lvMIP_Q).
\]
3.2. SOLVING THE MULTILEVEL CRITICAL NODE PROBLEM

3.2.1 Solving the Attack-Protect problem

A bilevel formulation for the Attack-Protect problem AP is modeled by the second and the third level in 3lvMIP:

\[
\min_{y \in \{0, 1\}^V} \sum_{v \in V} \alpha_v \\
\sum_{v \in V} y_v \leq \Phi
\]

where \( \alpha_v \) solves the defender’s problem

\[
\max_{x \in \{0, 1\}^V} \sum_{v \in V} \alpha_v \\
\sum_{v \in V} x_v \leq \Lambda \\
\alpha_v \leq 1 - y_v \quad \forall v \in V \\
\alpha_j \leq \alpha_i + x_j \quad \forall (i, j) \in A.
\]

This problem can be solved using an approach similar to the one seen in [23]. More precisely, by relaxing the third level variables, we can substitute the third level, which corresponds to a relaxed Fence problem

\[
\max \sum_{v \in V} \alpha_v \\
\sum_{v \in V} x_v \leq \Lambda \\
\alpha_v \leq 1 - y_v \quad \forall v \in V \\
\alpha_j - \alpha_i - x_j \leq 0 \quad \forall (i, j) \in A \\
x_v \geq 0 \quad \forall v \in V,
\]

(Primal)
with its corresponding dual problem,

\[
\begin{align*}
\min \quad & \Lambda p + \sum_{v \in V} (1 - y_v) h_v \\
& h_v + \sum_{(i,v) \in A} q_{(i,v)} - \sum_{(v,j) \in A} q_{(v,j)} = 1 \quad \forall v \in V \\
p - \sum_{(i,v) \in A} q_{(i,v)} \geq 0 \quad \forall v \in V \\
p, h_v, q_{(i,j)} \geq 0 \quad \forall v \in V, \forall (i,j) \in A.
\end{align*}
\]

(Dual)

Since, by strong duality, for a primal-dual optimal solution \(\sum_{v \in V} \alpha_v = \Lambda p + \sum_{v \in V} (1 - y_v) h_v\) holds, after relaxing the third level variables, (21vAP) can be rewritten as follows.

\[
\begin{align*}
\min \quad & \Lambda p + \sum_{v \in V} (1 - y_v) h_v \\
& \sum_{v \in V} y_v \leq \Phi \\
& h_v + \sum_{(i,v) \in A} q_{(i,v)} - \sum_{(v,j) \in A} q_{(v,j)} = 1 \quad \forall v \in V \\
p - \sum_{(i,v) \in A} q_{(i,v)} \geq 0 \quad \forall v \in V \\
p, h_v, q_{(i,j)} \geq 0 \quad \forall v \in V, \forall (i,j) \in A \\
y_v \in \{0, 1\} \quad \forall v \in V.
\end{align*}
\]

The above mathematical program can be linearize to a MILP exploiting the McCormick envelope, see [74], for the bilinear terms \((1 - y_v) h_v\). More precisely, we introduce the auxiliary variables \(\gamma_v\), so as \(\gamma_v = (1 - y_v) h_v\), for every \(v \in V\). Since we deal with a minimization problem, it suffices to consider the two under-estimators \(\gamma_v \geq 0\) and \(\gamma_v \geq h_v - |V| y_v\), where we use the valid upper bound \(h_v \leq |V|\). The latter upper bound follows by summing up all the inequalities \(h_v + \sum_{(i,v) \in A} q_{(i,v)} - \sum_{(v,j) \in A} q_{(v,j)} = 1\) for all nodes \(v \in V\) obtaining \(\sum_{v \in V} h_v = |V|\), since variables \(q_{(i,j)}\) cancel out for all \((i,j) \in A\). Finally, applying the latter linearization we obtain the
3.2. **SOLVING THE MULTILEVEL CRITICAL NODE PROBLEM**

following MILP.

\[
\begin{align*}
\min & \quad \Lambda p + \sum_{v \in V} \gamma_v \\
\sum_{v \in V} y_v & \leq \Phi \\
\sum_{(i,v) \in A} q(i,v) - \sum_{(v,j) \in A} q(v,j) & = 1 \quad \forall v \in V \\
p - \sum_{(i,v) \in A} q(i,v) & \geq 0 \quad \forall v \in V \\
\gamma_v + |V| y_v - h_v & \geq 0 \quad \forall v \in V \\
p, h_v, \gamma_v, q(i,j) & \geq 0 \quad \forall v \in V, \forall (i,j) \in A \\
y_v & \in \{0,1\} \quad \forall v \in V.
\end{align*}
\]

The optimal objective value of (rlxAP) provides only an upper bound to (2lvAP). This is due to the fact that the relaxation of the third level makes the defense response more effective against any possible attack. Moreover, even in case that the solution is integral, there is no guarantee it is also an optimal solution for (2lvAP), as already noted in [23]. The actual quality of the optimal solution \( y \) given by (rlxAP) can be estimated solving the third level (Fence problem) once the attack strategy \( V y \) is fixed. The optimal saved region obtained by the defensive strategy, i.e., the set of saved nodes, characterizes all the possible nodes that a potential better attack should take into account. More precisely, given any set of attacked nodes \( I \subseteq V \) and a corresponding best defensive strategy \( P \subseteq V \), corresponding to an optimal solution to the Fence problem with infected set \( I \), consider the set of saved nodes \( S \subseteq V \). Notice that \( P \subseteq S \) and \( I \cap S = \emptyset \). Any potentially more effective attack \( I^+ \) has to satisfies \( I^+ \cap S \neq \emptyset \). This is due to the fact that, if \( I^+ \cap S = \emptyset \) the strategy \( P \subseteq S \) would save at least all the nodes in \( S \) for the given attack \( I^+ \).

The previous consideration guarantees that, if an optimal solution \( y \) for (rlxAP) does not coincide with an optimal attack \( y_o \) for (2lvAP), the cut

\[
\sum_{v \in V} \chi^S(v) y_v \geq 1,
\]

(cutAP)
does not eliminate \( y_o \) from the feasible region when added to (rlxAP), and, on the other hand, it cuts off the current solution \( y \). By preceding arguments, the following procedure represents a possible approach to find an optimal solution to (2lvAP).

1. Solve (rlxAP). Let \( y \) be the indicator vector for the attack’s strategy in an optimal solution.

2. Given the attack \( y \), solve the corresponding Fence problem and find the optimal saved set \( S_y \).

3. Add (cutAP) corresponding to \( S_y \) to (rlxAP). If (rlxAP) is feasible, go to step 1; otherwise, return \( V^y \) and the corresponding protection strategy for the solution obtained in step 2.

When we say that we add a cut (cutAP) to (rlxAP), we implicitly assume we are modifying (rlxAP). In particular, the number of constraints in (rlxAP) increases after any iteration of the procedure presented above.

As mentioned before, the algorithm we are developing for the MCN problem does not always require to find an optimal solution from the subroutine that solves (AP). Instead, the value of a returned solution for (2lvAP) may just need to be strictly smaller than a given bound. Recall that this bound \( best \) is evaluated by the main algorithm \( MCN(V, A, \Omega, \Phi, \Lambda) \), and given as input \( \text{goal} = best - |D| \) to the subroutine solving (2lvAP). It represents the critical threshold that the value of an attack has to overtake in order to be added to \( Q \). Thus, we include two control flow statements in steps 1 and 2 of the high level procedure described above that interrupt the algorithm when the quality of the strategy found fulfills this condition, and return the strategy. Moreover, \( MCN(V, A, \Omega, \Phi, \Lambda) \) requires to know whether the returned strategy by the subroutine has been outputted because of optimality or because it attains the required quality. As we already said in the previous section, we store this information as a status flag, whether optimal (“opt”) or outmatching the required bound (“goal”).

We conclude this section by showing the subroutine for AP in Algorithm AP\( (V, A, \Phi, \Lambda, \text{goal}) \). Moreover, we denote by \( P(V, A, I, \Lambda) \) the subroutine
3.3. **COMPUTATIONAL RESULTS**

that, given a graph \((V, A)\), the set of infected nodes \(I\), and the protection
budget \(\Lambda\), returns the set of nodes that are saved by the optimal protection
strategy for the corresponding Fence problem. Concretely, \(P(V, A, I, \Lambda)\)
can be easily modeled as a MIP and solved by any MIP solver.

\[
\text{AP}(V, A, \Phi, \Lambda, \text{goal}) : \text{Subroutine for AP.}
\]

\[
\begin{align*}
\text{best} & \leftarrow |V|, \ I^\text{best} \leftarrow \emptyset. \\
\text{While (rlxAP) is feasible :} & \\
& \quad (I = \chi^y, \text{value}) \leftarrow \text{solve (rlxAP)}. \\
& \quad \text{If value} \leq \text{goal} - 1: \\
& \quad \quad \text{Return (I, “goal”).} \\
& \quad S \leftarrow P(V, A, I, \Lambda). \\
& \quad \text{If } |S| \leq \text{goal} - 1: \\
& \quad \quad \text{Return (I, “goal”).} \\
& \quad \text{If } |S| < \text{best}: \\
& \quad \quad \text{best} \leftarrow |S|, \ I^\text{best} \leftarrow I. \\
& \quad \quad \text{Add } \sum_{v \in V} \chi^S(v)y_v \geq 1 \text{ to (rlxAP).} \\
\text{Return (I}^\text{best}, \text{“opt”).}
\end{align*}
\]

3.3 **Computational Results**

In this section we present some computational experiment for the algorithm
we designed for the MCN problem. Algorithm \(\text{MCN}(V, A, \Omega, \Phi, \Lambda)\) was
implemented in Python 2.7.6, all MIPs have been solved with IBM ILOG
CPLEX 12.6.2.0 and experiments were conducted on Intel Xeon E3-1220
processor clocked at 3.10 GHz and 8 GB RAM, using a single core.
The test-set consists of randomly generated instances with the following two
graph topologies.

- **Randomly generated trees**: An instance is built by starting from a
  node and adding in each step a new node together with an edge that
connects this node to a randomly chosen node already in the tree.

- **Random graph**: An instance is generated by adding all the nodes at once and then choosing a uniformly random set of edges of a given size. This size is determined by the parameter *density*, which represents the fraction of chosen edges with respect to the total possible number of edges in a complete graph.

In both cases, we consider edges to be bidirected. Moreover, test instances vary with respect to the number of nodes and to the availability of different budgets, $\Omega$, $\Phi$, and $\Lambda$, for the different stages of the problem.

Our computational experiments investigate both the performance of the MCN algorithm in terms of running time and iterations, and the relative gain defined in (3.1). Recall that the relative gain measures the extent to which the three level approach impacts the quality of an optimal solution compared to the solution obtained by solving to optimality the different levels sequentially. The relative gain, as defined in (3.1) is a value in the interval $[0, 1]$, but here we present it as a percentage, namely a value in the interval $[0, 100]$.

For every regime we investigate, classified by the type of graph, the size, and the budget vector $(\Omega, \Phi, \Lambda)$, we generated and tested 20 instances. We then report the average values for these 20 computed instances. Every computation was aborted after a time limit of 2 hours. We reported the average values of the attributes we considered only for the instances successfully solved to optimality.

The following list summarizes the notations of the considered attributes reported in the tables.

- **type**: The type of the graph: *tree* for random trees and *rndgraph$(d)$* for random graphs of density $d$.
- **size**: the number of nodes in the graph.
- $\Omega$-$\Phi$-$\Lambda$: the budgets available for the three levels.
- **sol%**: percentage of instances optimally solved within time limit (2 hours).
3.3. COMPUTATIONAL RESULTS

- **time**: average total running time (seconds) for the solved instances.
- **AP\%**: average time spent by the last call to AP given as percentage of the total time.
- **it**: average number of iterations in MCN of the solved instances.
- **RG\%**: percentage of instances with a nonzero relative gain among the solved instances.
- **avg**: average relative gain for instances with nonzero relative gain.
- **max**: maximum relative gain observed for the solved instances.

Our computational analysis starts by comparing different instances with identical graph topology but with different size and vector of budgets. Figure 3.2 summarizes our results for random trees. For all graph classes we considered the sizes (number of nodes) 20, 40, 60, 80, and 100. The budgets in our computations range in the set \{1, 2, 3\}. As expected, instances get harder to solve as the number of nodes and the budgets increase. In particular, for trees of size 100, only roughly half of the instances could be solve to optimally. Moreover, a large budget for the second level \(\Phi\) seems to slow the performance down more than increasing the budget for the first and the third levels, \(\Omega\) and \(\Lambda\). Another observation is that the more difficult the instance is the bigger is the impact that the last call to AP has on the total time. Recall that the last call to AP is the one that has to return a solution to a corresponding AP problem with guarantee of optimality. For all the other calls, the solution are only required to match the quality of some given bound. Hence, Figure 3.2 gives evidence that for hard instances on trees, the algorithm suffers in performance when it has to achieve a guarantee of optimality in the last call to the AP subroutine. For what concerns the relative gain, it seems that, for sufficiently large trees, there is a high probability the instance has a nonzero relative gain if the attack budget \(\Phi\) is small. However, these relative gains are small. On the other hand, we notice higher relative gains for instances that have high budget for the second and the third level.

Figure 3.3 shows computational results on random graph of density 5\%, with sizes and budget vectors identical to ones used for the tree case. Based on
the number of instances solved, it seems that also in this case the algorithm’s performance declines when the attack budget $\Phi$ increases, while the increase of $\Omega$ and $\Lambda$ has a milder effect. Moreover, for sizes smaller than 40, we notice that, analogously to the tree case, the last call to $\text{AP}$ has a very high impact on the running time for hard instances. However, for sizes bigger than 60, this phenomenon disappears and it seems that the performance of the algorithm is not dependant on the time required for $\text{AP}$ to obtain a certified optimal solution.

Apart from small sizes, random graphs of density 5% show much higher relative gains compared to tree instances. In particular, for the case in which all budgets are 1, we notice very high relative gains: up to 40% per instance. In general, for other budget vectors, computations show many instances with nonzero relative gains and, on average, they are significant.

Finally, Figure 3.4 shows results for random graphs with different densities ranging from 5% to 15% for a fixed size of 40 nodes. On a very high level, it is hard to arrive at general conclusions for the affect of the graph’s density on the studied criteria, such as running time, performance and relative gain. However, if we restrict our investigation to particular fixed budget vectors, we can point out some interesting behavior. For example, for budgets $\Omega = 1, \Phi = 3, \Lambda = 3$ we notice a significant decrease in running times when the density increases from 5% to 15%. However, for $\Omega = 3, \Phi = 3, \Lambda = 3$, the algorithm perform better if the density is around $7 - 8\%$. Moreover, over all computations we perform, it seems that the largest relative gains have attained for budgets $\Omega = 3, \Phi = 1, \Lambda = 3$ with density around 8%.
3.3. **COMPUTATIONAL RESULTS**

Figure 3.2: Experiments on tree instances

<table>
<thead>
<tr>
<th>size</th>
<th>Ω-Φ-Λ</th>
<th>sol%</th>
<th>time(s)</th>
<th>AP%</th>
<th>it</th>
<th>RG%</th>
<th>avg</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1-1-1</td>
<td>100.0</td>
<td>1.5</td>
<td>60.0</td>
<td>3.4</td>
<td>10.0</td>
<td>7.1</td>
<td>7.1</td>
</tr>
<tr>
<td>20</td>
<td>2-2-2</td>
<td>100.0</td>
<td>15.1</td>
<td>91.4</td>
<td>5.8</td>
<td>30.0</td>
<td>6.7</td>
<td>6.7</td>
</tr>
<tr>
<td>20</td>
<td>3-3-3</td>
<td>100.0</td>
<td>227.3</td>
<td>98.2</td>
<td>11.1</td>
<td>50.0</td>
<td>6.8</td>
<td>7.1</td>
</tr>
<tr>
<td>20</td>
<td>1-3-3</td>
<td>100.0</td>
<td>109.0</td>
<td>98.4</td>
<td>4.5</td>
<td>25.0</td>
<td>11.6</td>
<td>16.7</td>
</tr>
<tr>
<td>20</td>
<td>3-1-3</td>
<td>100.0</td>
<td>3.0</td>
<td>76.7</td>
<td>5.0</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>3-3-1</td>
<td>100.0</td>
<td>61.5</td>
<td>95.4</td>
<td>8.4</td>
<td>10.0</td>
<td>7.7</td>
<td>7.7</td>
</tr>
<tr>
<td>40</td>
<td>1-1-1</td>
<td>100.0</td>
<td>2.2</td>
<td>68.2</td>
<td>3.4</td>
<td>10.0</td>
<td>3.7</td>
<td>3.7</td>
</tr>
<tr>
<td>40</td>
<td>2-2-2</td>
<td>100.0</td>
<td>70.9</td>
<td>95.2</td>
<td>6.0</td>
<td>25.0</td>
<td>3.3</td>
<td>3.6</td>
</tr>
<tr>
<td>40</td>
<td>3-3-3</td>
<td>5.0</td>
<td>2981.2</td>
<td>99.5</td>
<td>17.0</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>1-3-3</td>
<td>95.0</td>
<td>870.8</td>
<td>99.7</td>
<td>4.6</td>
<td>31.6</td>
<td>10.5</td>
<td>20.0</td>
</tr>
<tr>
<td>40</td>
<td>3-1-3</td>
<td>100.0</td>
<td>6.8</td>
<td>85.3</td>
<td>5.2</td>
<td>40.0</td>
<td>2.9</td>
<td>5.1</td>
</tr>
<tr>
<td>40</td>
<td>3-3-1</td>
<td>100.0</td>
<td>519.1</td>
<td>98.4</td>
<td>10.3</td>
<td>15.0</td>
<td>6.8</td>
<td>12.0</td>
</tr>
<tr>
<td>60</td>
<td>1-1-1</td>
<td>100.0</td>
<td>2.8</td>
<td>67.9</td>
<td>3.3</td>
<td>10.0</td>
<td>8.1</td>
<td>11.4</td>
</tr>
<tr>
<td>60</td>
<td>2-2-2</td>
<td>100.0</td>
<td>239.0</td>
<td>97.5</td>
<td>6.2</td>
<td>15.0</td>
<td>3.7</td>
<td>6.7</td>
</tr>
<tr>
<td>60</td>
<td>3-3-3</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>60</td>
<td>1-3-3</td>
<td>40.0</td>
<td>2241.9</td>
<td>99.7</td>
<td>4.5</td>
<td>25.0</td>
<td>6.1</td>
<td>9.4</td>
</tr>
<tr>
<td>60</td>
<td>3-1-3</td>
<td>100.0</td>
<td>12.5</td>
<td>88.8</td>
<td>5.2</td>
<td>55.0</td>
<td>1.7</td>
<td>1.8</td>
</tr>
<tr>
<td>60</td>
<td>3-3-1</td>
<td>65.0</td>
<td>1607.4</td>
<td>98.7</td>
<td>10.8</td>
<td>46.2</td>
<td>5.0</td>
<td>8.3</td>
</tr>
<tr>
<td>80</td>
<td>1-1-1</td>
<td>100.0</td>
<td>3.6</td>
<td>61.1</td>
<td>3.5</td>
<td>10.0</td>
<td>2.6</td>
<td>3.4</td>
</tr>
<tr>
<td>80</td>
<td>2-2-2</td>
<td>100.0</td>
<td>561.9</td>
<td>98.2</td>
<td>6.7</td>
<td>30.0</td>
<td>5.4</td>
<td>8.5</td>
</tr>
<tr>
<td>80</td>
<td>3-3-3</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>1-3-3</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>3-1-3</td>
<td>100.0</td>
<td>18.8</td>
<td>89.9</td>
<td>5.2</td>
<td>70.0</td>
<td>1.7</td>
<td>3.8</td>
</tr>
<tr>
<td>80</td>
<td>3-3-1</td>
<td>20.0</td>
<td>2341.4</td>
<td>98.3</td>
<td>10.2</td>
<td>25.0</td>
<td>6.1</td>
<td>6.1</td>
</tr>
<tr>
<td>100</td>
<td>1-1-1</td>
<td>100.0</td>
<td>4.3</td>
<td>60.5</td>
<td>3.7</td>
<td>15.0</td>
<td>5.1</td>
<td>8.3</td>
</tr>
<tr>
<td>100</td>
<td>2-2-2</td>
<td>100.0</td>
<td>915.6</td>
<td>98.2</td>
<td>6.6</td>
<td>25.0</td>
<td>3.3</td>
<td>4.3</td>
</tr>
<tr>
<td>100</td>
<td>3-3-3</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>1-3-3</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>3-1-3</td>
<td>100.0</td>
<td>29.3</td>
<td>91.1</td>
<td>5.3</td>
<td>45.0</td>
<td>1.8</td>
<td>5.2</td>
</tr>
<tr>
<td>100</td>
<td>3-3-1</td>
<td>10.0</td>
<td>3314.8</td>
<td>97.9</td>
<td>13.5</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 3.3: Experiments on random graphs of density 5%

<table>
<thead>
<tr>
<th>size</th>
<th>Ω-Φ-Λ</th>
<th>sol%</th>
<th>time(s)</th>
<th>AP%</th>
<th>it</th>
<th>RG%</th>
<th>avg</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1-1-1</td>
<td>100.0</td>
<td>2.6</td>
<td>80.8</td>
<td>3.0</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>2-2-2</td>
<td>100.0</td>
<td>32.9</td>
<td>97.3</td>
<td>5.9</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>3-3-3</td>
<td>100.0</td>
<td>209.0</td>
<td>99.1</td>
<td>8.6</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>1-3-3</td>
<td>100.0</td>
<td>521.4</td>
<td>99.8</td>
<td>4.2</td>
<td>25.0</td>
<td>7.7</td>
<td>12.5</td>
</tr>
<tr>
<td>20</td>
<td>3-1-3</td>
<td>100.0</td>
<td>2.9</td>
<td>75.9</td>
<td>5.0</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>3-3-1</td>
<td>100.0</td>
<td>164.4</td>
<td>98.5</td>
<td>10.7</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>1-1-1</td>
<td>100.0</td>
<td>2.6</td>
<td>53.8</td>
<td>3.9</td>
<td>70.0</td>
<td>17.5</td>
<td>30.0</td>
</tr>
<tr>
<td>40</td>
<td>2-2-2</td>
<td>100.0</td>
<td>58.2</td>
<td>94.0</td>
<td>6.2</td>
<td>70.0</td>
<td>9.9</td>
<td>19.2</td>
</tr>
<tr>
<td>40</td>
<td>3-3-3</td>
<td>5.0</td>
<td>3462.3</td>
<td>99.6</td>
<td>14.0</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>1-3-3</td>
<td>100.0</td>
<td>1850.3</td>
<td>99.4</td>
<td>4.6</td>
<td>60.0</td>
<td>9.6</td>
<td>19.0</td>
</tr>
<tr>
<td>40</td>
<td>3-1-3</td>
<td>100.0</td>
<td>6.7</td>
<td>85.1</td>
<td>5.0</td>
<td>75.0</td>
<td>3.8</td>
<td>7.9</td>
</tr>
<tr>
<td>40</td>
<td>3-3-1</td>
<td>100.0</td>
<td>605.7</td>
<td>97.0</td>
<td>11.4</td>
<td>45.0</td>
<td>10.2</td>
<td>15.8</td>
</tr>
<tr>
<td>60</td>
<td>1-1-1</td>
<td>100.0</td>
<td>2.9</td>
<td>30.0</td>
<td>4.2</td>
<td>15.0</td>
<td>27.1</td>
<td>40.0</td>
</tr>
<tr>
<td>60</td>
<td>2-2-2</td>
<td>90.0</td>
<td>701.3</td>
<td>2.0</td>
<td>14.1</td>
<td>77.8</td>
<td>8.0</td>
<td>14.3</td>
</tr>
<tr>
<td>60</td>
<td>3-3-3</td>
<td>15.0</td>
<td>3090.3</td>
<td>24.2</td>
<td>16.7</td>
<td>100.0</td>
<td>13.1</td>
<td>15.0</td>
</tr>
<tr>
<td>60</td>
<td>1-3-3</td>
<td>95.0</td>
<td>849.6</td>
<td>12.8</td>
<td>11.6</td>
<td>36.8</td>
<td>8.8</td>
<td>14.3</td>
</tr>
<tr>
<td>60</td>
<td>3-1-3</td>
<td>100.0</td>
<td>64.4</td>
<td>4.3</td>
<td>7.2</td>
<td>100.0</td>
<td>16.9</td>
<td>36.4</td>
</tr>
<tr>
<td>60</td>
<td>3-3-1</td>
<td>70.0</td>
<td>1457.7</td>
<td>5.7</td>
<td>22.1</td>
<td>42.9</td>
<td>9.1</td>
<td>10.0</td>
</tr>
<tr>
<td>80</td>
<td>1-1-1</td>
<td>100.0</td>
<td>5.4</td>
<td>22.2</td>
<td>4.0</td>
<td>10.0</td>
<td>14.6</td>
<td>16.7</td>
</tr>
<tr>
<td>80</td>
<td>2-2-2</td>
<td>70.0</td>
<td>1991.6</td>
<td>0.6</td>
<td>10.1</td>
<td>7.1</td>
<td>10.0</td>
<td>10.0</td>
</tr>
<tr>
<td>80</td>
<td>3-3-3</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>1-3-3</td>
<td>45.0</td>
<td>1374.7</td>
<td>3.6</td>
<td>9.4</td>
<td>33.3</td>
<td>12.5</td>
<td>12.5</td>
</tr>
<tr>
<td>80</td>
<td>3-1-3</td>
<td>95.0</td>
<td>707.2</td>
<td>0.3</td>
<td>6.9</td>
<td>68.4</td>
<td>8.7</td>
<td>18.8</td>
</tr>
<tr>
<td>80</td>
<td>3-3-1</td>
<td>55.0</td>
<td>2235.6</td>
<td>2.8</td>
<td>15.5</td>
<td>9.1</td>
<td>11.1</td>
<td>11.1</td>
</tr>
<tr>
<td>100</td>
<td>1-1-1</td>
<td>100.0</td>
<td>9.0</td>
<td>13.3</td>
<td>3.9</td>
<td>5.0</td>
<td>16.7</td>
<td>16.7</td>
</tr>
<tr>
<td>100</td>
<td>2-2-2</td>
<td>35.0</td>
<td>1018.6</td>
<td>1.3</td>
<td>7.7</td>
<td>42.9</td>
<td>13.1</td>
<td>14.3</td>
</tr>
<tr>
<td>100</td>
<td>3-3-3</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>1-3-3</td>
<td>45.0</td>
<td>2433.6</td>
<td>1.5</td>
<td>7.9</td>
<td>88.9</td>
<td>14.4</td>
<td>16.7</td>
</tr>
<tr>
<td>100</td>
<td>3-1-3</td>
<td>90.0</td>
<td>1731.5</td>
<td>0.1</td>
<td>6.3</td>
<td>38.9</td>
<td>8.0</td>
<td>9.1</td>
</tr>
<tr>
<td>100</td>
<td>3-3-1</td>
<td>10.0</td>
<td>1390.6</td>
<td>3.6</td>
<td>8.0</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 3.4: Experiments on random graphs of size 40 (1)

<table>
<thead>
<tr>
<th>type</th>
<th>Ω-Φ-Λ</th>
<th>sol%</th>
<th>time(s)</th>
<th>AP%</th>
<th>it</th>
<th>RG%</th>
<th>avg</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>rndgraph05 2-2-2</td>
<td>100.0</td>
<td>58.2</td>
<td>94.0</td>
<td>6.2</td>
<td></td>
<td>70.0</td>
<td>9.9</td>
<td>19.2</td>
</tr>
<tr>
<td>rndgraph06 2-2-2</td>
<td>100.0</td>
<td>27.4</td>
<td>67.2</td>
<td>7.7</td>
<td></td>
<td>90.0</td>
<td>12.3</td>
<td>28.6</td>
</tr>
<tr>
<td>rndgraph07 2-2-2</td>
<td>100.0</td>
<td>28.5</td>
<td>25.6</td>
<td>9.6</td>
<td></td>
<td>85.0</td>
<td>13.5</td>
<td>26.7</td>
</tr>
<tr>
<td>rndgraph08 2-2-2</td>
<td>100.0</td>
<td>54.1</td>
<td>12.8</td>
<td>10.5</td>
<td></td>
<td>50.0</td>
<td>12.6</td>
<td>21.4</td>
</tr>
<tr>
<td>rndgraph09 2-2-2</td>
<td>100.0</td>
<td>230.8</td>
<td>2.3</td>
<td>12.9</td>
<td></td>
<td>40.0</td>
<td>13.6</td>
<td>18.2</td>
</tr>
<tr>
<td>rndgraph10 2-2-2</td>
<td>100.0</td>
<td>254.5</td>
<td>1.6</td>
<td>11.9</td>
<td></td>
<td>35.0</td>
<td>15.5</td>
<td>25.0</td>
</tr>
<tr>
<td>rndgraph11 2-2-2</td>
<td>100.0</td>
<td>450.2</td>
<td>0.7</td>
<td>13.0</td>
<td></td>
<td>30.0</td>
<td>15.5</td>
<td>16.7</td>
</tr>
<tr>
<td>rndgraph12 2-2-2</td>
<td>100.0</td>
<td>236.0</td>
<td>1.2</td>
<td>11.3</td>
<td></td>
<td>65.0</td>
<td>16.2</td>
<td>16.7</td>
</tr>
<tr>
<td>rndgraph13 2-2-2</td>
<td>95.0</td>
<td>491.5</td>
<td>0.5</td>
<td>10.9</td>
<td></td>
<td>26.3</td>
<td>18.0</td>
<td>20.0</td>
</tr>
<tr>
<td>rndgraph14 2-2-2</td>
<td>100.0</td>
<td>118.2</td>
<td>2.3</td>
<td>9.8</td>
<td></td>
<td>55.0</td>
<td>18.2</td>
<td>20.0</td>
</tr>
<tr>
<td>rndgraph15 2-2-2</td>
<td>100.0</td>
<td>160.9</td>
<td>1.9</td>
<td>9.3</td>
<td></td>
<td>40.0</td>
<td>19.6</td>
<td>20.0</td>
</tr>
<tr>
<td>rndgraph05 3-3-3</td>
<td>5.0</td>
<td>3462.3</td>
<td>99.6</td>
<td>14.0</td>
<td></td>
<td>0.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rndgraph06 3-3-3</td>
<td>65.0</td>
<td>1831.2</td>
<td>98.0</td>
<td>13.4</td>
<td></td>
<td>69.2</td>
<td>9.6</td>
<td>17.4</td>
</tr>
<tr>
<td>rndgraph07 3-3-3</td>
<td>95.0</td>
<td>280.5</td>
<td>57.5</td>
<td>12.8</td>
<td></td>
<td>89.5</td>
<td>12.5</td>
<td>27.3</td>
</tr>
<tr>
<td>rndgraph08 3-3-3</td>
<td>70.0</td>
<td>631.6</td>
<td>9.5</td>
<td>16.5</td>
<td></td>
<td>85.7</td>
<td>12.2</td>
<td>25.0</td>
</tr>
<tr>
<td>rndgraph09 3-3-3</td>
<td>55.0</td>
<td>991.0</td>
<td>4.6</td>
<td>15.7</td>
<td></td>
<td>54.5</td>
<td>9.7</td>
<td>15.4</td>
</tr>
<tr>
<td>rndgraph10 3-3-3</td>
<td>35.0</td>
<td>1002.5</td>
<td>3.1</td>
<td>16.3</td>
<td></td>
<td>85.7</td>
<td>8.9</td>
<td>10.0</td>
</tr>
<tr>
<td>rndgraph11 3-3-3</td>
<td>35.0</td>
<td>1964.3</td>
<td>1.7</td>
<td>17.0</td>
<td></td>
<td>71.4</td>
<td>9.6</td>
<td>10.0</td>
</tr>
<tr>
<td>rndgraph12 3-3-3</td>
<td>30.0</td>
<td>2815.0</td>
<td>0.6</td>
<td>19.5</td>
<td></td>
<td>66.7</td>
<td>13.6</td>
<td>22.2</td>
</tr>
<tr>
<td>rndgraph13 3-3-3</td>
<td>20.0</td>
<td>2887.8</td>
<td>0.5</td>
<td>18.5</td>
<td></td>
<td>50.0</td>
<td>11.8</td>
<td>12.5</td>
</tr>
<tr>
<td>rndgraph14 3-3-3</td>
<td>35.0</td>
<td>2271.3</td>
<td>0.5</td>
<td>16.7</td>
<td></td>
<td>71.4</td>
<td>12.5</td>
<td>12.5</td>
</tr>
<tr>
<td>rndgraph15 3-3-3</td>
<td>50.0</td>
<td>2296.4</td>
<td>0.4</td>
<td>16.5</td>
<td></td>
<td>90.0</td>
<td>13.3</td>
<td>14.3</td>
</tr>
</tbody>
</table>
Figure 3.4: Experiments on random graphs of size 40 (2)

<table>
<thead>
<tr>
<th>type</th>
<th>Ω-Φ-Λ</th>
<th>sol%</th>
<th>time(s)</th>
<th>AP%</th>
<th>it</th>
<th>RG%</th>
<th>avg</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>rndgraph05</td>
<td>1-3-3</td>
<td>100.0</td>
<td>1850.3</td>
<td>99.4</td>
<td>4.6</td>
<td>60.0</td>
<td>9.6</td>
<td>19.0</td>
</tr>
<tr>
<td>rndgraph06</td>
<td>1-3-3</td>
<td>100.0</td>
<td>238.5</td>
<td>86.5</td>
<td>6.3</td>
<td>40.0</td>
<td>16.1</td>
<td>23.5</td>
</tr>
<tr>
<td>rndgraph07</td>
<td>1-3-3</td>
<td>100.0</td>
<td>109.2</td>
<td>37.3</td>
<td>8.4</td>
<td>50.0</td>
<td>10.9</td>
<td>23.1</td>
</tr>
<tr>
<td>rndgraph08</td>
<td>1-3-3</td>
<td>100.0</td>
<td>96.6</td>
<td>25.9</td>
<td>8.1</td>
<td>40.0</td>
<td>14.0</td>
<td>30.0</td>
</tr>
<tr>
<td>rndgraph09</td>
<td>1-3-3</td>
<td>100.0</td>
<td>145.1</td>
<td>11.4</td>
<td>9.4</td>
<td>40.0</td>
<td>14.0</td>
<td>25.0</td>
</tr>
<tr>
<td>rndgraph10</td>
<td>1-3-3</td>
<td>100.0</td>
<td>78.5</td>
<td>15.4</td>
<td>8.2</td>
<td>45.0</td>
<td>14.6</td>
<td>16.7</td>
</tr>
<tr>
<td>rndgraph11</td>
<td>1-3-3</td>
<td>100.0</td>
<td>361.8</td>
<td>2.7</td>
<td>8.4</td>
<td>45.0</td>
<td>15.6</td>
<td>16.7</td>
</tr>
<tr>
<td>rndgraph12</td>
<td>1-3-3</td>
<td>100.0</td>
<td>57.2</td>
<td>15.6</td>
<td>7.0</td>
<td>45.0</td>
<td>16.7</td>
<td>16.7</td>
</tr>
<tr>
<td>rndgraph13</td>
<td>1-3-3</td>
<td>100.0</td>
<td>87.2</td>
<td>9.2</td>
<td>7.5</td>
<td>20.0</td>
<td>16.7</td>
<td>16.7</td>
</tr>
<tr>
<td>rndgraph14</td>
<td>1-3-3</td>
<td>100.0</td>
<td>48.1</td>
<td>17.3</td>
<td>6.3</td>
<td>30.0</td>
<td>17.8</td>
<td>20.0</td>
</tr>
<tr>
<td>rndgraph15</td>
<td>1-3-3</td>
<td>100.0</td>
<td>37.3</td>
<td>23.9</td>
<td>5.5</td>
<td>10.0</td>
<td>20.0</td>
<td>20.0</td>
</tr>
<tr>
<td>rndgraph05</td>
<td>3-1-3</td>
<td>100.0</td>
<td>6.7</td>
<td>85.1</td>
<td>5.0</td>
<td>75.0</td>
<td>3.8</td>
<td>7.9</td>
</tr>
<tr>
<td>rndgraph06</td>
<td>3-1-3</td>
<td>100.0</td>
<td>7.1</td>
<td>77.5</td>
<td>6.2</td>
<td>75.0</td>
<td>6.1</td>
<td>13.2</td>
</tr>
<tr>
<td>rndgraph07</td>
<td>3-1-3</td>
<td>100.0</td>
<td>7.8</td>
<td>61.5</td>
<td>6.6</td>
<td>95.0</td>
<td>24.9</td>
<td>44.7</td>
</tr>
<tr>
<td>rndgraph08</td>
<td>3-1-3</td>
<td>100.0</td>
<td>29.7</td>
<td>7.7</td>
<td>6.9</td>
<td>95.0</td>
<td>24.4</td>
<td>41.9</td>
</tr>
<tr>
<td>rndgraph09</td>
<td>3-1-3</td>
<td>100.0</td>
<td>16.0</td>
<td>9.4</td>
<td>7.0</td>
<td>90.0</td>
<td>20.4</td>
<td>45.8</td>
</tr>
<tr>
<td>rndgraph10</td>
<td>3-1-3</td>
<td>100.0</td>
<td>31.7</td>
<td>5.0</td>
<td>7.4</td>
<td>70.0</td>
<td>11.9</td>
<td>17.6</td>
</tr>
<tr>
<td>rndgraph11</td>
<td>3-1-3</td>
<td>100.0</td>
<td>30.4</td>
<td>4.3</td>
<td>6.8</td>
<td>90.0</td>
<td>11.0</td>
<td>23.1</td>
</tr>
<tr>
<td>rndgraph12</td>
<td>3-1-3</td>
<td>100.0</td>
<td>42.7</td>
<td>2.8</td>
<td>7.1</td>
<td>60.0</td>
<td>11.2</td>
<td>23.1</td>
</tr>
<tr>
<td>rndgraph13</td>
<td>3-1-3</td>
<td>100.0</td>
<td>41.2</td>
<td>2.7</td>
<td>7.2</td>
<td>45.0</td>
<td>10.6</td>
<td>18.2</td>
</tr>
<tr>
<td>rndgraph14</td>
<td>3-1-3</td>
<td>100.0</td>
<td>37.3</td>
<td>2.9</td>
<td>7.0</td>
<td>40.0</td>
<td>10.3</td>
<td>11.1</td>
</tr>
<tr>
<td>rndgraph15</td>
<td>3-1-3</td>
<td>100.0</td>
<td>37.4</td>
<td>2.9</td>
<td>7.0</td>
<td>30.0</td>
<td>10.2</td>
<td>11.1</td>
</tr>
</tbody>
</table>
3.4 Conclusions

In this chapter we introduced a new three-stage model for protecting a network from a harmful spreading agent, based on a combination of the Critical Node and the Fence problems, which we call the Multilevel Critical Node problem. We devised an algorithm for this problem and tested its performance numerically on randomly generated graph structures. We made computational experiments on the relative gain of randomly generated instances, showing that this model often provides solutions that are significantly better than the ones obtained by solving the Critical Node problem and the Fence problem sequentially.

In this line of work, devising faster algorithms for MCN is an interesting direction for future works. Moreover, replacing the Fence problem with the Firefighter problem might be a possible way to describe more sophisticated models, which better capture some specific dynamics of network defending. Additionally, simultaneous vertex and edge removal, or only edge removal, can be considered instead.
Chapter 4

Conclusions

This thesis studied computational aspects of a number of well-known problems related to blocking harmful processes from spreading in networks. The results of this thesis can roughly be divided into contributions to the field of approximation algorithms for the Firefighter and related problems in tree networks, and contributions to modeling aspects of protection problems in the context of spreading phenomena.

For the Firefighter Problem in trees we devised the first polynomial time approximation scheme, thus basically settling its approximability status, assuming \( P \neq NP \). However, there is still a possibility that the Firefighter problem on trees admits an efficient polynomial time approximation scheme (EPTAS), namely an algorithm with running time \( f(\epsilon) \cdot \text{poly}(n) \). Since EPTASs are often practical alternative of PTASs, exploring this possibility is an interesting direction for future work.

For the Resource Minimization for Fire Containment problem we devised the first constant-factor approximation algorithm. It too is qualitatively best possible, as a polynomial \( 2 - \epsilon \) approximation for the problem does not exist, unless \( P \neq NP \). Our approximation guarantee is 12, and thus a constant gap still remains for this problem. We believe that many of the techniques developed for the latter approximation algorithms can be used to strengthen the bound of 12. This line of work is an interesting direction for future work.

The techniques developed here seem more general than just the Firefighter and the RMFC problems in trees. It is conceivable that they can also be applied to other restricted graph classes, for example. Furthermore, we believe that some techniques can also find applications to other problems.
In fact, very recently our techniques for the RMFC problem were applied to approximate the Non-Uniform $k$-Center Problem [56].

In the second part of this thesis we devised a new three-stage model for the protection problem of networks from a harmful spreading process, based on a combination of the Critical Node problem and the Fence problem. We presented an exact algorithm for this problem and analyzed it numerically. We showed that our three-stage model often provides solutions being significantly better than the ones obtained by a sequential approach. In this line of work, devising faster algorithms is an interesting direction for future work. Additionally, one can also replace the third stage problem with the more challenging Firefighter problem or considering edge removals instead.
Bibliography


