UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO DELAZERI

Models and Algorithms for Wildfire Suppression Problems

Thesis presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Advisor: Prof. Dr. Marcus Ritt

Porto Alegre
June 2025

CIP — CATALOGING-IN-PUBLICATION

Delazeri, Gustavo

Models and Algorithms for Wildfire Suppression Problems / Gustavo Delazeri. – Porto Alegre: PPGC da UFRGS, 2025.

127 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2025. Advisor: Marcus Ritt.

1. Combinatorial Optimization. 2. Operations Research. 3. Mixed-Integer Programming. 4. Mathematical Modeling. 5. Wildfire Management. 6. Heuristics. 7. Beam Search. I. Ritt, Marcus. II. Título.

“Time is the school in which we learn, time is the fire in which we burn.”

— DELMORE SCHWARTZ

AGRADECIMENTOS

I would like to start by thanking my advisor, Marcus Ritt. In 2019, I decided it was time to do something outside the regular undergraduate curriculum, and I applied for a scientific initiation scholarship under his supervision. That is one of the best decisions I have ever made. I can't thank him enough for all the guidance, book recommendations, life advice, career advice, and for always being available to discuss ideas. I hope I can help others half as much as Marcus has helped me over the years.

I would also like to thank UFRGS and the Institute of Informatics for the opportunity to study here since 2018. I am grateful to the professors and staff who continuously work to improve education and research at the university, even in the face of constant budget cuts. I was fortunate to benefit from the high-quality public education and research we have in Brazil, and I hope future generations will have the same opportunities. Speaking of important public institutions, I would like to thank CAPES for the financial support that made this research possible.

I once heard that the real scientific progress is the friends we make along the way. In this spirit, I thank my lab mates Victor, Marco, Gabriel, and Gustavo for all the interesting discussions, support and friendship. They made my time in the lab much more enjoyable, and I am grateful for the moments we shared. I would also like to extend my gratitude to André. His classes on planning and algorithms were very interesting, and I appreciate his helpfulness and availability for discussion.

I could not finish this section without mentioning my family. My parents, Nilso and Rosmeri, taught me the value of hard work and dedication by example. They always supported me in my decisions, even when they did not understand them. I am grateful for their love. My brother, Mauro, is certainly my greatest motivator and my best friend. If it were not for his insistence, my undergraduate degree and this dissertation would not exist. I hope I can be as good a brother to him as he is to me.

ABSTRACT

Wildfires cause major economic and environmental damage, and their frequency is rising. Effectively combating wildfires requires taking complex decisions under significant uncertainty and time pressure. Computational decision support tools can assist in this process. Among various Operations Research methods, graph-based modeling provides a suitable approach by integrating fire spread dynamics with the impact of suppression actions. However, progress in this area has been partly constrained by limitations in existing solution methods when applied to complex scenarios and by the reliance on benchmark instances often lacking scale or physical realism. This dissertation aims to advance graph-based wildfire suppression optimization. We introduce several contributions to address more realistic and complex problem instances: (1) An improved Mixed-Integer Programming (MIP) formulation that enhances solution quality and feasibility compared to previous models; (2) an Iterated Beam Search (IBS) algorithm that achieves a strong performance across many scenarios; (3) a fast Cut-based Heuristic (CUTH) that obtains competitive solutions in a few seconds, especially for large-scale problems; (4) a novel, instance generator using Rothermel's fire spread model for creating more realistic and systematically varied benchmarks; and (5) a new mathematical model incorporating operational factors such as resource heterogeneity, spatial constraints, safety buffers, and attack timing. Computational experiments using benchmarks from the literature and instances from the new generator demonstrate the effectiveness of the proposed methods.

Keywords: Combinatorial Optimization. Operations Research. Mixed-Integer Programming. Mathematical Modeling. Wildfire Management. Heuristics. Beam Search.

Modelos e Algoritmos para Problemas de Supressão de Incêndios Florestais

RESUMO

Incêndios florestais representam um desafio global crescente, aumentando em frequência e acarretando custos econômicos e ambientais substanciais. O combate eficaz a esses eventos exige a tomada de decisões complexas sob significativa incerteza e pressão de tempo. Ferramentas computacionais de apoio à decisão podem auxiliar nesse processo. Dentre os vários métodos da Pesquisa Operacional, a modelagem baseada em grafos fornece uma abordagem adequada ao integrar a dinâmica espacial da propagação do fogo com o impacto das ações de supressão. No entanto, o progresso nesta área tem sido parcialmente restringido pelas limitações dos métodos de solução existentes quando aplicados a cenários complexos e pela dependência de instâncias de benchmark que frequentemente carecem de escala ou realismo físico. Esta dissertação tem como objetivo avançar o estado da arte na otimização da supressão de incêndios florestais baseada em grafos. Introduzimos várias contribuições para abordar instâncias de problemas mais realistas e complexos: (1) Uma formulação aprimorada de Programação Inteira Mista (MIP) que resulta em melhor qualidade de solução e maior capacidade de encontrar soluções viáveis em comparação com modelos anteriores; (2) um algoritmo de Busca em Feixe Iterada (IBS) que alcança um ótimo desempenho na maioria das instâncias testadas; (3) uma Heurística Baseada em Cortes (CUTH), a qual obtém rapidamente soluções competitivas, especialmente para problemas de grande escala; (4) um novo gerador de instâncias que usa o modelo de propagação de fogo de Rothermel para criar benchmarks mais realistas e sistematicamente variados; e (5) um novo modelo matemático que incorpora fatores operacionais como heterogeneidade de recursos, restrições espaciais, restrições de segurança e de tempo de ataque. Experimentos computacionais usando benchmarks da literatura e instâncias do novo gerador demonstram a eficácia dos métodos propostos.

Palavras-chave: Otimização Combinatória. Pesquisa Operacional. Programação Inteira Mista. Modelagem Matemática. Gestão de Incêndios Florestais. Heurísticas. Busca em Feixe.

LIST OF ABBREVIATIONS AND ACRONYMS

CUTH	Cut-based Heuristic
ILS	Iterated Local Search
LBBD	Logic-based Benders Decomposition
MIP	Mixed-Integer Programming
OR	Operations Research
ROS	Rate of Spread
RS	Random Search
WAF	Wind Adjustment Factor

LIST OF SYMBOLS

V	Set of vertices
A	Set of arcs
s	Ignition vertex
t_{uv}	Fire travel time from vertex u to vertex v, for $uv \in A$
H	Time horizon of the problem
Δ	Delay introduced by a resource
k	Number of resources available
R	Set of resources
t_i	Release time of resource $i \in R$
N_v^-	Set of predecessors of vertex v
N_v^+	Set of successors of vertex v
$\mathbb{N}_v$	Extended neighborhood of vertex v
T	Ordered sequence of resource release times
R_t	Set of resources available at time t
$\alpha(t)$	First time after t when new resources become available
Λ	Allocation of resources
Λ_0	Empty allocation of resources
t_{uv}^Λ	Fire travel time from vertex u to vertex v in allocation Λ
a_v^Λ	Fire arrival time at vertex v in allocation Λ
p_v^Λ	Predecessor of vertex v in allocation Λ
B_t^Λ	Set of vertices burned by time t in allocation Λ
P^Λ	Set of vertices protected by allocation Λ
$\mathcal{P}_v^\Lambda$	Set of predecessors of vertex v in allocation Λ

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Wildfires are estimated to have caused global damage costs of about USD 69 billion in 2018–2023 (Munich Re, 2023; Joint Economic Committee, U.S. Senate, 2023). Their frequency and damage are likely to increase with climate change, with longer wildfire seasons, larger affected areas, and new locations of occurrence. They are at the same time harder to handle, since they coincide more frequently with dry air (United Nations, 2023; IPCC, 2023). Although deaths from wildfires are rare in comparison to other natural disasters, they destroy ecosystems, threaten homes, livelihoods, technical infrastructure such as railways and the electricity grid, and lead to a reversal of carbon capture (Reuters, 2023; IPCC, 2023).

Managing wildfire suppression efforts, particularly for large-scale fires, is a complex task. Decision-makers need to quickly interpret incomplete and continuously changing information about fire spread, weather conditions, and available resources. Coordination among various groups responsible for operations, planning, and logistics is also challenging due to conflicting demands and limited time (Martell, 1982; McLennan et al., 2006). Research indicates that effective wildfire management is significantly hindered by cognitive overload, uncertainty, and decision-making biases, such as persisting with ineffective strategies or misjudging risks (McLennan et al., 2006). Decision-making tools, such as mathematical models, can help managers better process complex information, systematically evaluate alternatives, and avoid common cognitive pitfalls (Martell et al., 2022).

Researchers in the field of Operations Research have been interested in the complex decisions involved in wildfire management since the early 1960s (Martell, 1982), and Jewell (1963) is one of the first works dealing with the application of operations research techniques to forest fire problems. Since then, we can find in the literature a variety of mathematical models that aim to capture decisions related to the process of preventing and suppressing a wildfire. Mendes & Alvelos (2023) categorize these models based on the type of decision they address. In vehicle routing models, for example, the goal is to determine which vehicles (e.g., water tankers) should visit which assets (e.g., factories) and when, in order to minimize the damage caused by the fire (Merwe et al., 2015). Covering models, on the other hand, address decisions about strategically locating resources, such as vehicles or stations, to maximize the potential for rapid initial attack across areas representing potential fire events (Dimopoulou; Giannikos, 2001). An important class of

models, which is the focus of this dissertation, are graph-based models.

Graph-based (or grid-based) models represent wildfire spread by discretizing the landscape into a grid of spatial units or cells. This discretization is then encoded as a graph, where each cell is a vertex and arcs connect vertices associated with adjacent cells. The arcs are labeled with the estimated fire travel time from the center of a cell to the center of an adjacent cell, which depend on local conditions such as fuel type, wind speed, and slope. Fire propagation across the landscape can then be approximated by calculating the shortest paths from an ignition point to other cells. An advantage of this approach is the ability to model how firefighting actions influence fire spread. If a fire suppression resource is deployed to a cell, fire spread through that cell is delayed, similar to the effect of a firebreak. Integrating fire spread and suppression actions into a single optimization model allows us to answer questions such as where should resources be deployed to minimize the total area burned within a given time horizon, or if there are enough resources to protect a critical asset in the event of a fire. We review graph-based models in detail in Chapter 2.

This dissertation focuses specifically on a graph-based model first defined by Alvelos (2018), and contributes through the development and evaluation of new algorithms to solve this model. The main contributions presented in this dissertation are:

1. An improved Mixed-Integer Programming formulation that combines efficient fire propagation constraints with time-dependent resource allocation, resulting in significantly better performance than previous formulations from the literature (Section 3.1).
2. An iterated beam search algorithm that demonstrates state-of-the-art performance, finding optimal or near-optimal solutions quickly on benchmark instances from the literature (Section 3.2).
3. A fast matheuristic based on computing vertex cuts in a graph, which provides competitive solutions in a few seconds, particularly for large-scale instances where most methods struggle (Section 3.3).
4. A new instance generator based on Rothermel's model for fire spread, capable of creating diverse benchmarks of instances with varying environmental factors, resource availability, and complexity (Chapter 4).
5. An extension of the basic model incorporating realistic operational constraints including resource heterogeneity, spatial limitations, safety constraints, and attack

expiration times (Chapter 5).

Contributions 2 and 5 were published in (Delazeri; Ritt, 2024b) and (Delazeri; Ritt, 2024a), respectively.

The remainder of this dissertation is structured as follows. Chapter 2 provides essential background on wildfire behavior modeling, Operations Research applications in fire management, and formally defines the graph-based suppression problem we will consider. Chapter 3 presents the core algorithmic contributions: the improved MIP formulation, the Iterated Beam Search, and the Cut-based Heuristic, along with a computational comparison on existing benchmark instances. Chapter 4 details the new instance generator and presents an extensive experimental study evaluating the algorithms under various conditions. Chapter 5 describes the extended MIP model that incorporates more realistic suppression features. Finally, Chapter 6 summarizes the dissertation and discusses potential directions for future research.

2 BACKGROUND

2.1 Operations Research in Forest Fire Management

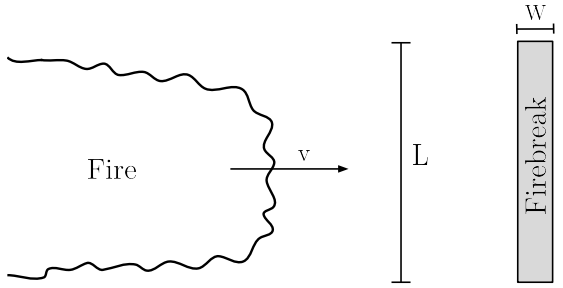


Figure 2.1 – Illustration of the fire containment model proposed by Jewell (1963). A fire front of length L moves at a constant velocity v, while firefighters construct a firebreak of width W and length L ahead of the fire to stop its spread. The model assumes that the firebreak is completed just as the fire reaches it.

Operations Research (OR) is concerned with the application of analytical and computational methods to aid decision-making. Its roots date back to the 1940s, when mathematicians worked on military planning during World War II (INFORMS, 2025). In wildfire management, OR has been used to determine optimal locations for firefighting aircraft to minimize response times (Lee et al., 2013), plan prescribed burns to reduce wildfire risk (Rachmawati et al., 2015), and allocate firefighting resources to protect critical infrastructure (Merwe et al., 2015).

To illustrate the application of OR in wildfire management, we discuss an analytical model for fire containment proposed by Jewell (1963). Consider a fire front of length L feet moving at a constant velocity of v feet per hour.[1] The chosen suppression strategy involves constructing a firebreak of width W feet and length L, with the expectation that once the fire's head is stopped, the flanks can be controlled more easily. Figure 2.1 illustrates this scenario.

Suppose x firefighters are deployed, each capable of clearing α square feet per hour. If the firebreak is positioned such that it is completed just as the fire reaches it, ensuring optimal timing of the suppression effort, the time required to complete the firebreak is given by

$$T = \frac{LW}{\alpha x},$$

[1]Throughout this dissertation, we use feet as the standard unit for length. For reference, 1 foot is approximately 0.3048 meters.

during which the fire spreads and burns an area of

$$A = (Tv)L$$

square feet.[2]

The total cost of fire suppression includes multiple components, each reflecting different aspects of firefighting operations. Fixed costs (C_f) represent investments made prior to the fire event, such as prevention programs, detection systems, and general preparedness infrastructure. Suppression mobilization costs (C_s) depend on the number of deployed forces and cover transportation, wages, and logistical support. Hourly suppression costs (C_h) scale with the total firefighting effort and include expenses related to labor, equipment usage, and supplies. Finally, the cost of values burned (C_b) captures economic losses due to fire damage, such as the market value of destroyed timber and costs associated with land restoration. Given these factors, the total suppression cost is expressed as

$$C(x) = C_f + C_s x + C_h x T + C_b A.$$

Let us consider an example where the fire front length L is 500 feet and the spread velocity is 1000 feet per hour. The firebreak width W is 5 feet, and the firefighters scraping rate α is 220 square feet per hour.[3] Fixed costs C_f account for R\$$20000$, the mobilization cost C_s is R\$$100$ per firefighter, and the hourly rate C_h is R\$$10$ per firefighter. The cost of values burned C_b is R\$$0.02$ per square foot. Figure 2.2 shows the total suppression cost as a function of the number of firefighters x. In this example, deploying 10 firefighters costs R\$$1000$ in terms of mobilization costs and R\$$11364$ in terms of burned area, with a total cost of R\$$32477$. A crew with 34 firefighters is the most cost-effective, with a total cost of R\$$26856$.

The analysis highlighted the trade-off between suppression expenses and damage costs. Increasing the number of firefighters reduces the burned area, which lowers the cost of values lost, but at the same time, it raises suppression costs. The model, however, relies on simplifying assumptions, such as constant fire spread velocity, perfect timing of the suppression effort, and homogeneous fuel conditions, which may not hold in real wildfire scenarios. In this dissertation, we will study a more complex model that accounts for the dynamics of fire spread and suppression. Before turning to our proposed model,

[2] If the firebreak construction is not optimally timed, the burned area A would be larger.

[3] In the metric system, we have $L \approx 152$ m, $W \approx 1.5$ m, $v \approx 0.08$ m/s, and $\alpha \approx 0.006$ m^2/s.

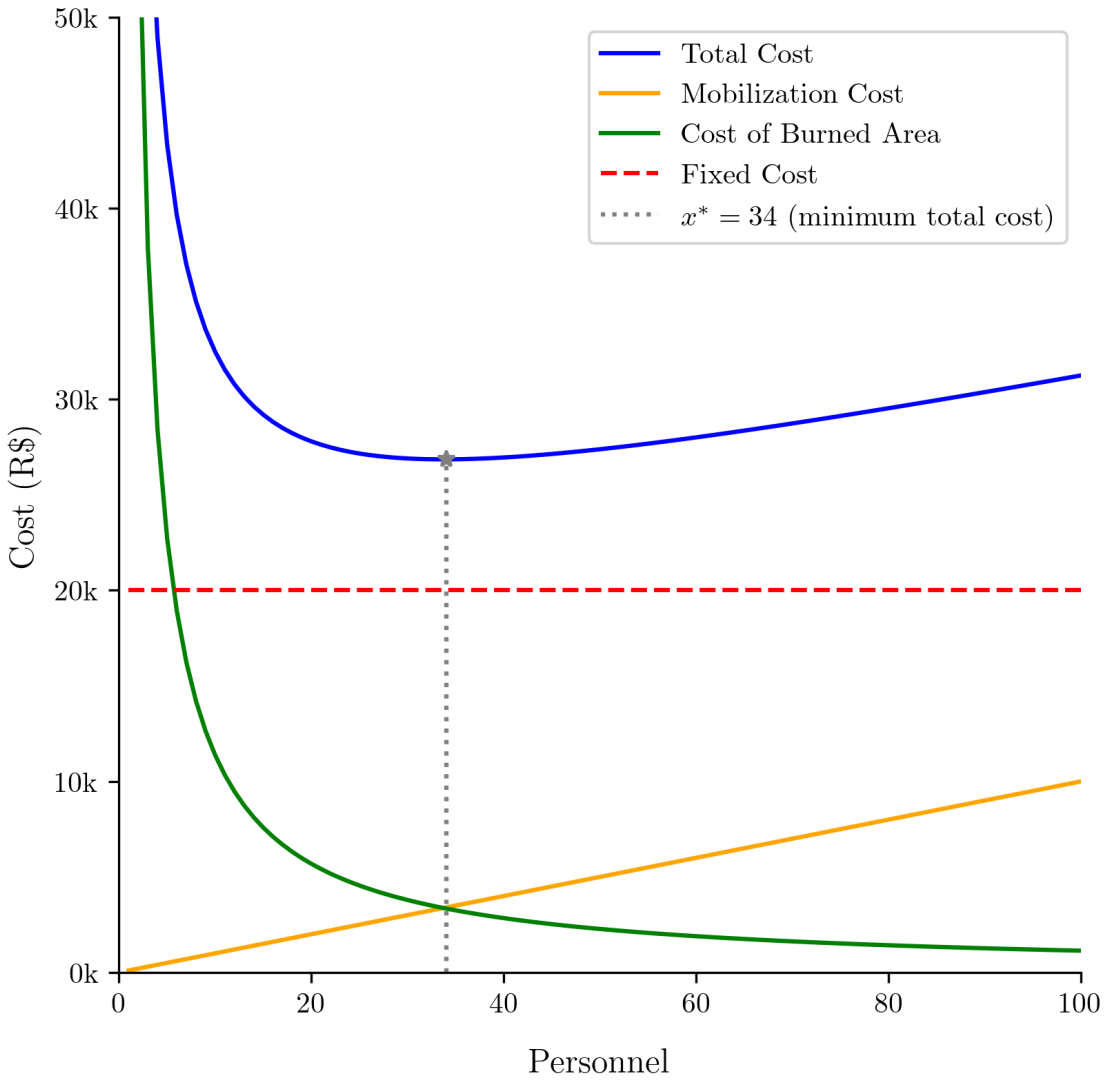


Figure 2.2 – Total suppression cost as a function of the number of firefighters x. The plot decomposes costs into fixed costs, mobilization costs, and the cost of burned area. Increasing x reduces the burned area cost but increases mobilization and hourly suppression costs. The most cost-effective crew size in this example is 34 firefighters, minimizing the total cost to R$26856.

we first review how fire behavior is typically modeled, which provides the foundation for the approach developed in this dissertation.

2.2 Understanding Wildfire Behavior

Researchers have studied fire behavior from different angles to better understand how fires spread and how they can be managed. According to Finney (1999), early studies focused on determining the shape of fires burning under relatively uniform conditions. Researchers observed that fire spread patterns ranged from nearly circular shapes in the absence of wind and slope to elongated ellipses under strong winds or steep terrain. By assuming a consistent fire shape and knowing the forward rate of spread, they could estimate how fire size and perimeter changed over time. As pointed out by Wagner (1969), for example, if the fire advances at a rate of v feet per minute at all points on its surface, as may be the case in a flat terrain with no wind and homogeneous fuel, then the burned area A after t minutes is $A = \pi(vt)^2$ square feet.

Fire behavior has also been studied based on where and how fires burn. Some fires spread along the surface, consuming grass, leaves, and small branches, while others climb into the canopy, becoming *crown fires* that are harder to control. The transition between these types of fires depends on wind, terrain, and fuel availability, and researchers have worked to understand when and why these shifts happen (Wagner, 1977). Knowing the likelihood of a crown fire occurrence helps in defining suppression strategies, such as whether aerial resources should be deployed, if firebreaks can be effective, or if evacuations are necessary (Andrews et al., 2011).

Besides fire shape and mode of spread, researchers have also studied numerical indicators to predict fire behavior in more detail. *Fireline intensity* measures how much energy a fire releases per length of the fire perimeter and per time, which is useful for fire managers to assess suppression difficulty and firefighter safety. For example, intense fires are more likely to break control lines, require more resources to contain, and pose greater risks to personnel. *Flame length*, closely related to fireline intensity, provides a measure of fire severity and is commonly used as a guideline for determining which suppression tactics are viable. Fires with shorter flames can often be suppressed with hand tools, while those with taller flames may require mechanized equipment or aerial resources (Andrews et al., 2011). These fire behavior indicators are incorporated into popular fire simulators such as FARSITE (Finney, 1998).

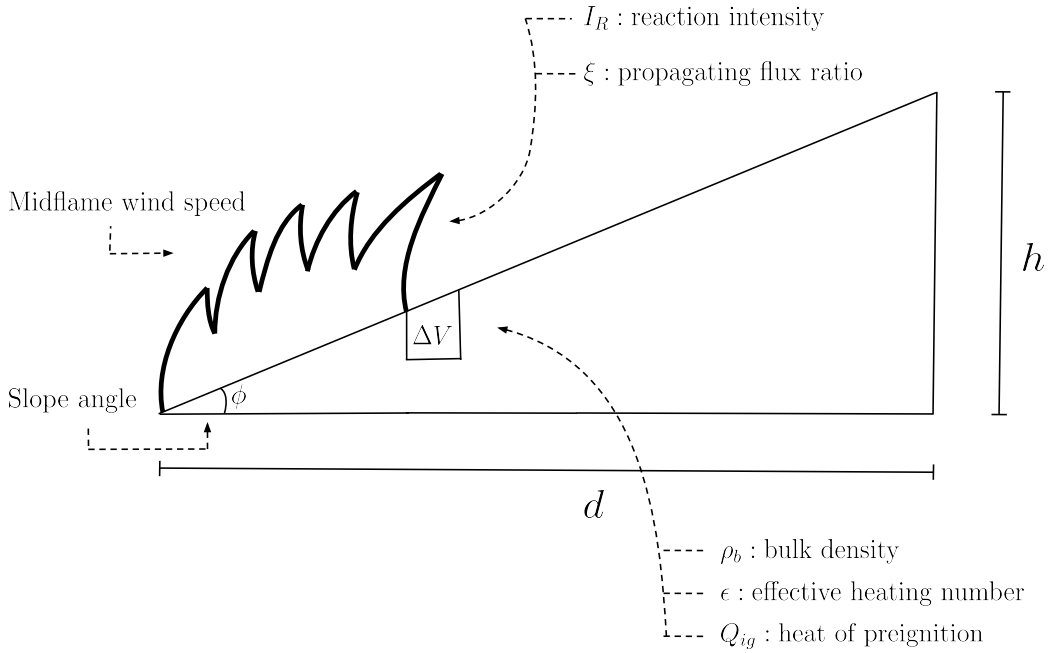


Figure 2.3 – Abstract setting behind Rothermel's model. As the fire front advances, the unit volume ΔV is heated until it ignites.

One of the most fundamental aspects of fire behavior is the rate of spread (ROS), which tells us how fast the fire is advancing. The rate of spread is influenced by multiple factors, including fuel properties, wind, and terrain slope. The next section introduces Rothermel's model of surface fire spread, a widely used tool to estimate ROS.

2.3 Rothermel's Model for Surface Fire Spread

In 1971, Frandsen (1971) proposed a theoretical model for fire spread based on the principle that fire propagates as a series of ignitions, i.e., unburned fuel is heated by the approaching fire front until it reaches ignition temperature. Figure 2.3 illustrates this principle, where we have a unit volume of fuel, denoted ΔV, and an approaching fire.[4] As the fire front advances, it heats the fuel volume ΔV until it ignites. Using the principle of conservation of energy, Frandsen (1971) derived the rate of spread of a fire as a ratio between the energy released by the fire as it gets closer to ΔV and the energy required to ignite ΔV.

Frandsen's model contains terms that are difficult to derive from first principles, which hinders its applicability in real settings. In 1972, Rothermel (1972) decomposed

[4] A unit volume of fuel is a small representative volume of the fuel bed that retains the same properties as the surrounding fuel.

Frandsen's equation into a few terms whose values could be estimated through experimentation in a laboratory using artificial fuel beds and wind tunnels. Rothermel's model is one of the most widely used tools in the prevention and combat of wildfires.

Rothermel's model defines the rate of spread R of a fire, in feet per minute, as

$$R = \frac{I_R \xi}{\rho_b \epsilon Q_{ig}}(1 + \Phi_w + \Phi_s). \tag{2.1}$$

Equation 2.1 has seven terms, which are influenced by the characteristics of the fuel particles, of the fuel bed, and of the environment.[5] The equation can be interpreted as describing the interaction between two main components, the no-wind, no-slope rate of spread $R_0 = \frac{I_R \xi}{\rho_b \epsilon Q_{ig}}$ (ft/min) and the slope-and-wind correction factor $(1 + \Phi_w + \Phi_s)$, giving us the following reformulation

$$R = R_0(1 + \Phi_w + \Phi_s). \tag{2.2}$$

Rothermel's model assumes fire spreads in the direction of both wind and slope, a situation known as *upslope headfire*. More generally, if the fire spreads in the same direction as the wind, it is called a *headfire*, otherwise, it is a *backfire*. Similarly, if it moves uphill, it is *upslope*, otherwise, it is *downslope*. There are four possible scenarios: upslope headfire, downslope headfire, upslope backfire, and downslope backfire.

To account for these effects, several extensions of Rothermel's model have been proposed. In this work, we adopt the formulation introduced by Albini and described by Weise & Biging (1997). Among the alternatives compared in their study, Albini's extension showed the best agreement with observed data. The rate of spread R according to Albini's model is given by

$$R = \begin{cases} R_0(1 + \Phi_w + \Phi_s) & \text{Upslope headfire;} \\ R_0(1 + \max\{0, \Phi_w - \Phi_s\}) & \text{Downslope headfire;} \\ R_0(1 + \max\{0, \Phi_s - \Phi_w\}) & \text{Upslope backfire;} \\ R_0 & \text{Downslope backfire.} \end{cases}$$

In the next sections, we explain each component of Equation 2.2 in detail.

[5] A fuel particle is discrete unit of combustible material, such as a leaf of grass or a piece of wood. A fuel bed is a collection of fuel particles.

(a) A dry grassland with fine, loosely packed grasses. The continuous fuel bed allows flames to efficiently transfer heat forward, resulting in a potentially high propagating flux ratio. The dominance of fine fuels (high surface-area-to-volume ratio) makes the impact of wind on fire spread particularly strong. Fine fuels are also associated with a high effective heating number.

(b) A landscape covered with fallen branches and logs. The heavy fuel load burns intensely, sustaining high heat output over time. As a result, this type of vegetation supports high reaction intensity. Due to the dominance of thick fuels (low surface-area-to-volume ratio), the effective heating number tends to be low.

(c) A sparse cover of shrubs creates a discontinuous fuel bed, potentially leading to a lower propagating flux ratio, as some energy is lost rather than driving forward combustion. Additionally, the low bulk density of the fuel further reduces reaction intensity, as less fuel is available per unit volume.

(d) A forest floor covered in compact litter. The dense, tightly packed fuel bed limits airflow, reducing the reaction intensity. The high bulk density of the fuel means there is more fuel available per unit volume.

Figure 2.4 – Examples of different fuel types and their relation with Rothermel's model. Sources: Price et al. (2024) and Scott & Burgan (2005).

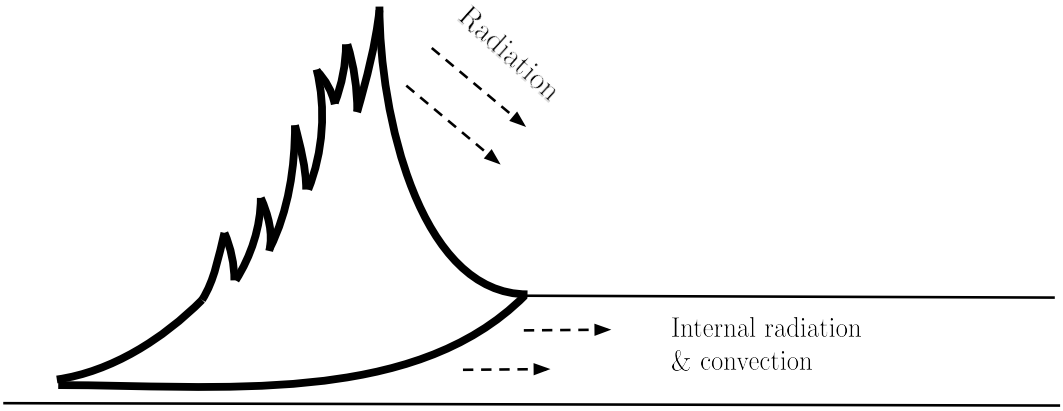


Figure 2.5 – Heat transfer in fire spread on flat terrain without wind or slope. Fire moves forward as heat from the flames warms and ignites unburned fuel. Some heat is transferred ahead through radiation and convection, while some is lost to the surroundings. Adapted from Rothermel (1972).

No-Wind, No-Slope Rate of Spread

As the name suggests, the no-wind, no-slope rate of spread R_0 describes how quickly fire propagates through a flat terrain in the absence of wind. It is defined as

$$R_0 = \frac{I_R \xi}{\rho_b \epsilon Q_{ig}}$$

where each parameter is defined as follows.

The **reaction intensity** I_R (Btu/ft^2/min) represents the rate at which energy is released per unit area of the fire front.[6] This depends on the amount of fuel available to burn and the physical and chemical properties of the fuel particles. For instance, a dense layer of fallen branches and logs, such as in Figure 2.4b, burns with high intensity, releasing a large amount of energy in a short period, resulting in a high I_R. In contrast, a cover of dry grass burns more gradually, producing a lower I_R, as illustrated by Figure 2.4a.

The **propagating flux ratio** ξ is a dimensionless coefficient that quantifies how much of the reaction intensity actually contributes to igniting the unburned fuel ahead. Not all the energy released by combustion is useful for fire spread, since some is lost to the surroundings. A fire in a continuous fuel bed, such as Figure 2.4a, may have a higher ξ than one burning over a sparse cover of shrubs, such as in Figure 2.4c, because more of the heat is effectively transferred forward.

The reaction intensity I_R and the propagating flux ratio ξ determine the energy

[6] We use Btu as the standard unit of energy. For reference, 1 Btu is about 1055 Joules, which is roughly the kinetic energy of a 30-kg object moving at 8.4m/s.

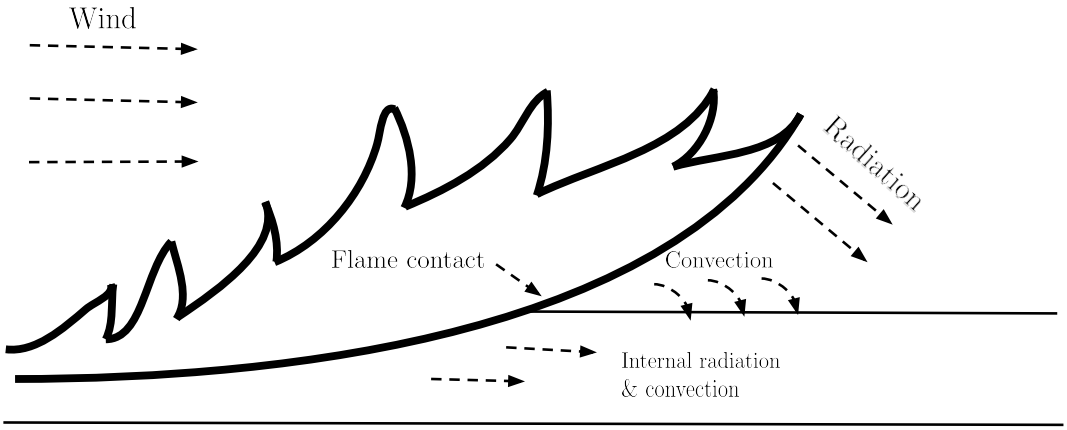


Figure 2.6 – Fire spread under the influence of wind. Wind pushes flames forward, increasing heat transfer to unburned fuel through radiation, convection, and direct flame contact. This accelerates ignition and increases the fire spread rate. Adapted from Rothermel (1972).

available for combustion, while the remaining terms of Equation 2.1 control how much of this energy is necessary to ignite the fuel ahead of the fire. We will take a closer look at these terms in the following paragraphs.

The **bulk density** ρ_b (lb/ft^3) represents the amount of fuel mass per unit volume of the fuel bed.[7] A compact layer of dry leafs, with many small fuel particles closely packed together (Figure 2.4d), has a higher ρ_b than a bed of scattered shrubs (Figure 2.4c). A higher bulk density means more fuel is available to burn in a given area, but it can also slow down the spread of fire due to reduced airflow.

The **effective heating number** ϵ is a dimensionless value that represents the proportion of each fuel particle that reaches ignition temperature during combustion. In fine fuels like grass, most of the fuel particle is quickly heated, so ϵ is close to 1 (Figure 2.4a). In contrast, thick branches heat more slowly, leading to a lower ϵ (Figure 2.4b).

The **heat of preignition** Q_{ig} (Btu/lb) is the amount of energy required to bring a unit mass of fuel to ignition, and depends on the moisture content of the fuel particles. Dry fuels, such as dead grass or wood with low moisture content, require less energy to ignite and thus have a lower Q_{ig}, while wet logs need significantly more energy before they start burning.

Wind Factor

The wind factor Φ_w is a dimensionless quantity that quantifies how wind enhances the spread rate of fire. As wind speed increases, flames are tilted forward into the unburned fuel, preheating and igniting it more quickly, as illustrated by Figure 2.6. This effect is captured by the equation

$$\Phi_w = 7.47 \cdot e^{-0.133 \cdot \sigma^{0.55}} U^{0.02526 \cdot \sigma^{0.54}} \left(\frac{\beta}{\beta_{op}} \right)^{-0.715 \cdot e^{(-3.59 \cdot 10^{-4}) \sigma}}$$

where U is the mid-flame wind speed (in ft/min), which represents the wind velocity at the level of the flames, and β, β_{op}, and σ are constants that depend on the fuel type.

It is important to note that the midflame wind speed U required by Rothermel's model is not the same as the speed obtained by standard wind measurements. Typically, wind speed is recorded at a height of 20 feet (6.1 meters) in open terrain, known as the 20-foot wind speed. However, wind velocity decreases significantly closer to the ground due to the presence of vegetation. Therefore, to obtain the midflame wind speed experienced by the fire front, the 20-foot wind speed must be adjusted downward. This is commonly achieved using a wind adjustment factor (WAF), which represents the ratio of midflame wind speed to the 20-foot wind speed (Andrews, 2012). The appropriate WAF depends on the fuel type and the surrounding vegetation, but common values lie between 0.1 and 0.5 (Andrews, 2012).

The term β represents the **packing ratio**, a dimensionless measure of how densely fuel particles are arranged within the fuel bed. It quantifies the trade-off between fuel availability and airflow in a fuel bed, where higher values indicate more fuel per unit volume but reduced oxygen penetration, and lower values indicate greater airflow but less fuel to sustain combustion. A low packing ratio can be found in scattered shrubs or dry grass (Figures 2.4a and 2.4c), while a high packing ratio can be found in compact litter (Figure 2.4d). The **optimum packing ratio** β_{op} is the value of β that maximizes the energy release for combustion. The ratio $\beta_{rel} = \beta / \beta_{op}$, known as the **relative packing ratio**, determines how efficiently a given fuel bed sustains fire spread. Finally, the **surface-area-to-volume ratio** σ (ft^2/ft^3) characterizes the fineness of the fuel particles. Fine fuels such as dry grass have high values of σ, leading to rapid ignition and combustion (Figure 2.4a),

[7] We use the pound as the standard unit of mass. For reference, 1 pound is approximately 0.453592 kilograms.

Table 2.1 – Wind factor (Φ_w) for different values of wind speed (U), relative packing ratio (β_{rel}), and surface-area-to-volume ratio (σ).

U	σ	β_{rel}	$C_w(\sigma, \beta_{rel})$	$B_w(\sigma)$	Φ_w
500	1000	1	0.019644	1.05	13.7
500	1000	5	0.008794	1.05	6.1
500	2000	1	0.001247	1.53	16.9
500	2000	5	0.000711	1.53	9.6
1000	1000	1	0.019644	1.05	28.3
1000	1000	5	0.008794	1.05	12.7
1000	2000	1	0.001247	1.53	48.9
1000	2000	5	0.000711	1.53	27.9

whereas logs and thick branches have much lower values (Figure 2.4b).

To make the dependence of Φ_w on wind speed more explicit, we define

$$C_w(\sigma, \beta_{rel}) = 7.47 \cdot e^{-0.133 \cdot \sigma^{0.55}} \beta_{rel}^{-0.715 \cdot e^{(-3.59 \cdot 10^{-4})\sigma}}$$
$$B_w(\sigma) = 0.02526 \cdot \sigma^{0.54}$$

Using these definitions, the wind factor can be rewritten as function of the wind speed U, given the value of σ and β_{rel}, as

$$\Phi_w(U; \sigma, \beta_{rel}) = C_w(\sigma, \beta_{rel}) U^{B_w(\sigma)}$$

It is known that wind speed is a key factor in wildfire behavior, and this effect is captured by the definition of Φ_w. Table 2.1 shows the wind factor for different values of wind speed, relative packing ratio, and surface-area-to-volume ratio. As we can see, in extreme cases, the wind factor can increase the spread rate by a factor of 40 or more.

As pointed out by Andrews (2018), some fire management systems impose a limit on the wind factor to prevent unrealistic spread rates. This typically occurs in fuel beds with low reaction intensity, such as in Figure 2.4a, where the wind factor can be particularly strong due to the high surface-area-to-volume ratio, but the fire spread rate is limited by the amount of energy available for combustion. There is no consensus, however, on what this limit should be or when it should be applied. In our instance model described in Section 4.1, we will not impose any limits on the value of Φ_w.

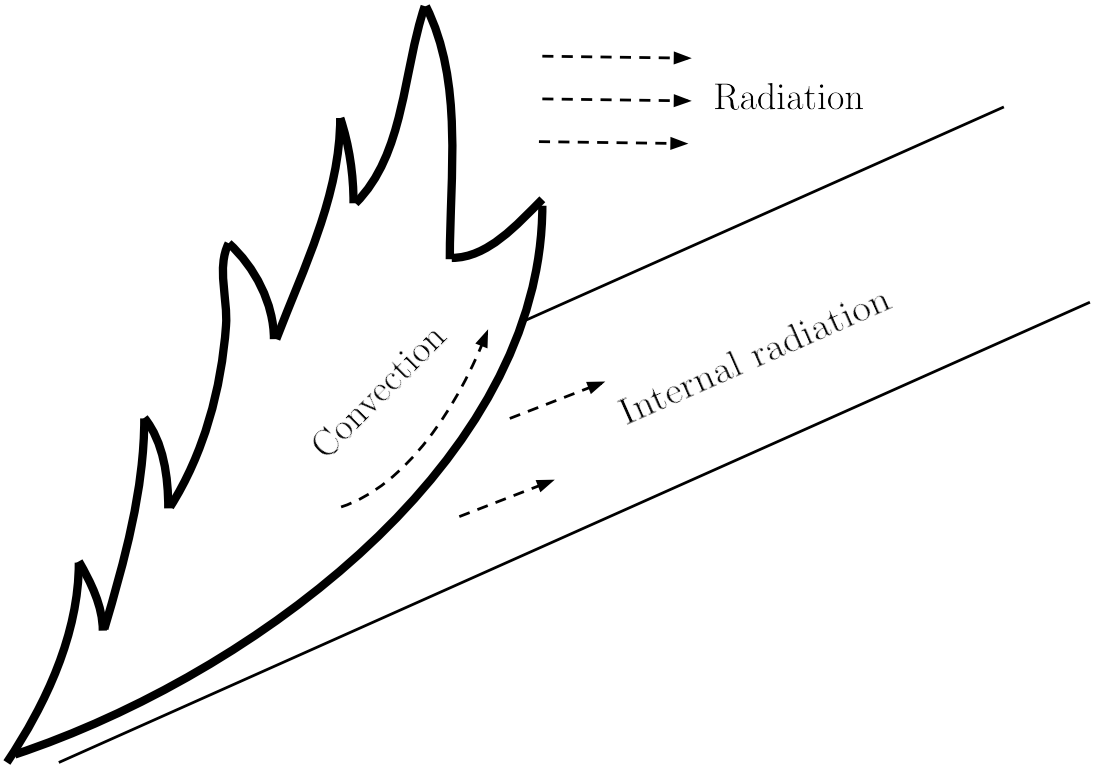


Figure 2.7 – Fire spread under the influence of slope. As the terrain steepens, flames tilt closer to the unburned fuel, increasing heat transfer through radiation and convection. This speeds up ignition and accelerates fire spread. Adapted from Rothermel (1972).

Slope Factor

The slope factor Φ_s is a dimensionless quantity that accounts for the influence of terrain steepness on fire spread. As illustrated by Figure 2.7, when fire burns on an inclined surface, the flames tilt, bringing them closer to the unburned fuel, which increases heat transfer and accelerates the fire spread rate. This effect is captured by the equation

$$\Phi_s = 5.275 \cdot \beta^{-0.3} \cdot \tan^2(\phi)$$

where ϕ is the slope angle, $\tan(\phi)$ represents the steepness of the terrain, and β is the packing ratio, explained in the last section. In a fuel bed with a packing ratio of $\beta = 0.005$, a fire on a 5° slope ($\tan\phi \approx 0.087$) spreads only slightly faster than on flat ground ($\Phi_s \approx 0.198$), while on a 30° slope ($\tan\phi \approx 0.577$), the spread rate can be significantly enhanced ($\Phi_s \approx 8.618$).

To make the dependence of Φ_s on the slope angle more explicit, we define

$$C_s(\beta) = 5.275 \cdot \beta^{-0.3}$$

which allows us to express the slope factor as a function of the tangent of the slope angle $A = \tan(\phi)$, given a value for β, as

$$\Phi_s(A; \beta) = C_s(\beta)A^2.$$

This explicit dependency on A will be useful in Section 4.1 where we introduce new instances for the problem defined in Section 2.5.1.1. It also helps illustrate why fires in mountainous regions are particularly challenging to control, as even small variations in terrain can lead to large changes in fire behavior.

2.4 Estimating Fuel Parameters

Rothermel's model requires several input parameters to estimate fire spread, such as reaction intensity, bulk density, and propagating flux ratio. These parameters can be measured through experiments, but collecting data for every possible fire scenario is time-consuming and impractical. Instead, fire scientists classify common types of vegetation and determine a set of representative parameter values for each. This predefined set of parameters is called a *fuel model*. Andrews (2018) compiled a list of 53 fuel models that are widely used in fire behavior prediction. Figure 2.8 shows an example of fuel model GR1, which represents short, sparse grass with low fuel load. In Section 4.1, we will use the data provide by Andrews (2018) to estimate reasonable values for R_0, β, β_{rel}, and σ.

The model described in Section 2.3 is the basic version of Rothermel's model, which assumes that the fuel bed is composed of homogeneous fuel particles with uniform physical and chemical properties. In reality, fuel beds are often heterogeneous, consisting of a mixture of particles with different characteristics. For example, a grassland may contain both fine grasses and shrubs, each contributing differently to fire behavior. It is also important to distinguish between dead and live fuels, as dead fuels generally have lower moisture content and different combustion properties than live fuels.

To account for heterogeneous fuel beds, Rothermel (1972) organized fuel particles into classes based on their size and whether they were dead or live fuels. He then introduced a set of weighting factors that adjust the values of input parameters according to the composition of the fuel bed. The effective surface-area-to-volume ratio of a given fuel model, such as GR1, would be a weighted average of the surface-area-to-volume ratios of each class of fuel particles. We refer the reader to Andrews (2018) for an explanation of

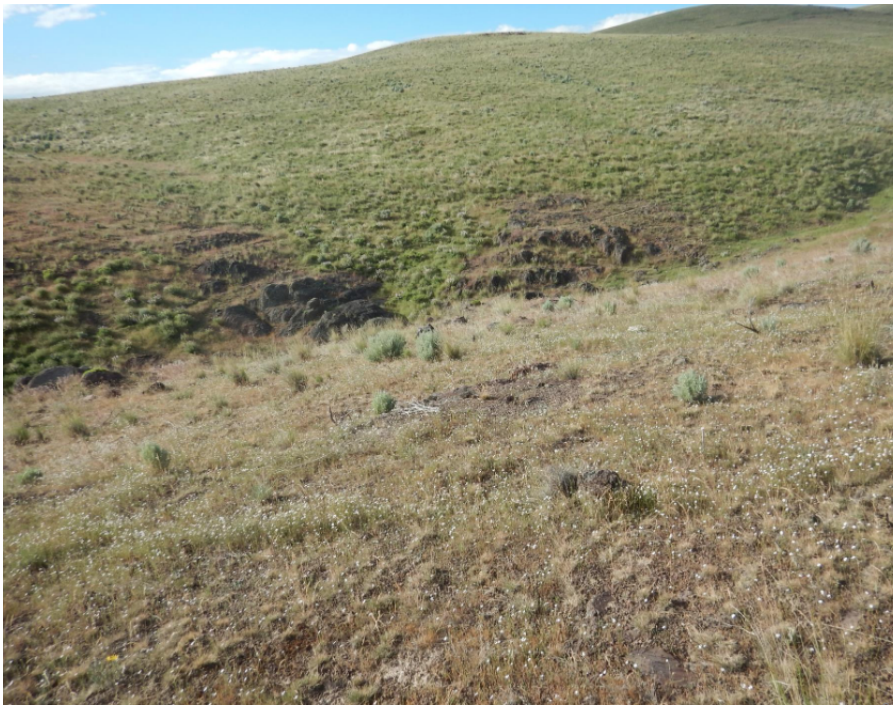


Figure 2.8 – Fuel model GR1, representing short, sparse, dry-climate grass. According to Andrews (2018), GR1 has a surface-area-to-volume ratio $\sigma \approx 2200$ ft^2/ft^3, bulk density $\rho_b \approx 0.02$ lb/ft^3, packing ratio $\beta \approx 0.0008$, and relative packing ratio $\beta_{rel} \approx 0.22$. The values were computed considering 1-hour fuels. Source: Price et al. (2024).

how these weighting factors are computed.

In the data compiled by Andrews (2018), dead fuels are divided into three time-lag classes: 1-hour, 10-hour, and 100-hour fuels. These classes reflect how quickly a fuel particle gains or loses moisture in response to environmental changes, and are a good proxy for the size class of a particle. To give some examples, 1-hour fuels include fine materials like grass, 10-hour fuels include twigs and small branches, and 100-hour fuels correspond to larger fuels such as logs and thick branches. Live fuels, on the other hand, are divided into herbaceous fuels (e.g., grasses and shrubs) and woody fuels (e.g., trees and bushes).

To give an example of the weighting factors in practice, let us consider fuel model GR1 of Figure 2.8. Andrews (2018) provides two values for the surface-area-to-volume ratio of GR1: $\sigma_{\text{dead}} = 2200$ for 1-hour dead fuels and $\sigma_{\text{live}} = 2000$ for live herbaceous fuels. The effective surface-area-to-volume ratio of the fuel bed, after correction using weighting factors, would be approximately 2054.

Since fine and dead fuels play the most significant role in determining fire spread rates, in this work, we only consider dead 1-hour fuels to estimate fire behavior parameters.

2.5 Graph-based Models for Wildfire Suppression

Finney (2002) demonstrated that fire growth over a landscape can be modeled as a search for the fastest path through a graph that represents the landscape. In this approach, the terrain is rasterized into a grid, where each cell corresponds to a spatial unit containing fire behavior parameters such as fuel type, wind influence, and slope. This grid is encoded as a graph where cells become vertices and arcs connect vertices representing adjacent cells. Arcs are labeled with the fire travel time from the center of a cell to the center of an adjacent cell, which can be estimated using a fire spread model such as Rothermel's. The fire arrival time at any given vertex is computed as the shortest travel time from the ignition point. When fire spread velocities are well estimated, this approach provides a computationally efficient way to approximate fire dynamics.

Graph-based models for wildfire suppression use this fire spread representation by incorporating suppression actions into the optimization problem. A set of firefighting resources is available over a given time horizon and can be allocated to vertices in the graph. As the fire spreads, resources are deployed to specific locations, where they act by increasing the travel time on outgoing arcs, slowing fire propagation locally. The objective is to determine a suppression strategy that optimizes a predefined criterion, such as minimizing the total area burned. In the next sections, we formally define this problem (Subsection 2.5.1), review existing models that follow this approach (2.5.2), and describe existing solution methods (Subsection 2.5.3).

2.5.1 Notation & Definitions

2.5.1.1 Problem Definition

We are given a directed graph $G = (V, A)$ with travel times t_{uv} on arcs $uv \in A$, which model the time required for fire to propagate from a vertex to a neighboring vertex. A directed graph permits modeling different fire travel times in opposite directions, which can occur due to factors like wind and terrain slope. Given an ignition vertex $s \in V$, the travel times define a shortest-path tree rooted at s in which each vertex $v \in V$ has an associated fire arrival time a_v and a predecessor p_v. Now assume we have k fire suppression resources which can be allocated to vertices $v \in V$, and each resource adds a delay Δ to the outgoing arcs of v. Each resource $i \in [k]$ is released at time t_i and can only be allo-

cated to a vertex v if $a_v \geq t_i$, i.e., if v is not burned yet.[8] We also assume that each vertex can receive at most one resource. Finally, we have a time horizon H and are interested in minimizing the vertices that burn before H.

The allocation of resources to vertices can be represented by an injective function $\Lambda : [k] \to V$. By definition, such an allocation changes the travel times on the arcs, but it can also change the topology and the arrival times of the shortest-path tree. As a result, given an allocation of resources Λ we denote the resulting fire propagation times by t^Λ, the fire arrival times by a^Λ, and the predecessor relation by p^Λ. The problem, then, is to find a feasible allocation of resources Λ that minimizes the number of burned vertices at time H, i.e.,

$$\sum_{v \in V} [a_v^\Lambda < H].$$

Figure 2.9 shows an example of an instance for our problem. The graph has nine vertices and the ignition vertex is s. We have three resources available at times 2, 3, and 4, i.e., $k = 3$, $t_1 = 2$, $t_2 = 3$, and $t_3 = 4$. The graph on the right-hand side depicts a possible allocation Λ, where vertices v_2, v_4, and v_6 receive the available resources, with $\Lambda_1 = v_2$, $\Lambda_2 = v_4$, and $\Lambda_3 = v_6$. Notice that vertices v_1 and v_3 cannot receive a resource since they burn at time 1 and the first resource is released at time 2. Due to the allocation, a delay Δ is added to the outgoing arcs of v_2, v_4, and v_6, which will increase the fire arrival time to vertices v_5, v_7, and v_8. Since the optimization horizon H is 5 and the delay Δ is 2, the allocation has an objective value of 6 because vertices s, v_1, v_2, v_3, v_4, and v_6 burn before time 5.

2.5.1.2 Notation

Given a directed graph $G = (V, A)$ and any vertex $v \in V$, we denote by N_v^+ and N_v^- the set of vertices that can be reached through outgoing arcs and incoming arcs of v, respectively. If G is a grid graph, the *extended neighborhood* $\mathbb{N}_v$ of v includes vertices reachable via outgoing arcs as well as diagonal connections from v. In Figure 2.9, we have $N_{v_4}^+ = \{v_5, v_7\}$, $N_{v_4}^- = \{v_1, v_3\}$ and $\mathbb{N}_{v_4} = \{v_5, v_7, v_8\}$.

Time instants where resources become available are represented by a sequence of times $T = (t_1, t_2, \dots)$, in ascending order.[9] We denote by $\alpha(t) = \min_{i>0|t_i>t} t_i$ the first time after t when new resources become available, with $\alpha(t) = H$ if no further resources

[8] We use $[n]$ to denote a set containing the first n natural numbers, i.e., $[n] = \{1, \dots, n\}$.

[9] Notice that here t_i denotes the ith time instant where one or more resources are released, and not the release time of resource $i \in [k]$. The context will always make clear the meaning of t_i.

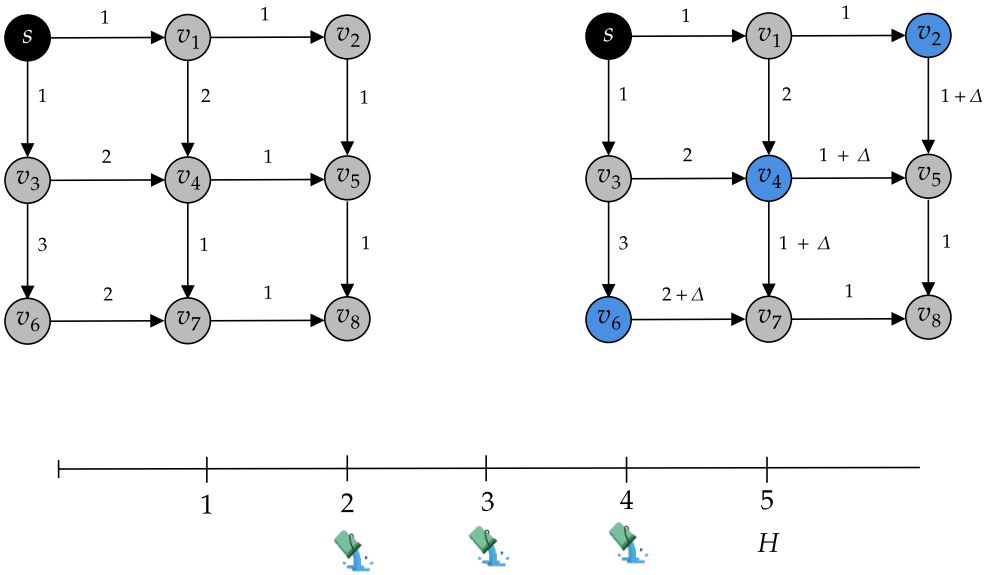


Figure 2.9 – Example of an instance for our problem. The ignition vertex is s and there are $k = 3$ resources available. The resources are released at times 2, 3, and 4, i.e., $t_1 = 2$, $t_2 = 3$, and $t_3 = 4$. The delay Δ introduced by a resource is 2 and the optimization horizon H is 5. On the right-hand side, the optimal allocation Λ is depicted, with $\Lambda_1 = v_2$, $\Lambda_2 = v_4$, and $\Lambda_3 = v_6$.

become available after t.[10] Finally, for each time $t \in [0, H]$, $R_t \subseteq [k]$ is the set of resources that are released at time t. In Figure 2.9, we have that $T = (2, 3, 4)$, $\alpha(1) = \alpha(1.5) = 2$, $\alpha(2) = 3$, and $\alpha(4) = H$. We also have $R_1 = \emptyset$, $R_2 = \{1\}$, $R_3 = \{2\}$, and $R_4 = \{3\}$.

When an allocation of resources Λ assigns a resource to a vertex $v \in V$, we say that v is *protected*. We denote by $P^\Lambda \subseteq V$ the set of vertices protected by Λ. In Figure 2.9, $P^\Lambda = \{v_2, v_4, v_6\}$. If $|P^\Lambda| = k$ we say that Λ is a *complete allocation*. Conversely, if $|P^\Lambda| < k$ we say that allocation Λ is *partial*. The special allocation that does not protect any vertex is denoted by Λ_0, i.e., $P^{\Lambda_0} = \emptyset$. Given an allocation Λ, a vertex $v \in V$, and a time $t \in T$, we will use the notation $(t, v) \in \Lambda$ to indicate that there exists a resource $i \in [k]$ allocated to vertex $v \in V$ whose release time is $t \in T$, i.e., $\Lambda_i = v$ and $t_i = t$. In the example of Figure 2.9, we have $(2, v_2) \in \Lambda$, $(3, v_4) \in \Lambda$, and $(4, v_6) \in \Lambda$. It also holds that $(2, v_1) \notin \Lambda$ (because $v_1 \notin P^\Lambda$) and $(2, v_6) \notin \Lambda$ (because v_6 is protected by the resource released at time 4).

Each allocation of resources Λ induces a new set of travel times t^Λ_{uv} on the arcs of G and, consequently, a new shortest-path tree rooted at s. For any vertex $v \in V$, we denote by $\mathcal{P}^\Lambda_v$ the set of predecessors of v in the shortest-path tree induced by Λ. In Figure 2.9, we have that $\mathcal{P}^{\Lambda_0}_{v_8} = \{s, v_1, v_2, v_5\}$ and $\mathcal{P}^{\Lambda_0}_{v_3} = \{s\}$. The ignition vertex s has

[10] Here we are assuming that all resource release times are smaller than H. This is as fair assumption, since a resource i with release time $t_i \geq H$ cannot save any vertices, and therefore it can be pre-processed out of the instance.

no predecessors, so $\mathcal{P}_s^\Lambda = \emptyset$ for any allocation Λ.

Finally, given an allocation Λ and a time $t \in [0, H]$, we define $B_t^\Lambda = \{v \in V \mid a_v^\Lambda < t\}$ as the set of vertices that are burned at t. In Figure 2.9, we have that $B_2^\Lambda = \{s, v_1, v_3\}$ and $B_5^\Lambda = \{s, v_1, v_2, v_3, v_4, v_6\}$. Note that our goal is to find an allocation Λ such that $|B_H^\Lambda|$ is minimized.

2.5.2 Related Work

In this section, we review the most relevant works in the literature that integrate fire spread with optimization. We focus on the works that use a graph-based representation of the fire spread in a deterministic setting, as this is the approach we follow in this dissertation. We also discuss selected works based on alternative representations when their modeling structure or objectives are closely related to ours.

To the best of our knowledge, the first work integrating spatial fire spread with optimization can be traced back to the linear programming model of Hof et al. (2000). They consider a grid of cells, where each cell has an amount of fuel available to burn, and the more fuel available in a cell, the faster the fire propagates through it. Suppression decisions determine how much fuel is removed from each cell, subject to a given suppression budget. Given an ignition cell, the objective is to find a suppression strategy that maximizes the fire arrival time to a target cell, which represents an area that should be protected from the fire.

Besides its pioneering nature, this model has several simplifying assumptions. First, it assumes that resources are available from the beginning of the fire event, so resource release times are not considered. Second, the fire propagation logic assigns two decision variables to each cell to represent the entry and exit times of the fire. Their difference is governed by a function of the fuel load, which is affected by the suppression strategy. As we show in Section B.1, this implies uniform fire travel times for all outgoing arcs of a cell, regardless of direction, preventing the model from capturing the effects of wind and slope.

Wei et al. (2011) build upon the work of Hof et al. (2000) by better modeling fire propagation. The fire travel times of the outgoing arcs are now dependent on the direction of the arc, which allows for modeling fire spread influenced by wind and slope. Similar to our problem, their MIP formulation considers a uniform delay Δ applied to the outgoing arcs of a vertex when a resource was allocated to the associated cell. The authors

also consider a time horizon H and are interested in the vertices that burn within this time horizon. However, differently from the problem we consider, they associate to each vertex $v \in V$ a value w_v, which is lost if vertex v burns, and the objective is minimizing the total value lost by burned vertices.

Wei et al. (2011) are the first to consider some form of safety in their model, by prohibiting the allocation of resources to cells with predicted flame lengths exceeding a predefined threshold. Nevertheless, they also assume that resources are available from the beginning of the fire event, similar to Hof et al. (2000). The authors acknowledge this limitation and try to overcome it by proposing heuristic methods to distribute the suppression effort across multiple time periods. In Section B.1 of the appendix, we show the model of Wei et al. (2011) using our notation.

Belval et al. (2015) contribute by integrating detailed fire behavior characteristics into the optimization. Their primary contribution is the explicit calculation of fireline intensity, in addition to fire arrival time, as dynamic responses to suppression decisions within the MIP model. Modeling fireline intensity allows them to distinguish between low-intensity fires, which are often ecologically beneficial, and high-intensity fires, which can be harmful. This distinction is used to formulate more complex objectives that focus only on suppressing high-intensity harmful fires, rather than simply minimizing the area burned.

The work of Belval et al. (2015) also presents significant differences from previous works. First, they consider that suppression actions completely block fire propagation, instead of just delaying it, as in Hof et al. (2000) and Wei et al. (2011). They also do not impose a limit on the number of resources available, but instead associate a cost to each resource, which is added to the objective function when the resource is used. Furthermore, their model does not have a time horizon, but instead the process evolves based on fire dynamics and suppression actions until all vertices are either burned or cannot be reached by the fire due to the suppression decisions. Similar to Hof et al. (2000) and Wei et al. (2011), the authors do not impose release times on resources.

All works to this point operate under the assumption that resources are available from the beginning of the fire event. Alvelos (2018) is the first, within this line of graph-based suppression models, to explicitly incorporate resource release times into the optimization model. However, this was one component of a broader set of contributions, which we will summarize next.

Alvelos (2018) proposed using linear programming duality principles applied to

the shortest path problem to establish a set of MIP constraints for fire spread. This approach transforms the calculation of minimum fire arrival times into a feasibility problem, ensuring that the resulting arrival times are accurate across the landscape regardless of the model's objective function. Building on this formulation for fire spread, and incorporating the resource release times, the author presents MIP models for four distinct wildfire suppression problems: (i) protecting specific areas (similar to Hof et al. (2000)), (ii) minimizing the total burned area by a given time (similar to Wei et al. (2011)), (iii) containing the fire based on eliminating new ignitions, and (iv) containing the fire based on covering the fire perimeter with resources. In Section 2.5.3.1, we will present the MIP model of Alvelos (2018) for the problem of minimizing the total burned area by a given time, as this formulation models the problem considered in this dissertation.

Besides the introduction of release times by Alvelos (2018), the suppression resources were still largely treated as homogeneous units. Avci et al. (2024) are the first to model distinct types of real-world suppression resources, including ground firefighting teams, firebreak construction teams, and helicopters. The authors associate unique operational constraints with each resource type, drawing inspiration from practical limitations. For example, their model considers that ground teams may require road access for support vehicles and must maintain specific safety distances from the fire front. Similar to Belval et al. (2015), a fire suppression resource completely blocks fire propagation. Similar to Wei et al. (2011), each cell has an associated value and the objective is to minimize the total loss of value.

Avci et al. (2024) also employs a considerably different modeling technique compared to the previous shortest-path based approaches. Instead of calculating continuous arrival times, they use a discrete-time, state-based approach, where the landscape is represented by cells transitioning between states (e.g., flammable, burning, burned, razed, protected by resource) over fixed time periods. Fire propagation occurs based on transition rules dependent on neighbor states in the preceding period. This state-based approach can handle time-varying conditions, like changes in wind direction. However, it introduces challenges related to the choice of time step resolution (affecting accuracy and model size) and ensuring the discrete-time process accurately reflects continuous fire spread dynamics.

All the models to this point assume a resource can be instantly allocated to any feasible location without considering the travel path or time required to reach it. This simplification can lead to operationally infeasible plans. For example, a single ground

brigade might be assigned sequential suppression tasks at locations too distant to travel between within the required timeframe. Furthermore, these models generally do not account for whether the path to a target suppression location remains accessible and safe. A route might become too dangerous to traverse due to the fire's progression, and it is possible that some cells become unreachable in practice. Recent research has focused on models that consider the routing aspects of suppression resources, ensuring that planned movements are both temporally feasible and safe along the entire path.

Granda et al. (2025) tackle the challenge of resource movement by formulating the wildfire suppression problem as a dynamic routing problem for a single brigade. Their MIP model determines an optimal continuous path (a sequence of adjacent cells) for the brigade to traverse, starting from an initial location. At cells visited along this path, the model decides whether the brigade should stop and perform suppression actions. The authors consider detailed timing constraints for the brigade, that account for both the travel time between cells and the duration required to perform suppression work at a specific cell. Furthermore, they incorporate temporal safety constraints ensuring the brigade completes its actions before the fire arrives to the cell. In their model, suppression resources completely block fire propagation. Similar to Belval et al. (2015), the model does not operate within a fixed optimization horizon, but instead the process evolves based on fire dynamics and suppression actions until fire containment is achieved. The authors consider three objectives: minimizing the loss of value due to burned cells, minimizing the cost of resources used, and minimizing the total operational time of the brigade.

Alvelos et al. (2025) follow a similar line of work, proposing a model for dispatching, positioning, and routing multiple, heterogeneous resources like ground crews and helicopters. Unlike Granda et al. (2025), this model determines the optimal route for each terrestrial unit from its base to a single selected attack position (i.e., the cell in which fire will be suppressed), rather than optimizing a continuous multi-stop path. Safety is incorporated through spatial constraints, requiring the path to remain outside a specified radius of the fire. Their model uses a finite planning horizon H, treats suppression as effectively blocking fire spread, and considers three objectives: minimizing burned area, suppression costs, and the travel time of the resources.

The experimental results presented in these studies, from the early work of Hof et al. (2000) to the more recent contributions, are often limited to relatively small grid sizes or simplified scenarios. The computational burden tends to increase significantly as models incorporate more operational realism. The detailed routing model of Granda

et al. (2025), for instance, was evaluated on grids no larger than 10x10. Likewise, the largest networks solved in Alvelos et al. (2025) contained approximately 1055 vertices after preprocessing.

2.5.3 Solution Methods

In this section, we review the solution methods proposed in the literature for the problem defined in Section 2.5.1.1. The first method is a mixed-integer programming (MIP) formulation, the second is a decomposition approach based on logic-based Benders decomposition, and the third is an iterated local search heuristic.

2.5.3.1 Mixed Integer Programming

A linear programming formulation for shortest-path trees is based on flow constraints. This idea is captured in model M_{tree}, which constructs a tree rooted at the ignition vertex s by routing flow from s to all other vertices in the graph. For each arc $uv \in A$, we define a real, non-negative variable x_{uv} that indicates how many paths in the tree use arc uv. The model sends exactly $|V| - 1$ units of flow out of the source s, as enforced by constraint (P.2), and ensures that each other vertex receives one unit of flow, via constraint (P.3).

$$
\begin{aligned}
(M_{tree}) \quad \text{min.} \quad & \sum_{uv \in A} t_{uv} x_{uv} && && \text{(P.1)} \\
\text{s.t.} \quad & -\sum_{v \in N_s^+} x_{sv} = -|V| + 1, && && \text{(P.2)} \\
& -\sum_{w \in N_v^+} x_{vw} + \sum_{u \in N_v^-} x_{uv} = 1, && v \in V \setminus \{s\}, && \text{(P.3)} \\
& x_{uv} \geq 0, && uv \in A. && \text{(P.4)}
\end{aligned}
$$

It is a well-known fact that the matrix of coefficients of formulation M_{tree} is totally unimodular, so the optimal solution is always integral. Moreover, the dual of this formulation has a natural interpretation in terms of fire arrival times. Observe that each primal variable x_{uv} is associated with an arc $uv \in A$, and its coefficient t_{uv} in the objective function corresponds to the time it takes for fire to travel from u to v. The dual

constraint associated with x_{uv} will therefore involve a bound on the difference between two dual variables, one for each endpoint of the arc. More precisely, the dual of M_{tree} has a variable $a_v \in \mathbb{R}$ for each vertex $v \in V$, which can be interpreted as the fire arrival time at that vertex. The dual program, $M_{arrival}$, is given below. Constraint (D.2) says that the fire arrival time at vertex v (a_v) is at most the arrival time at any adjacent vertex u (i.e., $uv \in A$) plus the fire travel time t_{uv} from u to v. Constraint (D.3) sets the fire arrival time at the ignition vertex s to zero.

$$(M_{arrival}) \quad \text{max.} \quad \sum_{v \in V} a_v \tag{D.1}$$

$$\text{s.t.} \quad a_v - a_u \leq t_{uv}, \qquad uv \in A, \tag{D.2}$$

$$a_s = 0. \tag{D.3}$$

Since M_{tree} and $M_{arrival}$ form a primal-dual pair, the theory of linear programming guarantees that any optimal solution satisfies the complementary slackness conditions (Vanderbei, 2020). Let $x^\star$ and $a^\star$ be optimal solutions to the primal and dual, respectively. Then, for each arc $uv \in A$, if the arc is not used in the shortest-path tree ($x^\star_{uv} = 0$), the corresponding dual constraint can be loose ($a^\star_v - a^\star_u < t_{uv}$). If $x^\star_{uv} > 0$, then the constraint must be tight, i.e., $a^\star_v - a^\star_u = t_{uv}$. This fact will be used next to derive an alternative formulation for fire arrival times that does not depend on the objective function.

While the dual program $M_{arrival}$ gives the correct fire arrival times at each vertex, this only holds when the objective is to maximize the sum of those times. To address this, Alvelos (2018) proposed a mixed-integer formulation that guarantees correctness of the arrival times regardless of the objective function. The idea is to combine the constraints of M_{tree} and $M_{arrival}$ into a MIP that also enforces the complementary slackness conditions. The formulation is shown below, where slack variables $s_{uv} \geq 0$ are introduced to capture the slack of the dual constraints, and binary variables $q_{uv} \in \{0, 1\}$ indicate whether an arc $uv \in A$ belongs to the shortest-path tree.

$$
\begin{aligned}
(M_{feasibility}) \quad \text{max.} \quad & 0 && && \text{(F.1)}\\
\text{s.t.} \quad & \text{(P.2)} - \text{(P.4)}, \quad \text{(D.3)} && &&\\
& a_v - a_u + s_{uv} = t_{uv}, && uv \in A, && \text{(F.2)}\\
& x_{uv} \leq (|V| - 1) q_{uv}, && uv \in A, && \text{(F.3)}\\
& s_{uv} \leq M(1 - q_{uv}), && uv \in A, && \text{(F.4)}\\
& s_{uv} \geq 0, && uv \in A. && \text{(F.5)}
\end{aligned}
$$

Constraints (P.2) to (P.4) ensure that the variables x_{uv} define a valid flow tree rooted at s, as in the original primal program M_{tree}. Constraint (D.3) sets the fire arrival time at the ignition vertex to zero, and constraint (F.2) defines the slackened version of the dual constraint (D.2) for each arc. Constraints (F.3) and (F.4) enforce complementary slackness. If an arc is used in the tree ($q_{uv} = 1$), the slack must be zero. If the slack is positive, then the arc must be excluded from the flow ($q_{uv} = 0$), which in turn implies $x_{uv} = 0$. In constraint (F.4), the constant M denotes an upper bound on fire the arrival times across all vertices. It is used to deactivate the constraint when $q_{uv} = 0$, allowing the slack to be positive.

Any feasible solution (x, a, s, q) to $M_{feasibility}$ is primal feasible (because of constraints (P.2)–(P.4)), dual feasible (because of constraints (D.3) and (F.2)), and satisfies the complementary slackness conditions (because of constraints (F.3) and (F.4)). Therefore, x and a are optimal solutions to the primal and dual programs, respectively, and a holds the correct fire arrival times at each vertex. On top of this feasibility problem, Alvelos (2018) formulated the problem we defined in Section 2.5.1.1. This formulation was later improved by Harris et al. (2023), and we present it below.

Following Harris et al. (2023), we introduce binary variables $y_v \in \{0, 1\}$ for each vertex $v \in V$, where $y_v = 1$ indicates that vertex v burns before time H. For each time $t \in T$ at which resources are released, and for each vertex $v \in V$, we define binary variables $r_{tv} \in \{0, 1\}$ indicating whether a resource released at time t is allocated to vertex v. The resulting model is shown below.

$$(M_{alvelos}) \quad \text{min.} \quad \sum_{v \in V} y_v \tag{A.1}$$

$$\text{s.t.} \quad \text{(P.2)} - \text{(P.4)}, \quad \text{(D.3)}, \quad \text{(F.3)} - \text{(F.5)}$$

$$a_v - a_u + s_{uv} = t_{uv} + \Delta \sum_{t \in T} r_{tu}, \qquad uv \in A, \tag{A.2}$$

$$\sum_{v \in V} r_{tv} \leq |R_t|, \qquad t \in T, \tag{A.3}$$

$$\sum_{t \in T} r_{tv} \leq 1, \qquad v \in V, \tag{A.4}$$

$$a_v \geq r_{tv} \cdot t, \qquad v \in V,\ t \in T, \tag{A.5}$$

$$y_v \geq 1 - a_v / H, \qquad v \in V. \tag{A.6}$$

Constraint (A.2) modifies the arrival time equations to include the suppression delay Δ, applied to outgoing arcs of any vertex that receives a resource. Constraints (A.3) to (A.4) model resource allocation: at most $|R_t|$ resources can be placed at time t, and each vertex receives at most one resource. Constraint (A.5) ensures that a resource can only be assigned to a vertex if it has not burned by the time of its release. Constraint (A.6) determines whether a vertex burns before the planning horizon H. The objective (A.1) minimizes the number of burned vertices.

2.5.3.2 Logic-based Benders Decomposition

Logic-based Benders decomposition (LBBD) is a decomposition technique used to solve large-scale mixed-integer programming problems. The idea is to decompose the problem into a master problem and a subproblem. The master problem is usually an integer programming problem responsible for making high-level decisions, while the subproblem encodes the implications of those decisions, such as the feasibility of the solution or the objective value. In our case, the master problem will decide which vertices receive a resource and at what time, and which vertices burn before time H. The subproblem will simply check the feasibility of those decisions. In the next paragraphs, we describe how to apply this technique to the problem defined in Section 2.5.1.1 following Harris et al. (2023).

The master problem MP for the fire suppression problem is depicted below. The variables r_{tv} indicate whether a resource released at time $t \in T$ is allocated to vertex

$v \in V$, and y_v indicates whether vertex v burns before time H, in the same way as in the MIP model $M_{alvelos}$ presented earlier. Constraint (A.3) ensures that the number of resources allocated at time t does not exceed the number of resources released at that time, while constraint (A.4) ensures that each vertex receives at most one resource. The objective function (A.1) minimizes the number of burned vertices.

$$
\begin{aligned}
(MP) \quad \min \quad & \sum_{v \in V} y_v && && \text{(A.1)} \\
\text{s.t.} \quad & \sum_{v \in V} r_{tv} \leq |R_t|, && t \in T, && \text{(A.3)} \\
& \sum_{t \in T} r_{tv} \leq 1, && v \in V, && \text{(A.4)} \\
& y_v, r_{tv} \in \{0,1\}, && t \in T,\ v \in V.
\end{aligned}
$$

The master problem MP is a relaxed version of the original problem in the sense that there is no link between the allocation decisions r_{tv} and the objective function value defined by the variables y_v. The idea is to build this link incrementally by solving MP and then checking if the solution is feasible for the original problem. If it is not feasible, we will update MP with additional constraints that enforce the correct relationship between the allocation decisions and the objective function value, and then solve it again. Once the optimal solution to MP is feasible for the original problem, we can stop the algorithm and return the solution. Optimality is guaranteed because the master problem is always a relaxation of the original problem.

Let (y, r) be an optimal solution to MP. We can check whether (y, r) is feasible for the original problem by simply running a shortest-paths algorithm on the graph modified by the allocation decisions r. Let Λ be the allocation defined by r. Two infeasibility scenarios may occur: i) variable y_v may indicate that a vertex did not burn ($y_v = 0$) when in fact it did burn ($a_v^\Lambda < H$), and ii) a variable r_{tv} may indicate that vertex v received a resource at time t ($r_{tv} = 1$), even though it burned before t ($a_v^\Lambda < t$). In both scenarios, we have to update MP with additional constraints to prevent this from happening again. We illustrate how to deal with infeasibility scenarios i) and ii) using the toy instance introduced in Section 2.5.1.1, which is also depicted in Figure 2.10.

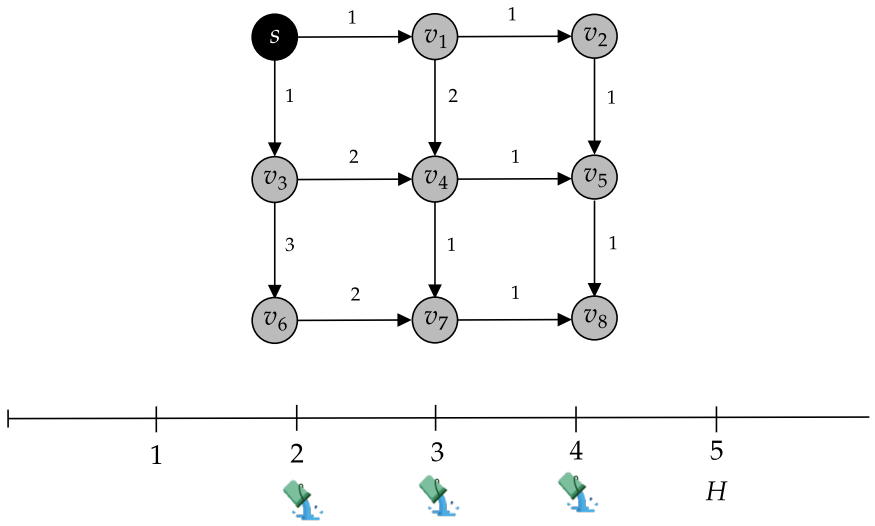


Figure 2.10 – Toy instance introduced in Section 2.5.1.1. A single resource is released at times 2, 3, and 4. The delay Δ introduced by a resource is 2 and the optimization horizon H is 5.

Let us consider scenario i) first, where the the optimal solution to MP sets y_v to 0 when $a_v^\Lambda < H$. For any vertex $v \in V \setminus \{s\}$, there are usually multiple distinct simple paths from s to v in the input graph G, some of them with cost less than H. The possible simple paths from the ignition vertex s to v_5, for example, are $p_1 = (s, v_3, v_4, v_5)$ with cost 4, $p_2 = (s, v_1, v_2, v_5)$ with cost 3, and $p_3 = (s, v_1, v_4, v_5)$ with cost 4. All three paths have a cost less than $H = 5$. Any allocation of resources that does not add at least one resource to the predecessors of v_5 in the paths p_1, p_2, and p_3 will not be able to save v_5 from burning before time H. This can be translated into the following constraints for MP:

$$y_{v_5} \geq 1 - \sum_{u \in \{s, v_3, v_4\}} \sum_{t \in \{2,3,4\}} r_{tu},$$

$$y_{v_5} \geq 1 - \sum_{u \in \{s, v_1, v_2\}} \sum_{t \in \{2,3,4\}} r_{tu},$$

$$y_{v_5} \geq 1 - \sum_{u \in \{s, v_1, v_4\}} \sum_{t \in \{2,3,4\}} r_{tu}.$$

In case ii), something similar can be done to impede MP from setting r_{tv} to 1 when $a_v^\Lambda < t$. Consider vertex v_5 again. If v_5 were to receive a resource at time 4, then any path from s to v_5 with cost less than 4 would have to be blocked by placing at least one resource on it before time 4. This can be done by adding the following constraint to MP:

$$r_{4,v_5} \leq \sum_{u \in \{s, v_1, v_2\}} \sum_{t \in \{2,3\}} r_{tu}.$$

In the example, the only path is (s, v_1, v_2, v_5), while in general there can be several. Furthermore, adding a single resource to each critical path (i.e., a path with cost less than H) is sufficient to ensure that the fire will not reach v_5 before time H. However, depending on the cost of the path and the delay Δ, it may be necessary to add more resources.

Let again (y, r) be the optimal solution to MP, and let Λ be the resource allocation induced by r. Suppose that Λ is not feasible for the original problem because $y_v = 0$ for some vertex $v \in V$ but $a_v^\Lambda < H$. This means that y_v may not be set to 0 unless the fire arrival time at v is delayed by at least $H - a_v^\Lambda$ time units, which can be done by placing at least $\mathcal{R} = \lceil (H - a_v^\Lambda)/\Delta \rceil$ *new resources* on the predecessors $\mathcal{P}_v^\Lambda$ of v under allocation Λ. We say *new resources* because it may be the case that some predecessors of v under allocation Λ are already protected by a resource. This is encoded by the following constraint:

$$y_v \geq 1 - \frac{1}{\mathcal{R}} \sum_{u \in \mathcal{P}_v^\Lambda} \sum_{\substack{t \in \overline{T}_v, \\ (t,u) \notin \Lambda}} r_{tu}, \tag{B.6}$$

where $\overline{T}_v = \{t \in T \mid t \leq a_v^\Lambda + (\mathcal{R} - 1)\Delta\}$. The double sum over all vertices and possible resource allocation times counts the number of resources that are newly allocated to the predecessors P_v^Λ. Therefore, (B.6) allows vertex v not to burn, i.e. $y_v = 0$, only if this number is $\mathcal{R}$ or more.

Now suppose that $r_{tv} = 1$ for some vertex $v \in V$ but $a_v^\Lambda < t$. In this case, the fire must be delayed by at least $\mathcal{F} = \lceil (t - a_v^\Lambda)/\Delta \rceil$ time units, and this can be accomplished by placing at least $\mathcal{F}$ new resources on the shortest path from s to v. The following constraint should be added to MP:

$$r_{tv} \leq \frac{1}{\mathcal{F}} \sum_{u \in \mathcal{P}_v^\Lambda} \sum_{\substack{t' \in \overline{T}_v, \\ (t',u) \notin \Lambda}} r_{t'u}, \tag{B.7}$$

where $\overline{T}_v = \{t \in T \mid t \leq a_v^\Lambda + (\mathcal{F} - 1)\Delta\}$.

Notice that both constraints only impose *necessary conditions* for vertex v to be declared saved ($y_v = 0$) or protected at time t ($r_{tv} = 1$). As we saw in the small example of Figure 2.10, it is often the case that there are multiple paths from s to v that have to be blocked to ensure that fire does not reach v before some given time.

The complete LBBD is given by Algorithm 1. The algorithm starts by solving the master problem MP (line 2), and the resource allocation Λ is then built using the

allocation variables r (line 3). If Λ is feasible for the original problem, it can be returned because it is also optimal (line 5). If Λ is not feasible, new constraints are added to MP to prevent the same infeasibility from occurring again. For each vertex $v \in V$, if v is protected by resource i in Λ, but it burns before time t_i (lines 8 to 10), constraint (B.7) is added to MP (line 11). If v was deemed saved ($y_v = 0$) but it burns before time H (line 14), we add constraint (B.6) (line 15). The algorithm continues until a feasible solution is found. By design, this solution will also be optimal for the original problem.

Algorithm 1: SIMPLEBENDERSDECOMPOSITION

```
1  while True do
2      y, r ← Solve MP
3      Λ ← Build allocation using r
4      if Λ is feasible for the original problem then
5          return Λ
6      end
7      for v ∈ V do
8          if v ∈ P^Λ then
9              Let i ∈ R such that Λ_i = v
10             if a_v^Λ < t_i then
11                 Add constraint B.7 to MP
12             end
13         end
14         if y_v = 0 and a_v^Λ < H then
15             Add constraint B.6 to MP
16         end
17     end
18 end
```

We refer to Algorithm 1 as SIMPLEBENDERSDECOMPOSITION because it simplifies the logic-based Benders decomposition method proposed by Harris et al. (2023). The main difference is that SIMPLEBENDERSDECOMPOSITION adds cuts to the master problem only after solving it to optimality. In contrast, modern implementations add cuts lazily during the branch-and-bound process, whenever a feasible integer solution to the master problem is found. This enables the solver to track and store incumbent solutions that are feasible for the original problem, allowing the algorithm to return the best one found if it stops early due to time limits. SIMPLEBENDERSDECOMPOSITION, by design, does not retain partial solutions and only terminates when optimality is proven.

2.5.3.3 *Iterated Local Search*

Combinatorial problems, such as the one we defined in Section 2.5.1.1, usually have a large set S of feasible solutions. To navigate S in search of good solutions, it is common to impose some structure on it in the form of a *neighborhood*. A neighborhood can be thought of as a function $f : S \to 2^S$ that maps each solution $s \in S$ to a set of solutions $f(s) \subseteq S$. Intuitively, the neighborhood of a solution s is the set of solutions that can be reached from s by applying a small change to it. For example, in the case of the fire suppression problem, we can define a neighborhood by allowing the addition or removal of resources from the allocation Λ. In this case, the neighborhood function would return all possible allocations that can be obtained by removing the resource of a protected vertex and adding it to another vertex.

Local search is an umbrella term for a family of algorithms that iteratively improve a solution s by exploring a given neighborhood. The simplest local search algorithm is hill climbing, which starts with an initial solution s_0 and iteratively explores its neighborhood $f(s_0)$. If a better solution $s_1 \in f(s_0)$ is found, the algorithm moves to s_1 and continues the process. This continues until a *local optimum* is found, i.e., a solution that is better (or not worse) than all its neighbors. Formally, we can think of a local search algorithm as a many-to-one mapping LOCALSEARCH : $S \to S^\star$ that maps solutions in S to a typically smaller set of solutions $S^\star \subseteq S$ that are local optima (Lourenço et al., 2003).

The space of local optima $S^\star$ of any local search algorithm must necessarily contain the optimal solution to the problem. In theory, one could even apply a local search algorithm to $S^\star$, which would map solutions to an even smaller set of solutions $S^{\star\star} \subseteq S^\star$ with better quality. The problem with this approach is that, for most problems, we do not know how to define a neighborhood $f : S^\star \to 2^{S^\star}$ over this space.

Iterated local search is a metaheuristic that tries to overcome this problem by exploring the space of local optima $S^\star$ stochastically. The idea is to apply a local search algorithm to a solution $s \in S$ to reach a local optimum $s' \in S^\star$. Solution s' is then randomly perturbed to generate a different solution $s'' \in S$. Local search is applied again to s'' and, hopefully, a new local optimum $s''' \in S^\star$ is found. Intuitively, the repeated application of local search and perturbation allows the algorithm to 'jump' from one local optimum to another.

Formally, the perturbation operator can be thought of as a function PERTURBATION : $S^\star \to S$ that maps local optima to new solutions. A good perturbation operator should be able to generate new solutions that are not too far from the local optimum, but also not

too close to it. This is important because if the perturbation is too small, the algorithm may get stuck in the same local optimum, while if it is too large, the iterated local search becomes similar to a random search.

In this section, we describe the iterated local search (ILS) proposed by Mendes & Alvelos (2023) to tackle the fire suppression problem. Algorithm 2 shows the main steps of the ILS algorithm. In line 1, MULTISTARTCONSTRUCTIVEHEURISTIC (Algorithm 4) constructs an initial solution $\Lambda^\star$, which is then improved by the local search algorithm LOCALSEARCH (Algorithm 5) in Line 2. At each iteration of the while loop (lines 3 to 8), the perturbation operator PERTURBATION (Algorithm 6) is applied to the current best solution $\Lambda^\star$, generating a new solution Λ. The local search algorithm is then applied to Λ, and if the new solution is better than the previous one, it becomes the new incumbent solution. In the next paragraphs, we briefly describe the main components of the ILS algorithm.

Algorithm 2: ITERATEDLOCALSEARCH

```
1  Λ* ← MULTISTARTCONSTRUCTIVEHEURISTIC()
2  Λ* ← LOCALSEARCH(Λ*)
3  while Termination criteria not met do
4  |   Λ ← PERTURBATION(Λ*)
5  |   Λ ← LOCALSEARCH(Λ)
6  |   if |B_H^Λ| < |B_H^Λ*| then
7  |   |   Λ* ← Λ
8  |   end
9  end
10 return Λ*
```

The MULTISTARTCONSTRUCTIVEHEURISTIC algorithm (Algorithm 4) builds an initial solution by iteratively selecting the available resource with the least release time and assigning it to a vertex that will burn soon. It starts with the empty allocation Λ_0 (line 4) and the set of available resources $R = [k]$ (line 5). At each iteration of the inner while loop (lines 6 to 14), it selects the resource with the earliest release time and identifies the set of vertices that can receive the resource. These vertices are those that are not burned when the resource is released and that are not protected by any other resource, or using our notation, those vertices in $V \setminus (B^\Lambda_{t_i} \cup P^\Lambda)$. The algorithm then builds a restricted candidate list of vertices C by selecting the first `MaxCandidates` vertices in order of their arrival times (line 8). If the candidates list C is not empty, the algorithm randomly selects a vertex from C and assigns the resource to that vertex (lines 10 to 13). This process continues until all resources are assigned, or no more candidates are available.

Each execution of lines 4 to 14 generates a new solution Λ, which is then used to update the incumbent $\Lambda^\star$ in lines 15 to 17. The algorithm terminates after `MaxTrials` solutions have been generated, and the best solution found is returned.

Algorithm 3: CANDIDATESLIST

Data: A set of vertices V, an allocation Λ, and a maximum number of candidates `NumVertices`.
Result: A restricted candidate list $C \subseteq V$ of size at most `NumVertices`.
1 Let $\sigma : \{1, \ldots, |C|\} \to C$ such that $a^\Lambda_{\sigma(1)} \leq \cdots \leq a^\Lambda_{\sigma(|C|)}$
2 $L \leftarrow \{\sigma(1), \ldots, \sigma(\min\{\texttt{NumVertices}, C\})\}$
3 **return** L

Algorithm 4: MULTISTARTCONSTRUCTIVEHEURISTIC

Result: An allocation Λ.
1 $\Lambda^\star \leftarrow \Lambda_0$
2 `trial` $\leftarrow 1$
3 **while** `trial` $<$ `MaxTrials` **do**
4 $\quad \Lambda \leftarrow \Lambda_0$
5 $\quad R \leftarrow [k]$
6 $\quad$ **while** $R \neq \emptyset$ **do**
7 $\quad\quad i \leftarrow \arg\min_{i \in R} t_i$
8 $\quad\quad R \leftarrow R \setminus \{i\}$
9 $\quad\quad C \leftarrow$ CANDIDATESLIST$(V \setminus (B^\Lambda_{t_i} \cup P^\Lambda), \Lambda,$ `MaxCandidates`$)$
10 $\quad\quad$ **if** $C \neq \emptyset$ **then**
11 $\quad\quad\quad v \leftarrow$ Select a vertex from C with uniform probability
12 $\quad\quad\quad \Lambda_i \leftarrow v$
13 $\quad\quad$ **end**
14 $\quad$ **end**
15 $\quad$ **if** $|B^\Lambda_H| < |B^{\Lambda^\star}_H|$ **then**
16 $\quad\quad \Lambda^\star \leftarrow \Lambda$
17 $\quad$ **end**
18 $\quad$ `trial` $\leftarrow$ `trial` $+ 1$
19 **end**
20 **return** $\Lambda^\star$

The LOCALSEARCH algorithm (Algorithm 5) tries to improve the current solution by moving resources from one vertex to another. The core of the algorithm is the for loop in lines 6 to 20, which implements the local search moves by iterating over all protected vertices $u \in P^\Lambda$. Each iteration starts by removing the resource i from the vertex u in Λ (lines 7 and 8), which later will be allocated to a new vertex v. The algorithm then builds a neighborhood N of vertices that are adjacent to the protected vertices in Λ. The neighborhood is built by taking the union of the extended neighborhoods of all protected

vertices in Λ (line 9). Line 10 then generates a restricted candidate list C of vertices that can receive the resource, excluding burned vertices and those already protected by another resource. The CANDIDATESLIST algorithm (Algorithm 3) is called to build this restricted candidate list of size at most `MaxNeighbors`, similarly to what is done in line 9 of MULTISTARTCONSTRUCTIVEHEURISTIC. Next, for each vertex $v \in C$, the algorithm assigns the resource i to v (line 12) and checks if the new solution is better than the incumbent solution Λ' (line 13). If the new solution is better, the incumbent Λ' is updated (line 14). If in a given iteration of the main loop the current solution Λ could not be improved, LOCALSEARCH terminates and returns the best solution found so far.

Algorithm 5: LOCALSEARCH

Data: A solution Λ.
Result: A (possibly) improved allocation.

1 $\Lambda' \leftarrow \Lambda$
2 **repeat**
3 **if** $|B_H^{\Lambda'}| < |B_H^{\Lambda}|$ **then**
4 $\Lambda \leftarrow \Lambda'$
5 **end**
6 **for** $u \in P^{\Lambda}$ **do**
7 Let $i \in [k]$ such that $\Lambda_i = u$
8 Remove resource i from u in Λ
9 $N \leftarrow \bigcup_{v \in P^{\Lambda}} \mathbb{N}_v$
10 $C \leftarrow$ CANDIDATESLIST$(N \setminus (B_{t_i}^{\Lambda} \cup P^{\Lambda}), \Lambda, \texttt{MaxNeighbors})$
11 **for** $v \in C$ **do**
12 $\Lambda_i \leftarrow v$
13 **if** Λ is feasible and $|B_H^{\Lambda}| < |B_H^{\Lambda'}|$ **then**
14 $\Lambda' \leftarrow \Lambda$
15 **end**
16 Remove resource i from v in Λ
17 **end**
18 $\Lambda_i \leftarrow u$
19 **end**
20 **until** $|B_H^{\Lambda'}| \geq |B_H^{\Lambda}|$
21 **return** Λ

The perturbation component (Algorithm 6) is responsible for modifying the current solution Λ to escape local optima. It implements three different perturbation strategies, and the selected strategy depends on the random number r generated in line 3 and on the probabilities $\texttt{prob}_1$ and $\texttt{prob}_2$, which are parameters. In the next paragraphs, we describe each strategy in detail.

The first strategy is to remove a resource from the current allocation (lines 4 to

9), and it is applied with probability $\texttt{prob}_1$. Line 4 builds a set R_{max} of currently used resources that have the latest release time. The algorithm then randomly selects a resource i from this set (line 5) and removes it from the allocation Λ (line 6).

The second strategy (lines 10 to 18) adds a new resource to the allocation, and it is applied with probability $\texttt{prob}_2$. The algorithm builds a set R_{min} of available resources that have the earliest release time (line 11) and randomly selects a resource i from this set (line 12). A restricted candidate list C of vertices that can receive the resource (line 13) is built, and the resource is assigned to one of these vertices (line 16). The size of C depends on a global parameter `MaxCandidates`, the same parameter used in MULTISTARTCONSTRUCTIVEHEURISTIC (Algorithm 4).

The third strategy (lines 19 to 41) moves a resource from one vertex to another, similar to LOCALSEARCH. At each iteration of the while loop, the algorithm attempts a move and continues until reaching either the maximum number of moves `MaxMods` or the maximum number of failed moves `MaxFails`. A move fails when the modified solution is infeasible. At the start of each iteration of the while loop, a protected vertex is randomly selected (line 21). The resource i assigned to this vertex is removed from the allocation (line 23) and a neighborhood N of vertices that are adjacent to the protected vertices in Λ is built (line 24). Equal to LOCALSEARCH (Algorithm 5), the neighborhood is built by taking the union of the extended neighborhoods of all protected vertices in Λ. The algorithm then generates a restricted candidate list C of vertices that can receive the resource, excluding burned vertices and those already protected by another resource (line 25). The size of C depends on the global parameter `MaxNeighbors`, which is the same parameter used in LOCALSEARCH. A vertex v from C is randomly selected (line 27) and resource i is allocated to it. If the new solution is feasible, the move is accepted and the number of successful moves is incremented (lines 30 and 31). If the new solution is infeasible, the algorithm counts the failed move (line 34). Other moves are explored until either `MaxMods` or `MaxFails` is reached.

Algorithm 6: PERTURBATION

Data: A solution Λ.
Result: A perturbation of Λ.

$R_{used} = \{i : i \in [k] \text{ and } \exists v \in P^{\Lambda}, \Lambda_i = v\}$
$R_{available} = [k] \setminus R_{used}$
$r \leftarrow U(0,1)$
if $r < \texttt{prob}_1$ and $R_{used} \neq \emptyset$ **then**
  $R_{max} \leftarrow \{i \in R_{used} : \forall j \in R_{used}, t_i \geq t_j\}$
  $i \leftarrow$ Pick a resource with uniform probability from R_{max}
  Let $v \in P^{\Lambda}$ such that $\Lambda_i = u$
  Remove resource i from u in Λ
else
  if $r < \texttt{prob}_1 + \texttt{prob}_2$ and $R_{available} \neq \emptyset$ **then**
    $R_{min} \leftarrow \{i \in R_{available} : \forall j \in R_{available}, t_i \leq t_j\}$
    $i \leftarrow$ Pick a resource with uniform probability from R_{min}
    $C \leftarrow$ CANDIDATESLIST$(V \setminus (B^{\Lambda}_{t_i} \cup P^{\Lambda}), \Lambda, \texttt{MaxCandidates})$
    if $C \neq \emptyset$ **then**
      $v \leftarrow$ Select a vertex from C with uniform probability
      $\Lambda_i \leftarrow u$
    end
  else
    $\texttt{mod} \leftarrow 0$, $\texttt{fail} \leftarrow 0$
    while $\texttt{mod} < \texttt{MaxMods}$ and $\texttt{fail} < \texttt{MaxFails}$ **do**
      $u \leftarrow$ Select a vertex from P^{Λ} with uniform probability
      Let $i \in [k]$ such that $\Lambda_i = u$
      Remove resource i from u in Λ
      $N \leftarrow \bigcup_{v \in P^{\Lambda}} \mathbb{N}_v$
      $C \leftarrow$ CANDIDATESLIST$(N \setminus (B^{\Lambda}_{t_i} \cup P^{\Lambda}), \Lambda, \texttt{MaxNeighbors})$
      if $C \neq \emptyset$ **then**
        $v \leftarrow$ Select a vertex from N with uniform probability
        $\Lambda_i \leftarrow v$
        if Λ is feasible **then**
          $\texttt{mod} \leftarrow \texttt{mod} + 1$
          $\texttt{fail} \leftarrow 0$
        else
          $\Lambda_i \leftarrow u$
          $\texttt{fail} \leftarrow \texttt{fail} + 1$
        end
      else
        $\Lambda_i \leftarrow u$
        $\texttt{fail} \leftarrow \texttt{fail} + 1$
      end
    end
  end
end
return Λ

3 ALGORITHMIC APPROACHES

In this chapter, we present algorithmic approaches to solve the problem defined in Section 2.5.1.1. The first approach is an improved version of the mixed-integer programming (MIP) formulation proposed by Alvelos (2018). The second is an iterated beam search based on our publication Delazeri & Ritt (2024b). The third is a novel matheuristic based on computing vertex cuts in a graph. We benchmark all algorithms on a set of instances from the literature and compare their performance in Section 3.4.

3.1 Mixed-Integer Programming Formulation

As we saw in Section 2.5.3.1, Alvelos (2018) was the first to propose a mixed-integer programming (MIP) formulation for the fire suppression problem considered in this dissertation. His formulation uses duality theory to compute exact fire arrival times at each vertex, something that was required in the context of his work. The resulting model, however, is computationally expensive to solve even for small instances, as reported by Harris et al. (2023) and Mendes & Alvelos (2023).

Before the work of Alvelos (2018), other authors had already modeled fire propagation in graphs using MIP. In particular, Hof et al. (2000) and Wei et al. (2011) formulated fire spread using inequalities of the form $a_v \leq a_u + t_{uv}$ for each arc $uv \in A$, where a_v is a real variable denoting the fire arrival time to vertex v. These constraints do not guarantee that the variable a_v will correspond to the actual fire arrival time at vertex v, since no mechanism enforces equality on the path taken by the fire. However, in their formulations, arrival times serve only to support decision variables that track whether a vertex burns within a given time horizon. Binary variables y_v indicate whether vertex v is reached before the planning horizon H, and are constrained by $y_v \geq 1 - a_v/H$. The objective minimizes the total number of burned vertices, $\sum_v y_v$. Because minimizing y_v requires $a_v \geq H$, the objective naturally incentivizes the model to increase a_v wherever possible. As a result, even though arrival times may be underestimated, the propagation constraints suffice for practical purposes.

These earlier works, however, did not consider suppression decisions with time restrictions on the availability of resources, which is a featured introduced by Alvelos (2018). In this chapter, we describe a MIP model that combines the simpler propagation constraints used by Hof et al. (2000) and Wei et al. (2011) with the time-constrained

suppression modeling of Alvelos (2018). The resulting formulation obtains significantly better results than the model of Alvelos (2018), and achieves competitive, and in some cases superior, performance compared to algorithms specifically designed for the problem, as shown in the experimental sections of this dissertation. Below, we provide the complete formulation for reference. The constraints are labeled according to the model in which they were introduced in Section 2.5.3.1.

$$\text{min.} \quad \sum_{v \in V} y_v \tag{A.1}$$

$$\text{s.t.} \quad a_s = 0 \tag{H.1}$$

$$a_v \leq a_u + t_{uv} + \Delta \sum_{t \in T} r_{tu} \qquad uv \in A \tag{H.2}$$

$$\sum_{v \in V} r_{tv} \leq |R_t| \qquad t \in T \tag{A.3}$$

$$\sum_{t \in T} r_{tv} \leq 1 \qquad v \in V \tag{A.4}$$

$$a_v \geq r_{tv} \cdot t \qquad t \in T,\ v \in V \tag{A.5}$$

$$y_v \geq 1 - a_v / H \qquad v \in V \tag{A.6}$$

$$y_v, r_{tv} \in \{0, 1\} \qquad v \in V,\ t \in T$$

$$a_v \in \mathbb{R}_{\geq 0} \qquad v \in V$$

The model combines propagation constraints from Hof et al. (2000) and Wei et al. (2011) with suppression constraints inspired by Alvelos (2018). Constraints (H.1)–(H.2) define fire arrival times: (H.1) sets the ignition vertex s to time zero, and (H.2) ensures that fire can only reach a vertex v from a predecessor u after a travel time t_{uv}, delayed by Δ if suppression is applied at u. Constraint (A.3) limits the number of suppression resources that can be used at each time t, while (A.4) ensures each vertex receives at most one resource. Constraint (A.5) prevents resources from being allocated to vertices that would already have burned by time t, and (A.6) defines whether vertex v burns before the planning horizon H. The objective (A.1) minimizes the total number of burned vertices.

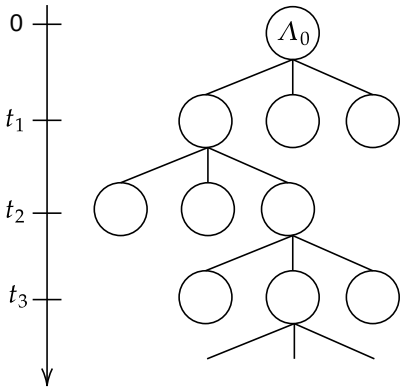


Figure 3.1 – Illustration of the beam search algorithm. Each node corresponds to a partial resource allocation, starting from the empty allocation Λ_0 at the root node. Each level of the tree corresponds to a decision point $t \in T$. In this example, the beam width is $\beta = 1$ and the beam factor is $\eta = 3$.

3.2 Iterated Beam Search

Beam search is a tree search algorithm that visits nodes in a breadth-first manner until a target node is reached. Starting from the root node, beam search keeps a list of β nodes and, at each level of the search tree, nodes in the list are expanded η times. In the literature, β is known as the beam width and η as the beam factor. A heuristic function is then used to rank the $\beta\eta$ expansions, and the best β nodes are selected to continue for the next iteration. Beam search has been extensively used to tackle optimization problems, including scheduling problems (Ow; Morton, 1988), p-median problems (Croce et al., 2004), and ship loading problems (Azevedo et al., 2014). Recently, beam search has also gained some attention in machine learning, specifically in scaling test-time computations in large language models (Snell et al., 2024).

In the context of our problem, each interior node of the search tree represents a partial allocation of resources, and leaf nodes are complete allocations. The root node is Λ_0, and for each $t \in T$, we expand the current set of allocations by applying the resources in R_t. The best leaf node is returned by the algorithm.

Algorithm 7 gives a high level view of our approach. In line 4, we create a set $\mathcal{A}$ containing only Λ_0, which will represent the current state of the search tree. For each time $t \in T$, we use the function STEP to expand each node in $\mathcal{A}$, and we store all the expansions of the current level in the set E. In line 7, we use a heuristic function to select the β best allocations to continue to the next iteration, and in line 9 we find the best leaf node in the search tree. The incumbent is updated in lines 10 and 11, and this procedure is repeated until the termination criteria is not met. The role of the variable z (lines 2, 6,

Algorithm 7: ITERATEDBEAMSEARCH

1 $\Lambda^* \leftarrow \Lambda_0$
2 $z \leftarrow 0$
3 **while** Termination criteria not met **do**
4 $\quad \mathcal{A} \leftarrow \{\Lambda_0\}$
5 $\quad$ **for** $t \in T$ **do**
6 $\quad\quad E \leftarrow \bigcup_{\Lambda \in \mathcal{A}} \text{STEP}(\Lambda, t, z)$
7 $\quad\quad \mathcal{A} \leftarrow \text{prune}(E, t)$
8 $\quad$ **end**
9 $\quad \Lambda \leftarrow \arg\min_{\Lambda \in \mathcal{A}} |B_H^\Lambda|$
10 $\quad$ **if** $|B_H^\Lambda| < |B_H^{\Lambda^*}|$ **then**
11 $\quad\quad \Lambda^* \leftarrow \Lambda$
12 $\quad$ **else**
13 $\quad\quad z \leftarrow (z+1) \mod z_{max}$
14 $\quad$ **end**
15 **end**
16 **return** Λ^*

and 13) will be explained in Subsection 3.2.2, while the procedure STEP used to expand an allocation (line 6) is described in Subsection 3.2.1. We will next define the heuristic function used to prune the search tree.

We propose two heuristic functions to evaluate a partial allocation of resources Λ. The first one, which we call h_1, is equal to the number of burned vertices at time H.

$$h_1(\Lambda) = \sum_{v \in V} [a_v^\Lambda < H].$$

Heuristic h_1 can be quite uninformative in the first few time instants, especially when the delay Δ is low and the optimization horizon H is large. In such situations, it is likely that the first few resources available cannot save any vertices, hence a comparison between two allocations is uninformative. In light of that, we propose a second heuristic, called h_2, which aims to measure how much delay an allocation Λ introduces in the network.

$$h_2(\Lambda) = \sum_{v \in V} \max\{H - a_v^\Lambda, 0\}.$$

As we will see in the experimental section, we can obtain better results by starting with h_2 as the guiding heuristic and then switching to h_1 at some point in time. We call the time at which we start using h_1 *transition instant*, and we denote it by $\hat{t}$. It is better to define the transition instant relative to the velocity with which the fire propagates. To this

end, we define the *free burning time* of an instance as the time at which the last vertex is burned assuming that no vertex is protected by a resource, i.e., the free burning time equals $\max_{v \in V} a_v^{\Lambda_0}$. We can now specify the transition instant as a percentage of the free burning time, and we denote this percentage by $\hat{p}$. To give an example, if the free burning time of an instance is 10 and $\hat{p} = 0.5$, we have that the transition instant $\hat{t}$ is equal to 5.

In summary, for each $t \in T$, if $t < \hat{t}$, we prune the search tree by selecting the β best partial allocations in E using h_2; otherwise, we use h_1. An exception occurs when $t < \hat{t}$ but t is the last decision point, meaning no further resources will be released. In this case, we also use h_1 to evaluate the objective value.

3.2.1 Expanding an Allocation of Resources

We now consider the problem of generating the expansions of a given allocation in the search tree, as is done in line 6 of ITERATEDBEAMSEARCH (Algorithm 7). As a first step, we will design a procedure to create a single expansion, and then embed this procedure in the function STEP (Algorithm 8).

Algorithm 8: STEP

Data: A partial allocation of resources Λ, a time t, the fire perimeter size z.
Result: A set with at most η expansions of Λ using the resources in R_t.

```
1  E ← ∅
2  repeat c|F_t^Λ(z)| times
3      Λ' ← Λ
4      F ← F_t^Λ(z)
5      N ← N_t^Λ(z)
6      for i ∈ R_t and F ≠ ∅ do
7          v ← Pick a vertex from F using probabilities p_v(N, F)
8          Λ'_i ← v
9          F ← F \ {v}
10         N ← (N ∪ ℕ_v) ∩ F
11     end
12     E ← E ∪ {Λ'}
13 end
14 E ← sort(E, t)
15 return First η expansions in E
```

Given an allocation of resources Λ and a time $t \in T$, we have a set of candidate vertices $C = V \setminus (B_t^\Lambda \cup P^\Lambda)$ which can receive a resource and $|R_t|$ resources available. Our goal is to select a subset of C of size $|R_t|$ to apply the resources in R_t. The simplest

approach would be to randomly select $|R_t|$ vertices from C with uniform probability, and we experiment with this approach in the experimental section. In the next paragraphs, we will propose a heuristic reduction rule that obtained state-of-the-art results on the benchmarks instances available in the literature.

Our heuristic reduction rule is based on two key observations about which vertices tend to receive a resource first in high quality solutions:

1. Vertices that are close to burned vertices;
2. Vertices that are neighbors of protected vertices.

Motivated by the first observation, we define the notion of *fire perimeter*, i.e., a set of vertices that are close to the current set of burned vertices. Since the arcs of an instance represent fire velocity instead of physical distance, our notion of closeness must be based on fire arrival time. We define the fire perimeter at a time t as

$$F_t^{\Lambda}(z) = \{v \in C \mid t \leq a_v^{\Lambda} \leq f(t, z)\}$$

for some non-negative integer z, where $f : T \times \mathbb{N}_0 \to [0, H]$ is[1]

$$f(t, z) = \left(\alpha^{\lceil (z+1)/2 \rceil}(t) + \alpha^{\lceil (z+2)/2 \rceil}(t)\right)/2.$$

Recall from Section 2.5.1.1 that $\alpha(t)$ is the earliest time after t in which new resources are released.

Intuitively, the fire perimeter at time t is the set of unprotected vertices whose fire arrival time is between t and some future time t', where $t' = f(t, z)$ for some non-negative integer z. The function $f(t, z)$ generates t' by averaging two future decision points drawn from the sequence $T^\star = (t_1, t_2, \ldots, H)$, which includes all resource release times and the optimization horizon. Depending on whether z is even or odd, $f(t, z)$ yields a value in $T^\star$ or a midpoint between two consecutive elements of $T^\star$. Increasing z gradually pushes t' forward in time, expanding the fire perimeter to include vertices that are expected to burn later.

We clarify how the fire perimeter heuristic works with an example. Suppose we have resources at times 10, 20, and 30, and the optimization horizon H is equal to 60. Assume that the current time is 10 and no resources were deployed yet, i.e., the current allocation is Λ_0. In this scenario, $F_{10}^{\Lambda_0}(0)$, $F_{10}^{\Lambda_0}(1)$, $F_{10}^{\Lambda_0}(2)$, $F_{10}^{\Lambda_0}(3)$ contain the vertices

[1] We write $\alpha^n(t)$ for the composition of α with itself n times, e.g., $\alpha^2(t) = \alpha(\alpha(t))$.

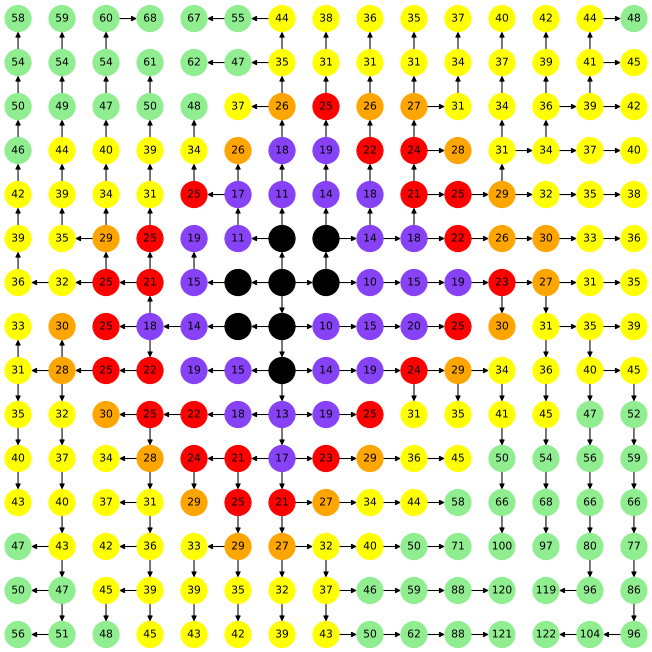


Figure 3.2 – A simple illustration of our definition of fire perimeter. In the example, we have resources at times 10, 20, and 30, and the optimization horizon H is equal to 60. We are at times 10 and no resources were deployed yet, i.e., the current allocation is Λ_0. The set $B_{10}^{\Lambda_0}$ is represented by the black colored vertices, $F_{10}^{\Lambda_0}(0)$ by purple vertices, $F_{10}^{\Lambda_0}(1)$ by red vertices, $F_{10}^{\Lambda_0}(2)$ by orange vertices, and $F_{10}^{\Lambda}(3)$ by yellow vertices. Notice that $F_{10}^{\Lambda_0}(0) \subset F_{10}^{\Lambda_0}(1) \subset F_{10}^{\Lambda_0}(2) \subset F_{10}^{\Lambda_0}(3)$.

that will burn between times 10 and $f(10,0) = 20$, $f(10,1) = 25$, $f(10,2) = 30$, and $f(10,3) = 45$, respectively. Figure 3.2 illustrates the example. Notice how z controls the size of the fire perimeter, and we use its value to control the degree of exploration of the search tree. In the next subsection, we will discuss how the value of z is updated during the execution of ITERATEDBEAMSEARCH (line 13).

When expanding a partial allocation, STEP only considers vertices that are in the fire perimeter $F_t^{\Lambda}(z)$. Among these, it gives preference to those that are adjacent to already protected vertices. To formalize this, we define the set $N_t^{\Lambda}(z)$ as the subset of $F_t^{\Lambda}(z)$ consisting of vertices that are neighbors of protected vertices:

$$N_t^{\Lambda}(z) = F_t^{\Lambda}(z) \cap \bigcup_{v \in P^{\Lambda}} \mathbb{N}_v.$$

We then assign a selection probability p_v to each vertex $v \in F_t^{\Lambda}(z)$, according to the following rule:

$$p_v(N, F) \propto \begin{cases} 1 + p\frac{|F|-|N|}{|N|}, & \text{if } v \in N, \\ 1 - p, & \text{if } v \in F \setminus N. \end{cases} \tag{3.1}$$

This distribution starts as uniform over the fire perimeter and shifts a fraction p of the

total probability mass from vertices in $F \setminus N$ to those in N. When $p = 0$, all vertices in F are equally likely to be selected. When $p = 1$, only vertices in N are selected. If N is empty, we fall back to uniform sampling over F.[2] As we will show in the experimental section, this bias towards vertices adjacent to protected vertices significantly affects the performance of the algorithm.

We are now ready to understand how a partial allocation of resources is expanded, as is done by lines 6 to 11 of STEP. While we have resources available and candidate vertices (line 6), we select a vertex in the fire perimeter F using probabilities $p_v(F, N)$ (line 7), and we protect it with a resource (line 8). We then update F and N by removing the selected vertex from F and adding its neighbors that are in F to N (lines 9 and 10). We repeat this process until all resources are used.

The execution of lines 2 to 12 of STEP create a number of expansions of Λ proportional to the size of the fire perimeter $F_t^{\Lambda}(z)$, for some constant $c \in \mathbb{Z}_+$. In line 14, we sort all the expansions in E using some heuristic function. Similarly to ITERATEDBEAMSEARCH, if $t < \hat{t}$, we use h_2, otherwise we use h_1. Finally, in line 15 we return the first η allocations in E. Note that in line 12 we do not check whether Λ' already is in E, hence it could be the case that $|E| < \eta$.

3.2.2 Dynamical Update of the Fire Perimeter Size

In the last subsection, we defined the notion of fire perimeter, which depends on an integer constant z. Setting z to a value that is too high may increase running times, since the number of iterations performed by STEP is directly proportional to the size of the fire perimeter. On the other hand, setting z to a value that is too low may impede the algorithm to find optimal solutions. To account for that, we propose to start with $z = 0$ and iteratively increase its value once an iteration of beam search (lines 5 to 8 of ITERATEDBEAMSEARCH) is not able to improve the current best solution. We observed in preliminary experiments that increasing z indefinitely does not improve performance and slows downs the algorithm in some cases, so we propose to define a maximum value for z and, once this value is reached, we cycle back to $z = 0$. Line 13 of ITERATEDBEAMSEARCH illustrates that. Note that, for a given choice of $z_{\max}$, the maximum value of z is $z_{max} - 1$.

[2] If both F and N are empty, which is unlikely but possible, we simply do not allocate any resources.

3.2.3 Implementation Details

There are two non-trivial operations in the beam search algorithm, namely the set operations in line 6 of ITERATEDBEAMSEARCH (Algorithm 7) and line 12 of STEP (Algorithm 8), and the update operation on the shortest path tree when we allocate a new resource in line 8 of STEP. As we will see in Section 3.3, a solution can be fully characterized by a boolean vector that indicates which vertices received a resource, which allows us to ignore the time at which the resources were allocated. We use this fact to maintain a set of solutions efficiently.

Regarding the update of the shortest path tree, the simplest approach is to compute everything from scratch using a shortest-path algorithm, such as Dijkstra's algorithm. In our implementation, however, we use the approach of Buriol et al. (2008) for incremental dynamic shortest paths. This algorithm simply identifies the set of vertices affected by the expansion and updates the shortest paths of only these vertices. Even though the theoretical complexity of the algorithm is the same as of Dijkstra's algorithms, it is faster in practice.

3.3 Cut-based Heuristic

A common strategy to suppress fire is to construct firebreaks, which are areas cleared of vegetation to slow or stop fire spread, similar to the example discussed in Section 2.1. In the context of our problem, a firebreak can be viewed as a set of protected vertices that form a barrier against fire propagation. Given the finite nature of our grid, effective solutions typically create firebreaks that separate burned vertices from unburned ones, functioning similarly to a vertex cut in a graph. Building on this idea, in this section, we propose a matheuristic that identifies vertex cuts in the graph and transforms them into feasible resource allocations.

We start by ignoring the time dimension of problem and assuming that, at a given time $\bar{t}$, all resources are released at the same time. Let $R = [k]$ be the set of resources. At time $\bar{t}$, we have a set of burned vertices $B_{\bar{t}}^{\Lambda_0}$ and a set of unburned vertices $V \setminus B_{\bar{t}}^{\Lambda_0}$. Our goal is to find a vertex cut $P \subseteq V \setminus B_{\bar{t}}^{\Lambda_0}$ that separates the burned vertices from the unburned ones. Among all possible cuts, we want the one that minimizes the number of vertices reachable by the burned vertices subject to the constraint that the size of the cut cannot exceed the number of available resources. Figure 3.3 illustrates this idea in the

example instance of Section 2.5.1.

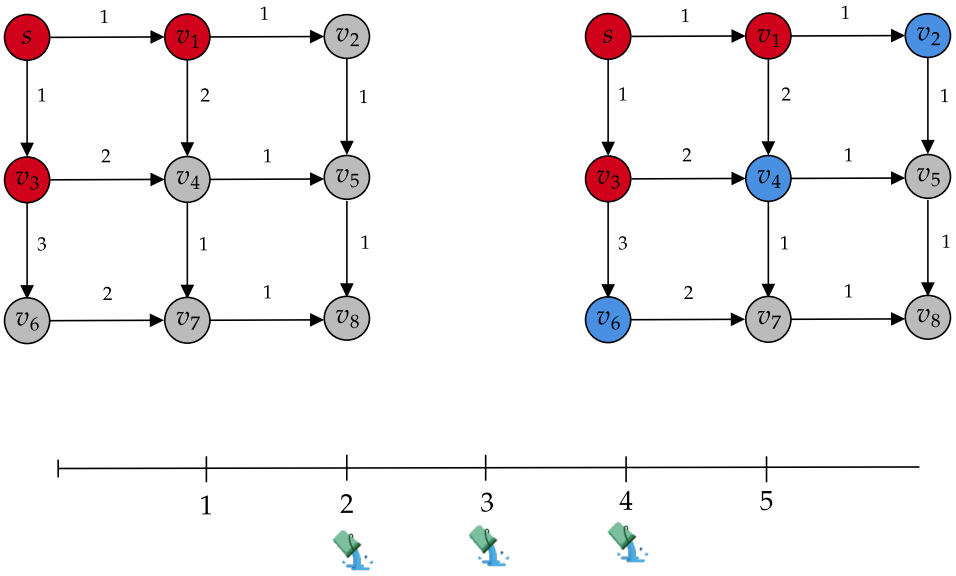


Figure 3.3 – Illustration of the cut-based heuristic applied to the example of Section 2.5.1. The ignition is s and there are three resources available, released at times 2, 3, and 4. The delay Δ is 2 and the optimization horizon H is 5. If we let $\bar{t} = 2$, then at the time $\bar{t}$ the set of burned vertices is $B_{\bar{t}}^{\Lambda_0} = \{s, v_1, v_3\}$ and we want to find a vertex cut P with at most three vertices that minimizes the number of vertices reachable by vertices in $B_{\bar{t}}^{\Lambda_0}$. The optimal vertex cut $P = \{v_2, v_4, v_6\}$ is shown in blue. Coincidentally, this is also the optimal solution to the instance.

Let P be a vertex cut. To create a feasible allocation, we have to assign to each vertex in P a resource in R such that the feasibility constraints are satisfied. Recall from Section 2.5.1 that an allocation Λ is feasible if it is an injective function from $[k]$ to V and, if $\Lambda_i = v$ for $i \in [k]$ and $v \in V$, then $a_v \geq t_i$, i.e., vertex v is not burned when i is released. As we will show later, the problem of deciding whether a given subset of vertices $P \subseteq V$ can be turned into a feasible allocation is solvable by a greedy algorithm.

Algorithm 9 gives a high level view of our approach. In lines 2 and 3 we define lower and upper bounds for the value of $\bar{t}$ as the first and last times when resources are released, and in lines 4 to 14 we perform a binary search to find the best allocation. At each iteration, we compute a cut parameterized by $\bar{t}$ and check if it can be transformed into a feasible allocation. If the answer is yes, we update the incumbent solution and set the upper bound to $\bar{t}$ (line 9). Otherwise, we set the lower bound to $\bar{t}$ (line 12). We repeat this process until the difference between the upper and lower bounds is smaller than a given tolerance ϵ (line 14). In the next two subsections, we will describe how to compute a cut and how to transform it into a feasible allocation.

Algorithm 9: CUTHEURISTIC

$\Lambda^* \leftarrow \Lambda_0$
$t_{lb} \leftarrow \min T$
$t_{ub} \leftarrow \max T$
do
  $\bar{t} \leftarrow (t_{lb} + t_{ub})/2$
  $P \leftarrow$ COMPUTECUT$(\bar{t})$
  $\Lambda \leftarrow$ GREEDYSCHEDULE(P)
  if Λ is feasible **then**
    Update incumbent if $|B_H^{\Lambda}| < |B_H^{\Lambda^*}|$
    $t_{ub} \leftarrow \bar{t}$
  else
    $t_{lb} \leftarrow \bar{t}$
  end
while $|t_{lb} - t_{ub}| > \epsilon$
return Λ^*

3.3.1 Cut Model

Given $\bar{t}$, and assuming that no resources were deployed yet, our goal is to find a vertex cut of size at most $|R|$ that minimizes the number of vertices reachable by vertices in $B_{\bar{t}}^{\Lambda_0}$. This can be accomplished by solving model M_{cut}, presented below. The model uses binary variables x_v to say whether vertex $v \in V$ is part of the cut, and binary variables y_v to track whether vertex v can be reached from the source vertices. Constraint (C.2) ensures that the source vertices are reachable. Burnt vertices cannot receive a resource, so Constraint (C.3) sets x_v to zero for all the source vertices. Constraint (C.4) is a propagation constraint, and it says that if vertex u can be reached from the sources, than its successors must also be reachable unless u is part of the cut. Finally, Constraint (C.5) says that the size of the vertex cut cannot exceed the number of available resources. The objective function minimizes the number of vertices that can be reached from the sources.

$$(M_{cut}) \quad \min \quad \sum_{v \in V} y_v \tag{C.1}$$

$$\text{s.t.} \quad y_v = 1 \qquad v \in B_{\bar{t}}^{\Lambda_0} \tag{C.2}$$

$$x_v = 0 \qquad v \in B_{\bar{t}}^{\Lambda_0} \tag{C.3}$$

$$y_u \leq y_v + x_u \qquad uv \in A \tag{C.4}$$

$$\sum_{v \in V} x_v \leq |R| \tag{C.5}$$

$$x_v, y_v \in \{0, 1\} \qquad v \in V \tag{C.6}$$

Algorithm 10 implements COMPUTECUT, which is called in Line 6 of CUTHEURISTIC (Algorithm 9). In Line 1, we solve M_{cut} parameterized by the time $\bar{t}$ passed as argument, and Line 4 builds a set P with all the vertices that are part of the cut.

Algorithm 10: COMPUTECUT

Data: A time $\bar{t}$.
Result: A set of vertices P.
1 $x, y \leftarrow$ Solve model M_{cut}
2 $P \leftarrow \{v \in V \mid x_v = 1\}$
3 **return** P

3.3.2 Finding a Feasible Schedule

Suppose we are given a set of vertices $P \subseteq V$ that received a resource, with $|P| \leq |R|$, and our task is to find a partial injective function $\Lambda : R \to P$ that does not assign a resource $i \in R$ to a vertex $v \in P$ if v is burned when i is released. We claim that a greedy rule that assigns the earliest available resource to the vertex in P with the earliest fire arrival time is optimal. For each resource $i \in R$, recall that t_i denotes its release time, and for every vertex $v \in V$, let a_v^P be the fire arrival time at v when the vertices in P are protected. Algorithm 11 implements the greedy rule.

The while loop in GREEDYSCHEDULE considers each resource in R and each vertex in P at most once. As a result, each resource is assigned to at most one vertex, and each vertex is protected by at most one resource. Assignments occur in Line 6, subject to the condition that the target vertex is not burned at the time the resource is released. This

Algorithm 11: GREEDYSCHEDULE

Data: Set of protected vertices $P \subseteq V$.
Result: A feasible allocation of resources Λ (if it exists).

```
1  Λ ← Empty allocation
2  while |P| > 0 do
3      i ← arg min_{i∈R} t_i
4      v ← arg min_{v∈P} a_v^P
5      if a_v^P ≥ t_i then
6          Λ_i ← v
7      else
8          return Infeasible solution
9      end
10     R ← R \ {i}
11     P ← P \ {v}
12 end
13 return Λ
```

is enough to guarantee that if Algorithm 11 returns an allocation, then this allocation is feasible.

We now show that if a feasible allocation exists, then the greedy rule is able to find it. The proof consists in transforming any feasible allocation $\Lambda^\star$ into one that is equal to the obtained by GREEDYSCHEDULE. Consider the first iteration of the while loop where Λ and $\Lambda^\star$ differ. Let v and i be the vertex and resource selected by lines 3 and 4 of GREEDYSCHEDULE, $w = \Lambda^\star_i$ be the vertex to which $\Lambda^\star$ assigns resource i, and j be the resource which is assigned to v by $\Lambda^\star$. We claim that $t_i \leq t_j \leq a_v^P \leq a_w^P$. From left to right, the first inequality holds because Line 3 selects the earliest available resource, the second inequality holds because $\Lambda^\star$ is a feasible allocation and j is assigned to v, and the last inequality holds because Line 4 selects the vertex with the earliest fire arrival time. If we swap i and j in $\Lambda^\star$, the allocation remains feasible. We can repeat this argument until we reach an allocation that is equal to the one obtained by GREEDYSCHEDULE.

3.4 Experiments

We now evaluate the algorithms described in this chapter. Section 3.4.1 introduces the benchmark instances used in the experiments, and Section 3.4.2 specifies the parameter values adopted for all algorithms. Sections 3.4.3 and 3.4.4 examine some key components of the beam search algorithm. The former analyzes the effect of the transition

Table 3.1 – Instances used in the experiments.

	Resources per time instant							
Group	10	20	30	40	50	60	H	Δ
LA	3	3	3	3	0	0	70	50
LB	3	3	3	3	3	3	70	30

instant, while the latter evaluates the impact of the heuristic rules for expanding an allocation. Section 3.4.5 assesses the performance and scalability of the proposed MIP model, and Section 3.4.6 reports results for the cut-based heuristic. Lastly, in Section 3.4.7, we compare the proposed algorithms with algorithms from the literature.

All the experiments were done on a platform with a 3.5 GHz AMD Ryzen 9 3900X 12-Core processor, 32 GB of main memory, and Ubuntu Linux 20.04. Our algorithms were implemented in C++ and compiled with GCC 13.3.1 with maximum optimization. Some algorithms require a solver to run, and we used Gurobi 10.0.3 with default settings for this purpose.

3.4.1 Instance Set

We consider the set of instances proposed by Harris et al. (2023), consisting of 16 problems defined on 20×20 grid graphs. In all instances, the fire starts at the center of the grid, and the optimization horizon is fixed at 70. The instances are divided into two groups, LA and LB, each containing 8 instances. Within each group, all instances share the same number of resources released over time and the same magnitude of delay caused by each resource. The instance set summarized in Table 3.1. The optimal solution is known for every instance, so we report algorithm performance in terms of the absolute deviation from the optimal objective value. Appendix B.1 lists the optimal objective value and number of vertices for each instance.

Fire travel times are modeled using direction-dependent uniform distributions to capture the influence of wind. Directions aligned with the wind are assigned shorter propagation times, while opposing directions are slower. For each arc, the travel time is sampled from a distribution determined by the arc's direction. For example, to model fire propagation times under the influence of a south-east wind, arcs pointing south use $U(2, 4)$ and arcs pointing east use $U(4, 6)$, while arcs pointing north and west use $U(7, 9)$ and $U(6, 8)$, respectively. See Table 5 in Harris et al. (2023) for details.

Table 3.2 – Description of parameter values.

Parameter	Value	Description
β	50	Beam width
η	70	Beam factor
c	30	See Algorithm 8
p	0.5	See Equation 3.1
z_{max}	3	See Algorithm 7
ϵ	0.01	See Algorithm 9

3.4.2 Parameter Values

In all the experiments below, the beam search and the cut-based heuristic use the same set of parameter values, shown in Table 3.2. These values were selected based on preliminary experiments. The iterated local search algorithm of Mendes & Alvelos (2023) also relies on several parameters, which we set according to the authors' recommendations.

As we saw in Section 3.2, beam search also has a parameter $\hat{p}$ which defines the transition instant, the time at which we switch the guiding heuristic used to prune the search tree. In the next section, we conduct an experiment to find the best transition instant for the instances we are considering.

3.4.3 Transition Instant and Heuristic Functions

In Section 3.2, we introduced the concept of transition instant for the beam search algorithm. The transition instant is the point at which the algorithm switches from using heuristic h_2 to heuristic h_1 for guiding the search, and we define it as a fraction $\hat{p}$ of the free-burning time. To analyze its impact on performance, we conducted an experiment varying $\hat{p}$ from 0 to 1. When $\hat{p} = 0$, the algorithm uses only h_1; when $\hat{p} = 1$, it uses only h_2, except at the final decision point where the objective function is directly evaluated. Intermediate values combine both heuristics, with the switch occurring at the time defined by $\hat{p}$.

Table 3.3 summarizes the results of our experiment, where beam search was run for 600 seconds on each of the 16 instances, with 10 replications per instance. Performance is evaluated using the absolute deviation from the optimal objective value (δ), and Table 3.3 shows the mean and the standard deviation of δ for each instance group. Since each group contains eight instances, and we perform 10 replications per instance,

the means are computed over 80 different executions. In the last three columns, we also report the percentage of the 80 runs in which the optimal solution was found (OPT(%)).

We use the absolute deviation as our performance metric because the 16 instances are structurally similar, in the sense that they have nearly the same number of vertices and their optimal objective values are of the same order of magnitude, as Table B.1 in the appendix shows. For any instance I, let $A(I)$ be the objective value obtained by the algorithm, and $A^{\star}(I)$ the optimal objective value. The absolute deviation is then calculated as $\delta = A(I) - A^{\star}(I)$. Table 3.3 reports the average and the standard deviation of this quantity computed over 80 runs.

Results are similar across all values of $\hat{p}$, but $\hat{p} = 0.4$ achieves the best average performance, both in terms of solution quality and frequency of finding the optimal solution. The case $\hat{p} = 0$, where only h_1 is used, obtains the worst results and exhibits slightly higher variance due to a few outliers. The values $\hat{p} = 0.8$ and $\hat{p} = 1$ lead to identical results, as in both cases the transition instant happens after the last decision point.

Given these results, we use $\hat{p} = 0.4$ as the transition instant in all subsequent experiments of this chapter. As we will see in Section 4.2, however, when the algorithm is evaluated on more challenging instances, its performance improves with higher values of $\hat{p}$.

Table 3.3 – Performance of beam search for different values of the transition instant $\hat{p}$, which defines the point at which the algorithm switches from heuristic h_2 to heuristic h_1. When $\hat{p} = 0$, only h_1 is used to guide the search; when $\hat{p} = 1$, only h_2 is used, except at the final decision point. Intermediate values combine both heuristics. We report the average and standard deviation of the absolute deviation from the optimal objective value (δ), computed over 80 runs (8 instances and 10 replications per instance). The OPT(%) column indicates the percentage of runs in which the optimal solution was found.

	δ						OPT(%)					
	0	0.2	0.4	0.6	0.8	1	0	0.2	0.4	0.6	0.8	1
LA	1.8 ± 4.8	0.1 ± 0.3	0.0 ± 0.0	0.2 ± 0.5	0.2 ± 0.5	0.2 ± 0.5	85.0	91.2	100.0	88.8	88.8	88.8
LB	1.1 ± 2.2	0.3 ± 0.4	0.2 ± 0.4	0.3 ± 0.8	0.4 ± 0.8	0.4 ± 0.8	62.5	72.5	75.0	77.5	72.5	72.5

3.4.4 Fire Perimeter

In Section 3.2.1, we introduced some heuristic rules for expanding an allocation of resources. Recall that, given a partial allocation Λ and a time $t \in T$, we have $|R_t|$ resources available to be allocated and a set of vertices $C = V \setminus (B_t^\Lambda \cup P^\Lambda)$ that can receive resources. Set C contains all vertices that are not burned at time t and that do not have a resource allocated to them. Our heuristic rules are based on considering only a subset $F \subseteq C$ of vertices that will burn soon. We call F the fire perimeter. Among the vertices in F, we create a set $N \subseteq F$ of vertices that are adjacent to vertices protected by Λ. We then select $|R_t|$ vertices from F probabilistically, giving more preference to vertices that are also in N. This preference is controlled by a parameter p, and when p is equal to one we only select vertices that are in N. When p is equal to zero, we select vertices from F uniformly.

To evaluate the effectiveness of the rules, we conducted an experiment comparing four versions of the algorithm. We denote by IBS$_{FN}$ the standard version of beam search, which uses sets F and N to expand an allocation. We denote by IBS$_{CN}$ the same algorithm, but without the fire perimeter heuristic. In IBS$_{CN}$, we still bias the selection of vertices towards those that are adjacent to protected vertices, but the pool of candidates is C instead of F. We also consider IBS$_F$, which is identical to IBS$_{FN}$, but with $p = 0$, i.e., we select vertices from F uniformly. Finally, IBS$_C$ is a version of beam search identical to IBS$_{CN}$, but with $p = 0$. We set $\hat{p} = 0.4$ for all four versions, as this transition instant obtained the best results in the previous experiment. Each algorithm was run 10 times on each of the 16 instances for 600 seconds, and results are presented Table 3.4 using the same performance metrics as before.

The first two columns of Table 3.4 show that removing the fire perimeter heuristic (moving from IBS$_{FN}$ to IBS$_{CN}$) slightly increases the average absolute deviation. However, this removal significantly reduces the frequency with which optimal solutions are found, as indicated by the decrease in OPT(%). In contrast, eliminating the bias toward vertices adjacent to protected vertices (from IBS$_{FN}$ to IBS$_F$) leads to a large deterioration in solution quality: deviations from optimality become substantially larger, and no optimal solutions are identified across all 160 runs. This impact is even stronger when comparing IBS$_{CN}$ to IBS$_C$, which is expected since the search space of IBS$_C$ is larger than that of IBS$_{CN}$.

Overall, the data suggests that the fire perimeter heuristic is beneficial mainly for

Table 3.4 – Impact of the heuristic rules used to expanding an allocation of resources. We compare four algorithm variants: with (IBS_{FN}, IBS_{F}) and without (IBS_{CN}, IBS_{C}) the fire perimeter heuristic, and with ($p = 0.5$) or without ($p = 0$) bias towards vertices adjacent to protected vertices. We report the average and standard deviation of the absolute deviation from the optimal objective value (δ) and the percentage of runs in which the optimal solution was found (OPT(%)).

	δ				OPT(%)			
	IBS_{CN}	IBS_{FN}	IBS_{C}	IBS_{F}	IBS_{CN}	IBS_{FN}	IBS_{C}	IBS_{F}
LA	0.1 ± 0.3	0.0 ± 0.0	26.3 ± 5.5	11.1 ± 5.4	87.5	100.0	0.0	0.0
LB	1.1 ± 0.9	0.2 ± 0.4	19.5 ± 4.1	7.5 ± 3.2	31.2	75.0	0.0	0.0

increasing the likelihood of identifying optimal solutions, though its removal has only a minor impact on solution quality. On the other hand, maintaining a bias toward vertices adjacent to protected vertices is critical. Removing this bias drastically worsens solution quality and completely eliminates the algorithm's capability to identify optimal solutions.

3.4.5 MIP Model

We now analyze the performance of the MIP model introduced in Section 3.1. Previous works proposing algorithms for the fire suppression problem often compare their results with the MIP model of Alvelos (2018). As noted by Mendes & Alvelos (2023) and Harris et al. (2023), and confirmed by our experiments below, this model obtains inferior solutions compared to heuristic approaches and struggles to find feasible solutions for some instances within typical time limits. As we will demonstrate in this section and in Section 3.4.7, the MIP formulation described in Section 3.1 is competitive with the best algorithms proposed in the literature and in this work.

We evaluate the performance of the proposed MIP formulation using the experimental setup from previous sections, with a time limit of 600 seconds per run and 10 replications for each instance. The results are summarized in Table 3.5. The table shows solution-quality metrics similar to previous experiments: the absolute deviation from the optimal objective value (δ) and the percentage of optimal solutions found (OPT(%)). We also report two metrics for evaluating MIP performance. The first metric is the MIP gap, which measures the relative difference between the best feasible solution found (incumbent) and the best lower bound found by the solver during execution. For any instance I, let $A(I)$ be the objective value of the best integer solution and $L(I)$ the best lower bound found within the time limit. The MIP gap is then calculated as $\frac{A(I)-L(I)}{L(I)}$. A large MIP gap

indicates difficulties for the solver in proving optimality. The second metric is the Root gap, defined as the relative deviation between the objective value of the linear programming relaxation at the root node and the objective value of the optimal solution. Let $R(I)$ denote the root relaxation value and $A^\star(I)$ the optimal objective value. The Root gap is $\frac{A^\star(I)-R(I)}{R(I)}$. A large Root gap indicates that the root relaxation is weak, which can lead to poor performance of the branch-and-bound algorithm in closing the optimality gap.

Analyzing the results for the proposed MIP model in Table 3.5, we observe that it generally achieves lower solution quality than the best-performing beam search variants, both in terms of absolute deviation (δ) and the percentage of optimal solutions found (OPT(%)). Nevertheless, it consistently produces close-to-optimal solutions, as reflected in the low average values of δ (0.2 for LA, 3.4 for LB). The data also indicates that the solver struggles to prove optimality within the time limit, particularly in group LB, as shown by the relatively high average MIP gap values (7.0% for LA, 21.7% for LB). The Root gap is also substantial in both groups (360.5% for LA, 265.3% for LB), indicating that the formulation's initial relaxation is weak.

To highlight the improvements achieved by the proposed reformulation, we also evaluated the original MIP model proposed by Alvelos (2018) using the the same experimental setup. The results are presented in Table 3.6. Note that this table includes an additional column, FEAS(%), indicating the percentage of runs in which the solver found at least one feasible integer solution within the time limit. As specified in the table caption, for runs where no feasible solution was found, we assumed the worst-case outcome (all vertices burn) when calculating δ and the MIP Gap.

The performance of the model of Alvelos (2018), shown in Table 3.6, differs significantly from the proposed formulation. Firstly, the previous model frequently fails to find any feasible solution within the 600-second time limit, achieving only 75.0% feasibility for LA instances and 50.0% for LB instances. The proposed model, while not always finding the optimum, consistently found feasible solutions in all runs. Secondly, the solution quality of the previous model is significantly worse, with average deviations (δ) of 60.9 and 58.4 for LA and LB groups, respectively, compared to 0.2 and 3.4 for the proposed model. This difference is also reflected in the fact that the previous model found the optimal solution in 0.0% of the runs, whereas the proposed model found the optimum in 86.2% of LA runs and 23.8% of LB runs. Thirdly, the MIP gaps reported for the previous model are extremely high (over 100%). These large average gaps are partly due to runs where no feasible solution was obtained, but also indicate that even when feasible

Table 3.5 – Performance of the MIP model. The columns show the absolute deviation from the optimal objective value (δ), the average relative gap between the best integer solution and the best lower bound found by the solver (MIP Gap (%)), the average relative gap between the root relaxation value and the optimal objective value (Root Gap. (%)), and the percentage of runs in which the optimal solution was found (OPT(%)).

	δ	MIP Gap (%)	Root Gap. (%)	OPT(%)
LA	0.2 ± 0.5	7.0 ± 6.5	360.5 ± 112.5	86.2
LB	3.4 ± 3.3	21.7 ± 6.1	265.3 ± 86.3	23.8

Table 3.6 – Performance of the MIP model of Alvelos (2018). The columns show the absolute deviation from the optimal objective value (δ), the average relative gap between the best integer solution and the best lower bound found by the solver (MIP Gap (%)), the average relative gap between the root relaxation value and the optimal objective value (Root Gap. (%)), the percentage of runs where a feasible solution was found (FEAS(%)), and the percentage of runs in which the optimal solution was found (OPT(%)). In runs where a feasible solution was not found, we considered that all vertices burned when computing δ and MIP Gap (%).

	δ	MIP Gap (%)	Root Gap. (%)	FEAS(%)	OPT(%)
LA	60.9 ± 26.4	101.2 ± 30.6	401.2 ± 102.0	75.0	0.0
LB	58.4 ± 29.8	119.3 ± 41.8	320.7 ± 82.5	50.0	0.0

solutions were found, the gap between the solution and the lower bound remained very large within the time limit. While both models exhibit large root gaps (indicating weak initial relaxations), the values for the previous model (401.2% and 320.7%) are worse than those for the proposed model (360.5% and 265.3%).

In summary, while the proposed MIP model typically produces solutions of slightly lower quality compared to the best beam search variants, it represents a substantial improvement over the original formulation by Alvelos (2018). It consistently finds near-optimal solutions and proves optimality in about half of the runs within the time limit. The high root relaxation and MIP gap values indicate that future work should focus on developing stronger formulations or introducing valid inequalities to improve solver performance.

3.4.6 Cut-Based Heuristic

In this section, we analyze the performance of the cut-based heuristic introduced in Section 3.3. Following previous experiments, we set a time limit of 600 seconds and do 10 replications per instance. The results are summarized in Table 3.7.

Table 3.7 shows the absolute deviation from the optimal objective value (δ), the percentage of optimal solutions found (OPT(%)), and the total running time in seconds

Table 3.7 – Performance of the cut-based heuristic. We report the average and standard deviation of the absolute deviation from the optimal objective value (δ), the percentage of runs in which the optimal solution was found (OPT(%)), and the total execution time in seconds (Time (s)).

	δ	OPT(%)	Time (s)
LA	37.62 ± 11.29	0.0	3.75 ± 5.23
LB	61.25 ± 5.90	0.0	0.00 ± 0.00

(Time (s)). In terms of solution quality, the cut-based heuristic performs worse than the other algorithms evaluated in this work, as shown by the values of δ and OPT(%). The algorithm performs particularly poorly on instances in the LB group, which is explained by the fact that the cut-based heuristic assumes that a firebreak separating the ignition from the other vertices will stop the fire from spreading. This assumption is not valid for instances in this group, where the delay Δ caused by a resource is small compared to the optimization horizon. On the other hand, the average execution time of the cut-based heuristic is very low. As shown in Table 3.7, the algorithm often finishes in less than 4 seconds, and in many cases, it completes almost instantaneously. As we will see in Chapter 4, low running times are obtained even for large instances, and solution quality compared to other algorithms improves significantly.

3.4.7 Comparison with the Literature

Table 3.8 – List of algorithms considered in the experiment.

Algorithm	Section
IBS	Section 3.2
LBBD	Section 2.5.3.2
ILS	Section 2.5.3.3
MIP	Chapter 3.1
CUTH	Section 3.3
RS	Appendix B.3

In this section, we compare the performance of all algorithms evaluated in this work and in the literature. The comparison includes the best variant of the beam search algorithm (IBS), the logic-based Benders decomposition of Harris et al. (2023) (LBBD), the iterated local search of Mendes & Alvelos (2023) (ILS), the MIP model described in Section 3.1 (MIP), the cut-based heuristic introduced in Section 3.3 (CUTH), and a random search over the space of feasible solutions (RS), which will be used as a baseline. Table 3.8 provides a summary of all methods considered.

All algorithms were executed on the same computational platform described at the beginning of this chapter. For LBBD and ILS, we use the original implementation provided by Harris et al. (2023). Each algorithm was run 10 times on each of the 16 instances, with a time limit of 600 seconds per run. The results are presented in Table 3.9.

The results show that IBS is the best-performing algorithm in this set of instances. It finds the optimal solution in all 10 replications for 14 out of the 16 instances. LBBD ranks second, solving 9 instances to optimality in all replications, followed by MIP with 5 instances, and ILS with only one. The cut-based heuristic and the random search failed to find any optimal solution. This ranking is consistent when we look at average solution quality: IBS obtains the lowest average deviation from the optimal solution, followed by LBBD, MIP, and ILS. CUTH and RS have the worst performance.

The results largely agree with conclusions from previous works regarding relative algorithm performance. In Harris et al. (2023), LBBD outperforms ILS when given a much larger time limit of 7200 seconds. Here, with only 600 seconds, LBBD still significantly outperforms ILS. Regarding exact methods, while Mendes & Alvelos (2023) and Harris et al. (2023) reported challenges with their MIP models failing to find feasible solutions for several instances within the time limit, the MIP formulation described in Section 3.1 proved more effective in our tests. It successfully found feasible solutions for all instances and performed competitively with LBBD and ILS.

As expected, the cut-based heuristic has the worst performance among all methods considered. When the delay Δ is small, even the random search obtains better results. As we will see in the next chapter, however, CUTH becomes more competitive in other instance classes.

To better understand the evolution of the algorithms over time, Figure 3.4 shows the performance profile of all methods. The x-axis shows the elapsed time in seconds, and the y-axis shows the percentage of the 160 executions (16 instances and 10 replications) in which the algorithm found an optimal solution within that time.

The plot shows that IBS quickly finds high-quality solutions: within the first 100 seconds, it finds the optimal solution in roughly 80% of the runs. In contrast, LBBD and MIP reach only about 40% in the same time interval, and ILS stay at 20%. RS and CUTH do not find any optimal solutions. After 300 seconds, IBS's curve plateaus, while LBBD and MIP continue to make progress, as they explore the search space systematically. Within 600 seconds, IBS has found the optimal solution in 88% of the runs, followed by LBBD (70%), MIP (60%), and ILS (25%).

In summary, the beam search algorithm clearly outperforms all other methods in terms of solution quality and time to optimality. LBBD ranks second, followed by the MIP model, which improves on previous formulations by finding feasible solutions for all instances within the time limit. ILS performs modestly, while CUTH and RS serve as baselines for comparison.

Table 3.9 – Comparison between the proposed algorithms and the algorithms in the literature. We report the average and standard deviation of the absolute deviation from the optimal objective value (δ) and the percentage of runs in which the optimal solution was found (OPT(%)). The averages are calculated over 10 runs.

	δ						OPT(%)					
	IBS	CUTH	MIP	ILS	LBBD	RS	IBS	CUTH	MIP	ILS	LBBD	RS
LA0	0.0 ± 0.0	45.0 ± 0.0	0.2 ± 0.4	0.8 ± 2.5	0.0 ± 0.0	68.1 ± 2.8	100	0	80	90	100	0
LA1	0.0 ± 0.0	54.0 ± 0.0	0.0 ± 0.0	2.0 ± 3.4	0.0 ± 0.0	72.8 ± 3.1	100	0	100	70	100	0
LA2	0.0 ± 0.0	38.0 ± 0.0	1.2 ± 0.9	0.9 ± 0.9	0.0 ± 0.0	62.4 ± 2.0	100	0	30	40	100	0
LA3	0.0 ± 0.0	39.0 ± 0.0	0.2 ± 0.4	10.1 ± 5.2	0.0 ± 0.0	59.0 ± 2.1	100	0	80	0	100	0
LA4	0.0 ± 0.0	45.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	77.0 ± 1.6	100	0	100	100	100	0
LA5	0.0 ± 0.0	29.0 ± 0.0	0.0 ± 0.0	4.6 ± 4.6	0.0 ± 0.0	64.2 ± 2.1	100	0	100	30	100	0
LA6	0.0 ± 0.0	34.0 ± 0.0	0.0 ± 0.0	5.8 ± 5.8	0.0 ± 0.0	64.9 ± 1.0	100	0	100	40	100	0
LA7	0.0 ± 0.0	17.0 ± 0.0	0.0 ± 0.0	6.1 ± 4.1	0.0 ± 0.0	58.1 ± 1.6	100	0	100	10	100	0
LB0	1.0 ± 0.0	67.0 ± 0.0	3.8 ± 3.0	1.0 ± 1.4	6.6 ± 3.2	61.5 ± 1.5	0	0	20	60	0	0
LB1	0.0 ± 0.0	67.0 ± 0.0	2.4 ± 1.6	13.6 ± 7.3	0.0 ± 0.0	68.1 ± 1.0	100	0	10	10	100	0
LB2	1.0 ± 0.0	62.0 ± 0.0	2.8 ± 1.7	8.6 ± 6.3	2.3 ± 2.4	56.0 ± 2.0	0	0	10	10	40	0
LB3	0.0 ± 0.0	58.0 ± 0.0	1.9 ± 4.3	11.4 ± 5.1	1.1 ± 2.3	52.5 ± 1.4	100	0	80	0	80	0
LB4	0.0 ± 0.0	68.0 ± 0.0	6.4 ± 3.0	6.7 ± 1.4	7.6 ± 5.1	64.2 ± 3.6	100	0	0	0	10	0
LB5	0.0 ± 0.0	58.0 ± 0.0	3.9 ± 2.9	12.2 ± 4.7	3.8 ± 3.6	52.3 ± 1.6	100	0	0	0	30	0
LB6	0.0 ± 0.0	59.0 ± 0.0	4.4 ± 3.8	12.8 ± 4.8	5.2 ± 4.5	54.5 ± 1.5	100	0	20	0	30	0
LB7	0.0 ± 0.0	51.0 ± 0.0	1.5 ± 3.1	2.9 ± 2.8	4.5 ± 2.3	48.4 ± 2.0	100	0	50	0	0	0
LA	0.0 ± 0.0	37.6 ± 11.3	0.2 ± 0.5	3.8 ± 4.9	0.0 ± 0.0	65.8 ± 6.5	100.0	0.0	86.2	47.5	100.0	0.0
LB	0.2 ± 0.4	61.2 ± 5.9	3.4 ± 3.3	8.6 ± 6.3	3.9 ± 4.0	57.2 ± 6.6	75.0	0.0	23.8	10.0	36.2	0.0

3.4.8 Discussion

This chapter presented an extensive evaluation of the algorithms proposed in this work, including a beam search algorithm (IBS), a new MIP formulation (MIP), and a cut-based heuristic (CUTH). These were compared against methods from the literature, namely logic-based Benders decomposition (LBBD) and iterated local search (ILS), using a standard set of benchmark instances.

Among all algorithms, IBS achieved the best overall performance, often finding optimal solutions quickly and in most replications. In Sections 3.4.3 and 3.4.4, we examined the impact of its main components, including the transition instant and the heuristic rules used to expand an allocation. The results showed that beam search performed similarly across different values of the transition instant, with $\hat{p} = 0.4$ giving slightly better outcomes in the tested instances. The fire perimeter heuristic improved the frequency of optimal solutions, although the algorithm remained effective without it. In contrast, the bias towards vertices adjacent to protected vertices was essential. Removing this bias led to a clear drop in solution quality.

The MIP formulation introduced in Section 3.1 demonstrated a significant improvement over the formulation proposed by Alvelos (2018). Specifically, the proposed model consistently found feasible solutions for all instances within the time limit, whereas the previous model failed to do so in many runs (up to 50% failure rate in the LB group). Furthermore, the proposed model achieved solution quality very close to optimal (average δ of 0.2–3.4), unlike the extremely poor solutions obtained with the previous model (average δ often exceeding 50). While the previous model never found an optimal solution, the proposed formulation found the optimal solution in about half of the runs. However, large MIP and root gaps indicate that the formulation can still be improved, especially to help the solver prove optimality.

The cut-based heuristic had the worst performance in terms of solution quality, particularly in instances where the delay Δ is small. On the other hand, it had the shortest running times, often completing in under 4 seconds. As we will see in Chapter 4, the cut-based heuristic becomes more competitive in larger instances, where it can find solutions of similar quality to those obtained by IBS and better than those obtained by LBBD and ILS.

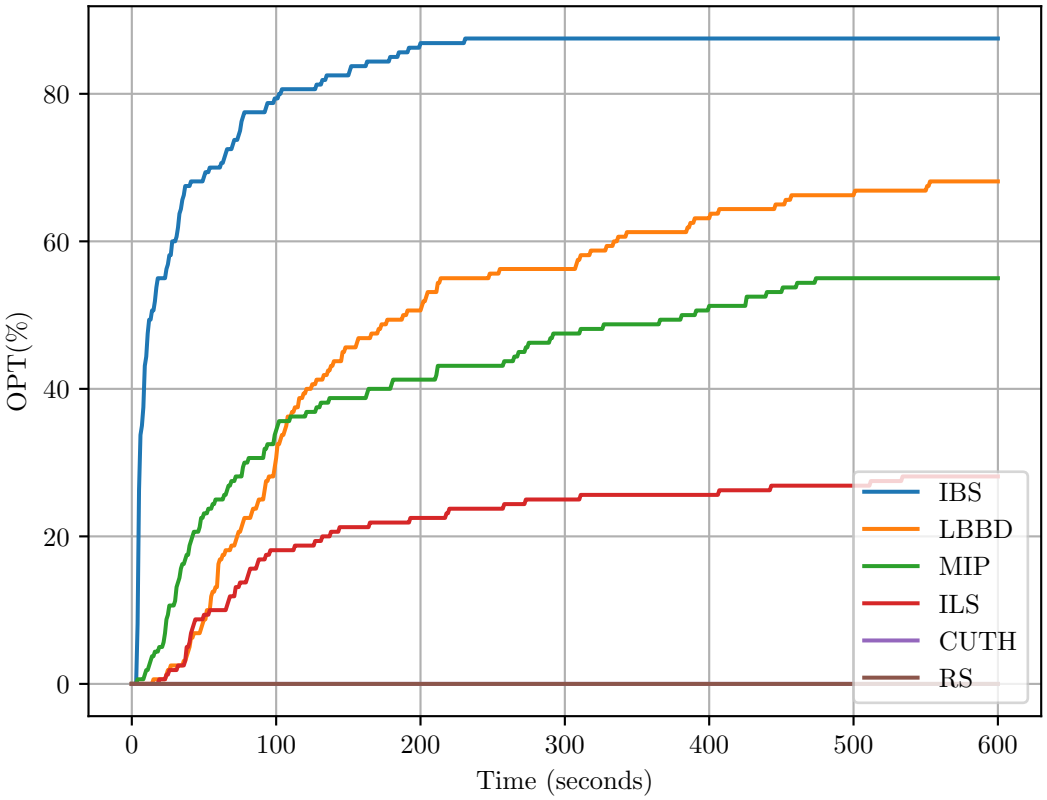


Figure 3.4 – Performance profile considering all 160 executions (16 instances and 10 replications). The x-axis shows time in seconds and the y-axis shows the percentage of the runs in which the algorithm found the optimal solution within that time.

4 A NEW INSTANCE GENERATOR

As we saw in Section 3.4, existing instances in the literature do not present a challenge to current methods. The beam search algorithm proposed in this dissertation, for example, finds the optimal solution of almost all instances in less than five minutes and, as reported by Harris et al. (2023), the logic-based Benders decomposition is able to prove optimality given a few hours of computation.

Besides the fact that the existing instances are not challenging, they also have other drawbacks, which we summarize below:

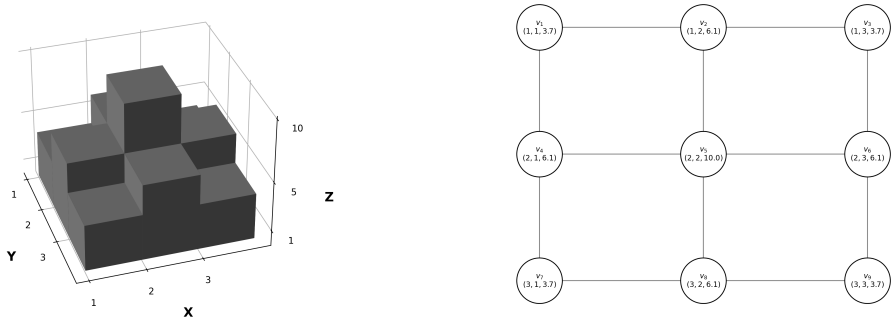
- Lack of spatial correlation. Fire travel times have been assigned randomly and independently to each vertex, ignoring the spatial correlations expected in real fire spread dynamics.
- No physical interpretation. Numerical parameters, such as fire travel times and the optimization horizon, lack a clear physical basis, making it difficult to connect computational results with real-world wildfire suppression scenarios.
- Unreachable vertices. The earliest arrival time to some vertices is after the optimization horizon, allowing them to be pre-processed out of the problem, which reduces the complexity of the instances.
- Absence of a systematic generation process. There is no well-defined method for creating new instances of varying difficulty.

To address these issues, we propose a new instance generation process based on Rothermel's fire spread model (see Section 2.3), which provides a physically meaningful computation of fire travel times. Our proposed model supports flexible configurations, including different grid sizes, delay values, and resource availability, allowing for controlled variation in problem complexity. In Subsection 4.2, we will use this model to study how the algorithms discussed in this dissertation behave on instances of varying difficulty.

4.1 Instance Model

Landscape

We consider a landscape represented as a two-dimensional square grid of cells, where each cell has an associated height. Figure 4.1a shows an example. This landscape



(a) Landscape. (b) Graph representation.

Figure 4.1 – Visualization of a discretized 3x3 landscape (left) and its graph representation (right).

Table 4.1 – Grid sizes and the associated distance between adjacent cells in the xy plane.

Category	$n \times n$	Distance d in the xy plane
Small	20×20	1312 ft (≈ 400 m)
Medium	30×30	875 ft (≈ 266 m)
Large	40×40	656 ft (≈ 200 m)
Huge	80×80	328 ft (≈ 100 m)

is encoded by a graph $G = (V, A)$, where each vertex corresponds to a cell, and arcs represent adjacencies between neighboring cells. Neighborhood relations are defined in the xy-plane, and each vertex can have up to four neighbors: two horizontal (left and right) and two vertical (up and down). For any vertex $v \in V$, we denote its spatial position as the tuple $(v_x, v_y, v_z) \in \mathbb{R}^3$, where v_x and v_y are the coordinates in xy-plane and v_z is the height. Figure 4.1b illustrates that. The Euclidean distance between any two vertices $u, v \in V$ in the three-dimensional space is denoted by $d(u, v)$.

We define three different grid sizes and each grid represents a square landscape of approximately 15806 acres ($\approx$ 64 km^2). Table 4.1 lists the distances between two adjacent cells in the xy-plane for each grid size. Since all landscapes cover approximately the same area, the grid size determines the resolution of the landscape. In all instances, the fire starts from a single ignition vertex located at the center of the grid.

Slope

As discussed in Section 2.3 and illustrated by Figure 2.3, the slope factor Φ_s in Rothermel's model depends on the tangent of the slope angle ϕ. More specifically, given

two adjacent vertices $u, v \in V$, the tangent of the slope from u to v is

$$A_{uv} = \frac{v_z - u_z}{d},$$

where d is the distance between u and v in the xy-plane.

While the x and y coordinates of each vertex are determined by the grid size, the z coordinate is generated using Perlin noise (Perlin, 2002). Perlin noise is a gradient-based noise function commonly used in procedural terrain generation to produce natural-looking variations in elevation. In our instance generation process, each vertex $v \in V$ is assigned a Perlin noise value P_v in the range $[0, 1]$, which influences its elevation. The noise values are then mapped to the interval $[0, h_{max}]$ feet, where h_{max} is the maximum value for the height. The greater the value of h_{max}, the more stretched the elevation values become, resulting in steeper slopes. We use the value of h_{max} to define slope categories, which are shown in Table 4.2.

To define the slope categories, we considered the maximum slope angle that could arise under a linear elevation across the landscape. Specifically, we assumed that elevation increases uniformly from one corner of the grid to the adjacent corner along one side. Although these two corner vertices are not adjacent, the resulting slope angle serves as a proxy for the steepest slopes we expect to observe between adjacent vertices. Let us call this angle θ. Since all landscapes span the same physical width (approximately 26240 ft), the maximum slope angle computed between two opposite corners of the grid along one side depends only on the elevation range defined by the slope category, and not on the grid size. For a given category with maximum height h_{max}, the slope angle is given by $\theta = \arctan(h_{max}/26240)$. Using the values $h_{max} = 6560$, 13120, and 26240 ft for the Flat, Moderate, and Steep categories, respectively, we obtain slope angles of approximately 14.5°, 27.3°, and 45.0°. These angle values correspond closely to widely adopted slope classifications in wildfire behavior literature. For example, training materials published by the National Wildfire Coordinating Group, a United States organization that creates training materials for agencies that manage wildfires, describe slopes around 0° as flat, 20° as moderate, and 30° as steep (National Wildfire Coordinating Group, 2024).

The slope factor Φ_s in Rothermel's model depends not only on the tangent of the slope angle A_{uv}, but also on the packing ratio β. Using the data compiled by Alvelos (2018) on 53 fuel models, we built a histogram of the values of β, depicted in Figure 4.2. As shown in the figure, small values of β, particularly in the range near 0.005, are common across the dataset. For the remainder of this work, we assume $\beta = 0.005$ as a

Table 4.2 – Slope categories and their associated maximum height h_{max}.

Category	h_{max} (feet)
Flat	6560
Moderate	13120
Steep	26240

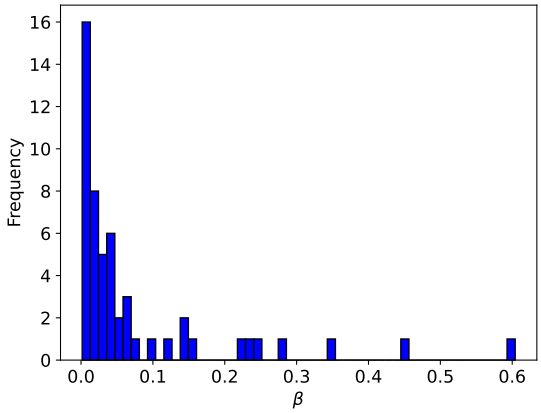


Figure 4.2 – Histogram of values for β considering the 53 fuel models listed by Andrews (2018).

representative value when computing $\Phi_s(A_{uv})$.

Wind

The problem defined in Section 2.5.1.1 assumes static fire travel times, meaning wind effects remain constant over time rather than varying dynamically. To incorporate wind influence, we define a time-invariant wind field. This field is characterized by a predominant wind direction, represented by a unit vector $\mathbf{w} \in \mathbb{R}^2$. To introduce local variations, we assign a wind velocity vector $\mathbf{w}_{uv}$ to each pair of adjacent vertices $u, v \in V$. This vector accounts for wind conditions around the corresponding grid cells and is derived by perturbing the primary wind direction $\mathbf{w}$ in both magnitude and orientation. Figure 4.3 illustrates an example wind field for the small grid introduced in Figure 4.1.

To introduce variability, we apply two Perlin noise functions to the wind vector $\mathbf{w}$. The first noise, mapped to the interval $[-\frac{\pi}{4}, \frac{\pi}{4}]$, introduces directional deviations from the main wind direction. The second noise determines wind speed and is mapped to an interval based on the wind category.

To define the wind categories that determine the magnitude of the local wind vectors $\mathbf{w}_{uv}$, we employ the Modified Beaufort Scale (Encyclopaedia Britannica, 2025). This scale provides a standardized approach to classify the standard 20-foot wind speeds into

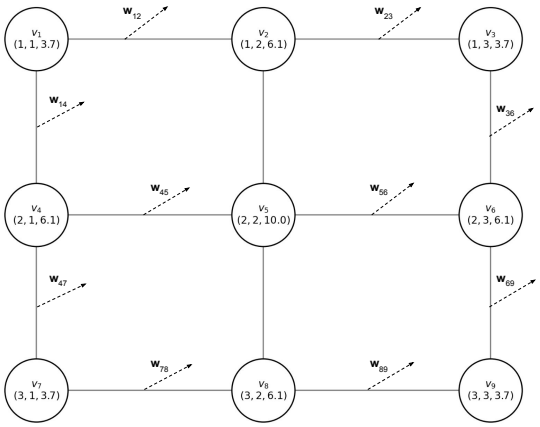


Figure 4.3 – Illustration of the wind field used in the instance model. Each vector $\mathbf{w}_{uv}$ represents the wind velocity between adjacent vertices u and v, incorporating both directional and magnitude perturbations relative to the predominant wind direction. These variations are introduced using Perlin noise.

Table 4.3 – Wind categories based on wind speed.

Category	Beaufort Scale (ft/min)	Adjusted Wind Speed (ft/min)
Light	[315, 650]	[94.5, 195.0]
Moderate	[1083, 1555]	[324.9, 466.5]
Strong	[2126, 2717]	[637.8, 815.1]

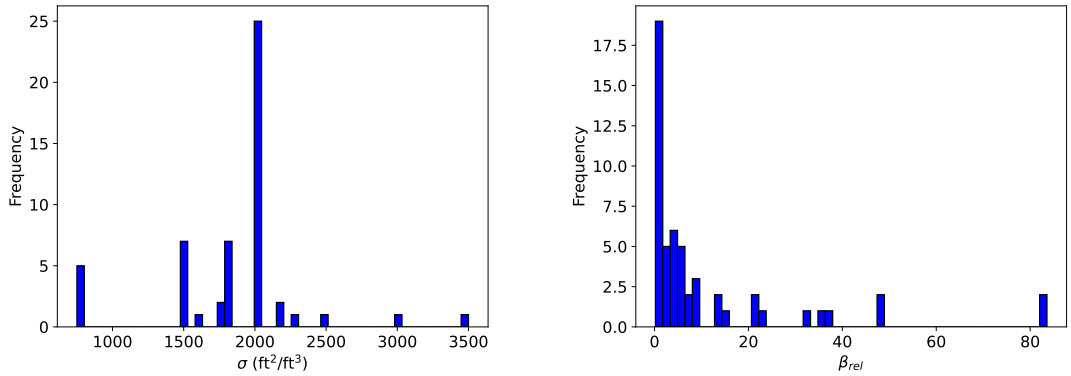
discrete ranges (e.g., ‘Light’, ‘Moderate’, ‘Strong’). Since Rothermel’s model requires the midflame wind speed as input, we convert the 20-foot wind speed ranges defined by the Beaufort scale using a wind adjustment factor (WAF) of 0.3. This specific WAF is selected as it represents a common value used to estimate midflame wind speed under many typical field conditions (Andrews, 2012). Table 4.3 presents the selected wind categories, their corresponding 20-foot wind speed ranges based on the Beaufort scale, and the resulting adjusted midflame wind speed ranges used in our instance generation.

For a given pair of adjacent vertices, let π_{uv} represent the rotational perturbation and s_{uv} the wind speed scaling factor. The resulting wind velocity vector is computed as follows

$$\mathbf{w}_{uv} = s_{uv} R(\pi_{uv}) \mathbf{w},$$

where $R(\pi_{uv})$ is the rotation matrix that rotates the vector $\mathbf{w}$ by an angle π_{uv}. Since wind conditions are assumed to be symmetric between adjacent vertices, we enforce $\mathbf{w}_{uv} = \mathbf{w}_{vu}$, which implies $s_{uv} = s_{vu}$ and $\pi_{uv} = \pi_{vu}$.

For each pair of adjacent vertices $u, v \in V$, let $\mathbf{n}_{uv} \in \mathbb{R}^2$ denote the unit direction



(a) Histogram of surface-area-to-volume ratio σ. (b) Histogram of relative packing ratio β_{rel}.

Figure 4.4 – Histogram of values for σ and β_{rel} considering the 53 fuel models compiled by Andrews (2018).

vector from u to v in the xy-plane. The wind speed component along this direction is given by

$$U_{uv} = \mathbf{w}_{uv}^T \mathbf{n}_{uv}.$$

As with the slope factor, the wind factor Φ_w in Rothermel's model depends not only on wind speed U_{uv}, but also on the surface-area-to-volume ratio σ and the relative packing ratio β_{rel}. To choose appropriate values for these parameters, we analyzed the 53 fuel models compiled by Andrews (2018). Figure 4.4 shows the distribution of σ and β_{rel} across the dataset. The histogram for σ (Figure 4.4a) shows a clear mode at 2000 ft^2/ft^3, and the histogram for β_{rel} (Figure 4.4b) indicates that most values lie close to 1. Based on this, we set $\sigma = 2000$ ft^2/ft^3 and $\beta_{rel} = 1$ in all our computations of the wind factor $\Phi_w(U_{uv})$.

Fire Propagation

Having defined the landscape, and the slope and wind components of our instance model, we now turn to the computation of fire spread velocity and propagation time.

Rothermel's model defines a rate of spread assuming that the parameters related to the fuel bed and the environment are static. As a result, we assume that the composition of the fuel inside each cell is homogeneous in the sense that the no-wind, no-slope rate of spread remains constant within the cell. For each vertex $v \in V$, then, we define $R_0(v)$ as the base fire spread rate in the absence of other factors. Similar to our approach for wind and slope, we built a histogram for the value of R_0 considering the 53 fuel models

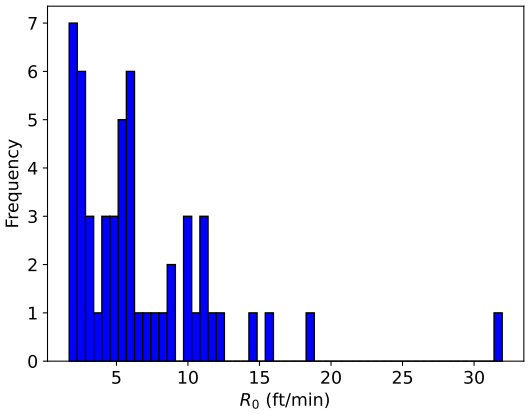


Figure 4.5 – Histogram of values for R_0 considering the 53 fuel models compiled by Andrews (2018).

compiled by Andrews (2018), depicted in Figure 4.5. As the data shows, most values are concentrated between 1 ft/min and 20 ft/min. We generate values of $R_0(v)$ using Perlin noise, mapping the noise to the range $[1, 20]$ ft/min.

Recall from Section 2.3 that the rate of spread of a fire, according to Rothermel's model with Albini's extensions, depends on whether the wind blows with or against the direction of fire propagation, and whether the fire moves upslope or downslope. These two factors (wind and slope) affect the flame tilt and thus the amount of heat transferred to unburned fuel. In our model, this is captured by examining the signs of U_{uv} and A_{uv}. When $U_{uv} \geq 0$, the fire spreads with the wind (headfire). When $U_{uv} < 0$, it spreads against the wind (backfire). Likewise, $A_{uv} \geq 0$ corresponds to upslope spread, while $A_{uv} < 0$ corresponds to downslope spread. The rate of spread $R(u; \mathbf{n}_{uv})$ inside cell u in the direction of $\mathbf{n}_{uv}$ is given by

$$R(u; \mathbf{n}_{uv}) = \begin{cases} R_0(u)(1 + \Phi_w(U_{uv}) + \Phi_s(A_{uv})) & \text{if } A_{uv} \geq 0, U_{uv} \geq 0 \\ R_0(u)(1 + \max\{0, \Phi_w(U_{uv}) - \Phi_s(A_{uv})\}) & \text{if } A_{uv} < 0, U_{uv} \geq 0 \\ R_0(u)(1 + \max\{0, \Phi_s(A_{uv}) - \Phi_w(|U_{uv}|)\}) & \text{if } A_{uv} \geq 0, U_{uv} < 0 \\ R_0(u) & \text{if } A_{uv} < 0, U_{uv} < 0 \end{cases}$$

To determine the fire propagation time from the center of cell u to the center of cell v, we compute the harmonic mean of the fire spread velocities in the given direction. For any arc $uv \in A$, then, the fire propagation time t_{uv} is computed as

$$t_{uv} = d(u, v) \frac{R(u; \mathbf{n}_{uv}) + R(v; \mathbf{n}_{uv})}{2R(u; \mathbf{n}_{uv})R(v; \mathbf{n}_{uv})}.$$

Optimization Horizon

Previous work has defined instances in an ad hoc manner, often specifying the optimization horizon H as an absolute value (e.g. $H = 70$ in the instances used in Section 3.4). This approach can lead to inconsistencies in instance difficulty. As noted by Harris et al. (2023), in the 20×20 grid instances generated by Mendes & Alvelos (2023), only a small subset of the 400 vertices burned before H. As a result, most vertices can be removed from the instance since they never contribute to the objective function. The instances proposed by Harris et al. (2023), which we used in our experiments in Section 3.4, suffer from the same issue to a lesser extent.

To address this issue, we define the optimization horizon relative to the fire spread velocity rather than as an arbitrary fixed value. More specifically, we use the free burning time, which represents the time at which the last vertex burns when no suppression resources are deployed. Formally, the free burning time of an instance is given by

$$t_{\text{free}} = \max\{a_v^{\Lambda_0} : v \in V\},$$

where $a_v^{\Lambda_0}$ denotes the fire arrival time at vertex v under the assumption that no resources are allocated. We then define the optimization horizon as

$$H = 1.1 t_{\text{free}}.$$

A multiplier less than or equal to 1 would result in some vertices never burning, allowing them to be trivially removed from the instance. Setting H slightly beyond t_{free} ensures that all vertices remain relevant during the optimization. The value of 1.1 was chosen arbitrarily.

Resources

With the landscape, the fire propagation model, and the optimization horizon described in the previous sections, we now define the fire suppression resources. Specifically, we must determine the number of available resources k, the sequence of times $T = (t_1, t_2, \ldots, t_{|T|})$ at which resources become available, and, for each $t_i \in T$, the subset $R_t \subseteq [k]$ of resources released at that time. Additionally, we must specify the delay Δ

Table 4.4 – Delay values for resources.

Category	Δ
Low	$H/3$
Medium	$H/2$
High	H

Table 4.5 – Number of resources based on grid size n.

Category	k
Few	$n/2$
Moderate	n
Many	$2n$

imposed by a deployed resource.

To ensure consistency across different instances, we define the delay values relative to the optimization horizon H, similar to how H is defined in terms of the free burning time. Table 4.4 presents the three possible delay categories. When the delay is high ($\Delta = H$), a resource completely blocks fire spread through its outgoing arcs, since any fire arrival time passing through a protected vertex will be at least H. For low and medium delay values, resources do not fully block fire spread but still significantly slow it down.

The number of available resources k is defined based on the grid size n. Table 4.5 shows the number of available resources for each grid size. An instance with grid size 20×20 and many resources, for example, has 40 available resources. Our decision to define the number of available resources relative to the dimension of the grid is based on the observation that, on planar graphs, optimal allocations of resources often form fire suppression lines. In instances with a moderate amount of resources, then, there are enough resources to construct a firebreak that spans the dimension of the landscape. In contrast, an instance with few resources may require a different allocation strategy.

We now turn our attention to the sequence of times T at which resources become available. We will start by defining the size of T, i.e., the number of decision points where resources are released. This quantity is important because it is directly related to size of the MIP model defined in Section 3.1 and with the master problem of the Benders decomposition proposed by Harris et al. (2023) (see Section 2.5.3.2). It also influences the depth of the search tree of the beam search we proposed in Section 3.2.

For each possible number of decision points $|T|$, the total number of available resources k is distributed as evenly as possible across the time instants $t \in T$. Since k

Table 4.6 – Number of decision points.

Category	$\|T\|$
Few	5
Moderate	10
Many	20

may not be divisible by $|T|$, an equal division is not always possible. In such cases, we assign one resource to each time instant until all k resources are allocated. This process starts with a vector of $|T|$ ones and repeatedly increments elements from left to right in a cyclic manner until the total sum reaches k. To avoid biasing instances towards higher suppression capacity in earlier time instants, the resulting allocation vector is randomly shuffled. For example, if $k = 5$ and $|T| = 3$, the resulting (unshuffled) allocation is $[2, 2, 1]$. It is also possible that the number of resources is less than the number of decision points, i.e., $k < |T|$. In this case, we randomly select k decision points from $|T|$ with uniform probability and assign one resource to each of them. The remaining decision points will not have any resources available.

Lastly, we define the release times T. As in previous sections, T is specified relative to a quantity that captures the progression of the fire. More specifically, release times are defined in terms of the percentage of the landscape that has burned by a given time. This allows us to better understand the state of the wildfire when suppression resources arrive. By anchoring the first release time to a burn percentage, we want to avoid to extremes: releasing resources too early, when the fire perimeter is small enough to be easily contained, or too late, when most of the landscape has already burned and the remaining problem becomes trivial.

Recall from Section 2.5.1.2 that $B_t^{\Lambda_0}$ denotes the set of vertices that are burned at time t considering a free burning process. We define $q(p)$ as the time when $p\%$ of the vertices burned under a free burning process

$$q(p) = \sup \left\{ t : 100 \frac{|B_t^{\Lambda_0}|}{|V|} \leq p \right\}.$$

Decision points are equally spaced over the time interval from the first to the last release time, both of which are specified via $q(p)$ for different values of p. Tables 4.7 and 4.8 summarize the possible configurations.

We illustrate how release times are defined with an example. Consider an instance with few decision points (Table 4.6), an early first release time (Table 4.7) and a late last

Table 4.7 – First release times.

Category	First Release Time
Early	$q(5)$
Late	$q(10)$
Very late	$q(20)$

Table 4.8 – Last release times.

Category	Last Release Time
Very early	$q(60)$
Early	$q(70)$
Late	$q(80)$
Very late	$q(95)$

release time (Table 4.8). Such instance has five equally-spaced decision points starting from $t_1 = q(5)$ and ending at $t_5 = q(80)$, i.e.

$$T = \left(t_1, \frac{3t_1 + t_5}{4}, \frac{t_1 + t_5}{2}, \frac{t_1 + 3t_5}{4}, t_5 \right).$$

4.2 Experiments

In this section, we use the proposed instance generator to assess how different experimental factors affect the performance of the algorithms. Our goal is to understand under which conditions each algorithm performs best, how they compare to each other in terms of solution quality, and how good are the solution they produce in terms of fire suppression. Subsection 4.2.1 describes the experimental setup, including default values for each experimental factor and the parameter settings used by the algorithms. In the following subsections, we analyze the results of our experiments, focusing on how instance size, suppression capacity, environmental factors, and the resource arrival window influence algorithmic performance. We conclude in Subsection 4.3.

4.2.1 Experimental Setup

The instance generator defines eight experimental factors, which are listed in Table 4.10. In our experiments, we will study the effect of experimental factors in pairs, meaning that we will vary two factors at a time while keeping the others fixed. To this end, we organize the eight factors into four groups, as depicted in Table 4.10. Our rationale for this grouping is that factors in the same group are likely to affect algorithm behavior in similar ways. For example, the grid size and the number of decision points are both related to the size of the models solved by MIP and LBBD, while environmental factors like wind and slope are related to the velocity with which fire spreads.

Regarding the default parameter values, we use 30×30 grids as our default because, for larger grids, algorithms like LBBD and ILS typically fail to make progress within reasonable time limits, making it difficult to evaluate how their performance is affected by other experimental factors. The default number of decision points is set to 10, following the study of Avci et al. (2024), who modeled real-world suppression efforts with around 10 decisions over a 10-hour horizon. Since our test instances often span 24 to 36 hours, using 10 to 20 decision points is reasonable. We set the delay to high, meaning that suppression resources completely block fire spread at the protected vertex.[1] Estimating the real delay caused by a suppression resource is outside the scope of this

[1] When delay is set to high, we have $\Delta = H$. Any vertex that has a predecessor protected by a resource will have its arrival time greater than H, hence it will not burn. In practice, we can think of this as a complete block of fire spread.

Table 4.9 – Time limits for the different grid sizes. The time limit is set to 1.5 seconds per grid cell.

Grid Size	Time Limit (s)
20×20	600
30×30	1350
40×40	2400
80×80	9600

Table 4.10 – Default parameters for each experimental factor.

Group	Parameter	Default Value
Instance Size	Grid Size	30×30
	Decision points	Medium
Suppression Capacity	Delay	High
	# of resources	Moderate
Environment	Slope	Moderate
	Wind	Light
Release Window	First release time	Early
	Last release time	Very late

dissertation. The number of resources is set to moderate, which means that it is equal to the dimension of the grid (i.e. n resources for an $n \times n$ grid). We have chosen this level as default because, as we will see in Subsection 4.2.3, in scenarios with few resources the effect of other experimental factors is weak. The slope and wind are set to moderate values to avoid extreme fire spread velocities in the baseline case. The effect of extreme velocities is analyzed separately in later sections. Finally, the first release time is set to early, meaning that only a few of the vertices burned when the first batch of resources is released. The last release time is set to very late, meaning that resources are available until the end of the time horizon.

The algorithm parameters are the same as those used in Section 3.4, except for time limits. Since the new instances include larger grids, we scale the time limit with grid size: we allocate 1.5 seconds per grid cell. This results in the limits shown in Table 4.9. All experiments were run on the same hardware and software environment as in Section 3.4.

We aim to answer two questions in this section: (i) how do the algorithms compare in terms of solution quality? and (ii) how effective are their solutions in suppressing fire spread? For this purpose, the following performance metrics are used:

- **Relative Gap to the Best-Known Value (BKV):** The difference between a solu-

tion's objective value and the best value obtained across all algorithms, divided by the best value.

- **Vertices Saved (%):** The percentage of grid vertices that do not burn in the best-known solution.

To be more precise, for any algorithm A and instance I, let $A(I)$ be the objective value obtained by the algorithm and $A^\star(I)$ be the best objective value known for instance I. Also, let $V(I)$ be the number of vertices in instance I. The relative gap to the best-known value is defined as $\frac{A(I)-A^\star(I)}{A^\star(I)}$. The percentage of vertices saved by the best-known solution is $1 - \frac{A^\star(I)}{V(I)}$.

4.2.2 Instance Size

These two factors are related in the sense that they affect the number of variables and constraints of the models solved by MIP, LBBD, and CUTH. They also affect the size of the fire perimeter of IBS, and the depth of the search tree. The instance generator defines four levels for grid size, namely small (20×20), medium (30×30), large (40×40), and huge (80×80), and three levels for the number of decision points, namely few (5), medium (10), and many (20). We run 10 replications per instance for grid sizes from 20×20 to 40×40, and 3 replications for 80×80. For IBS, we report results for different values of transition instants $\hat{p}$, since in the experiments of Section 3.4 the results were similar for all values of $\hat{p}$.

Table 4.11 shows the results of the experiment. The first group of columns shows the average relative gap to the best-known value (BKV) obtained across all replications. In parentheses, we also report the number of replications in which the BKV was reached. The second group of columns shows the percentage of vertices that do not burn in the best-known solution. The best result for each instance is highlighted in bold.

We start by analyzing how the algorithms compare to each other in terms of solution quality. The best average gaps are obtained by IBS with $\hat{p} = 1$ (or $\hat{p} = 0.8$, which produces the same results).[2] In 7 out of 12 instances, IBS with $\hat{p} \in \{0.8, 1\}$ matches or outperforms all other methods. This includes all cases with 20 decision points and half the cases with 10 decision points. In fact, for instances with 20 decision points and grid sizes up to 40×40, beam search reached the BKV in all 10 replications.

[2]The results for $\hat{p} = 0.8$ and $\hat{p} = 1$ are equal because in both cases the transition instant occurs after the last release time, which implies that heuristic h_2 is used in all decision points except the last one.

In contrast to the experiments of Section 3.4, where all values of $\hat{p}$ produced similar results, Table 4.11 shows that the choice of $\hat{p}$ has a clear impact when the instances are more complex. Averaged over all 12 instances, the average relative gap drops from 31.0% for $\hat{p} = 0$ to 12.5% for $\hat{p} = 0.4$, and further down to 6.25% for $\hat{p} = 1$. For the next experiments, we will use $\hat{p} = 1$ as the default value for IBS.

In Section 3.4, when we compared the algorithms on the instances of Harris et al. (2023), MIP was among the best performing methods. Here we see a similar picture, with MIP ranking first in two instances and reaching the BKV at least once in five instances. Compared against LBBD, which is also an exact method, MIP obtains a smaller gap in all 12 configurations. Compared against IBS, MIP has a better gap in three instances with 5 decision points (grids 30×30, 40×40, and 80×80), and in one instance with 10 decision points (grid 80×80). MIP performs better than IBS specially in instances with 5 decision points, but the raw data suggests this is due more to a drop in the performance of beam search when the number of decision points is small than to any improvement of MIP on such instances. We will discuss this effect at the end of this section.

The experiments on the instances of Harris et al. (2023) in Section 3.4 showed that CUTH ranked among the worst algorithms. Table 4.11 presents a different picture. In the 80×80 grid, CUTH performs well compared to other algorithms, obtaining the best average gap in 2 out of 3 cases. For instances with 5 and 10 decision points, MIP comes close in second, with gaps of 3.3% and 3.2%. Even though the average gap of CUTH and MIP are close, it takes a lot longer for MIP to match the best solution found by CUTH. For example, in the 80×80 grid with 5 decision points, MIP finds a better solution in only 1 out of 3 replications and takes 2325 seconds to do so, while CUTH reaches its best solution in just 2 seconds. In fact, the dominance of CUTH over other algorithms is more pronounced if we consider shorter time limits. To quantify this difference, we checked the best solution found by each algorithm within the first 300 seconds of execution. In the 80×80 grid, the average gap of CUTH in that window is 11.1%, compared to 18.1% for MIP. In the 40×40 grid, CUTH also performs better, with an average gap of 25.9%, while MIP reaches 32.5%. For smaller instances, MIP clearly outperforms CUTH in the 20×20 grid (21.5% vs. 32.5%), and has a slight advantage in the 30×30 grid as well (32.6% vs. 33.9%).

Aside from CUTH ranking higher than in previous experiments, another difference is how ILS and LBBD behave compared to earlier results. The experiments of Section 3.4 showed that LBBD ranked close to IBS in terms of solution quality, and that

ILS was not the best algorithm, but still was far better than CUTH. The data, however, indicates that ILS and LBBD consistently obtain the worst average gaps. In 8 of the 12 instances, their results are worse than CUTH, and in 7 of them, both methods lose to the random search. For ILS, these results may be explained by the weak implementation available in the literature. The raw data shows that, as the grid size increases, ILS spends almost all of its time constructing the initial solution, which does not allow us to draw conclusions about the performance of the local search and the perturbation components. LBBD, on the other hand, is arguably not affected by the implementation, since most of the time is spent solving the master problem using a commercial solver.

We now take a closer look at the results for IBS. When the number of decision points is small, IBS ranks first only in 20×20, and for larger grid sizes, MIP and CUTH are better alternatives. As the number of decision points increases, however, the performance of IBS improves significantly, while other methods do not benefit as much. This can be seen both in the relative gap and in the number of vertices saved by the best-known solution. For instance, in the 30×30 grid, increasing the number of decision points from 5 to 20 leads to an increase in saved vertices from 25.8% to 37.1%, even though the suppression capacity remains unchanged. This result is somewhat counterintuitive. Adding more decision points means releasing the same number of resources across more time instants, introducing additional constraints into the model. For example, in the 30×30 grid with 5 decision points, the release schedule distributes 6 resources at each decision time. With 10 decision points, only 3 resources are released at a time, and with 20 decision points, only 1 or 2. These more restrictive schedules should, in principle, lead to solutions with higher objective value (fewer vertices saved). However, IBS seems to exploit this finer temporal resolution to find better solutions. Looking at the raw data, this improvement is clear. In the 30×30 grid, IBS finds solutions with objective value 741 when the number of decision points is 5, and improves this to 615 with 20 decision points, more than a 17% gain.

The performance drop of IBS in instances with few decision points may be explained by the randomized approach used to expand allocations. Consider a 30×30 grid with 5 and 10 decision points, and let F_5 and F_{10} be the fire perimeters at the first decision point. Since the first and last release times are fixed, we have $F_{10} \subseteq F_5$. With a moderate number of resources, 6 resources are allocated per decision point in the 5-point case, and 3 in the 10-point case. Assuming $|F_5| = |F_{10}| = 50$, there are nearly 16 million combinations to choose 6 vertices, compared to just 19600 for 3. Since IBS samples

a fixed number of expansions (e.g. 1500 for $|F| = 50$), the chance of sampling a good allocation drops sharply as the number of resources increases. This makes it far more likely for IBS to find high-quality allocations with 3 resources than with 6. Besides, the example assumes that the fire perimeter has the same size in both cases, which is unlikely. In practice, the fire perimeter is larger in the 5-decision-point case, which means that the number of combinations to choose 6 vertices is even larger.

To better understand the differences in solution quality, Figure 4.6 shows the best solutions found by each algorithm in selected instances with different grid sizes and numbers of decision points.

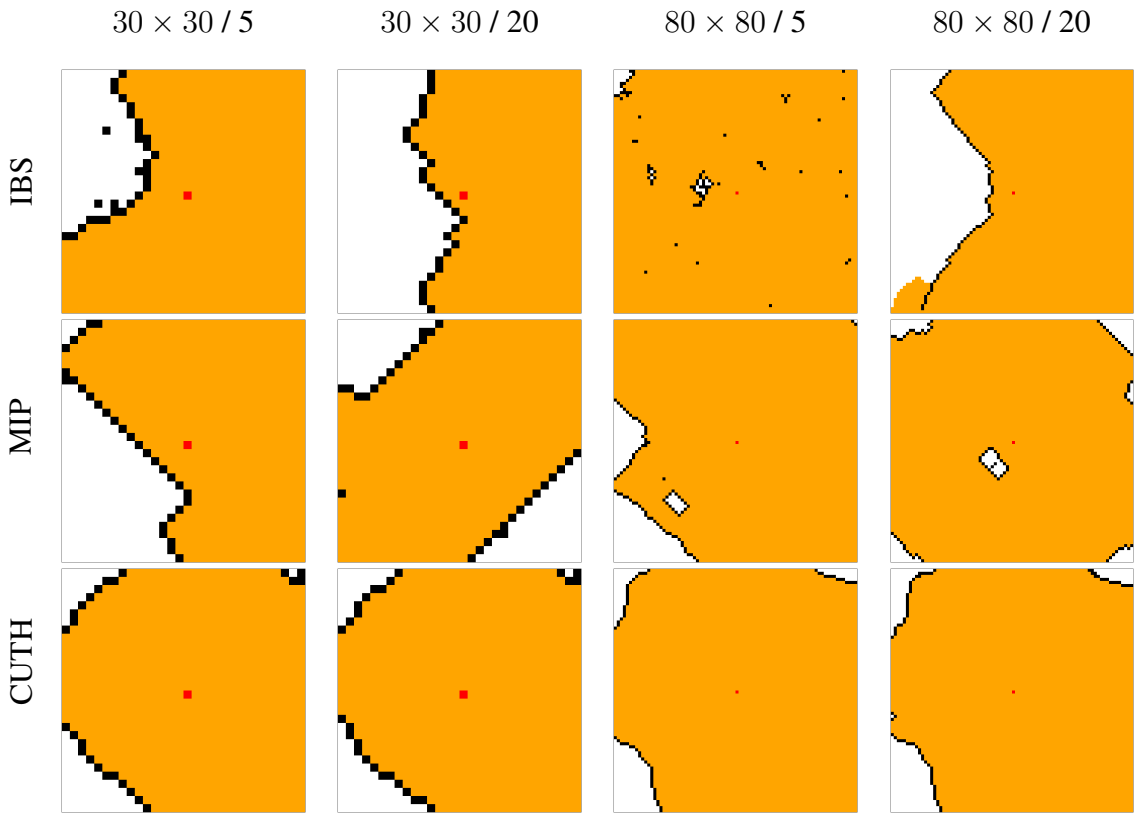


Figure 4.6 – Solutions for a sample of the 12 instances considered in the experiment on instance size. Rows correspond to algorithms (IBS, MIP, CUTH), and columns to instance configurations: (i) 30×30 grid, 5 decision points; (ii) 30×30 grid, 20 decision points; (iii) 80×80 grid, 5 decision points; (iv) 80×80 grid, 20 decision points. Each cell illustrates the best solution found by the algorithm for the corresponding instance. The red cell is the ignition point, the black cells are the ones that are protected by resources, and the orange cells are the ones that burned.

Table 4.11 – Results for different grid sizes and numbers of decision points. The first group of columns reports the average relative gap to the best-known value (BKV) across all replications. In parentheses, we report the number of replications in which the BKV was reached. We performed 10 replications for grid sizes 20×20 to 40×40, and 3 for 80×80. The last column (Saved Vertices, %) shows the percentage of vertices that do not burn in the best-known solution. Best results for each instance are in bold.

		Average Relative Gap to BKV (%)											Saved Vertices (%)
			IBS										
$n \times n$	$\|T\|$	CUTH	0	0.2	0.4	0.6	0.8	1	ILS	LBBD	MIP	RS	
20×20	5	30.9 (0)	17.7 (0)	5.2 (0)	3.8 (0)	3.5 (0)	**3.1** (0)	**3.1** (0)	24.9 (0)	15.9 (0)	10.3 (1)	33.3 (0)	29.5
	10	32.6 (0)	37.4 (0)	14.0 (0)	3.8 (0)	1.8 (0)	**0.0** (10)	**0.0** (10)	18.7 (0)	30.1 (0)	19.3 (1)	38.1 (0)	31.8
	20	34.1 (0)	43.0 (0)	37.2 (0)	8.1 (0)	0.4 (0)	**0.0** (10)	**0.0** (10)	29.1 (0)	36.1 (0)	22.2 (0)	40.4 (0)	32.5
30×30	5	23.4 (0)	25.4 (0)	27.2 (0)	26.1 (0)	17.3 (0)	17.3 (0)	17.3 (0)	29.2 (0)	26.3 (0)	**14.4** (1)	29.6 (0)	25.8
	10	33.7 (0)	38.1 (0)	1.6 (0)	1.3 (0)	1.3 (0)	**0.6** (2)	**0.6** (2)	44.0 (0)	38.7 (0)	26.7 (0)	41.0 (0)	31.7
	20	44.6 (0)	54.1 (0)	4.9 (0)	1.1 (0)	**0.0** (10)	**0.0** (10)	**0.0** (10)	58.5 (0)	58.7 (0)	37.8 (0)	54.1 (0)	37.0
40×40	5	17.1 (0)	24.5 (0)	25.9 (0)	26.3 (0)	26.8 (0)	26.8 (0)	26.8 (0)	30.4 (0)	29.9 (0)	**10.7** (1)	27.1 (0)	23.4
	10	29.7 (0)	38.4 (0)	40.1 (0)	**0.6** (1)	0.7 (0)	1.2 (0)	1.2 (0)	44.4 (0)	44.2 (0)	30.7 (0)	41.2 (0)	30.8
	20	31.0 (0)	42.4 (0)	42.6 (0)	25.9 (0)	0.1 (0)	**0.0** (10)	**0.0** (10)	46.0 (0)	46.1 (0)	32.9 (0)	42.5 (0)	31.6
80×80	5	**2.6** (0)	7.6 (0)	7.7 (0)	7.9 (0)	8.0 (0)	8.1 (0)	8.1 (0)	8.9 (0)	8.9 (0)	3.3 (1)	8.5 (0)	8.2
	10	**0.0** (3)	5.6 (0)	5.7 (0)	6.0 (0)	6.6 (0)	5.7 (0)	5.7 (0)	7.4 (0)	7.3 (0)	3.2 (0)	7.0 (0)	6.9
	20	30.8 (0)	38.4 (0)	38.4 (0)	39.3 (0)	16.1 (1)	**12.2** (0)	**12.2** (0)	40.7 (0)	40.8 (0)	36.8 (0)	40.4 (0)	29.0

4.2.3 Suppression Capacity

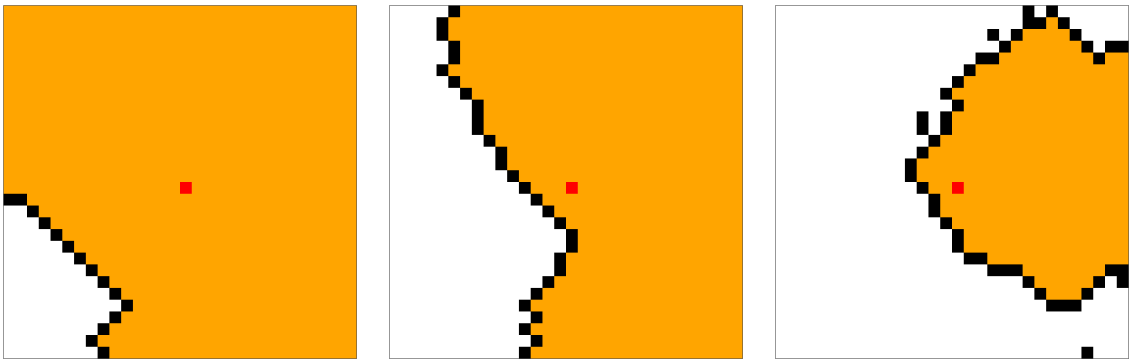
Suppression capacity is related to the number of available resources and the delay they introduce when allocated to a vertex. In the MIP model, varying the delay and the number of resources affects the right-hand side and the matrix of coefficients. For LBBD, lower delay values lead to more variables being introduced in each cut, which increases the size of the master problem. For CUTH, lower delays degrade the assumption that a vertex cut blocks fire propagation entirely. The proposed instance generator defines three levels for the delay (low, medium, and high) and three levels for the number of resources (few, moderate, and many), leaving us with nine instances. As in the previous experiment, we use the default settings for all other experimental factors, vary each suppression parameter independently, and run 10 replications per instance. From now on, we report results for IBS considering $\hat{p} = 1$, since it consistently outperforms the other values in the previous experiment. Table 4.12 shows the results of the experiment.

Let us again start by discussing the overall performance of the algorithms. The results show a clear division in algorithm performance depending on the level of suppression capacity. In all three instances with few resources, CUTH obtains the best average gap. Its performance worsens with higher delay, but it still ranks first. IBS and MIP show higher gaps in these instances. When the number of resources is moderate, IBS performs best in the medium and high delay cases, while MIP leads in the low delay instance. This pattern repeats with many resources: MIP obtains the lowest gap when delay is low, but IBS dominates in the medium and high delay cases.

Among all algorithms, MIP adapts better to variations in suppression capacity, particularly when the number of resources is either very small or very large, and when the delay is low. Compared to CUTH, MIP achieves a lower average gap in 6 out of the 9 instances, and in the remaining 3, the difference is small. When compared to IBS, MIP outperforms it in 5 of the 9 cases. MIP is also the only method to reach the BKV in any instance with low delay or many resources.

In the last section, we saw that IBS outperforms other methods by a large margin on the default instance. It obtained an average gap of 0.6% on the 30×30 grid with 10 decision points, while MIP obtained 26.7% and CUTH 33.7%. As Table 4.12 shows, when there are few resources, IBS is outperformed even by random search. We analyze this further in the next paragraph.

The main assumption behind the fire perimeter heuristic is that suppression efforts



(a) (b) (c)

Figure 4.7 – Illustration of the best-known solution for the three instances with high delay. The red cell is the ignition point, the black cells are the ones that are protected by resources. The orange cells are the ones that burned. (a) Few resources, (b) Moderate resources, and (c) Many resources.

usually are concentrated in cells that are close to the fire. On instances with few resources, however, the best strategy is usually to build a firebreak at some corner of the grid, which is not necessarily close to the fire. This is illustrated by Figure 4.7a, which depicts the best-known solution for the three instances with high delay. In the case with few resources (Figure 4.7a), the firebreak is built in the bottom left corner of the grid, which is far from the ignition point (depicted in red). To construct a solution like this, IBS would have to use a larger fire perimeter, something that could be achieved by increasing the value of the parameter z_{max}. In preliminary experiments, however, we have found that a larger fire perimeter is detrimental to performance in large grids, and the likely reason is the same used to explain the performance drop of IBS in instances with few decision points, which we analyzed in the previous section.

Figure 4.8 provides a visual comparison of the best solutions found by each algorithm in selected instances with different suppression capacities.

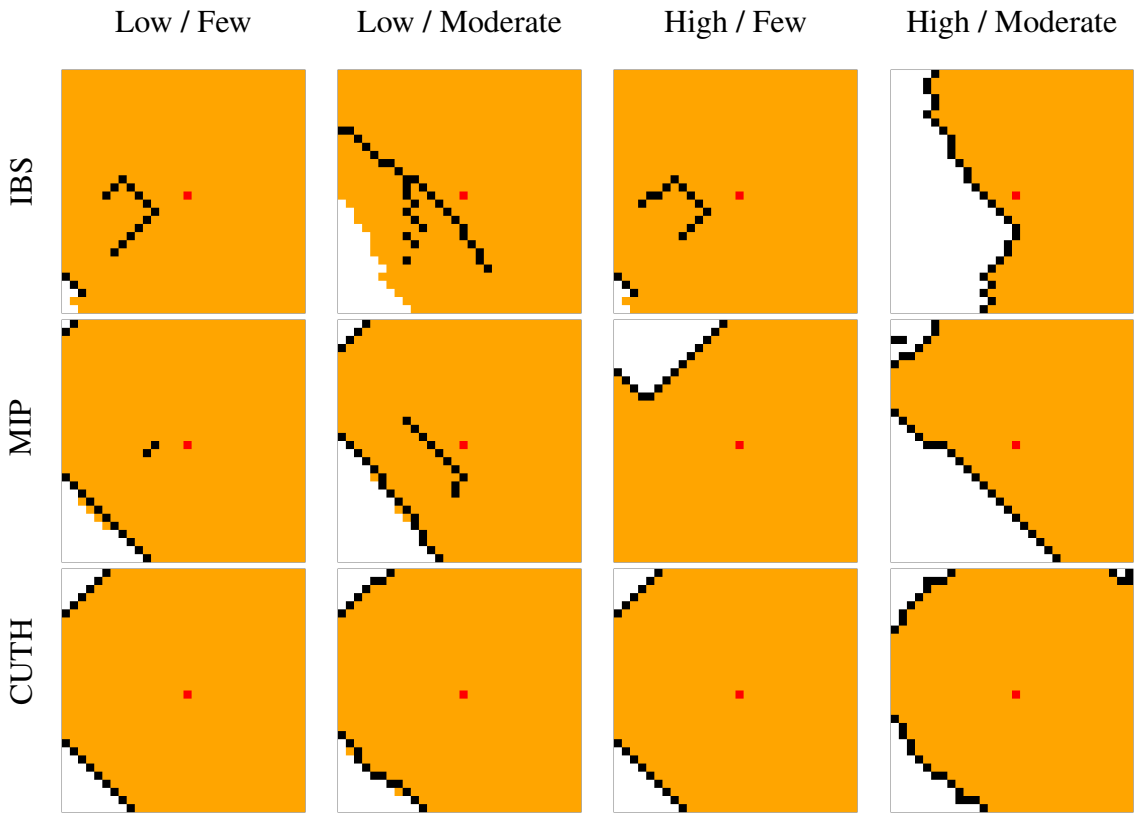


Figure 4.8 – Solutions for a sample of the nine instances. Rows correspond to algorithms (IBS, MIP, CUTH), and columns to instance configurations: (i) Low delay, Few resources; (ii) Low delay, Moderate resources; (iii) High delay, Few resources; (iv) High delay, Moderate resources. Each cell illustrates the best solution found by the algorithm for the corresponding instance. The red cell is the ignition point, the black cells are the ones that are protected by resources, and the orange cells are the ones that burned.

Table 4.12 – Results grouped by delay level and number of available suppression resources. The first group of columns reports the average relative gap to the best-known value (BKV) across all replications. In parentheses, we report the number of replications in which the BKV was reached. We ran 10 replications for each configuration. The last column (Saved Vertices, %) shows the percentage of vertices that do not burn in the best-known solution. Best results for each instance are in bold.

		Average Relative Gap to BKV (%)						Saved Vertices (%)
# of resources	Delay	CUTH	IBS	ILS	LBBD	MIP	RS	
Few	Low	**0.1** (0)	5.4 (0)	4.0 (0)	3.0 (0)	0.5 (2)	4.2 (0)	5.8
	Medium	**0.5** (0)	5.8 (0)	2.9 (1)	4.6 (0)	2.0 (0)	4.6 (0)	6.1
	High	**4.3** (0)	9.8 (0)	6.6 (1)	9.3 (0)	5.5 (0)	8.6 (0)	9.6
Moderate	Low	5.8 (0)	5.5 (0)	12.4 (0)	12.1 (0)	**2.4** (1)	9.2 (0)	11.7
	Medium	10.9 (0)	**0.6** (2)	19.9 (0)	13.3 (0)	8.3 (0)	17.0 (0)	17.7
	High	33.7 (0)	**0.6** (2)	44.0 (0)	38.1 (0)	26.7 (0)	41.0 (0)	31.7
Many	Low	66.7 (0)	10.3 (0)	75.3 (0)	77.4 (0)	**6.4** (1)	69.4 (0)	44.0
	Medium	91.2 (0)	**21.6** (0)	119.6 (0)	120.8 (0)	22.3 (1)	114.1 (0)	55.8
	High	125.1 (0)	**7.8** (0)	157.7 (0)	129.5 (0)	17.0 (1)	152.1 (0)	62.4

4.2.4 Environmental Factors

Before any resource is allocated, the only two experimental factors that affect fire propagation are slope and wind, which will be analyzed in this section. The instance generator defines three levels for slope (flat, moderate, and steep), and three levels for wind (light, moderate, and strong), giving us nine instances. All other parameters are set to their default values. Table 4.14 shows the results, averaged over 10 replications per instance.

Across all instances, IBS clearly dominates the other algorithms. It obtains the lowest average gap in every configuration and is the only method that reaches the best-known value in any of the replications. Moreover, the frequency with which it reaches the BKV increases with slope steepness. For example, IBS reaches the BKV 3 times in the three flat instances, compared to 11 times in the steep ones. We examine this in more detail in the next paragraph.

While the average gap of IBS remains stable across the nine instances, the table shows that, for the other algorithms, it increases as the slope becomes more steep. For example, the average gap of MIP increases from 28.8% (flat) to 32.8% (moderate) and 41.6% (steep). The same trend is observed for CUTH, whose average gap increases from 32.9% (flat) to 38.1% (moderate) and 51.9% (steep). The raw data reveals that IBS finds better solutions as the slope steepens, whereas MIP shows little improvement. Table 4.13 reports the average objective values obtained by IBS and MIP for the different slope levels, and it is clear that the average objective value of IBS is decreasing as the slope increases. The likely reason is that, in instances with higher fire spread velocities (due to strong wind or steep slope), the length between the first and last release time is smaller, which means that the fire perimeter is smaller. As we saw in the last section, IBS benefits from a smaller fire perimeter.

Figure 4.9 shows how the solutions produced by each algorithm change under different combinations of wind and slope.

Table 4.13 – Average objective values of IBS and MIP across wind levels, grouped by slope.

Slope	IBS	MIP
Flat	609.33	780.93
Moderate	594.80	782.50
Steep	549.37	774.30

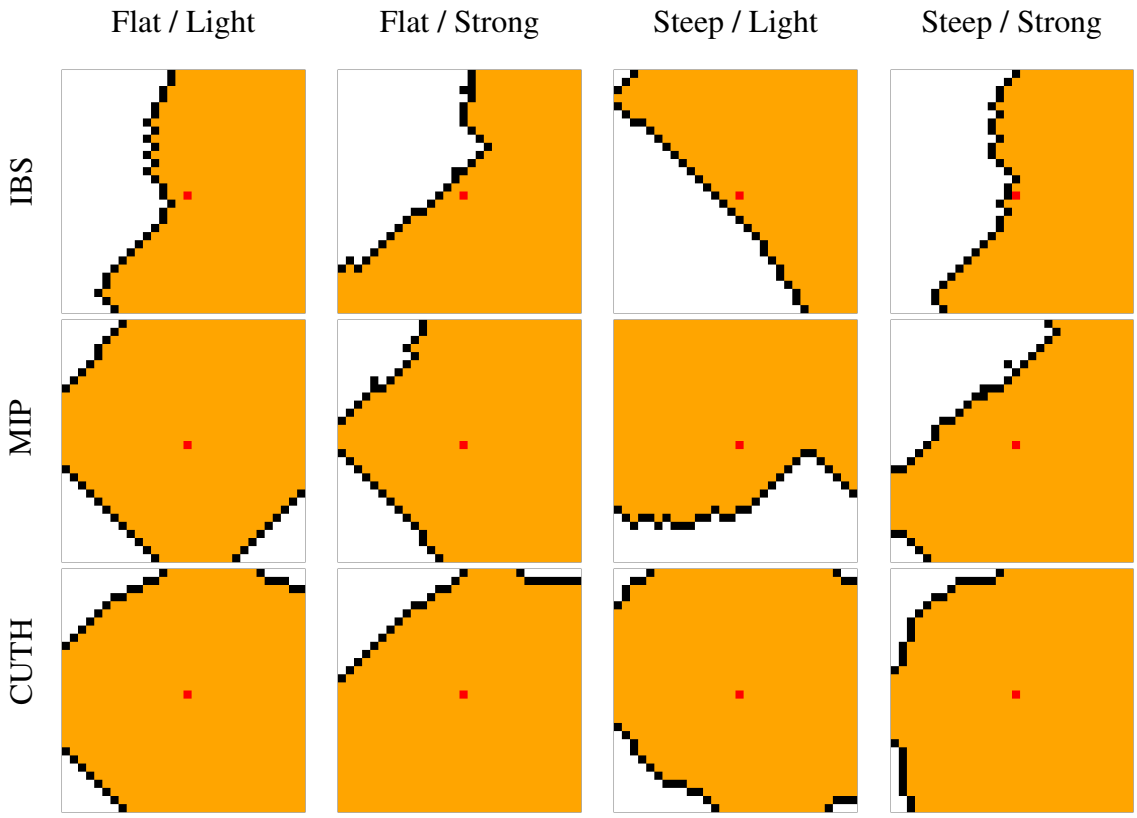


Figure 4.9 – Solutions for a sample of the 12 instances considered in the experiment on environmental factors. Rows correspond to algorithms (IBS, MIP, CUTH), and columns to instance configurations: (i) Flat slope, Light wind; (ii) Flat slope, Strong wind; (iii) Steep slope, Light wind; (iv) Steep slope, Strong wind. Each cell illustrates the best solution found by the algorithm for the corresponding instance. The red cell is the ignition point, the black cells are the ones that are protected by resources, and the orange cells are the ones that burned.

Table 4.14 – Results for varying slope and wind categories. The first group of columns reports the average relative gap to the best-known value (BKV) over all replications, with the number of times the BKV was matched shown in parentheses. Each instance was tested with 10 replications. The final column (Saved Vertices, %) indicates the percentage of vertices that did not burn in the best-known solution. Best results for each instance are in bold.

		Average Relative Gap to BKV (%)						Saved Vertices (%)
Slope	Wind	CUTH	IBS	ILS	LBBD	MIP	RS	
Flat	Light	34.1 (0)	**0.4** (1)	47.3 (0)	42.6 (0)	31.0 (0)	43.5 (0)	32.6
	Moderate	32.2 (0)	**0.5** (1)	46.4 (0)	43.8 (0)	26.4 (0)	40.4 (0)	32.1
	Strong	32.4 (0)	**0.6** (1)	48.6 (0)	48.8 (0)	29.0 (0)	43.4 (0)	33.2
Moderate	Light	33.7 (0)	**0.6** (2)	44.1 (0)	38.8 (0)	26.7 (0)	41.0 (0)	31.7
	Moderate	46.7 (0)	**1.2** (2)	60.4 (0)	61.2 (0)	40.4 (0)	54.9 (0)	38.1
	Strong	34.1 (0)	**0.7** (1)	49.5 (0)	50.3 (0)	31.5 (0)	43.6 (0)	33.6
Steep	Light	47.5 (0)	**0.7** (1)	58.2 (0)	56.0 (0)	32.8 (0)	54.1 (0)	37.3
	Moderate	57.1 (0)	**0.1** (5)	70.9 (0)	65.1 (0)	50.2 (0)	66.3 (0)	42.0
	Strong	51.0 (0)	**0.1** (5)	60.8 (0)	58.8 (0)	41.7 (0)	55.3 (0)	38.1

4.2.5 Resource Arrival Window

In this last experiment, we analyze how the length between the first and last times when resources are released affects the performance of the algorithms. Recall that the sequence of release times $T = (t_1, \ldots, t_{|T|})$ is composed of equidistant time points defined in terms of the first release time t_1 and the last release time $t_{|T|}$. The number of elements in T is an experimental factor, and we have studied it in Section 4.2.2. The first release time t_1 is defined as the first time when a percentage of the vertices burned if fire would propagate without any resources. The instance generator defines three levels for the first release time, namely early (5%), late (10%), and very late (20%). The last release time $t_{|T|}$ is defined in the same way, and the instance generator defines four levels: very early (60%), early (70%), late (80%), and very late (95%).

Increasing the first release time means that the fire has more time to propagate before any resources are released, resulting in a larger fire perimeter which, in real life, would be more difficult to control. In terms of instance complexity, this could result in a smaller search space for algorithms like MIP and LBBD, since several variables could have their values fixed. The impact of the last release time, however, is more difficult to predict. On one hand, the closer $t_{|T|}$ is to t_1, the more concentrated the resources are in time, which could make it easier to control the fire. On the other hand, a lower $t_{|T|}$ makes it more distant to the horizon H, which means that fire has more time to propagate after the last resource is released, which could make it more difficult to control the fire. Similar to the previous experiments, we do 10 replications per instance. The results are shown in Table 4.15 with the usual notation.

Similarly to the experiments on slope and wind, IBS achieves the lowest average gap in the 12 instances and reaches the best-known value at least once in 8 of them. The only cases where IBS fails to reach the BKV are when the last release time is set to very early, as well as one instance with first release time very late and last release time early. One interesting case is the instance with a late first release time and a very early last release time. MIP reaches the BKV in 5 out of 10 replications but still ends up with a much higher average gap (23.8%) than IBS (0.8%). A look at the raw data shows that MIP, in half the replications, obtains an objective value of 451, which is equal to the BKV. In other five replications, the average objective value is 665.4. Such a variation is not observed in IBS, whose standard deviation across the 10 replications is only 0.85.

The data also shows that LBBD reaches the best-known value in two instances

with very late first release time, when the last release time is either very early or early. These are the only cases where LBBD reaches the BKV across all experiments. This might be related to the fact that the search space becomes simpler when the release window is narrow and 20% of the vertices are already burned at t_1, which can reduce the number of active variables in the master problem. As we saw in Section 3.4, LBBD was among the best algorithms in the simpler instances used in previous work.

To illustrate the solutions in different release window settings, Figure 4.10 presents selected examples for each algorithm.

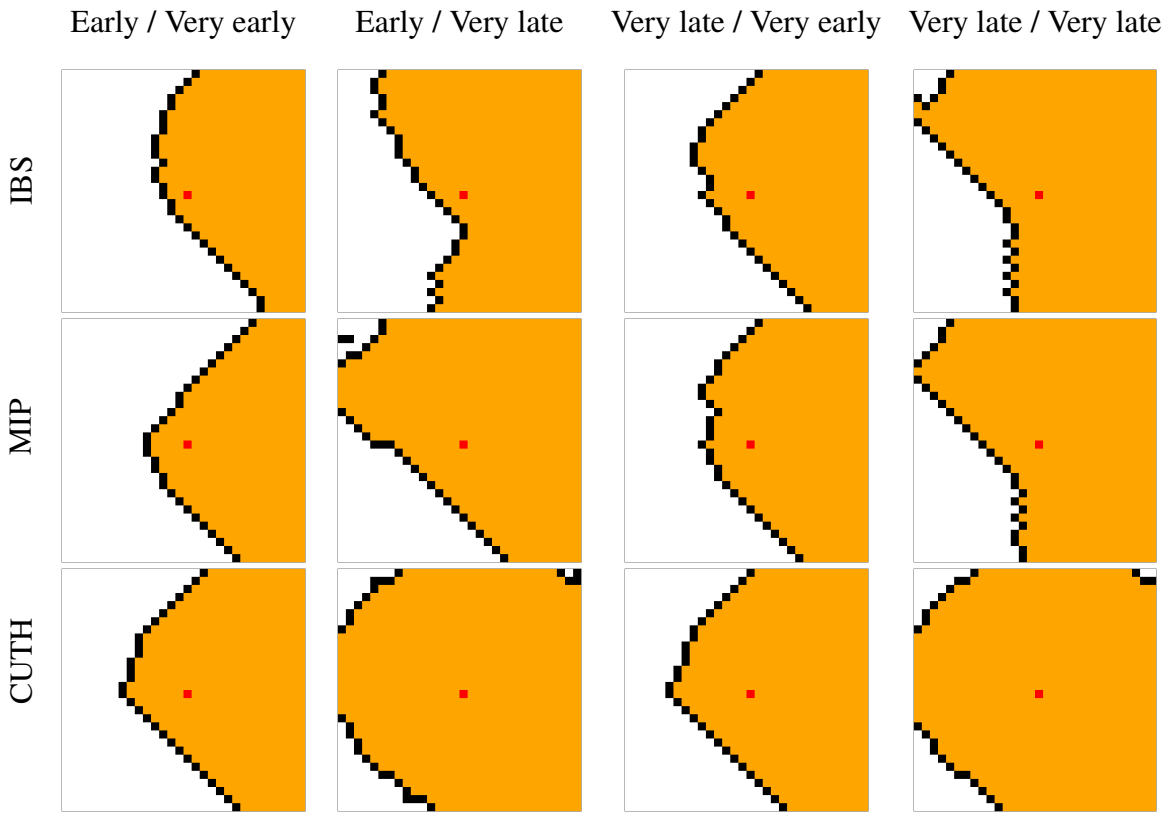


Figure 4.10 – Solutions for a sample of the 12 instances considered in the experiment on resource arrival window. Rows correspond to algorithms (IBS, MIP, CUTH), and columns to instance configurations: (i) Early first release time and very early last release time; (ii) Early / Very late; (iii) Very late / Very early; (iv) Very late / Very late. Each cell illustrates the best solution found by the algorithm for the corresponding instance. The red cell is the ignition point, the black cells are the ones that are protected by resources, and the orange cells are the ones that burned.

Table 4.15 – Results for different first and last release times. The first group of columns reports the average relative gap to the best-known value (BKV) across all replications. In parentheses, we report the number of replications in which the BKV was reached. We ran 10 replications for each configuration. The last column (Saved Vertices, %) shows the percentage of vertices that do not burn in the best-known solution. Best results for each instance are in bold.

		Average Relative Gap to BKV (%)						Saved Vertices (%)
First Release Time	Last Release Time	CUTH	IBS	ILS	LBBD	MIP	RS	
Early	Very Early	20.2 (0)	**1.9** (0)	104.9 (0)	94.0 (0)	43.0 (1)	102.5 (0)	51.6
	Early	38.1 (0)	**0.1** (5)	96.2 (0)	87.4 (0)	59.4 (0)	92.3 (0)	49.2
	Late	50.8 (0)	**0.1** (4)	89.5 (0)	79.7 (0)	66.3 (0)	82.3 (0)	47.3
	Very Late	33.7 (0)	**0.6** (2)	44.1 (0)	38.8 (0)	26.7 (0)	41.0 (0)	31.7
Late	Very Early	22.8 (0)	**0.8** (0)	98.3 (0)	90.9 (0)	23.8 (5)	95.1 (0)	49.9
	Early	28.0 (0)	**0.0** (10)	82.1 (0)	74.7 (0)	48.8 (0)	78.5 (0)	45.2
	Late	38.5 (0)	**0.0** (8)	71.4 (0)	59.5 (0)	42.9 (0)	67.6 (0)	41.8
	Very Late	29.2 (0)	**0.1** (4)	40.1 (0)	32.9 (0)	25.3 (0)	35.9 (0)	29.1
Very Late	Very Early	14.6 (0)	**0.3** (0)	75.8 (0)	65.9 (1)	36.3 (0)	73.4 (0)	43.6
	Early	18.4 (0)	**2.7** (0)	68.0 (0)	48.7 (1)	32.4 (0)	64.5 (0)	40.8
	Late	25.0 (0)	**0.0** (10)	54.3 (0)	43.3 (0)	20.4 (0)	50.6 (0)	35.4
	Very Late	20.5 (0)	**0.0** (10)	29.8 (0)	21.7 (0)	13.6 (1)	26.3 (0)	24.0

4.3 Discussion

The results across all experiments indicate that IBS is the most effective method overall, but it is not the best choice in all scenarios. In particular, it performs poorly in instances with few resources, few decision points, or low delay. In the first two cases, we believe this is due to the fire perimeter heuristic combined with the random sampling approach used to expand allocations. In the case of low delay, the reasons are less clear.

MIP shows strong results in extreme cases, especially when suppression capacity is low or high. It is also the only method to reach the BKV in instances with low delay or many resources. MIP does not always dominate in terms of average gap, but it adapts better than other methods to variations in delay and number of resources. This is somewhat expected, since it explores the search space systematically.

The matheuristic CUTH, which ranked poorly in earlier experiments, becomes competitive with other algorithms as instance size increases. In the 80×80 grid, it outperforms MIP in all three instances and IBS in two instances, but using considerably less time. When we restrict the time limit to 300 seconds, CUTH's performance compared to other algorithms is even better.

ILS and LBBD consistently show the worst performance across all experiments. ILS likely suffers from a weak implementation, as in large instances most of the execution time is spent building the initial solution. LBBD, in turn, does not scale well to the larger instances introduced in this work.

5 EXTENSIONS TO THE BASIC MODEL

Despite the computational difficulties already present in the basic model, it simplifies or omits many real-world aspects of wildfire suppression. This chapter explores how several such features, inspired by recent literature, can be incorporated into the basic combinatorial problem. We will introduce model features reflecting resource heterogeneity, spatial limitations, safety constraints, and expiration time constraints for attacks. This chapter is based on our publication Delazeri & Ritt (2024a).

Table 5.1 – Parameters of the model.

Parameter	Description
$I \subseteq V$	Ignition vertices
R	Set of resources
V_i^b	Set of vertices that can be the base location of resource $i \in R$
V_i^p	Set of vertices that can be protected by resource $i \in R$
$d(v, u)$	Distance between vertices v and u, $u, v \in V$.
w_v	Value of vertex $v \in V$
t_i	Release time of resource $i \in R$
c_i	Number of vertices that can be protected by resource $i \in R$
r_i	Range of resource $i \in R$ (i.e., maximum distance)
z_i^b	Safety time for the base location of resource $i \in R$
z_i^p	Safety time for the vertices protected by resource $i \in R$
d_i^b	Safety distance for the base location of resource $i \in R$
d_i^p	Safety distance for vertices protected by resource $i \in R$
δ_i	Delay caused by resource $i \in R$

Table 5.2 – Variables of the model.

Variable	Type	Description
a_v	Continuous	Fire arrival time at vertex $v \in V$
Δ_v	Continuous	Delay added to the outgoing arcs of vertex v
y_v	Binary	Indicates if vertex $v \in V$ burns before time H
s_{iv}	Binary	Indicates if vertex $v \in V_i^b$ is the base location of resource $i \in R$
r_{iv}	Binary	Indicates if fire at vertex $v \in V$ is being suppressed by resource $i \in R$

Fire Propagation and Delay

Fire arrival times a_v are calculated using propagation constraints similar to the basic model. The core constraint

$$a_v \leq a_u + t_{uv} + \Delta_u, \qquad uv \in A,$$

establishes that fire can reach vertex v from a predecessor u after the travel time t_{uv} plus any delay Δ_u resulting from suppression at u. Instead of a single ignition vertex s, the extended model considers a set of ignition vertices $I \subseteq V$, for which the fire arrival times are set to zero, i.e., $a_u = 0$ for $u \in I$.

Unlike the basic model, this formulation requires exact fire arrival times to properly handle some extensions, such as resource expiration times discussed later. Therefore, we introduce binary variables p_{uv} to indicate if u is the unique predecessor of v in the shortest path forest. The following constraints enforce this and ensure a_v represents the exact fire arrival time:

$$\begin{aligned} a_v &\geq a_u + t_{uv} + \Delta_u - M(1 - p_{uv}), && uv \in A, v \in V \setminus I, \\ \sum_{u \in N_v^-} p_{uv} &= 1, && v \in V \setminus I. \end{aligned}$$

Here, M is an upper bound on the fire arrival time at a vertex. We can set M to $(|V| - 1) \cdot t_{max} + C_{sum} \cdot \delta_{max}$, where $t_{max} = \max_{uv \in A} t_{uv}$, $C_{sum} = \sum_{i \in R} c_i$, and $\delta_{max} = \max_{i \in R} \delta_i$.

In this extended model, resources are heterogeneous, meaning different resources can have different characteristics. One such characteristic is the delay they impose on fire spread. The total delay Δ_v added to the outgoing arcs of a vertex v is therefore calculated based on the specific resource i assigned to protect it ($r_{iv} = 1$) and its delay value δ_i. The constraint is formulated as

$$\Delta_v = \sum_{i \in R : v \in V_i^p} \delta_i r_{iv}, \qquad v \in V.$$

Base Location and Range

Wildfire suppression resources often face spatial limitations in where they can operate from. Ground crews, for example, might need to stay close to accessible roads to ensure water supply from tankers, a constraint considered in the work of Avci et al. (2024). Aerial resources might have fewer restrictions on their positioning but still operate relative to specific target areas or landing zones. To incorporate such spatial restrictions, this model introduces the concept of a base location for each resource.

We define a binary variable s_{iv} which equals 1 if vertex v is selected as the operational base location for resource i. Each resource type can have a different set of permissible base locations, denoted by the parameter $V_i^b \subseteq V$. For instance, V_i^b for a ground crew could be limited to vertices representing road access points, while for a helicopter, V_i^b might encompass all vertices V. Each resource i must be assigned to at most one base location from its allowed set:

$$\sum_{v \in V_i^b} s_{iv} \leq 1, \qquad i \in R.$$

Furthermore, resources typically have a limited range from their base location. We model this using a parameter r_i, the maximum range for resource i. A resource i based at location u ($s_{iu} = 1$) can only be assigned to suppress fire at vertex v ($r_{iv} = 1$) if v is within this range:

$$r_{iv} \leq \sum_{u \in V_i^b : d(v,u) \leq r_i} s_{iu}, \qquad i \in R, v \in V_i^p.$$

Here, $d(v, u)$ represents the Euclidean distance between vertices, and V_i^p is the set of vertices that resource i can potentially protect.

Resource Capacity

Another difference from the basic model is how resource capacity is handled. In the extended formulation, a single resource i can be assigned to protect multiple vertices, up to a maximum limit defined by its capacity parameter c_i. This allows for modeling different types of resources, such as crews of varying sizes. The capacity limit is enforced

by the constraint

$$\sum_{v \in V_i^p} r_{iv} \leq c_i, \qquad i \in R,$$

where $r_v^i = 1$ indicates resource i protects vertex v within its set of protectable vertices V_i^p.

While one resource can protect multiple vertices, the model retains the constraint from the basic model ensuring that each vertex v can be protected by at most one resource

$$\sum_{i \in R : v \in V_i^p} r_{iv} \leq 1, \qquad v \in V.$$

Resource Deployment Time

In the basic model, where resources are homogeneous, it is generally optimal to deploy a resource as soon as it becomes available at its release time t_i. Allowing deployment at later times in that simpler context, as was possible in the original formulation by Alvelos (2018), could introduce unnecessary symmetries without improving the solution, as noted by Harris et al. (2023).

However, the extended model includes additional features (detailed in subsequent sections) which can make delaying the deployment of a resource beyond its release time t_i potentially necessary or optimal. As a result, we introduce a continuous variable d_i to represent the actual deployment time for resource i. This deployment time d_i must respect the resource's release time t_i and cannot occur after the planning horizon H, as any deployment after H would be ineffective for the objective function considered

$$t_i \leq d_i \leq H, \qquad i \in R.$$

Safety Constraints

Recent research has focused on modeling safety constraints into the models to approximate the real-world behavior of fire brigades. For instance, Avci et al. (2024) enforced a minimum spatial distance between ground resources and the fire front. Routing models like Granda et al. (2025) and Alvelos et al. (2025) prevent brigades from moving

through areas already reached by fire. Granda et al. (2025) further include a temporal constraint, ensuring that the time needed for a brigade to reach a location and complete its work is less than the fire's arrival time at that location. Alvelos et al. (2025) uses a safety radius around the resource's path, requiring that area to be free of fire at the time the resource is present.

The extended model proposes a safety mechanism that considers both spatial and temporal dimensions. It uses safety distance parameters (d_i^b for base locations, d_i^p for protected vertices) to define a safety zone around the resource's operational point, similar in concept to the safety radii used by Alvelos et al. (2025). However, it also adds a temporal buffer (z_i^b or z_i^p). This requires that the defined spatial zone remains fire-free not just at the moment of deployment, but for a specified duration after deployment.

Specifically, if vertex v is chosen as the base location for resource i ($s_{iv} = 1$), which is deployed at time d_i, then the fire arrival time a_u at any vertex u within the safety distance d_i^b must be greater than or equal to the deployment time plus the safety buffer ($d_i + z_i^b$). A similar condition applies if resource i is protecting vertex v ($r_{iv} = 1$), using the parameters d_i^p and z_i^p. These conditions are enforced through the following constraints, formulated using a 'big-M' approach, where $M_i^b = H + z_i^b$ and $M_i^p = H + z_i^p$ serve as sufficiently large constant:

$$-M_i^b(1 - s_{iv}) + d_i + z_i^b \leq a_u, \qquad i \in R, v \in V_i^b, u \in V : d(u,v) \leq d_i^b$$
$$-M_i^p(1 - r_{iv}) + d_i + z_i^p \leq a_u, \qquad i \in R, v \in V_i^p, u \in V : d(u,v) \leq d_i^p$$

Expiration Constraints

Previous work, such as those of Belval et al. (2015), Avci et al. (2024), Granda et al. (2025), and Alvelos et al. (2025), treat suppression actions as creating a complete block to fire propagation. This is often representative of indirect tactics like constructing firebreaks, where the suppression action remains effective regardless of when the fire arrives.

However, direct suppression attacks, like applying water or retardant, are typically most effective only when the fire is imminent at the target location. Applying water long before the fire arrives offers little benefit. To capture this aspect of direct attacks, we introduce an expiration time parameter e_i for each resource. This parameter defines the maximum allowable interval between deploying the resource and the fire's arrival.

If resource i protects vertex v ($r_{iv} = 1$), it ensures the fire arrives at v (a_v) no later than e_i time units after the resource was deployed at time d_i:

$$M(1 - r_{iv}) + d_i + e_i \geq a_v, \qquad i \in R, v \in V_i^p.$$

The constant M is an upper bound on the fire arrival time, similar to the one used in the fire propagation constraints.

Objective Function

Previous work employs various objective functions. The basic model presented earlier focused on minimizing the number of vertices burned. Other approaches have minimized total value lost to the fire, resource costs, or the total time taken by the suppression operation (Belval et al., 2015; Avci et al., 2024; Granda et al., 2025; Alvelos et al., 2025). Some works even consider multiple objectives (Granda et al., 2025; Alvelos et al., 2025).

The extended model focuses on minimizing the total value lost due to fire within the planning horizon H. Each vertex v is assigned a value w_v, which allows us to differentiate between areas of varying importance. This generalizes the basic model's objective of minimizing the number of burned vertices. As in the basic model, the binary variable y_v indicates if vertex v burns before time H.

$$y_v \geq 1 - a_v/H, \qquad v \in V.$$

The objective function then seeks to minimize the total weighted value of the burned vertices

$$\text{min.} \quad \sum_{v \in V} w_v y_v.$$

Discussion

The improved MIP formulation introduced in Section 3.1 is already computationally demanding. On a 30×30 grid with ten decision points, a commercial solver needs 900 seconds to reach an average objective of 778.9 (Section 4.2). Adding only the ex-

act fire arrival variables p_{uv}, the first step toward the extended formulation, prevents the same solver, on identical hardware, from finding a feasible solution after 7200 seconds. This slowdown is not unique. Other extended models in the literature, such as the routing approaches of Granda et al. (2025) and Alvelos et al. (2025), are tractable only for small instances.

Because current exact and heuristic methods still struggle even with the basic model, improving algorithms for this simpler problem remains the logical first target. We therefore introduce the extended formulation primarily as a reference point for future algorithmic developments. Algorithm components, potentially first validated on the basic model, can eventually be applied to formulations that more closely represent real-world practice.

6 CONCLUSIONS AND FUTURE WORK

Effectively managing wildfire suppression requires handling numerous decisions under uncertainty and time constraints, and computational tools can assist in this process. This dissertation addressed the optimization problem of allocating suppression resources with release times on a graph-based representation of a landscape to minimize the total burned area by a given horizon. Progress solving this type of problem has been partly constrained by limitations in existing solution methods when applied to complex scenarios and by a reliance on benchmark instances lacking scale or physical realism. In the next paragraphs, we highlight the main contributions of this dissertation and discuss future research directions.

Previous Mixed-Integer Programming (MIP) formulations were computationally expensive and sometimes failed to find feasible solutions. We demonstrated that a simpler MIP model, combining earlier propagation ideas with time-constrained resources, results in a competitive algorithm compared to the state-of-the-art. The improved MIP formulation significantly outperformed the formulation by Alvelos (2018), obtained better results than the logic-based Benders decomposition of Harris et al. (2023) in most scenarios, and in some cases even obtained the best solutions among all available methods.

Few tailored algorithms existed for this problem in the literature, and this dissertation explored two distinct novel approaches. First, given that the problem's structure of sequential decision-making aligns well with constructive algorithms, we designed an Iterated Beam Search (IBS). Our experiments demonstrated that IBS is the best-performing algorithm in most tested scenarios. Second, drawing inspiration from the common real-world strategy of constructing firebreak lines, we developed a fast Cut-based Heuristic (CUTH). This heuristic identifies relevant vertex cuts in the graph and transforms them into feasible resource allocations. CUTH runs in a few seconds and its relative performance improves significantly as the grid size increases.

Instances available in the literature were few, small, lacked physical realism, and did not allow systematic algorithm evaluation. To overcome this, we created a new instance generator using Rothermel's model, which enables controlled experiments to understand algorithm behavior under diverse conditions. As demonstrated by our experiments, the instance generator revealed strengths and weaknesses of current methods, such as how the performance of IBS strongly depends on the number of decision points, and how LBBD, which was a state-of-the-art algorithm, struggles on slightly larger instances

than the ones available in the literature.

The basic graph-based model studied in this dissertation ignores several operational details of fire suppression resources. While our algorithmic contributions focused on this basic model, which already is difficult to solve for large instances, we also presented an extended mathematical formulation. This extension demonstrates how factors such as resource heterogeneity, spatial limitations, safety constraints, and attack expiration times can be integrated into the basic MIP formulation. Operational factors like these have been incorporated into related graph-based suppression models in recent literature (see Section 2.5.2).

Several avenues for future research remain, particularly concerning improving solution quality. Our experiments revealed that the Iterated Beam Search (IBS) algorithm obtains better solutions as the number of decision points increases, even though this adds more constraints on resource allocation. Understanding the mechanisms behind this phenomenon could lead to the design of an enhanced beam search that performs consistently well across scenarios with both few and many decision points. Another possible direction would be to improve the Iterated Local Search (ILS) algorithm. The algorithm to construct initial solutions is computationally expensive and struggles to find good solutions in large instances. This algorithm could be replaced by the cut-based heuristic (CUTH) proposed in this dissertation, which is faster and produces better solutions.

Future efforts could also focus on obtaining better bounds for the problem. Although the proposed MIP formulation showed significant improvement, its large root gaps show that its linear programming relaxation is weak. Strengthening the MIP formulation, perhaps through valid inequalities or alternative modeling approaches, could improve the solver's ability to find and prove optimal solutions.

Lastly, future research could involve designing algorithms to solve the extended model from Chapter 5. Adapting the methods developed in this dissertation, or the algorithms from the literature, to the constraints of the extended model would likely require significant modifications. The extended model can serve as a testbed for future algorithm development, where heuristic components can be evaluated not only for their performance on the basic problem, but also for how easily they can be adapted to handle the additional constraints.

REFERENCES

ALVELOS, F. Mixed integer programming models for fire fighting. In: GERVASI, O. et al. (Ed.). **Computational Science and Its Applications – ICCSA 2018**. Cham: Springer International Publishing, 2018. p. 637–652. ISBN 978-3-319-95165-2.

ALVELOS, F.; MARTO, M.; MENDES, A. Dispatching, positioning and routing resources for wildfire initial attack. **Networks**, Wiley, April 2025. ISSN 1097-0037. Ahead of print. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1002/net-.22278>.

ANDREWS, P. L. **Modeling wind adjustment factor and midflame wind speed for Rothermel's surface fire spread model**. Fort Collins, CO, 2012. Available from Internet: <https://www.fs.usda.gov/rm/pubs/rmrs_gtr266.pdf>.

ANDREWS, P. L. **The Rothermel Surface Fire Spread Model and Associated Developments: A Comprehensive Explanation**. Fort Collins, CO, 2018. 132 p. Available from Internet: <http://dx.doi.org/10.2737/RMRS-GTR-371>.

ANDREWS, P. L.; HEINSCH, F. A.; SCHELVAN, L. **How to Generate and Interpret Fire Characteristics Charts for Surface and Crown Fire Behavior**. Fort Collins, CO, 2011. 48 p. Available from Internet: <https://www.fs.usda.gov/rm/pubs/rmrs_gtr253-.pdf>.

AVCI, M. G. et al. The wildfire suppression problem with multiple types of resources. **European Journal of Operational Research**, v. 316, n. 2, p. 488–502, 2024. ISSN 0377-2217.

AZEVEDO, A. T. de et al. Solving the 3d container ship loading planning problem by representation by rules and meta-heuristics. **International Journal of Data Analysis Techniques and Strategies**, Inderscience Publishers, Geneva, v. 6, n. 3, p. 228–260, 2014.

BELVAL, E. J.; WEI, Y.; BEVERS, M. A mixed integer program to model spatial wildfire behavior and suppression placement decisions. **Canadian Journal of Forest Research**, v. 45, n. 4, p. 384–393, April 2015. ISSN 1208-6037.

BURIOL, L. S.; RESENDE, M. G. C.; THORUP, M. Speeding up dynamic shortest-path algorithms. **INFORMS Journal on Computing**, v. 20, n. 2, p. 191–204, 2008. ISSN 1526-5528.

CROCE, F. D.; GHIRARDI, M.; TADEI, R. Recovering beam search: Enhancing the beam search approach for combinatorial optimization problems. **Journal of Heuristics**, Springer Science and Business Media LLC, v. 10, n. 1, p. 89–104, January 2004. ISSN 1381-1231.

DELAZERI, G.; RITT, M. A general model for wildfire suppression problems. In: **Anais do Simpósio Brasileiro de Pesquisa Operacional**. Galoa, 2024. (SBPO 2024, Vol 56, 2024). ISSN 2965-1476. Available from Internet: <http://dx.doi.org/10.59254/sbpo-2024-193493>.

DELAZERI, G.; RITT, M. Iterated beam search for wildland fire suppression. In: SMITH, S.; CORREIA, J.; CINTRANO, C. (Ed.). **Applications of Evolutionary Computation**. Cham: Springer Nature Switzerland, 2024. p. 273–286. ISBN 978-3-031-56852-7.

DIMOPOULOU, M.; GIANNIKOS, I. Spatial optimization of resources deployment for forest-fire management. **International Transactions in Operational Research**, Wiley, v. 8, n. 5, p. 523–534, September 2001. ISSN 1475-3995.

Encyclopaedia Britannica. **Beaufort Scale**. 2025. Encyclopaedia Britannica, online ed. Available from Internet: <https://www.britannica.com/science/Beaufort-scale>.

FINNEY, M. A. **FARSITE: Fire Area Simulator-model development and evaluation**. Fort Collins, CO, 1998. Accessed: May 2, 2025. Available from Internet: <https://www-.fs.usda.gov/rm/pubs/rmrs_rp004.pdf>.

FINNEY, M. A. Mechanistic modeling of landscape fire patterns. In: MLADENOFF, D. J.; BAKER, W. L. (Ed.). **Spatial Modeling of Forest Landscape Change: Approaches and Applications**. New York: Cambridge University Press, 1999. p. 186–209. ISBN 9780521631228.

FINNEY, M. A. Fire growth using minimum travel time methods. **Canadian Journal of Forest Research**, v. 32, n. 8, p. 1420–1424, 2002. ISSN 1208-6037.

FRANDSEN, W. H. Fire spread through porous fuels from the conservation of energy. **Combustion and Flame**, Elsevier BV, v. 16, n. 1, p. 9–16, February 1971. ISSN 0010-2180.

GRANDA, B.; VITORIANO, B.; FIGUEIRA, J. R. A mathematical programming approach for a wildfire suppression problem. **Operational Research**, Springer Science and Business Media LLC, v. 25, n. 1, January 2025. ISSN 1866-1505.

HARRIS, M. G.; FORBES, M. A.; TAIMRE, T. Logic-based benders decomposition for wildfire suppression. **Computers & Operations Research**, Elsevier BV, v. 160, p. 106392, December 2023. ISSN 0305-0548.

HOF, J. et al. A timing-oriented approach to spatial allocation of fire management effort. **Forest Science**, Springer Science and Business Media LLC, v. 46, n. 3, p. 442–451, August 2000. ISSN 1938-3738.

INFORMS. **Operations Research: Frequently Asked Questions**. 2025. Accessed: March 11, 2025. Available from Internet: <https://www.informs.org/About-INFORMS-/What-is-Operations-Research>.

IPCC. **Climate Change 2022: Impacts, Adaptation and Vulnerability: Working Group II Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change**. Cambridge: Cambridge University Press, 2023.

JEWELL, W. S. Forest Fire Problems–A Progress Report. **Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), v. 11, n. 5, p. 678–692, October 1963. ISSN 1526-5463.

Joint Economic Committee, U.S. Senate. **Climate-exacerbated wildfires cost the U.S. between $394 to $893 billion each year in economic costs and damages**. 2023. Available from Internet: <https://web.archive.org/web/20231114011935/https://www-.jec.senate.gov/public/index.cfm/democrats/reports?id=E31AF93E-34C7-4C35-A416-533FF796369B>.

LEE, Y. et al. Deploying initial attack resources for wildfire suppression: spatial coordination, budget constraints, and capacity constraints. **Canadian Journal of Forest Research**, v. 43, n. 1, p. 56–65, January 2013. ISSN 1208-6037.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. A. (Ed.). **Handbook of Metaheuristics**. Boston, MA: Springer US, 2003. p. 320–353. ISBN 978-0-306-48056-0.

MARTELL, D. L. A review of operational research studies in forest fire management. **Canadian Journal of Forest Research**, v. 12, n. 2, p. 119–140, 1982. ISSN 1208-6037.

MARTELL, D. L.; THOMPSON, M. P.; CALKIN, D. E. Wildfire management analytics: Bridging the gap between research and practice. **OR/MS Today**, v. 49, n. 5, October 2022. Available from Internet: <https://pubsonline.informs.org/do/10.1287/orms.2022-.05.11/full/>.

MCLENNAN, J. et al. Decision making effectiveness in wildfire incident management teams. **Journal of Contingencies and Crisis Management**, Wiley, v. 14, n. 1, p. 27–37, March 2006. ISSN 1468-5973.

MENDES, A. B.; ALVELOS, F. Iterated local search for the placement of wildland fire suppression resources. **European Journal of Operational Research**, v. 304, n. 3, p. 887–900, 2023. ISSN 0377-2217.

MERWE, M. van der et al. A mixed integer programming approach for asset protection during escaped wildfires. **Canadian Journal of Forest Research**, v. 45, n. 4, p. 444–451, April 2015. ISSN 1208-6037.

Munich Re. **Wildfires and bushfires – Climate change increasing wildfire risk**. 2023. Available from Internet: <https://www.munichre.com/en/risks/natural-disasters-/wildfires.html>.

National Wildfire Coordinating Group. **Field Manager's Course Guide: S-290 Intermediate Wildland Fire Behavior**. 2024. Available from Internet: <https://web-.archive.org/web/20250503025950/https://fs-prod-nwcg.s3.us-gov-west-1.amazonaws-.com/s3fs-public/publication/pms437-1.pdf>.

OW, P. S.; MORTON, T. E. Filtered beam search in scheduling. **International Journal of Production Research**, Taylor & Francis, v. 26, n. 1, p. 35–62, January 1988. ISSN 1366-588X.

PERLIN, K. Improving noise. **ACM Transaction on Graphics**, Association for Computing Machinery, New York, NY, USA, v. 21, n. 3, p. 681–682, July 2002. ISSN 0730-0301.

PRICE, S. et al. **Field guide for classifying standard fire behavior fuel models in sagebrush steppe**. 2024. Available from Internet: <https://web.archive.org-/web/20241219181516/https://greatbasinfirescience.org/wp-content/uploads/2024/12-/FuelBedCharacterization_Final_GBFSE_2.0.pdf>.

RACHMAWATI, R. et al. A model for solving the prescribed burn planning problem. **SpringerPlus**, Springer Science and Business Media LLC, v. 4, n. 1, October 2015. ISSN 2193-1801.

REUTERS. **Death toll from Hawaii wildfires drops to 97**. 2023. <https://www.reuters.com/world/us/death-toll-hawaii-wildfires-drops-97-missing-is-now-31-hawaii-governor-2023-09-15>, Accessed: January 19, 2024.

ROTHERMEL, R. C. A mathematical model for predicting fire spread in wildland fuels. Ogden, UT, p. 40, 1972. Available from Internet: <https://web.archive.org/web-/20250405160256/https://www.fs.usda.gov/rm/pubs_int/int_rp115.pdf>.

SCOTT, J. H.; BURGAN, R. E. **Standard fire behavior fuel models: a comprehensive set for use with Rothermel's surface fire spread model**. Fort Collins, CO, 2005. Available from Internet: <https://web.archive.org/web/20250503034522/https://www.fs-.usda.gov/rm/pubs_series/rmrs/gtr/rmrs_gtr153.pdf>.

SNELL, C. et al. **Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters**. 2024. Available from Internet: <https://arxiv.org/abs-/2408.03314>.

United Nations. **As Wildfires Increase, Integrated Strategies for Forests, Climate and Sustainability Are Ever More Urgent**. 2023. <https://www.un.org/en/un-chronicle/wildfires-increase-integrated-strategies-forests-climate-and-sustainability-are-ever-0>, Accessed: Jan 19, 2024.

VANDERBEI, R. J. **Linear Programming: Foundations and Extensions**. Cham: Springer International Publishing, 2020. ISSN 2214-7934. ISBN 9783030394158.

WAGNER, C. E. V. A simple fire-growth model. **The Forestry Chronicle**, v. 45, n. 2, p. 103–104, April 1969. ISSN 1499-9315.

WAGNER, C. E. V. Conditions for the start and spread of crown fire. **Canadian Journal of Forest Research**, v. 7, n. 1, p. 23–34, 1977. ISSN 1208-6037.

WEI, Y.; RIDEOUT, D. B.; HALL, T. B. Toward efficient management of large fires: A mixed integer programming model and two iterative approaches. **Forest Science**, Springer Science and Business Media LLC, v. 57, n. 5, p. 435–447, October 2011. ISSN 1938-3738.

WEISE, D. R.; BIGING, G. S. A qualitative comparison of fire spread models incorporating wind and slope effects. **Forest Science**, v. 43, n. 2, p. 170–180, May 1997. ISSN 1938-3738.

APPENDIX A — RESUMO EXPANDIDO

A.1 Modelos e Algoritmos para Problemas de Supressão de Incêndios Florestais

Os incêndios florestais representam um desafio global crescente, com aumento notável em sua frequência e intensidade nas últimas décadas. Eventos extremos, muitas vezes exacerbados pelas mudanças climáticas, resultam em perdas econômicas substanciais, danos ambientais severos e custos sociais elevados, incluindo a perda de vidas e propriedades. Gerenciar e combater esses incêndios é uma tarefa complexa, exigindo a tomada de decisões críticas sob condições de incerteza, pressão temporal e rápida evolução do fogo.

A dificuldade no combate a incêndios florestais não reside apenas nos desafios logísticos e ambientais, mas também na significativa carga cognitiva imposta aos tomadores de decisão. A necessidade de processar rapidamente grandes volumes de informação dinâmica, avaliar múltiplos cursos de ação e antecipar o comportamento do fogo em terrenos complexos coloca uma pressão imensa sobre as equipes de gerenciamento. Essa complexidade inerente e a carga cognitiva associada podem ser substancialmente mitigadas pelo uso da tecnologia, particularmente através de ferramentas computacionais projetadas para auxiliar na análise de cenários e na tomada de decisões.

Uma das principais áreas científicas dedicadas ao desenvolvimento de tais ferramentas computacionais de apoio à decisão é a Pesquisa Operacional (PO). A PO utiliza modelos matemáticos, algoritmos e técnicas de otimização para analisar problemas complexos e identificar as melhores soluções ou estratégias possíveis. Tais problemas geralmente envolvem a alocação eficiente de recursos limitados sob diversas restrições. Existe uma longa e estabelecida linha de trabalho em Pesquisa Operacional voltada especificamente para a criação de métodos e ferramentas computacionais para apoiar as diversas fases do gerenciamento e combate a incêndios florestais.

Embora diversas técnicas de PO tenham sido aplicadas ao gerenciamento de incêndios florestais, uma abordagem promissora é a modelagem baseada em grafos. Este tipo de modelo permite integrar a dinâmica espacial da propagação do fogo com o impacto das ações de supressão de forma explícita. Essa capacidade de capturar a interação entre o avanço do incêndio e as intervenções de combate possibilita responder a perguntas importantes, como avaliar se a quantidade de recursos disponíveis é suficiente para

proteger determinadas áreas-alvo ou identificar quais pontos de intervenção geram maior impacto na contenção do fogo. Dentro deste paradigma de modelagem, esta dissertação foca especificamente no problema de otimizar a alocação de recursos de supressão, cuja disponibilidade é restrita ao longo do tempo, com o objetivo de minimizar a área total queimada até um determinado horizonte de planejamento.

A.1.1 Contribuições da Dissertação

Trabalhos anteriores que abordaram este problema específico se limitaram a poucas instâncias e de pequena escala. Além disso, essas instâncias eram geralmente baseadas em dados sintéticos, com pouca conexão com a física real da propagação do fogo. Isso dificulta a avaliação sistemática dos métodos propostos, bem como sua aplicabilidade em cenários práticos. Esta dissertação visa avançar o estado da arte nesta área, apresentando um conjunto de contribuições algorítmicas e metodológicas projetadas para enfrentar problemas de supressão de incêndios mais complexos e realistas.

A primeira contribuição algorítmica é o desenvolvimento de um algoritmo Iterated Beam Search (IBS). O IBS é uma heurística baseada em busca em árvore que explora de forma inteligente o espaço de soluções. Ele mantém um conjunto limitado (o 'feixe' ou beam) das melhores alocações parciais de recursos a cada passo de tempo e expande essas alocações usando heurísticas específicas (h_1 e h_2) e um conceito de 'perímetro de fogo' para priorizar vértices candidatos. Os experimentos computacionais demonstraram que o IBS obteve a melhor qualidade de solução na maioria dos cenários testados, superando significativamente os métodos existentes na literatura e outras abordagens desenvolvidas neste trabalho.

A segunda contribuição algorítmica é uma nova heurística denominada Cut-based Heuristic (CUTH). Inspirada na estratégia prática de construção de linhas de contenção de fogo, a CUTH funciona identificando cortes de vértices relevantes no grafo que separam a área queimada da área não queimada. Esses cortes são então convertidos em alocações factíveis de recursos por meio de um procedimento guloso. Uma vantagem da CUTH é sua velocidade: ela encontra soluções competitivas em questão de segundos, mesmo para instâncias grandes. Os resultados mostram que seu desempenho relativo a outros métodos melhora consideravelmente com o aumento do tamanho da instância, tornando-a uma alternativa promissora para cenários de grande escala onde outros métodos se tornam computacionalmente inviáveis.

A terceira contribuição algorítmica consiste em uma formulação aprimorada de Programação Inteira Mista (MIP) para o problema. Combinando restrições de propagação de fogo mais simples de trabalhos anteriores com a modelagem de recursos com restrição temporal introduzida por Alvelos (2018), a nova formulação demonstrou ser significativamente superior aos modelos MIP anteriores. Em contraste com formulações prévias que frequentemente falhavam em encontrar soluções viáveis dentro de limites de tempo razoáveis, o modelo proposto consistentemente encontrou soluções de alta qualidade para todas as instâncias testadas, provando-se competitivo e, em alguns casos, até superior às heurísticas especializadas e outros métodos exatos.

Uma limitação identificada na literatura foi a carência de instâncias de benchmark adequadas para avaliar rigorosamente os algoritmos de supressão. As instâncias existentes são pequenas, baseadas em dados de propagação aleatórios sem correlação espacial ou base física, e não permitem variar sistematicamente as características do problema para testar a robustez dos algoritmos. Essa lacuna dificulta a comparação justa entre diferentes abordagens e a compreensão de como os algoritmos se comportariam em condições diversas. A falta de instâncias desafiadoras e realistas impede o progresso sistemático na área.

Para suprir essa lacuna, a contribuição metodológica desta dissertação foi o desenvolvimento de um novo gerador de instâncias baseado em modelos físicos de propagação do fogo, especificamente o modelo de Rothermel. Este gerador permite criar paisagens com topografia e condições de vento variáveis, e calcular tempos de propagação do fogo mais realistas. Além disso, permite configurar flexivelmente o tamanho da grade, a quantidade e a distribuição temporal dos recursos de supressão, e o atraso causado por eles. A utilização deste gerador nos experimentos permitiu uma análise mais profunda do desempenho dos algoritmos, revelando suas forças e fraquezas sob diferentes condições.

Apesar dos avanços algorítmicos e metodológicos focados no modelo básico de supressão, é importante reconhecer que este modelo ainda simplifica consideravelmente a complexa realidade das operações de combate a incêndios. Ele ignora diversas restrições operacionais fundamentais na prática, como as restrições de posicionamento e de alcance operacional das brigadas e, mais importante, considerações de segurança para as equipes de combate. Planos de supressão gerados sem considerar tais fatores podem ser irrealistas ou inviáveis de implementar.

Visando aproximar a modelagem da realidade operacional, esta dissertação também apresenta um modelo matemático estendido que incorpora vários desses fatores.

Embora a resolução computacional deste modelo estendido não tenha sido o foco principal, sua formulação demonstra como integrar aspectos como heterogeneidade de recursos (diferentes tipos de equipes ou equipamentos com capacidades, alcances e efetividades distintas), limitações espaciais (necessidade de operar a partir de locais de base específicos), restrições de segurança (exigência de zonas espaciais e temporais livres de fogo ao redor das operações) e tempo de validade para ataques diretos (como lançamento de água, que só é eficaz se o fogo chegar logo após a aplicação).

A.1.2 Conclusões

Em suma, esta dissertação abordou o problema de otimização da alocação de recursos para supressão de incêndios florestais em modelos baseados em grafos, apresentando contribuições tanto no desenvolvimento de algoritmos quanto na metodologia de avaliação. Foram propostos dois novos algoritmos heurísticos (IBS e CUTH) com desempenhos complementares, uma formulação MIP aprimorada que supera modelos anteriores, e um novo gerador de instâncias baseado em física que permite testes mais rigorosos e realistas. Além disso, foi delineado um modelo estendido que incorpora restrições operacionais relevates na literatura. Espera-se que essas contribuições representem um passo importante no desenvolvimento de ferramentas computacionais mais robustas, eficazes e aplicáveis para apoiar a tomada de decisão no combate aos incêndios florestais.

APPENDIX B — SUPPLEMENTARY DETAILS

B.1 Related Work

This note details the connection between the models proposed by Hof et al. (2000) and Wei et al. (2011) and the graph-based shortest-path formulation used in this dissertation.

Hof et al. (2000) model the problem on an $m \times n$ grid of cells, indexed by row i and column j. Their goal is to maximize the arrival time of fire at a specific cell (m, n), given an ignition starting at cell (a, b). The variables include T°_{ij} to denote the time the fire front ignites (enters) cell (i, j), T'_{ij} to denote the time the fire front leaves cell (i, j), F_{ij} to denote the fuel available for combustion in cell (i, j) after suppression treatment, and X_{ij} to denote the proportion of cell (i, j) receiving treatment. The authors use Ω_{ij} to denote the set of neighboring cells (h, k) that can potentially ignite cell (i, j), $f_{ij}(\cdot)$ to denote a function relating available fuel in cell (i, j) to the fire duration within that cell, $\overline{F}_{ij}$ to denote the available fuel in cell (i, j) before treatment, γ_{ij} to denote the proportion of fuel that can be removed by treatment in cell (i, j), and $\overline{X}$ to denote the total treatment budget. The formulation presented by Hof et al. (2000) is as follows

$$\begin{aligned}
\text{max.} \quad & T^{\circ}_{mn} && && \text{(H.0)}\\
\text{s.t.} \quad & T^{\circ}_{ab} = 0 && && \text{(H.1)}\\
& T^{\circ}_{ij} \leq T'_{hk} && \forall i, j,\ \forall (h, k) \in \Omega_{ij} && \text{(H.2)}\\
& T'_{ij} - T^{\circ}_{ij} = f_{ij}(F_{ij}) && \forall i, j && \text{(H.3)}\\
& F_{ij} = \overline{F}_{ij} - \gamma_{ij}\overline{F}_{ij}X_{ij} && \forall i, j && \text{(H.4)}\\
& 0 \leq X_{ij} \leq 1 && \forall i, j && \text{(H.5)}\\
& \sum_i \sum_j X_{ij} \leq \overline{X} && && \text{(H.6)}
\end{aligned}$$

Constraints H.1 sets the ignition time at the starting cell (a, b) to 0. Constraint H.2 ensures a cell ignites only after the fire leaves a neighboring cell capable of igniting it. Constraint H.3 defines the time the fire takes to pass through cell (i, j) as a function of the available fuel F_{ij}. Constraint H.4 calculates the available fuel after treatment X_{ij} reduces the initial fuel $\overline{F}_{ij}$. Constraint H.5 limits the treatment proportion to the interval $[0, 1]$, while constraint H.6 restricts the total treatment budget to $\overline{X}$. The objective function

maximizes the time the fire reaches the target cell (m, n).

Notice that we can merge the equality constraints H.3 and H.4 into constraint H.2, leaving us with

$$T^{\circ}_{ij} \leq T^{\circ}_{hk} + f_{hk}(\overline{F}_{hk} - \gamma_{hk}\overline{F}_{hk}X_{hk}) \qquad \forall i, j,\ \forall (h,k) \in \Omega_{ij} \tag{H.7}$$

In practice, $f_{ij}(\cdot)$ is a linear function of the decision variable X_{ij}, parameterized by the amount of available fuel $\overline{F}_{ij}$ and the proportion of fuel that can be removed by treatment γ_{ij}. For each cell (i, j), let α_{ij} and β_{ij} denote the coefficients of the linear function $f_{ij}(\cdot)$. We obtain the following constraint for the fire propagation

$$T^{\circ}_{ij} \leq T^{\circ}_{hk} + \alpha_{hk}X_{hk} + \beta_{hk} \qquad \forall i, j,\ \forall (h,k) \in \Omega_{ij} \tag{H.8}$$

We can reinterpret this model on a graph $G = (V, A)$, where each cell (i, j) corresponds to a vertex $v \in V$ and the set of arcs A represents the adjacency relation defined by Ω. Let us replace T°_{ij} with a_v to denote fire arrival time at vertex v, X_{ij} with r_v to denote the resource allocation decision at vertex v, and $\overline{X}$ with k to denote the total treatment budget. If we let $s \in V$ denote the ignition vertex and $t \in V$ denote the target vertex, we can rewrite the model as follows:

$$\begin{aligned}
(M_{hof}) \quad \text{max.} \quad & a_t && && \text{(H.9)}\\
\text{s.t.} \quad & a_s = 0 && && \text{(H.10)}\\
& a_v \leq a_u + \alpha_u r_u + \beta_u && uv \in A && \text{(H.11)}\\
& 0 \leq r_v \leq 1 && v \in V && \text{(H.12)}\\
& \sum_{v \in V} r_v \leq k && && \text{(H.13)}
\end{aligned}$$

It is easy to see that the model above is a simplified version of the one presented in Section 3.1. Besides the fact that the suppression variables r_v do not have a temporal dimension (resources do not have release times), we can also see that fire propagation from a vertex u to a vertex v does not depend on the direction of the arc uv, but only on whether u has a resource or not.

We now analyze the model proposed by Wei et al. (2011). The notation is the same used in the previous model and in Section 3.1, with a few additions. Each vertex v has a weight w_v, which is the value lost if the vertex burns, and a predicted flame length

F_v, which could be estimated using a fire simulator. There is also a flame length threshold FL that indicates the maximum flame length that can be suppressed, and a maximum number of resources k that can be allocated. Wei et al. (2011) do not consider resource release times, so the resource allocation variables r_v do not have a time dimension, as in the MIP model described in Section 3.1. The model is as follows:

$$
\begin{aligned}
(M_{wei}) \quad \text{min.} \quad & \sum_{v \in V} w_v y_v && && \text{(W.1)} \\
\text{s.t.} \quad & a_s = 0 && && \text{(W.2)} \\
& a_v \leq a_u + t_{uv} + \Delta r_u && uv \in A && \text{(W.3)} \\
& y_v \geq 1 - a_v / H && v \in V && \text{(W.4)} \\
& \sum_{v \in V} r_v \leq k && && \text{(W.5)} \\
& r_v = 0 && v \in V, F_v > FL && \text{(W.6)} \\
& r_v, y_v \in \{0, 1\} && v \in V &&
\end{aligned}
$$

The objective function minimizes the total value lost in the vertices that burn. Constraints W.2 and W.3 are the usual fire propagation constraints. Constraint W.4 sets variable y_v to 1 if the vertex v burns before the horizon H, and to 0 otherwise. Constraint W.5 ensures that the total number of resources allocated to the vertices v is not greater than k. As mentioned in Section 2.5.2, Wei et al. (2011) are the first to consider firefighter safety constraints in the model. They do this by fixing the value of r_v to 0 if the predicted flame length F_v at vertex v is greater than the threshold FL, as shown in constraint W.6.

B.2 Benchmark Instances

Table B.1 reports the number of vertices and the optimal objective value for each instance. The number of vertices is always below 400, the size of a full 20×20 grid. This reduction results from the instance generation process: since the optimization horizon was arbitrarily defined, some vertices burn after this horizon and are excluded.

Table B.1 – Number of vertices and optimal objective value for each instance.

ID	$\|V\|$	Optimal Obj.
L0A	289	189
L1A	294	189
L2A	282	190
L3A	294	207
L4A	317	216
L5A	312	226
L6A	331	239
L7A	327	246
L0B	289	195
L1B	294	196
L2B	282	196
L3B	294	213
L4B	317	226
L5B	312	235
L6B	331	249
L7B	327	253

B.3 Random Search Algorithm

Algorithm 12 presents a pseudo-code for the Random Search algorithm. The idea is to build an allocation Λ incrementally from the first decision point to the last. At each decision point $t \in T$, we build a set C of candidate vertices are not burned yet and that are not protected by Λ. We then randomly select $|R_t|$ vertices from C and update the allocation Λ accordingly. If the new allocation Λ is better than the current best allocation Λ^*, we update Λ^* with Λ. The algorithm stops when the termination criteria are met (e.g., a maximum running time).

Algorithm 12: RANDOMSEARCH

$\Lambda^* \leftarrow \Lambda_0$
while Termination criteria not met **do**
  $\Lambda \leftarrow \Lambda_0$
  for $t \in T$ **do**
    $C \leftarrow V \setminus (B_t^\Lambda \cup P^\Lambda)$
    for $i \in R_t$ **do**
      $v \leftarrow$ Pick a random element from C
      $\Lambda_i \leftarrow v$
      $C \leftarrow C \setminus \{v\}$
    end
  end
  if $|B_H^\Lambda| < |B_H^{\Lambda^*}|$ **then**
    $\Lambda^* \leftarrow \Lambda$
  end
end
return Λ^*